Fortezza Program Overview

The National Security Agency

Version 4.0a
February, 1996

This document provides the reader with a broad overview of the Fortezza program. It introduces the Fortezza Crypto Card, the enabling technologies, common messaging protocols, important technical terms, and various application possibilities. Greater detail on all the topics can be found in other Fortezza documents noted in the Reference appendix.

Table Of Contents

List of Figures

1.0 Introduction

Businesses today demand accurate and secure handling of electronic information. The National Security Agency's Fortezza program addresses this demand by providing the technology to enable value-added security services for Unclassified but Sensitive information. Fortezza technology provides data integrity, originator authentication, non-repudiation (undeniable proof of one's identity), and confidentiality (data privacy). Fortezza personalizes security through an individualized cryptographic device, a PC Card called the Fortezza Crypto Card (the Card). The Card contains the user's unique cryptographic key material and related information, and executes the cryptologic algorithms. A sophisticated infrastructure has been designed to generate, distribute, and control the cryptographic keys, control the integrity of the data on the Card, and disseminate required cryptographic and system information. Fortezza interfaces and specifications are designed with an "open system" philosophy. This permits seamless integration of the Fortezza technology into most data communication hardware platforms, operating systems, software application packages, and computer network configurations and protocols.

1.1 In The Beginning

The National Security Agency (NSA) developed the Fortezza program for the Department of Defense (DoD) in response to the growing need for economical and secure electronic messaging. The DoD is incorporating the Fortezza technology into its Defense Message System (DMS) to secure its Unclassified but Sensitive information. The Fortezza technology satisfies the DMS security architecture with a user friendly, inexpensive, cryptographic mechanism that provides writer to reader message confidentiality, integrity, authentication, non-repudiation, and access control to messages, components, and systems. While the DMS exposed the DoD to the need for the Fortezza technology, the same security requirements are valid today for civilian agencies, commercial businesses, and private citizens.

1.2 What's In A Name

Fortezza has had several name changes through the years. The original name, circa 1991, for the Fortezza program was the Pre Message Security Protocol (PMSP). The cryptographic device was a "smart card." In 1993, the name was changed to "MOSAIC," and the cryptographic device changed to a PC Card and named the "Tessera Crypto Card." The program was then incorporated into a larger NSA program, Multi-Level Information Systems Security Initiative (MISSI), in 1994. The program name was changed again to "Fortezza" and the PC Card renamed the "Fortezza Crypto Card."

2.0 Data Security And Fortezza

2.1 Data Security And Fortezza: Why We Need Them

The increasing availably and use of electronic data presents new problems for individuals and businesses. The parties involved in the exchange of information can no longer use a persons voice, handwriting, or face to recognize the other party. However, the recipient must still have confidence in the integrity of the information and the identity its originator. Developers of electronic messaging and data handling products must provide security services so parties can have confidence in the information. These services must be resolutely enforced yet unobtrusively performed. Fortezza, with the Card and the supporting software, provides the developer with a flexible, modular, security “tool-kit.”

2.2 Security Services For Electronic Data: What We Need

Accurate and secure data must have four security attributes:

- Data Integrity
- User Authentication
- User Non-repudiation
- Data Confidentiality

2.2.1 Data Integrity

The data integrity attribute indicates the data has been processed by both the originator and the recipient, through a “Hash” function. The data in the message are read through a mathematical algorithm which uses every bit in the message to form a uniformly sized string of bits unique to that message. Any change in the message, even a single bit, will cause the recipient’s Hash value to differ from the sender’s Hash value. To provide integrity over the Hash value, a method to secure the value and verify the originator of the Hash function is required. This requires the message to have the user authentication attribute.

2.2.2 User Authentication

User Authentication assures the recipient of the originator’s identity by cryptographically processing the data with an algorithm which incorporates parameters unique to the originator. The mechanism to perform this check must assure that the data could only be sent from the declared author (the author’s identity has not been forged). The algorithm must produce a result that is easy to verify yet difficult to forge. Authenticating the originator of a message is performed by the hash and digital signature functions.

2.2.3 User Non-repudiation

Non-repudiation is a condition whereby the author of the data (i.e., a message) cannot repudiate (disavow) the validity of the result used to authenticate the identity of that user. The technique used to identify the author must be strong enough so the authenticity of the message originator can be proven to a third party (someone other than the recipient). Non-repudiation is accomplished by digital signatures.

2.2.4 Data Confidentiality

Confidentiality provides data privacy by encrypting and decrypting data, whereby only the intended recipient can read a message. Encrypted data renders the sensitive data, nonsensitive. Thus, encrypted data needs less physical data protection. To provide confidentiality, a technique must be established to provide a unique “key” for encryption of the data and the capability to transmit the key and other necessary information to the recipient to decrypt the data. The key provides a variable for each encryption session. This means that multiple encryption of the same plaintext will result in different cipher (encrypted) text. Some algorithms also require an Initialization Vector (IV), for added variability.

2.3 Fortezza Cryptographic Functions: What The Card Can Do

2.3.1 Hash

The Hash function provides a check for data integrity. The Card performs a mathematical Hash algorithm on the data, producing a 160 bit (20 byte) Hash value. This Hash value will be unique for any given message because any change in the data, even a single bit, will change the Hash Value. The Card uses the SHA-1 (Federal Information Processing Standard) (FIPS 180-1) Hash algorithm.

2.3.2 Digital Signature

The digital signature allows a message originator to sign (cover) data (e.g., the Hash value). This provides the recipient with the means to verify the identity of the originator (user authentication and non-repudiation). The digital signature is somewhat analogous to superimposing a user’s thumbprint over some data. Any change in the data will cause a change in the “thumbprint.” A recipient can then use the cryptography to verify the originator’s digital signature. This verifies the originator’s identity. It also may allow the recipient to verify the integrity of the message if the signature covers the Hash value. The Card uses the Digital Signature Standard (DSS) (FIPS-186), Digital Signature Algorithm (DSA).

2.3.3 Confidentiality

Encryption and decryption is implemented using a single, shared “key” between the originator and the recipient. The “key” is a random number generated by the Card and provided as an input, along with the message data and an IV for encryption and decryption. The Card uses the SKIPJACK algorithm (FIPS-185). The “key” is securely transmitted to the recipient by a secure Key Exchange.

2.3.4 Key Exchange

The Key Exchange process wraps (similar to encrypt) the key necessary to implement the encryption algorithm. The wrapped key can then be safely transmitted to the recipient. The key exchange process allows only the intended recipient to unwrap the needed key to decrypt data. The Card uses the Key Exchange Algorithm (KEA).

2.4 Fortezza Cryptography Architecture: How The Card's Algorithms Work

2.4.1 Asymmetric Cryptography

Asymmetric cryptography is often referred to as "Public Key Cryptography." Each participant has a pair of keys, a public key, "y", and a private key, "x." The y value is mathematically derived from the x value. The strength of the cryptography is the mathematical infeasibility of deriving a private key, or unknown x from a public key, or known y. Asymmetric cryptography is used in the Card for digital signatures and key exchanges. The following examples should shed some light on public key cryptography and how the technique is used in Fortezza.

2.4.1.1 Hash And Digital Signature

While an entire message can be signed, the user should first hash the message and then sign that Hash value, since processing time is smaller for 160 bits. The signature is calculated on the data and a random number generated on the Card. The random value assures that signatures on the same data will be different.

This process assures the recipient of the integrity of the data and originator. The Hash provides the data integrity. The digital signature over the Hash value authenticates the originator (signer) and provides integrity over the Hash value.

Figure 1 shows how Fortezza implements this principle. UserA desires data integrity and user authentication security services for a message. UserA executes the Hash function on the message. UserA then executes the digital signature algorithm over the Hash value. The digital signature requires the Hash value, UserA's private key x for the DSA, $UserA_{(x,DSA)}$, and a random number (generated internally by the Card). The resulting signature value is two cascaded 20 byte parameters, "r" and "s." This value is sent with the message to the recipient, UserB. UserB receives the message and independently performs the Hash function on the message. UserB uses that Hash value, the digital signature value, and UserA's public key, $UserA_{(y,DSA)}$ to verify that the message was signed by the expected originator and the data was not altered.

The Fortezza user's private value, $x_{(DSA)}$, is stored on the user's card and is not user readable. The user's public value, $y_{(DSA)}$, should be publicly available (discussed in the Key Management section).

2.4.1.2 Timestamp

The Card uses the hash and sign functionality to provide an application a timestamp function. This function uses the real time clock on the Card to provide a secure method of determining when a message was processed by the originator (at least when the timestamp was applied). The process concatenates the original

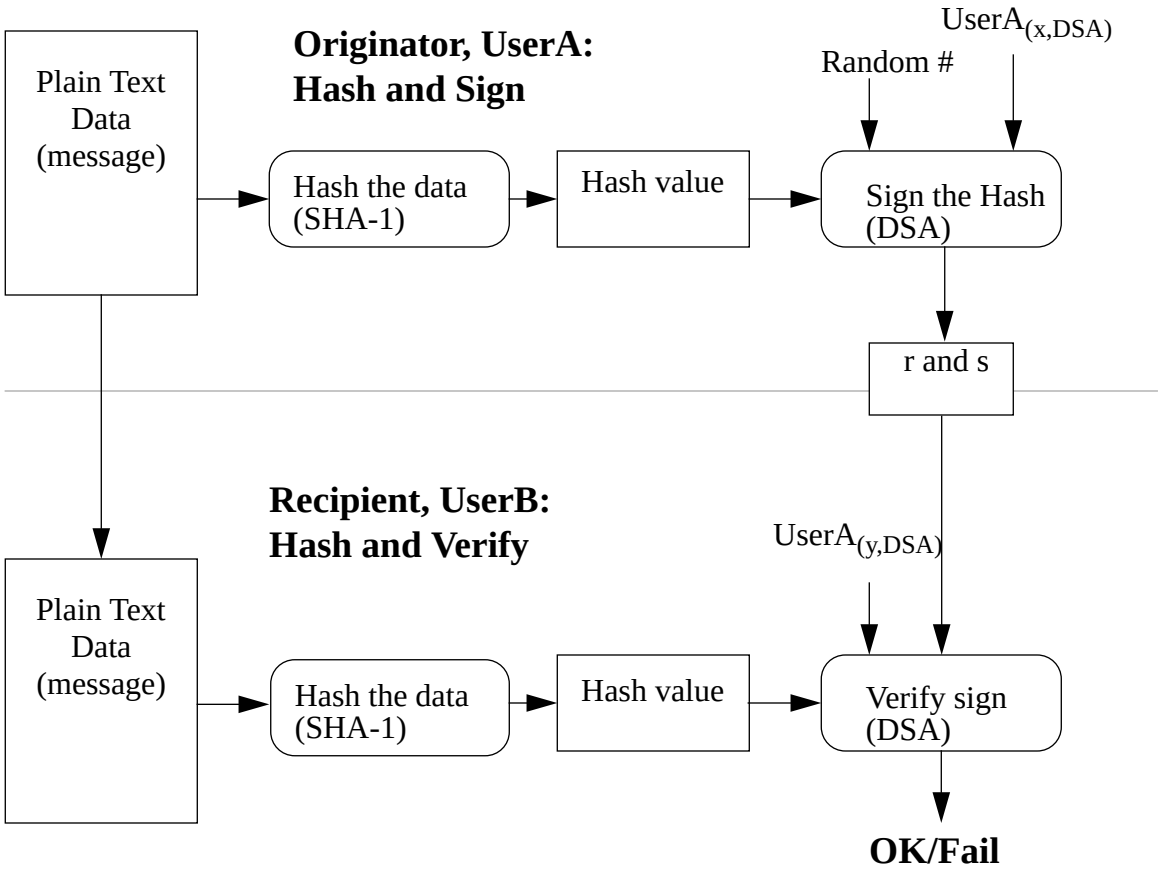


Figure 1: Hash and Digital Signatures

message Hash value with the Card's internal time (so the recipient knows it was "this data at this time"). The Card then performs a hash on that value, and signs that Hash value with a private $x_{(DSA)}$ that is embedded in all Fortezza Crypto Cards (Using an internal DSA key prevents the originator from modifying the timestamp). The result can be verified by the recipient.

2.4.1.3 Key Exchange

Asymmetric cryptography is used to secure the transmission of the Message Encryption Key (MEK), which is the key used by the originator to encrypt a message and by the recipient to decrypt that message (symmetric cryptography will be discussed in the next section). Figure 2 shows how the originator securely sends the MEK to the recipient by generating a Token Encryption Key (TEK). The TEK is produced by the KEA, which uses the sender's $x_{(KEA)}$, the recipient's $y_{(KEA)}$, and two random numbers as input. The TEK wraps ("encrypts") the MEK. The recipient unwraps

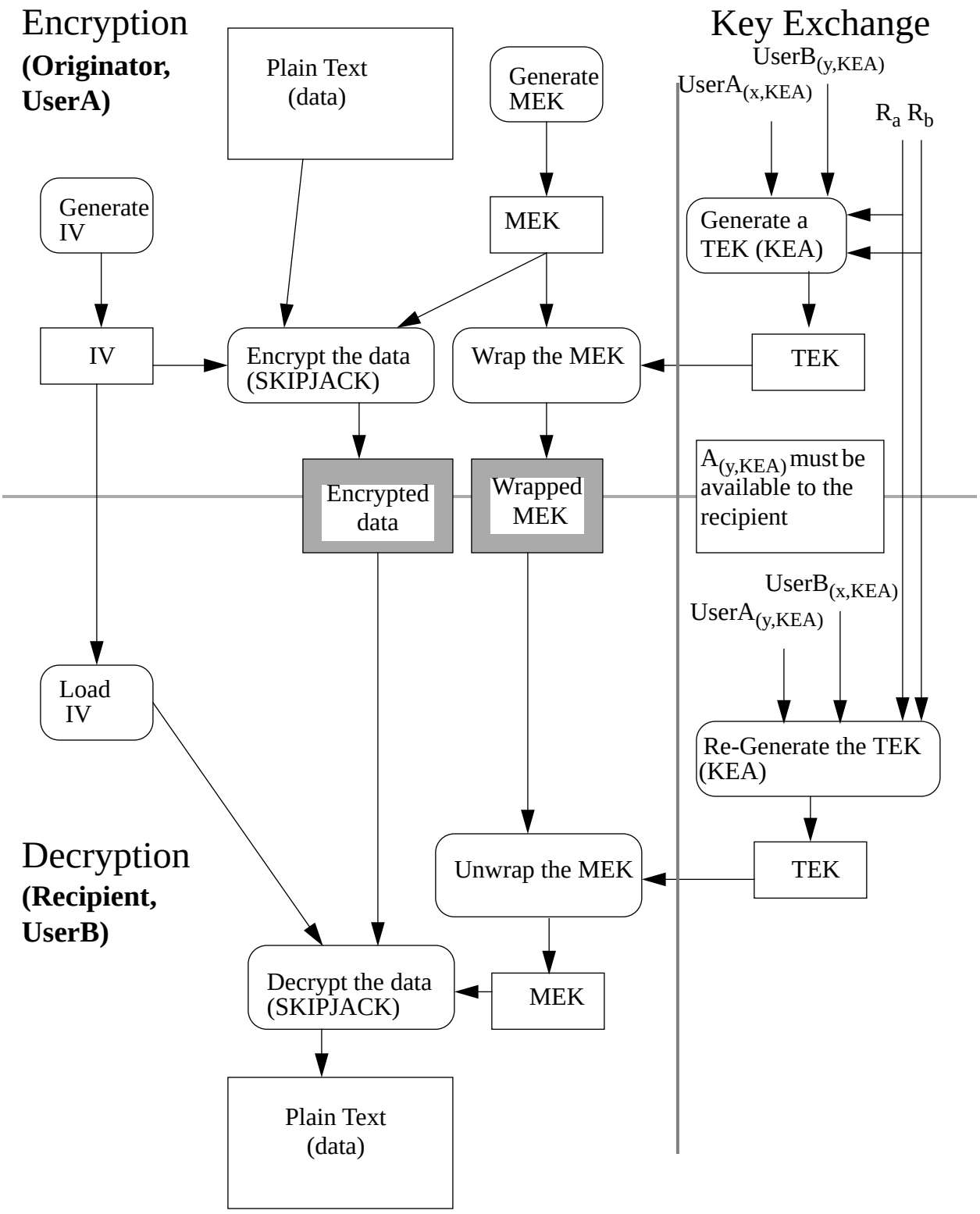


Figure 2: Encryption, Decryption and Key Exchange

("decrypts") the MEK after generating the same TEK, by using its $x_{(KEA)}$ and the originator's $y_{(KEA)}$. This exchange permits only the recipient to recover the MEK, verify the sender of the MEK, and decrypt the message.

The procedure is significant because the time-consuming process of encrypting a large message or file is performed just once per session, regardless of the number of recipients. The faster TEK generation and MEK wrap procedure will be performed for each recipient.

The right side of Figure 2 shows a typical key exchange. UserA desires data confidentiality (i.e., encryption of the message). UserA generates the MEK, then generates a TEK using $UserA_{(x,KEA)}$, the recipient's (UserB), $UserB_{(y,KEA)}$ and two random numbers (R_a and R_b). UserA then wraps the MEK with the TEK and transmits it, the message, and the random numbers to UserB. UserB regenerates the TEK with its private value, $UserB_{(x,KEA)}$, the originator's public value, $UserA_{(y,KEA)}$, and the random numbers. UserB then unwraps the MEK and uses that value to perform the symmetric decryption of the data.

The Fortezza user's private value, $x_{(KEA)}$, is stored on the user's card and is not user readable. The user's public value, $y_{(KEA)}$, should be publicly available (discussed in the key management section).

2.4.2 Symmetric Cryptography

Symmetric cryptography uses one key to encrypt and decrypt data. The originator must ensure that the recipient has the same key for decryption. This key must remain secure between the intended parties or the security of the message will be compromised. The Card handles several encryption modes and requires an IV for each mode.

2.4.2.1 Encryption and Decryption

The left side of Figure 2 shows the process of symmetric encryption. The originator generates an MEK and an IV on the Card. These parameters are used with the SKIPJACK algorithm to synchronize with the eventual decryption process and add some randomness to the encryption process. The originator then wraps the MEK with the TEK and transmits the wrapped MEK and the IV to the recipient.

2.4.2.2 Archive Data

Fortezza provides several methods to archive data (such as e-mail). One technique is similar to the aforementioned process, whereby the user generates and uses an MEK. In another method, the user encrypts and decrypts the data using a special key called the Storage Key (Ks), stored internally in the user's card. Each card has

a Ks unique to that card. This feature relieves the user from the key management and the security processing required in asymmetric cryptography.

3.0 Common Messaging Protocols and Fortezza

The Card is not a stand-alone message processor, it simply performs cryptologic processes on the given data. This architecture allows the Card to be used in most computer and network environments. But, for the Card to be useful, it must be integrated into an application. The software developer must understand how to implement the Card's functions and the appropriate standards and protocols to provide secure electronic messages. These standards and protocols address the messaging formats, security concerns, and infrastructure requirements (such as a directory to help "find" people). Below are some, though not all, messaging protocols commonly associated with the Fortezza program.

3.1 Simple Mail Transport Protocol (SMTP)

SMTP is an e-mail protocol defined by the Request For Comment (RFC) 821. It evolved over the years from earlier mail standards. SMTP is limited because the character set is limited to a subset of ASCII characters, the lines are restricted to no greater than 1000 characters, and the body length is limited. Users must convert non-ASCII data such as facsimile, digitized voice, and pictures into standard ASCII using such common methods as "base 64 bit encoding" and "uuencoding." SMTP usually relies on the Transport Control Protocol /Internet Protocol (TCP/IP) stack. This limited protocol still dominates the e-mail market. However, many e-mail developers are moving toward two more flexible messaging formats.

3.2 Multipurpose Internet Mail Extension (MIME)

MIME provides a mechanism to allow the transmission of non-ASCII data using the SMTP RFC 821 protocol. RFC 1521 defines the MIME specification. SMTP developers are migrating to the MIME specification because it improves the reliability and interoperability of e-mail. The format will also support new, non-e-mail applications. MIME provides Fortezza compliant applications, such as e-mail, with the required flexibility to transmit encrypted data through SMTP or X.400 message systems, and the ability to send cryptographically altered messages across computer platforms, and to define the content (message) as a Fortezza compliant message.

3.3 X.400

X.400 represents a set of protocol recommendations, established by the International Telecommunications Union- Telecommunications Standardizations Section (ITU-T) (formerly the Consultive Committee on International Telephone and Telegraph (CCITT)), which define how messages comprising many types of data besides text are communicated between different computer systems. It does not describe how an application is to appear to a user nor does it define the transport mechanism. The X.400 environment is encompassed in the Message Handling System (MHS). The MHS (see Figure 3) consists of: the User Agent (UA), which processes messages; the Message Store (MS), which can store, list, summarize, delete, retrieve, and

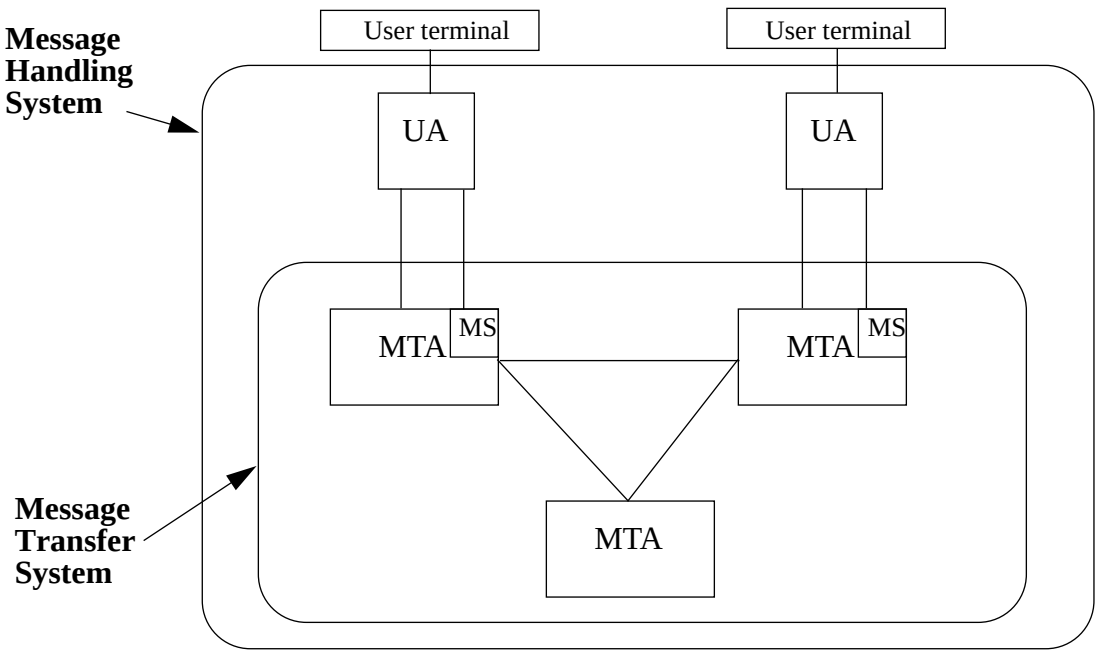


Figure 3: X.400 Message Handling System

manage messages for the UA; the Message Transfer Agent (MTA), which routes the messages; and the Message Transfer System (MTS), which is the MTA backbone. Although the X.400 specifies an entire transport system, the components noted handle how the messages are formatted by an application, not how the actual bits are transmitted. An X.400 message is designated as P2 or P22 if it complies with the 1984 or 1988 recommendations, respectively. The ITU-T releases new specifications as needed.

3.4 Allied Communications Protocol 123 (ACP 123)

ACP 123 is a superset of the 1988 X.400 interpersonal message type P22. It defines increased options and the policies and procedures required in military messaging. ACP 123 also identifies the services and protocols necessary to ensure interoperability between the Military MHS and the commercial X.400 MHS. DMS e-mail will comply with ACP 123. An ACP 123 body part is being designated as P772, although this designation has not been approved by the ITU-T.

3.5 X.500

X.500 is the ITU-T recommendation for directory service. The X.500 Directory (the Directory) can be conceptualized as a telephone book for computer networks. The Directory stores, and makes available many “attributes” of an entity (the entity can be a person or a machine). These attributes may include the entity’s e-mail address,

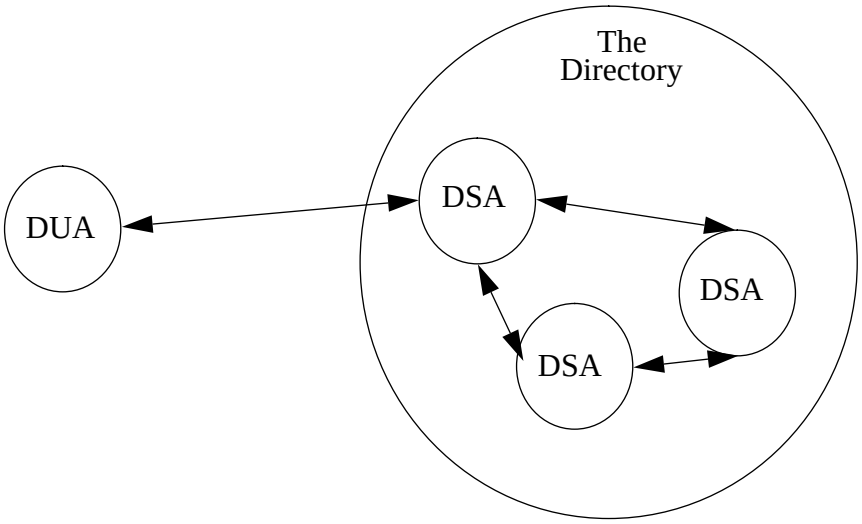


Figure 4: X.500 Directory

Fortezza related information, and other information the user wishes to make available. Figure 4 shows how architecture of the Directory is a distributed database (over multiple physical locations). The arrangement of the database is called the Directory Information Tree (DIT). Together the entire database is referred to as the Directory Information Base (DIB). The component that handles all requests to the Directory from users is the Directory System Agent (DSA). The DSA may handle a request for information itself, ask another DSA, or advise the requester to contact another DSA. The Directory User Agent (DUA) formulates, formats, and sends requests to the DSA, then presents the results to the user. The user's interface to the DUA is vendor dependent. The X.500 recommendation does not specify the relationship between an X.400 UA and the DUA. This is consistent with the ITU-T recommendations in that the implementation of the recommendations are not specified. Fortezza certificates (explained later) use the ITU-T X.509 Recommendation for the implementation of security services for the Directory and its users.

3.6 Abstract Syntax Notation.1 (ASN.1)

While not a message protocol, ASN.1 is critical to ITU-T and Fortezza applications. ASN.1 is a data encoding (not encryption) scheme that allows data to be transferred to a wide variety of applications. The ASN.1 encoding rules provide applications with standardized "tags" which identify the data structure or object (e.g., a bit map, a message's header information, or a directory request). Messaging protocols and the Fortezza program require a number of these data structures and objects. Fortezza also requires the use of an extremely structured enforcement of ASN.1 called Distinguished Encoding Rules (DER), since any variance to the data bits caused by an implementation of ASN.1 would ruin the data integrity. Special ASN.1 compilers (e.g., DSET) relieve developers of the often complex encoding and decoding rules.

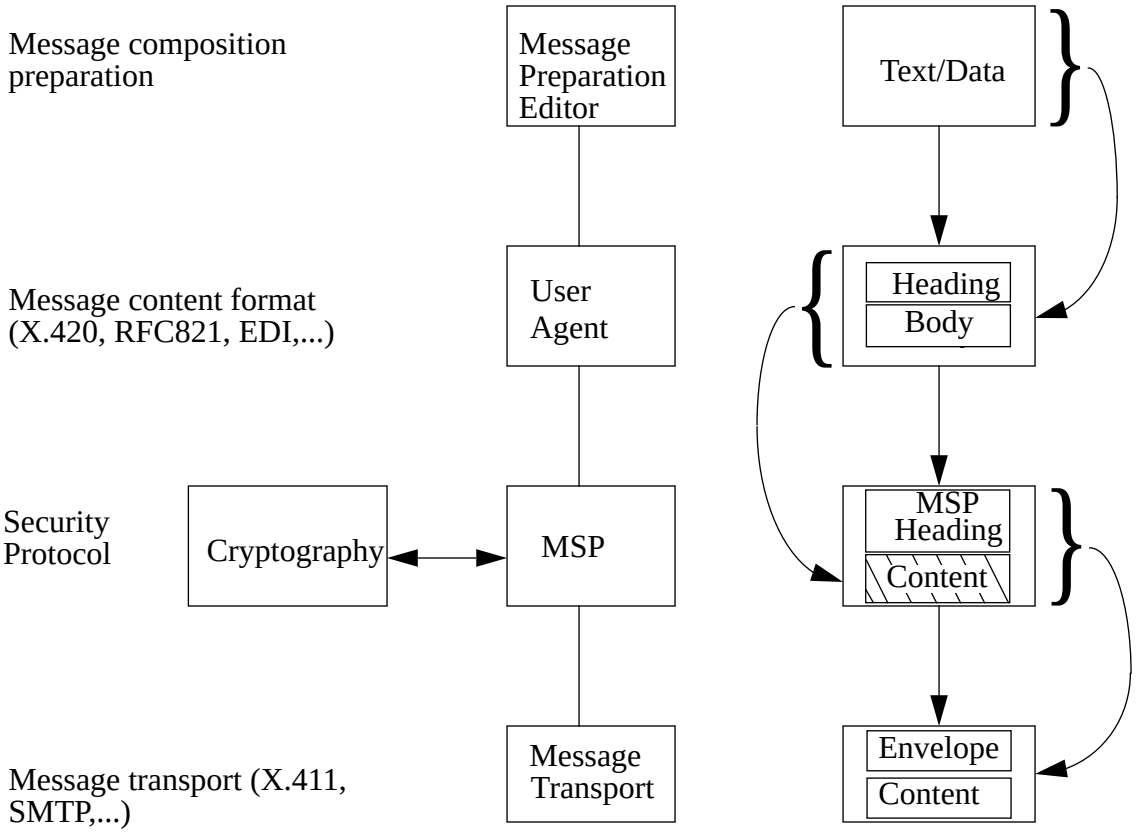


Figure 5: MSP Architecture

3.7 Secure Data Network System (SDNS)

The SDNS is a set of protocols developed jointly by commercial vendors and the NSA, and managed within the NSA's MISSI program. The combined effort provides the basis for improved security technology, interoperable security standards, and cost-effective security components for computers and telecommunications networks and messaging systems. The most relevant documents for implementing Fortezza are the SDN.701, SDN. 702, and SDN.704.

3.7.1 SDN.701: Message Security Protocol (MSP)

The MSP specifies how the required message security services (defined earlier) are integrated into messages. This protocol, while not specifically designed for Fortezza, does handle the Fortezza cryptography and key management architecture. Figure 5 shows the path of a message as it evolves from a plain message to an MSP formatted and secured final message, complete with header information. The MSP specification is applicable for many messaging environments, including, though not limited to: the X.400 MHS, MIME, Electronic Data interchange (EDI), and file transfer. A message using the MSP is being referenced as P42 (formerly it was P48), though the ITU-T has not approved the designation.

3.7.2 SDN.702: SDNS Directory Specifications

SDN.702 defines how the MSP is integrated into the Directory. The MSP (SDN.701) and the Fortezza key management implementation requires public access to many objects and attributes. SDN.702 specifies the format of such objects, attributes, and features as the user's public cryptographic information, Fortezza infrastructure objects, and Fortezza key management control information.

3.7.3 SDN.704: MIME/SMTP with MSP

This specification defines the message format for an MSP protected SMTP/MIME message. The format details how the MSP processes the MIME message, the resulting format of the message, and how the recipient is notified that the message has been processed by the MSP. Fortezza enhanced products must use this specification to ensure e-mail interoperability.

4.0 The Fortezza System

4.1 Key Management Components

A practical implementation of a cryptographic system requires an efficient infrastructure. This is the behind-the-scenes processing that makes the entire system function and enforces trust between participants. The key management concept employed in Fortezza provides a robust yet practical solution. It incorporates cryptographic key generation, distribution, verification, revocation, expiration, notification, and authentication. The principle component in the key management architecture is the user's certificate.

4.1.1 Certificates

Public cryptographic information is passed among users through a data structure called a certificate. The Fortezza certificate structure is based on the certificate defined by the ITU-T X.509 Recommendation. The ASN.1 encoded "Fortezza"/X.509 certificate contains, at a minimum, the user's unique identifier and public key information, bound together by a cryptographic mechanism that can be authenticated. The certificate is then digitally signed by its Issuer. Fortezza uses two lists to control the revocation of certificates.

4.1.2 Certificate Revocation List (CRL)

The CRL is a record of all revoked certificates produced by a common Issuer (an Issuer is defined in the Certificate Authority section). It is based on the CRL described in the ITU-T X.509 Recommendation of 1988. A certificate is revoked and placed on a CRL by its Issuer. A certificate is revoked when any data in it changes before it expires, such as when a user moves and changes address, accidentally destroys the Card (this must be verified), or the certificate is no longer needed. CRL's are available from the Directory, are ASN.1 encoded, and are updated as local policy warrants.

4.1.3 Compromised Key List (CKL)

The CKL is a list with the Key Material Identifier (KMID) of every user with compromised key material. Key material is compromised when a card and its PIN are uncontrolled or the user has become a threat to the security of the system. All certificates on a CKL are also placed on a CRL. CKL's are distributed by the Policy Creation Authority (the Policy Creation Authority is explained in the section 4.2.3), are ASN.1 encoded, and are updated as local policy warrants.

4.2 Certificate Authentication Hierarchy

The Fortezza key management concept is based on the multi-level hierarchy recommended in X.509. Figure 6 summarizes these levels. After defining an Issuer,

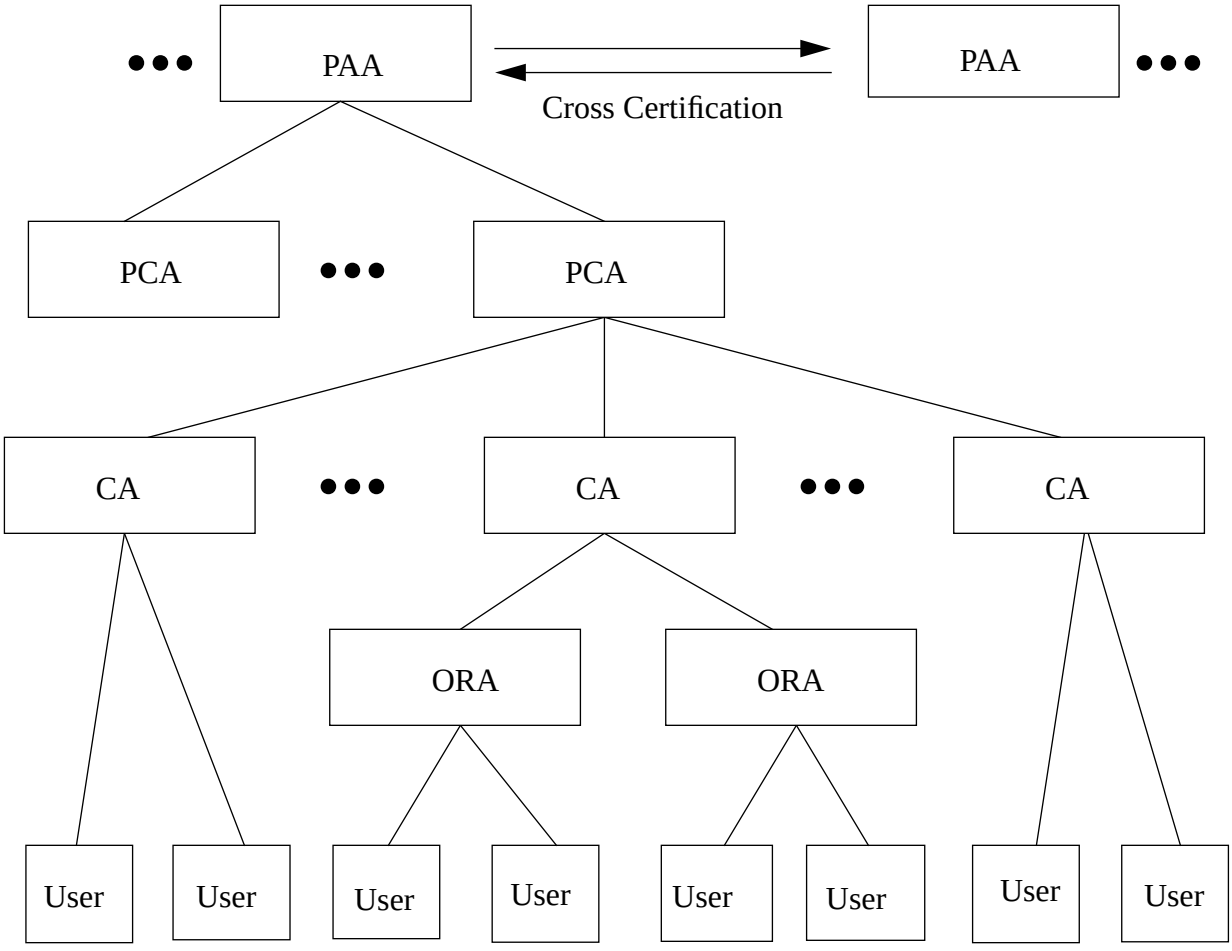


Figure 6: Authentication Hierarchy

the discussion will proceed with the Certificate Authority, then the Policy Creation Authority, the Policy Approval Authority, the User, and finally, the Organizational

Registration Authority. While this seems out of order from Figure 6, the discussions build on each preceding section.

4.2.1 Issuer

An Issuer is an entity which provides certificates to subordinates. An Issuer is responsible for validating the identity of the entity receiving the certificate. The Issuer then signs the certificate, binding the user to itself. A number of entities in the Fortezza system are Issuers, as we'll see below.

4.2.2 Certificate Authority (CA)

The CA is responsible for establishing, authenticating, maintaining, and possibly revoking each immediate subordinate user's key material, certificate, and hardware. The CA is an Issuer because of its responsibility to authorize new users to the system. Upon an authorized request from a potential user, a CA generates the cryptographic material, creates and signs a certificate, posts the certificate to the Directory, loads the personality and cryptographic data onto the user's card, and gives the Card to the user. The CA will provide maintenance for its users by rekeying the device periodically, posting new certificates to the Directory, maintaining a database of user information, maintaining CRL's, and distributing CKL's to its users. There can be multiple CA layers in the hierarchy. The CA was previously called a Local Authority (LA). The workstation used by the CA is known as the Certificate Authority Workstation (CAW), which was previously called the Local Authority Workstation (LAW).

4.2.3 Policy Creation Authority (PCA)

The PCA is the highest layer in the hierarchy with revocation authority. All entities under the PCA are part of its "domain." The PCA is also an Issuer. It is responsible for creating, authenticating, rekeying, and revoking CAs in its domain (i.e., those CAs it issued). The PCA will maintain and post an X.509 CRL containing revoked CA's, and maintain and distribute the CKL's to the CA's. The PCA was previously called the Root.

4.2.4 Policy Approval Authority (PAA)

Although the PAA is above the PCA in the hierarchy its functions are limited to creating PCAs and assuring each PCA's uniqueness. The PAA will not maintain a CRL or CKL, and cannot revoke any certificates. The PAA was previously called the Root Registry. Future responsibilities for the PAA include Cross Certification, whereby Users under different PAA's can securely communicate.

4.2.5 User

The bottom layer is the User. A User is provided with the Card, which has been loaded with keying material from the CA. The User is responsible for properly handling its card and notifying its CA when its card or certificate needs to be revoked. The UA must have access to the latest CRL and CKL list. This may be accomplished via directory inquiries or local caching (storing) of these lists.

4.2.6 Organizational Registration Authority (ORA)

The ORA is an appointed intermediary between an CA and a subset of its users. The ORA will assist the CA in requesting and distributing keys and cards. The ORA does not sign User certificates. The ORA was previously called the Organizational Notary (ON).

4.2.7 How the Authentication Hierarchy Works.

The architecture gives a recipient the ability to authenticate the validity of the information in an originator's certificate. This is accomplished when the recipient validates the originator's certificate by verifying certificate Issuer's signature, then checks the validity of the Issuer's certificate by verifying the signature of its certificate Issuer. The recipient continues verifying Issurers until the recipient reaches the trusted Issuer for the recipient's hierarchy (e.g., PAA certificate). At that point the recipient has complete trust in the validity of the originator's certificate. The path of a User's Issuers is called the User's Certification Path.

4.3 System Modules

The Fortezza program can be conceptualized as three intersecting rings (see Figure 7). The rings provide a graphical representation of the various modules in a Fortezza compliant application.Linking the modules are software interfaces specified in Interface Control Documents (ICDs) and programmer's guides. Supporting the three rings are a number of interdependent entities which were discussed earlier: the key management protocols and procedures, the CAW, and the Directory.

4.3.1 User Agent (Ring 1)

The first ring represents the UA. The UA is the message preparation system (e.g., e-mail) and is usually defined by protocols. The system integrator must decide on which security services will be available to the user and how these services will be presented. The UA has the responsibility for handling the lower transmission layers, which can be based on the Open Systems Interface (OSI) or TCP/IP, if necessary. The security interface to the Security Process is specified in an ICD (e.g., the MSP ICD). The UA should have the capability to load, maintain, and verify certificates, and to check a certificate against revocation lists.

4.3.2 User Agent/Security Protocol Interface (MSP ICD)

The interface between the UA and the Security Protocol software abstracts the security protocol from the UA. The Government has developed, and will provide to developers, an MSP ICD, as Government Furnished Information (GFI), for integrating applications with a Government-developed implementation of MSP.

4.3.3 Security Protocol (Ring 2)

The second ring represents the Security Protocol. This ring provides a logical (and often physical) distinction between the generic UA and the implementation of security services. This ring isolates the UA from the details of the security software. The security protocol software must process

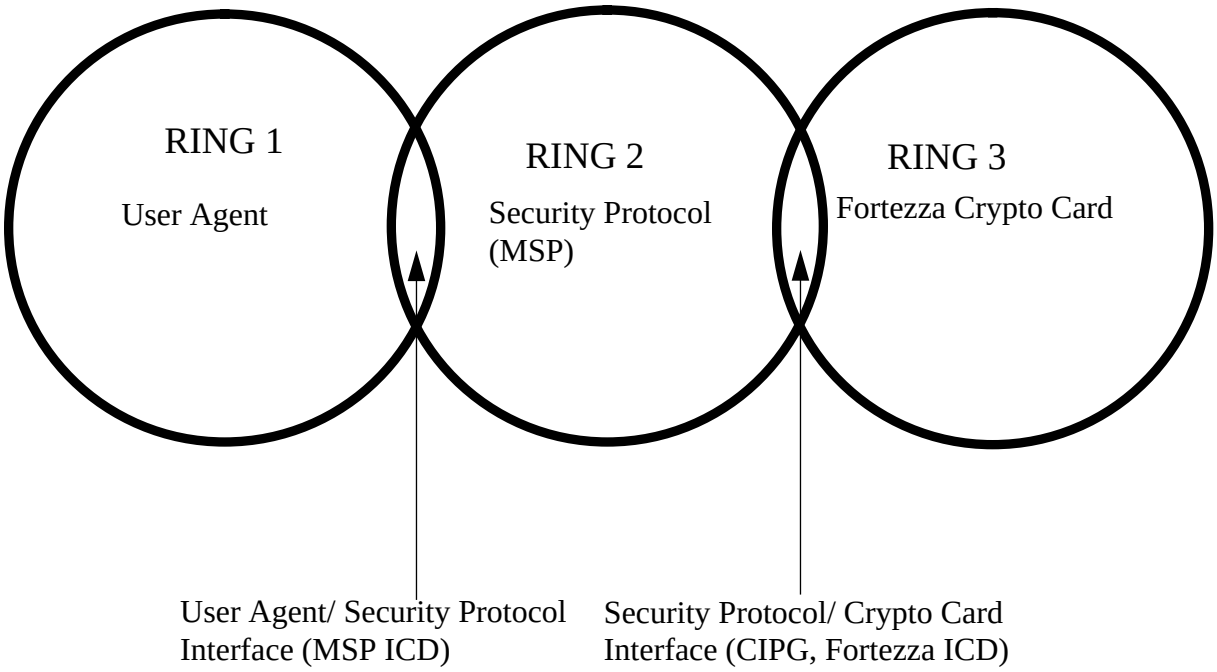


Figure 7: Architecture Overview

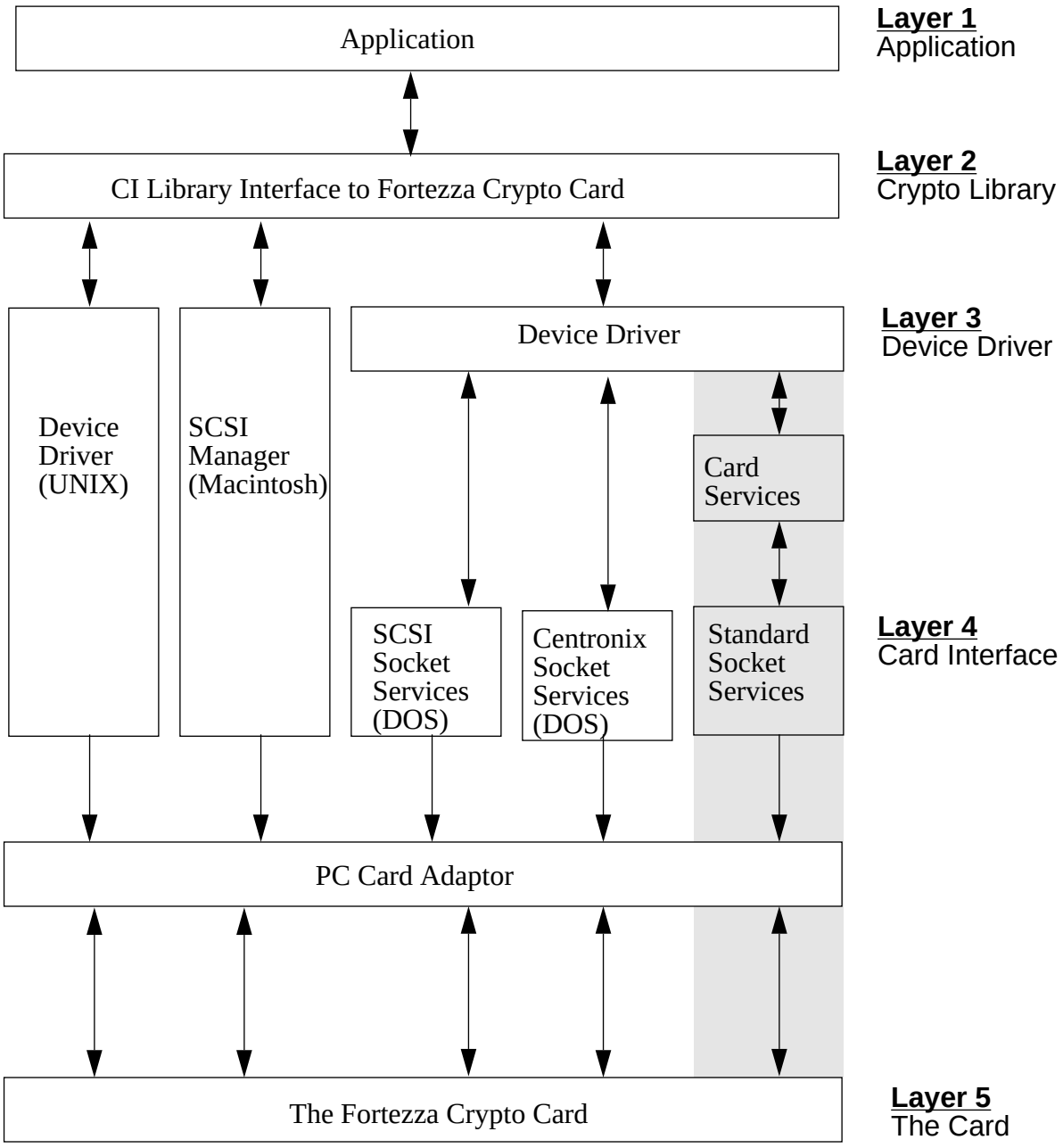
UA messages, and header information, and invoke the requested security services.The Government will provide to developers an implementation of MSP, as GFI.

4.3.4 Security Protocol /Crypto Card Interface (CI Library)

The “Cryptologic Interface Programmers Guide For The Fortezza Crypto Card (CIPG)” specifies the logical interface to the Card. This describes the function set used by the Cryptographic Interface (CI) Library to interface with the Card. The CI Library provides the application with a software interface to the Card’s functions while abstracting the specific data formats, protocols, and PCMCIA interfacing requirements (as defined in the Fortezza ICD and PC Card specifications, respectively). The CIPG document and CI Library software (for a number of operating systems) are available as GFI.

4.3.5 Crypto Card (Ring 3)

The third ring represents the physical cryptographic device, the Fortezza Crypto Card. The Card fulfills the security services specified for the Fortezza system. It stores the private key information for each user personality, the certificates of its Issuers, and the public keys needed for cryptography. It performs the digital signature and hash algorithms, public/private key exchange functions, encryption, and decryption. The interface to the external device depends on the hardware platform and its configuration, and the



Shaded area indicates a full PC Card compliant path and is Fortezza's final implementation goal.

Figure 8: Fortezza Crypto Card Interface

operating system. Figure 8 illustrates the software necessary to integrate the card to multiple platforms. The proper CI Library, device driver, Card Services, Socket Services (if applicable) software, and hardware must all be correctly configured for proper operation.

4.4 Personal Computer Memory Card International Association (PCMCIA)

The Card's adherence to the PC Card standards, as developed by the PCMCIA, allows the Card to be integrated into different hardware platforms and operating systems with relative ease. To enable the Card, the host (workstation) must have its hardware and software configured properly to handle PC Cards. Since PC Cards are a relatively new technology, this is often a concern. Note in Figure 8 the required components and the differences in the various operating systems. The shaded area highlights a fully standardized PC Card stack.

4.4.1 Device Driver

The device driver provides the CI Library an interface to the lower layer components in the stack. The device driver is operating system dependent. In environments that do not yet support the PC Card stack (e.g., UNIX), the device driver provides much of the functionality of the equivalent lower layers in the PC Card stack. As more operating systems adopt the PC Card stack, the architecture for the CI Library and device drivers will need to change. The Government will provide to developers the appropriate device driver and the CI Library software for a number of operating systems, as GFI.

4.4.2 Card Services (CS)

CS allocates and controls operating system and hardware resources for the PC Card. CS requires an application to register as a client with CS. This allows the CS to handle multiple applications and to notify each application of any changes to the PC Card or its status (ex., the card is inserted or removed). CS functions include directing power requirements, resetting the socket (the socket is the hardware device that connects to the PC Card), mapping available host memory to the PC Card, assigning interrupts, reading card configuration information (called Tuples), and performing the reads and writes with the card's memory.

4.4.3 Socket Services (SS)

SS is the lowest level access to a PC Card. It is the software interface between the host and the hardware used to manage PC Card sockets (readers). Its responsibilities include detecting interrupts from the PC Card, setting power levels, and mapping the PC Card memory to a specific area of available memory in the host. PC Card Readers that do not fully use the PCMCIA stack, such as SCSI or parallel (Centronix) readers, require each reader manufacturer to supply its own SS. It is important that integrators match the proper SS with the appropriate reader in these cases.

4.4.4 Adaptor

The adaptor is the hardware, usually a Large Scale Integrated Circuit, that communicates with the PC Card. It performs the access timing, power control, interrupt routing, and memory allocation in the PC Card. This hardware can be found on the host motherboard or the PC Card reader.

4.4.5 PC Card Reader

The PC Card reader is the component that transmits the data to and from the PC Card. Its characteristics are controlled by SS (since it is the PC Card "socket"). Card readers exist for several interface protocols, such as serial, parallel, internal PCMCIA, or SCSI.

4.4.6 PC Card (Fortezza Crypto Card)

PC Cards are designed as either memory or Input/Output (I/O) based. I/O based cards rely on interrupts, which cannot be supported on all configurations (namely SCSI readers). Memory based PCMCIA cards basically just read in and write out data. The Card is a memory-based PC Card: it reads data from the host, processes it, then makes the data available to the host. The physical forms for PC Cards are designated Type 1, Type 2, and Type 3, with the difference being the thickness of the card. The Fortezza program places no restrictions on the form type, unless it affects the security functionality.

5.0 Applications

The utility of Fortezza can be appreciated from the number of possible applications which can use its services. The number of applications is limited only by one's imagination.

5.1 File and Media Encryption

Applications can be written to allow the Card to secure user files or storage mediums. A file can be secured through encryption using the users local storage key, or via the standard MEK generation. Applications can also be written to allow the Card to secure entire media storage equipment. It can secure a computer hard drive, floppy Disk, CD-ROM, and any other media storage device. The security service can test the integrity of the data with a hash and sign service or full data encryption.

5.2 Access Control/Strong Authentication

One application for Fortezza is access control. Access control is a policy set to control who or what can gain access to something, such as data in a database, use of a computer network, or entry to a building. Identification and Authentication (I&A) are elements of the supporting policy required for the integrity of that policy. In order to be effective, I&A mechanisms must uniquely identify the subject and resolutely prevent forgery. Traditionally, user I&A has been based on the use of "something you know" (knowledge- e.g., a password), "something you are" (characteristic- e.g., a thumbprint), and "something you have" (ownership- e.g., a card). Fortezza satisfies the "something you have," with the Card, and "something you know," with the user's Personal Identification Number (PIN). This approach is

stronger than just a traditional, simple password mechanism, which is only "something you know." This I&A technique is known as "Strong Authentication."

5.3 Remote Network Services

Applications can be written to allow the Card to secure Telnet, File Transport Protocol (FTP), and File Transfer, Access, and Management (FTAM), or even network layer communications. Fortezza can be incorporated into these protocols if an application is developed. The Card can provide strong authentication and data confidentiality.

5.4 Directory Service

Fortezza can be used to provide strong authentication capability for the Directory. This process will consist of the DUA initiating a session (a DUA always initiates a session) and sending a signed value to the DSA. The DSA verifies the signature and determines access privilege to the Directory. The exact process of authentication can be designed to meet a local policy.

5.5 World Wide Web (WWW)

The popularity of the WWW, along with increased security concerns, is creating a need for secure browsing. The Card can be used to secure access to Web sites and provide confidentiality for financial transactions. Products could include secure Hyper Text Transport Protocol (HTTP).

5.6 Firewalls

A Firewall is a collection of components placed between two networks that collectively have the following properties: all traffic from inside to outside (and vice-versa) must pass through the firewall; and, only authorized traffic (as defined by the local security policy) will be allowed to pass. The firewall must be tamperproof, and not bypassable. Typically, Firewalls will provide the following capabilities: access control (through packet filtering), network service restrictions, user authentication, and audit logging. The Card can be used to support the I&A requirements and to assure that required messages are properly cryptographically processed.

5.7 Client /Server

Fortezza can be used in a client/server environment. However, to truly provide writer-to-reader security, the process must be invoked by the end user. If the end user cannot operate the Card because the terminal does not have processing capabilities, writer-to-reader security cannot be achieved without additional network security devices and a trusted operating system at the host.

5.8 Electronic Messaging

Applications can be written to allow the Card to secure electronic mail (e-mail, EDI, Electronic Commerce (EC), and FAX). These applications can take advantage of all of Fortezza's capabilities.

References

Cryptologic Interface Programmers Guide for the Fortezza Crypto Card, Revision 1.52, NSA, November, 1995

Data Communication Networks Message Handling Systems, Recommendations X.400-X.420, Volume VIII, CCITT, Geneva 1989

Data Communication Networks Message Handling Systems, Recommendations X.500-X.521, Volume VIII, CCITT, Geneva 1989

Defense Message System (DMS) Concept of Operations (CONOPS), DISA, April 2, 1993

Fortezza Message Security Protocol Software Interface Control Document, Version 3.01, NSA, November 28, 1995

Interface Control Document for the Fortezza Crypto Card, Revision P1.5, NSA, December 22, 1994

The ISO Development Environment: User's Manual Volume 5: QUIPU, Version 7.0, ISODE, July 19, 1991

(MISSI) Key, Privilege, and Certificate Management Plan, Version 3.1 Draft, NSA, September 11, 1995

PCMCIA Standards, Version 2.1, PCMCIA, November 1992

SDN.701, Message Security Protocol (MSP), Revision 3.0, NSA, March 23, 1994

SDN.702, SDNS Directory Specifications for Utilization with SDNS Message Security Protocol, Revision 2.7, NSA, August 8, 1995

SDN.704, MIME/SMTP With MSP, Draft, NSA

Tannenbaum, Andrew S.: *Computer Networks*, Englewood Cliffs, NJ: Prentice Hall, 1988

Acronyms

ACP	Allied Communications Publication
API	Application Programming Interface
ASN.1	Abstract Syntax Notation 1
CA	Certificate Authority
CAW	Certificate Authority Workstation
CCITT	Consultive Committee on International Telephone and Telegraph
CI	Cryptologic Interface
CIPG	Cryptologic Interface Programmers Guide
CKL	Compromised Key List
CRL	Certificate Revocation List
CS	Card Services
DER	Distinguished Encoding Rules
DIB	Directory Information Base
DISA	Defense Information System Agency
DISN	Defense Information System Network
DIT	Directory Information Tree
DMS	Defense Message System
DoD	Department of Defense
DSA	Digital Signiture Algorithm
DSA	Directory System Agent
DSS	Digital Signature Standard
DUA	Directory User Agent
EC	Electronic Commerce
EDI	Electronic Data Interchange
FIPS	Federal Information Processing Standard
FTP	File Transfer Protocol
FTAM	File Transfer, Access, and Management
GFI	Government Furnished Information
HTTP	Hyper Text Transport Protocol
I&A	Identification and Authentication
ICD	Interface Control Document
ISO	International Standards Organization
ITU-T	International Telecommunications Union- Telecommunications Standardizations Section
IV	Initialization Vector
KEA	Key Exchange Algorithm
KMID	Key Material Identifier
KMS	Key Management System
Ks	Storage Key
LA	Local Authority
LAW	Local Authority Workstation
MEK	Message Encryption Key
MHS	Message Handling System
MIME	Multipurpose Internet Mail Extension

MISSI	Multi- level Information System Security Initiative
MLA	Mail List Agent
MSP	Message Security Protocol
MTA	Message Transport Agent
MTS	Message Transport System
NSA	National Security Agency
NIST	National Institute of Standards and Technology
ON	Organizational Notary
ORA	Organizational Registration Authority
OSI	Open Systems Interface
OUA	Organizational User Agent
P2	Interpersonal Message Content Type, X.400, 1984
P22	Interpersonal Message Content Type, X.400, 1988
P48, P42	Message Security Protocol
P772	Military Messaging Type
PAA	Policy Approval Authority
PC	Personal Computer
PCA	Policy Creation Authority
PCMCIA	Personal Computer Memory Card International Association
PIN	Personal Identification Number
PMSP	Pre Message Security Protocol
RFC	Request For Comment
SDNS	Secure Data Network System
SHA	Secure Hash Algorithm
SMTP	Simple Mail Transport Protocol
SS	Socket Services
TCP/IP	Transport Control Protocol /Internet Protocol
TEK	Token Encryption Key
UA	User Agent
WWW	World Wide Web

Subject Index