NEAR EAST UNIVERSITY

Faculty of Engineering

Department of Computer Engineering

CRYPTOGRAPHY AND NETWORK SECURITY

Graduation Project

COM- 400

Student: Muhammad Muazzam yazdani (2002140)

Supervisor: Prof. Dr. Fakhreddin MAMEDOV

Nicosia – 2002

ACKNOWLEDGEMENT

"First of all, I want to pay my regards to my supervisor "Prof. Dr. Fakhreddin MAMEDOV" (Dean, faculty of Engineering) and all persons who have contributed in the preparation of my project to complete it successfully. I am also thankful to who helped me a lot in my crises and gave me full support toward the completion of my project.

I would like to thank my parents who gave their lasting encouragement in my studies, so that I could be successful in my life time. I specially thank to my mother whose prayers have helped me to keep safe from every dark region of life. Special thank to my father who help me in joining this prestigious university and helped me to make my future brighter

I am also very much grateful to my all companions and colleagues who gave their precious time to help me to encourage me their ever devotion and all valuable information which I really need to complete my project.

Further I am thankful to Near East University academic staff and all those persons who helped me or encouraged me incompletion of my project. Thanks!"

ABSTRACT

Cryptography protects a message or file from being read by an eavesdropper who has no other means of access to either the original text of what is protected, or the key with which it is encrypted. We use a program to encrypt the text; the program will change the letters into symbols and other weird characters, so when someone opens the file they cannot read it. The interconnection of networks is an increasing trend in government and private industry. There is the obvious danger that connections made in such an extended network may increase the risk of a security compromise, with the owners unaware of the risk. Network connections should therefore be protected, at a level based on the risk. The assumption must be that the connecting parties are to a certain degree hostile and have to be strictly constrained to the access for which the connection was agreed.

Although cryptography is fascinating and glamorous, because of its association with such things as espionage, diplomacy, and the higher levels of the military, it has a limited but important role in the area of network security.

TABLE OF CONTENTS

INTRODUCTION

Communication and information technology are making a dramatic impact on society and commerce. Digital information can be efficiently stored, processed and communicated, allowing substantial improvements in production and wealth. By connecting providers and suppliers around the world, and allowing them to interact via automated mechanisms, technology is opening amazing opportunities, mostly the result of removing barriers to communication and commerce. However, with this come risks of illegitimate, malicious use and access of information, by an adversary abusing the ease of storage, processing and communication. There are risks and threats associated with the existing commercial and social mechanisms. Such as expose of secret information from storage or communication, e.g. credit card numbers or medical records. Modification in information stored or communicated, e.g. moving funds illegitimately. Duplicating and selling copyrighted text or music and last is misrepresent herself when communicating, creating false image, and using this to cheat.

Cryptography is not a trivial area. Since its goal is to govern the use of information, preventing unauthorized use, simulations and experimentation cannot test cryptographic mechanisms. Furthermore, weaknesses are often hard to find, and often finding a weakness involves substantial innovation and ingenuity. In fact, there is a branch of cryptography, called cryptanalysis, dedicated to breaking cryptographic mechanisms and their applications. The ultimate test of any cryptographic mechanism is when a very large effort by dedicated researchers and by actual adversaries fails to find a weakness in it. However, this is rarely a useful test for new mechanisms and systems. This makes precise definitions and proofs of security extremely important.

My first Chapter is all about the introduction as cryptography is the art of limiting the use and access of information, to address such threats. And what functions involve in this technique and then main encryption and decryption of data.

In my second chapter I have explained various functions techniques used in cryptography in detail. It includes ciphers, a technique use to code data then we have hash function and the authentication methods and threats to the cryptography as how some one can break through to check the secure information

My third chapter is about the cryptographic encryption and decryption, where two parties exchange messages encrypted/decrypted by some secret, and an adversary eavesdropping on their messages cannot learn the encrypted content (plaintext). In fact, the term `cryptography` comes from this first goal, of protecting the confidentiality of information. I have explained in detail how we will assign the keys and the data can be encrypted and decrypted by using different techniques.

My last fourth chapter is about the network security. As cryptography are the techniques and network security is overall security of the information on the network. I have explained in detail about the network and about OSI layer model then what protocols are and how they have threat for different attacks. Then I have explained about the firewall and how they make the network security possible

1. INTRODUCTION TO CRYPTOGRAPHY

1.1 Overview

To introduce cryptography, an understanding of issues related to information security in general is necessary. Network security manifests itself in many ways according to the situation and requirement. Regardless of who is involved, to one degree or another, all parties to a transaction must have confidence that certain objectives associated with network security have been met. Some of these objectives are listed in Table 1.1. Often the objectives of on security cannot solely be achieved through mathematical algorithms and protocols alone, but require procedural techniques and abidance of laws to achieve the desired result. One of the fundamental tools used in network security is the signature. It is a building block for many other services such as no repudiation, data origin authentication, identification, and witnessing, to mention a few. Achieving network security in an electronic society requires a vast array of technical and legal skills. There is, however, no guarantee that all of the network security objectives deemed necessary can be adequately met. The technical means is provided through cryptography. Cryptography is not the only means of providing network security, but rather one set of techniques

1.2 Cryptography

Cryptography is the study of mathematical techniques related to aspects of network security such as confidentiality, data integrity, entity authentication, and data origin authentication.

The following are the goals of the Cryptography

1. Confidentiality is a service used to keep the content of information from all but those authorized to have it. There are numerous approaches to providing confidentiality, ranging from physical protection to mathematical algorithms.

Table 1.1: Some information security objectives.

Privacy or confidentiality	Keeping information secret from all but those who are authorized to see it.
Data integrity ensuring	Information has not been altered by unauthorized or unknown means.
Entity authentication or identification	Corroboration of the identity of an entity (e.g., a person, a computer terminal, a credit card, etc.).
Message authentication	Corroborating the source of information; also known as data origin authentication.
Signature	A means to bind information to an entity.
Authorization	Conveyance, to another entity, of official sanction to do or be something.
Validation	A means to provide timeliness of authorization to use or manipulate information or resources.
Access control	Restricting access to resources to privileged entities.
Certification	Endorsement of information by a trusted entity.
Time stamping	Recording the time of creation or existence of information.
Witnessing	Verifying the creation or existence of information by an entity other than the creator.
Receipt	Acknowledgement that information has been received.
Confirmation	Acknowledgement that services has been provided.
Ownership	A means to provide an entity with the legal right to use or transfer a resource to others.
Anonymity	Concealing the identity of an entity involved in some process.
Non-repudiation	Preventing the denial of previous commitments or actions.
Revocation	Retraction of certification or authorization.

2. Data integrity is a service which addresses the unauthorized alteration of data. To assure data integrity, one must have the ability to detect data manipulation by unauthorized parties.
3. Authentication is a service related to identification. This function applies to both entities and information itself. Aspect of cryptography is usually subdivided into two major classes: entity authentication and data origin authentication.
4. Non-repudiation is a service which prevents an entity from denying previous commitments or actions.

A fundamental goal of cryptography is to adequately address these four areas in both theory and practice. Cryptography is about the prevention and detection of cheating and other malicious activities. A number of basic cryptographic tools (primitives) used to provide network security. Examples of primitives include encryption schemes hash functions, and digital signature schemes. Figure 1.1 provides a schematic listing of the primitives considered and how they relate.

These primitives should be evaluated with respect to various criteria such as:

1. Level of security. This is usually difficult to quantify. Often it is given in terms of the number of operations required to defeat the intended objective.
2. Functionality. Primitives will need to be combined to meet various network security objectives. Which primitives are most effective for a given objective will be determined by the basic properties of the primitives.

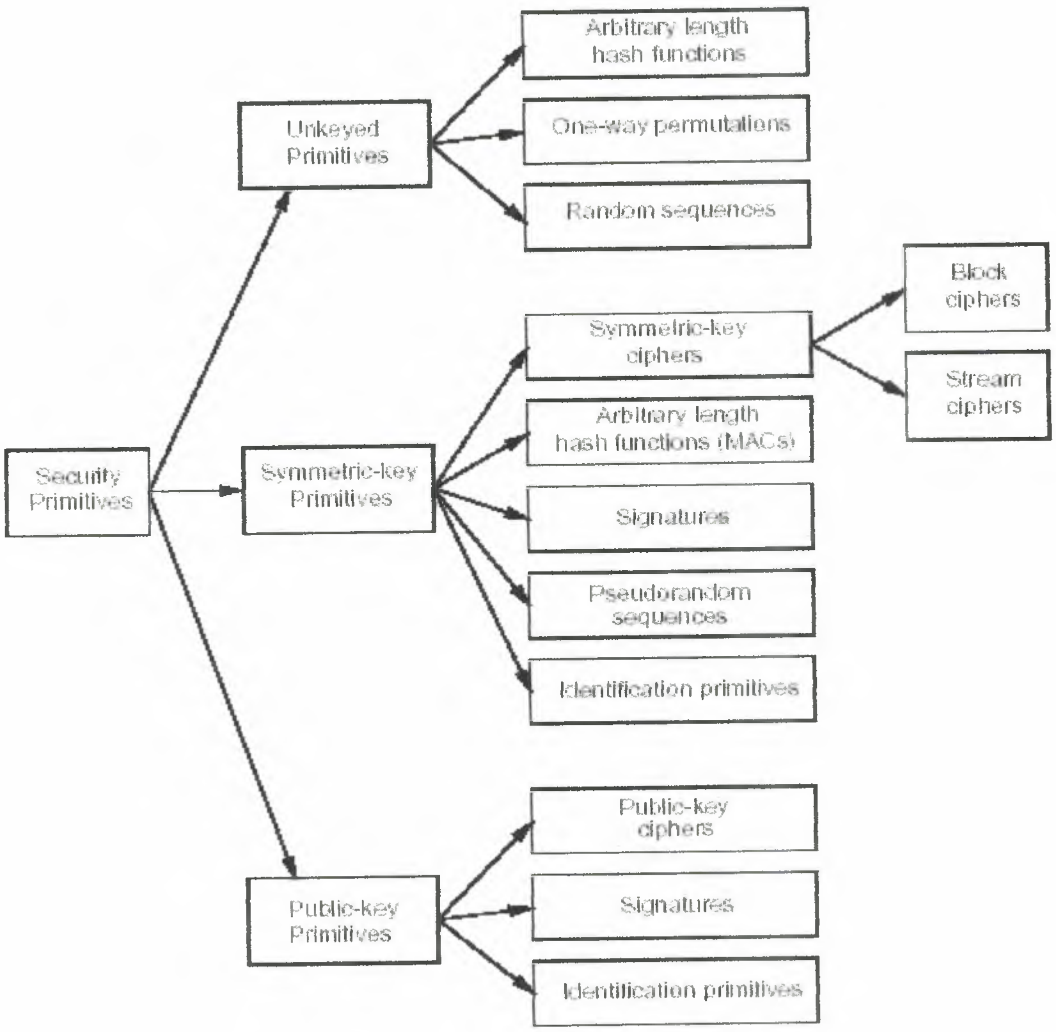


Figure 1.1: A taxonomy of cryptographic primitives.

3. Methods of operation. Primitives, when applied in various ways and with various inputs, will typically exhibit different characteristics; thus, one primitive could provide very different functionality depending on its mode of operation or usage.
4. Performance. This refers to the efficiency of a primitive in a particular mode of operation.
5. Ease of implementation. This refers to the difficulty of realizing the primitive in a practical instantiation. This might include the complexity of implementing the primitive in either a software or hardware environment.

The relative importance of various criteria is very much dependent on the application and resources available. For example, in an environment where computing power is limited one may have to trade off a very high level of security for better performance of the system as a whole.

1.3 Basic Functions and Concepts

A familiarity with basic mathematical concepts used in cryptography will be useful. One concept which is absolutely fundamental to cryptography is that of a function in the mathematical sense. A function is alternately referred to as a mapping or a transformation.

1.3.1 Function

A set consists of distinct objects which are called elements of the set. For example, a set X might consist of the elements a, b, c, and this is denoted $X = \{a; b; c\}$. If x is an element of X (usually written $x \in X$) the image of x is the element in Y which the rule *f* associates with x; the image y of x is denoted by $y = f(x)$. Standard notation for a function *f* from set X to set Y is $f: X \rightarrow Y$.

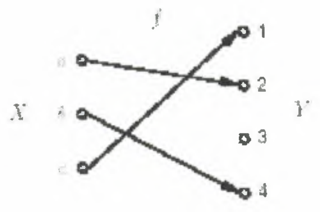


Figure 1.2: A function *f* from a set X to a set Y.

- 1-1 Functions: A function is 1 - 1 (one-to-one) if each element in the co domain Y is the image of at most one element in the domain X.
- Onto function: A function is onto if each element in the co domain Y is the image of at least one element in the domain.
- Bijection: If a function f: X→Y is 1-1 and Im (f) = Y, then f is called a bijection.
- One-way functions: A function f from a set X to a set Y is called a one-way function if f (x) is easy to compute for all $x \in X$ but for essentially all elements $y \in$ Im (f) it is "computationally infeasible" to find any $x \in X$ such that $f(x) = y$.
- Trapdoor one-way functions: A trapdoor one-way function is a one-way function f: X→Y with the additional property that given some extra
- Permutations: Let S be a finite set of elements. A permutation p on S is a bijection from S to itself (i.e., p: S→S).
- Involutions: Involutions have the property that they are their own inverses. (i.e., f: S→ S).

1.3.2 Basic Terminology and Concepts

The scientific study of any discipline must be built upon exact definitions arising from fundamental concepts. Where appropriate, strictness has been sacrificed for the sake of clarity.

1.3.2.1. Encryption Domains and Co-domains

- $\mathcal{A}$ denotes a finite set called the alphabet of definition.
- $\mathcal{M}$ denotes a set called the message space. $\mathcal{M}$ consists of strings of symbols from an alphabet. An element of $\mathcal{M}$ is called a plaintext message or simply a plaintext.
- $\mathcal{C}$ denotes a set called the cypertext space. $\mathcal{C}$ consists of strings of symbols from an alphabet; differ from the alphabet of $\mathcal{M}$. An element of $\mathcal{C}$ is called a cypertext.

1.3.2.2 Encryption and Decryption Transformations

- $\mathcal{K}$ denotes a set called the key space. An element of $\mathcal{K}$ is called a key.
- Each element $e \in \mathcal{K}$ uniquely determines a bijection from $\mathcal{M}$ to C, denoted by $\mathcal{E}e$.
- $\mathcal{D}_d$ denotes a bijection from C to $\mathcal{M}$ and $\mathcal{D}_d$ is called a decryption function.
- The process of applying the transformation $\mathcal{E}e$ to a message $m \in \mathcal{M}$ is usually referred to as encrypting m or the encryption of m.
- The process of applying the transformation $\mathcal{D}_d$ to a cypertext c is usually referred to as decrypting c or the decryption of c.
- The keys e and d are referred to as a key pair and denoted by (e; d).

1.3.2.3 Achieving Confidentiality

An encryption scheme may be used as follows for the purpose of achieving confidentiality. Two parties Alice and Bob first secretly choose or secretly exchange a key pair (e; d). At a subsequent point in time, if Alice wishes to send a message $m \in \mathcal{M}$ to Bob, she computes $c = \mathcal{E}e\,(m)$ and transmits this to Bob. Upon receiving c, Bob computes $\mathcal{D}_d(c) = m$ and hence recovers the original message m.

The question arises as to why keys are necessary. If some particular encryption/decryption transformation is exposed then one does not have to redesign the entire scheme but simply change the key. Figure 1.3 provides a simple model of a two-party communication using encryption.

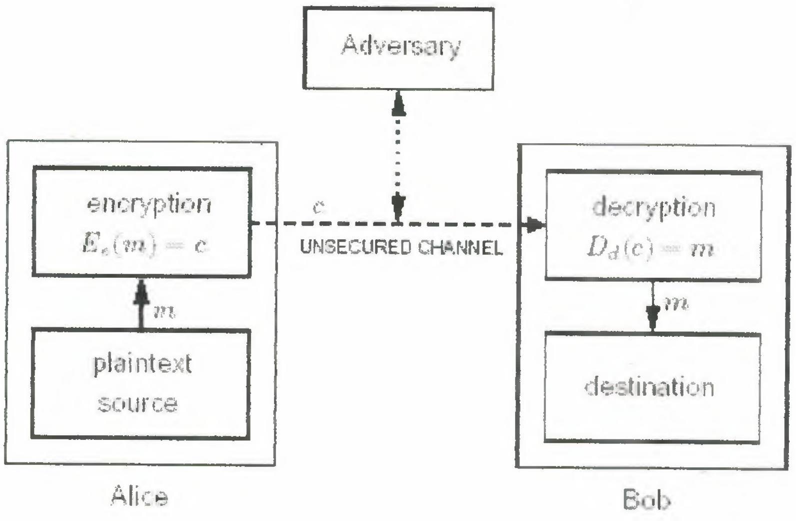


Figure 1.3: Schematic of a two-party communication.

1.3.2.4 Communication Participants

Referring to Figure 1.3, the following terminology is defined.

- An entity or party is someone or something which sends, receives, or manipulates information. An entity may be a person, a computer terminal, etc.
- A sender is an entity in a two-party communication which is the legitimate transmitter of information.
- A receiver is an entity in a two-party communication which is the intended recipient of information.
- An adversary is an entity in a two-party communication which is neither the sender nor receiver, and which tries to defeat the information security service being provided between the sender and receiver.

1.3.2.5. Channels

A channel is a means of conveying information from one entity to another. A physically secure channel is one which is not physically accessible to the adversary. An unsecured channel is one from which parties other than those for which the information is intended can reorder, delete, insert, or read. A secured channel is one from which an adversary does not have the ability to reorder, delete, insert, or read. A secured channel may be secured by physical or cryptographic techniques.

1.3.2.6 Security

A fundamental principle in cryptography is that the sets $\mathcal{M}$; $\mathcal{C}$; $\mathcal{K}$; $\{E_e: e \in \mathcal{K}\}$, $\{D_d: d \in \mathcal{K}\}$ are public knowledge. When two parties wish to communicate securely using an encryption scheme, the only thing that they keep secret is the particular key pair $(e; d)$, which they must select. One can gain additional security by keeping the class of encryption and decryption transformations secret but one should not base the security of the entire scheme on this approach. An encryption scheme is said to be breakable if a third party, without prior knowledge of the key pair $(e; d)$ can systematically recover plaintext from corresponding cypertext within some appropriate time frame. An encryption scheme can be broken by trying all possible keys to see which one the communicating parties are using. This is called an exhaustive search of the key space.

Frequently cited in the literature are Kerckhoffs' desiderata, a set of requirements for cipher systems. They are given here essentially as Kerckhoffs originally stated them:

1. The system should be, if not theoretically unbreakable, unbreakable in practice.
2. Compromise of the system details should not inconvenience the correspondents.
3. The key should be remember able without notes and easily changed.
4. The cryptogram should be transmissible by telegraph.
5. The encryption apparatus should be portable and operable by a single person.
6. The system should be easy, requiring neither the knowledge of a long list of rules nor mental strain.

1.3.2.7 Network Security in General

So far the terminology has been restricted to encryption and decryption with the goal of privacy in mind. Network security is much broader, encompassing such things as authentication and data integrity.

- A network security service is a method to provide specific aspect of security.
- Breaking a network security service implies defeating the objective of the intended service.
- A passive adversary is an adversary who is capable only of reading information from an unsecured channel.

- An active adversary is an adversary who may also transmit, alter, or delete information on an unsecured channel.

1.4 Symmetric-key Encryption

Consider an encryption scheme consisting of the sets of encryption and decryption transformations $\{\mathcal{E}_e : e \in \mathcal{K}\}$ and $\{\mathcal{D}_d : d \in \mathcal{K}\}$, respectively, where $\mathcal{K}$ is the key space. The encryption scheme is said to be symmetric-key if for each associated encryption/decryption key pair $(e; d)$, it is computationally easy to determine d knowing only e, and to determine e from d. Since $e = d$ in most practical symmetric-key encryption schemes, the term symmetric key becomes appropriate.

A two-party communication using symmetric-key encryption can be described by the block diagram of Figure 1.4, with the addition of the secure channel.

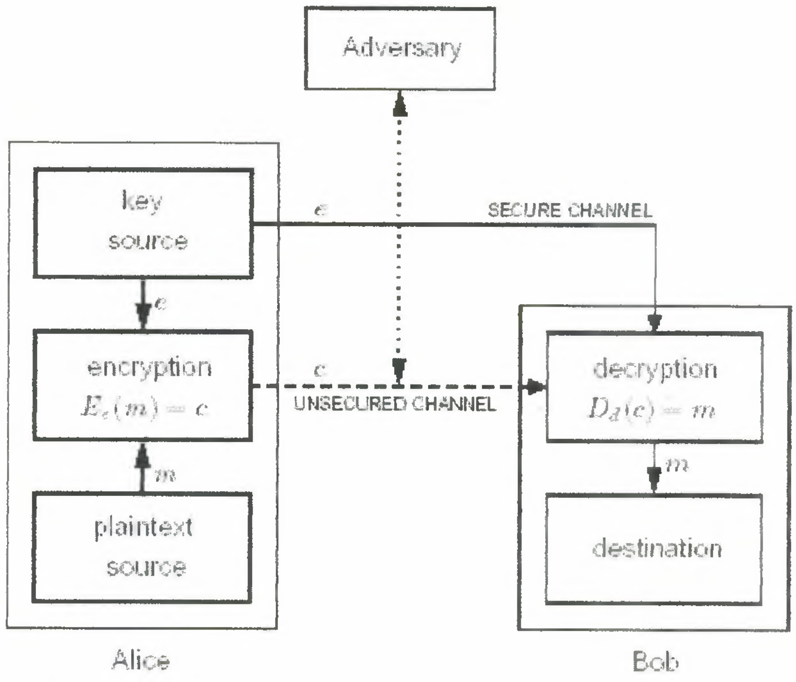


Figure 1.4: Two-party communication using encryption, with a secure channel

One of the major issues with symmetric-key systems is to find an efficient method to agree upon and exchange keys securely. It is assumed that all parties know the set of encryption/decryption transformations there are two classes of symmetric-key encryption schemes which are commonly distinguished, block ciphers and stream ciphers.

1.4.1 Block Ciphers

A block cipher is an encryption scheme which breaks up the plaintext messages to be transmitted into strings (called blocks) of a fixed length t over an alphabet $\mathcal{A}$, and encrypts one block at a time. Most well-known symmetric-key encryption techniques are block ciphers. Two important classes of block ciphers are substitution ciphers and transposition ciphers

1.4.2 Stream Ciphers

Stream ciphers form an important class of symmetric-key encryption schemes. They are, in one sense, very simple block ciphers having block length equal to one. What makes them useful is the fact that the encryption transformation can change for each symbol of plaintext being encrypted. In situations where transmission errors are highly probable, stream ciphers are advantageous because they have no error propagation. They can also be used when the data must be processed one symbol at a time

1.4.3 The Key Space

The size of the key space is the number of encryption/decryption key pairs that are available in the cipher system. A key is typically a compact way to specify the encryption transformation to be used. For example, a transposition cipher of block length t has $t!$ Encryption functions from which to select. Each can be simply described by a permutation which is called the key.

1.5 Digital Signatures

A cryptographic primitive who is fundamental in authentication, authorization, and non-repudiation is the digital signature. The purpose of a digital signature is to provide a means for an entity to bind its identity to a piece of information. The process of signing entails transforming the message and some secret information held by the entity into a tag called a signature.

1.5.1. Nomenclature and Set-up

The transformations $\mathcal{S}_{\mathcal{A}}$ and $\mathcal{V}_{\mathcal{A}}$ provide a digital signature scheme for $\mathcal{A}$.

- $\mathcal{M}$ is the set of messages which can be signed.
- $\mathcal{S}$ is a set of elements called signatures, possibly binary strings of a fixed length.
- $\mathcal{S}_{\mathcal{A}}$ is a transformation from the message set $\mathcal{M}$ to the signature set $\mathcal{S}$, and is called a signing transformation for entity $\mathcal{A}$.
- $\mathcal{V}_{\mathcal{A}}$ is a transformation from the set $\mathcal{M} \times \mathcal{S}$ to the set {true, false} $\mathcal{V}_{\mathcal{A}}$ is called a verification transformation for $\mathcal{A}$'s signatures, is publicly known, and is used by other entities to verify signatures created by $\mathcal{A}$.

1.6 Public-key Cryptography

The concept of public-key encryption is simple and elegant, but has far-reaching consequences. Let $\{\mathcal{E}e\colon e \in \mathcal{K}\}$ be a set of encryption transformations, and let $\{\mathcal{D}_d\colon d \in \mathcal{K}\}$ be the set of corresponding decryption transformations, where $\mathcal{K}$ is the key space. Consider any pair of associated encryption/decryption transformations $(\mathcal{E}e; \mathcal{D}d)$ and suppose that each pair has the property that knowing $\mathcal{E}e$ it is computationally infeasible, given a random ciphertext $c \in \mathcal{C}$, to find the message $m \in \mathcal{M}$ such that $\mathcal{E}e(m) = c$. This property implies that given e it is infeasible to determine the corresponding decryption key d. $\mathcal{E}e$ is being viewed here as a trapdoor one-way function with d being the trapdoor information necessary to compute the inverse function and hence allow decryption. This is unlike symmetric-key ciphers where e and d are essentially the same.

The encryption method is said to be a public-key encryption scheme if for each associated encryption/decryption pair $(e; d)$, one key e (the public key) is made publicly available, while the other d (the private key) is kept secret. For the scheme to be secure, it must be computationally infeasible to compute d from e. To avoid ambiguity, a common convention is to use the term private key in association with public-key cryptosystems, and secret key in association with symmetric-key cryptosystems

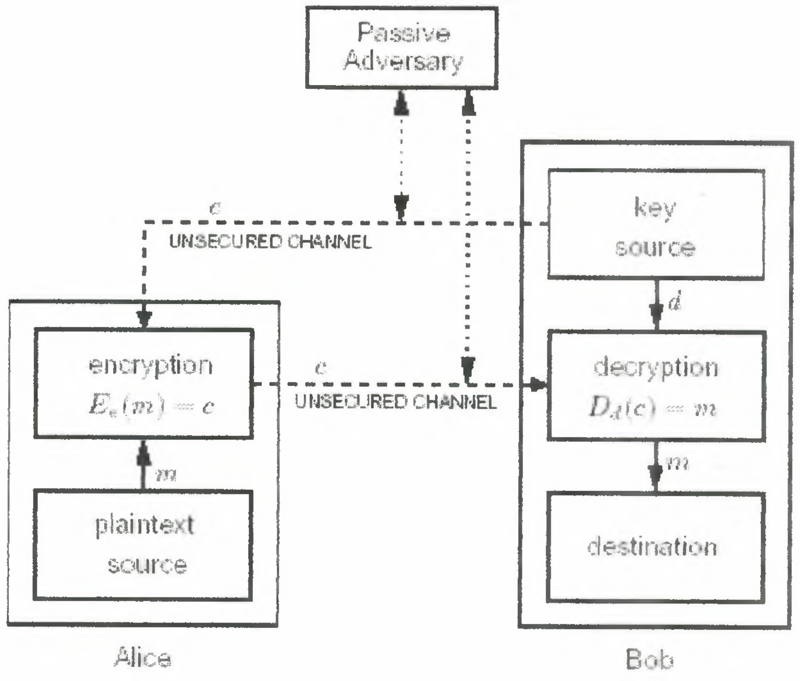


Figure 1.5: Encryption using public-key techniques.

1.7 Hash Functions

One of the fundamental primitives in modern cryptography is the cryptographic hash function, often informally called a one-way hash function. A simplified definition for the present discussion follows. A hash function is a computationally efficient function mapping binary strings of arbitrary length to binary strings of some fixed length, called hash-values. For a hash function which outputs n-bit hash-values and has desirable properties, the probability that a randomly chosen string gets mapped to a particular n-bit hash-value (image) is 2^{-n}. The basic idea is that a hash-value serves as a compact representative of an input string. To be of cryptographic use, a hash function *h* is typically chosen such that it is computationally infeasible to find two distinct inputs which hash to a common value and that given a specific hash-value y, it is computationally infeasible to find an input x such that *h*(x) = y. The most common cryptographic uses of hash functions are with digital signatures and for data integrity Hash functions are typically publicly known and involve no secret keys. When used to detect whether the message input has been altered, they are called modification detection codes (MDCs). Related to these are hash functions which involve a secret key, and provide data origin authentication as well as data integrity; these are called message authentication codes (MACs).

1.8 Protocols, Mechanisms

A cryptographic protocol is a distributed algorithm defined by a sequence of steps precisely specifying the actions required of two or more entities to achieve a specific security objective. As opposed to a protocol, a mechanism is a more general term encompassing protocols, algorithms and non-cryptographic techniques to achieve specific security objectives. Protocols play a major role in cryptography and are essential in meeting cryptographic goals. Encryption schemes, digital signatures, hash functions, and random number generation are among the primitives which may be utilized to build a protocol.

1.8.1 Protocol and Mechanism Failure

A protocol failure or mechanism failure occurs when a mechanism fails to meet the goals for which it was intended. Protocols and mechanisms may fail for a number of reasons:

1. Weaknesses in a particular cryptographic primitive which may be amplified by the protocol or mechanism.
2. Claimed or assumed security guarantees which are overstated or not clearly understood.
3. The oversight of some principle applicable to a broad class of primitives such as encryption.

When designing cryptographic protocols and mechanisms, the following two steps are essential:

1. Identify all assumptions in the protocol or mechanism design.
2. For each assumption, determine the effect on the security objective if that assumption is violated.

1.9 Classes of Attacks and Security Models

Over the years, many different types of attacks on cryptographic primitives and protocols have been identified. The attacks these adversaries can mount may be classified as follows:

1. A passive attack is one where the adversary only monitors the communication channel. A passive attacker only threatens confidentiality of data.
2. An active attack is one where the adversary attempts to delete, add, or in some other way alter the transmission on the channel.

A passive attack can be further subdivided into more specialized attacks for deducing plaintext from ciphertext.

1.9.1 Attacks on Encryption Schemes

The objective of the following attacks is to systematically recover plaintext from ciphertext, or even more drastically, to deduce the decryption key.

1. A ciphertext-only attack is one where the adversary tries to deduce the decryption key or plaintext by only observing ciphertext.
2. A known-plaintext attack is one where the adversary has a quantity of plaintext and corresponding ciphertext.
3. A chosen-plaintext attack is one where the adversary chooses plaintext and is then given corresponding ciphertext.
4. An adaptive chosen-plaintext attack is a chosen-plaintext attack wherein the choice of plaintext may depend on the ciphertext received from previous requests.
5. A chosen-ciphertext attack is one where the adversary selects the ciphertext and is then given the corresponding plaintext. One way to mount such an attack is for the adversary to gain access to the equipment used for decryption
6. An adaptive chosen-ciphertext attack is a chosen-ciphertext attack where the choice of ciphertext may depend on the plaintext received from previous requests.

1.9.2 Attacks on Protocols

The following is a partial list of attacks which might be mounted on various protocols. Until a protocol is proven to provide the service intended, the list of possible attacks can never be said to be complete.

1. Known-key attack. In this attack an adversary obtains some keys used previously and then uses this information to determine new keys.
2. Replay. In this attack an adversary records a communication session and replays the entire session, or a portion thereof, at some later point in time.
3. Impersonation. Here an adversary assumes the identity of one of the legitimate parties in a network.
4. Dictionary. This is usually an attack against passwords. An adversary can take a list of probable passwords; hash all entries in this list, and then compare this to the list of true encrypted passwords with the hope of finding matches.
5. Forward search. This attack is similar in spirit to the dictionary attack and is used to decrypt messages.
6. Interleaving attack. This type of attack usually involves some form of impersonation in an authentication protocol.

2. CRYPTOGRAPHY FUNCTIONS

2.1 Overview

In this chapter basic functions involved in cryptography are explained. Functions which are used in the encryptions and decryption of the text such ciphers mainly block cipher and stream ciphers. Hash functions are also one of the important encryption functions. It is also explained that how the attacks are being done on cryptography and what are the authentication methods are being used so for.

2.2 Block Ciphers

The most important symmetric algorithms are block ciphers. The general operation of all block ciphers is the same - a given number of bits of plaintext (a block) are encrypted into a block of ciphertext of the same size. Thus, all block ciphers have a natural block size - the number of bits they encrypt in a single operation. This stands in contrast to stream ciphers, which encrypt one bit at a time. Any block cipher can be operated in one of several modes.

2.2.1 Iterated Block Cipher

An iterated block cipher is one that encrypts a plaintext block by a process that has several rounds. In each round, the same transformation or round function is applied to the data using a subkey. The set of subkeys are usually derived from the user-provided secret key by a key schedule. The number of rounds in an iterated cipher depends on the desired security level and the consequent trade-off with performance. In most cases, an increased number of rounds will improve the security offered by a block cipher, but for some ciphers the number of rounds required to achieve adequate security will be too large for the cipher to be practical or desirable.

2.2.2 Electronic Codebook (ECB) Mode

ECB is the simplest mode of operation for a block cipher. The input data is padded out to a multiple of the block size, broken into an integer number of blocks, each of which is encrypted independently using the key. In addition to simplicity, ECB has the advantage of allowing any block to be decrypted independently of the others. Thus, lost data blocks do not affect the decryption of other blocks. The disadvantage of ECB is that it aids known-plaintext attacks. If the same block of plaintext is encrypted twice with ECB, the two resulting blocks of ciphertext will be the same.

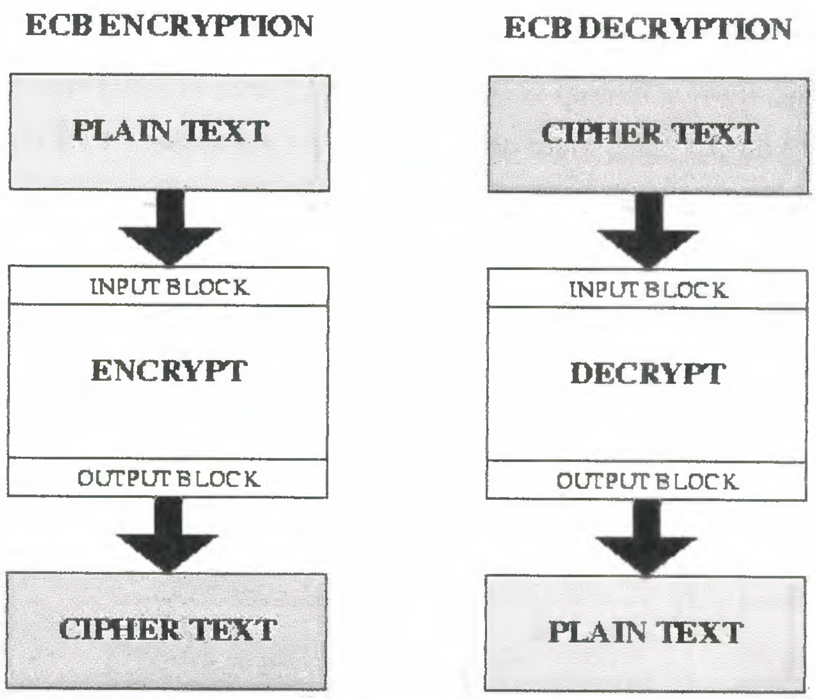


Figure 2.1: Shows a ECB Encryption/Decryption Model

2.2.3 Cipher Block Chaining (CBC) Mode

CBC is the most commonly used mode of operation for a block cipher. Prior to encryption, each block of plaintext is XOR-ed with the prior block of ciphertext. After decryption, the output of the cipher must then be XOR-ed with the previous ciphertext to recover the original plaintext. The first block of plaintext is XOR-ed with an initialization vector (IV), which is usually a block of random bits transmitted in the clear. CBC is more secure than ECB because it effectively scrambles the plaintext prior to each encryption step. Since the ciphertext is constantly changing, two identical blocks of plaintext will encrypt to two different blocks of ciphertext. The disadvantage of CBC is that the encryption of a data block becomes dependent on all the blocks prior to it. A lost block of data will also prevent decoding of the next block of data. CBC can be used to convert a block cipher into a hash algorithm. To do this, CBC is run repeatedly on the input data, and all the ciphertext is discarded except for the last block, which will depend on all the data blocks in the message. This last block becomes the output of the hash function.

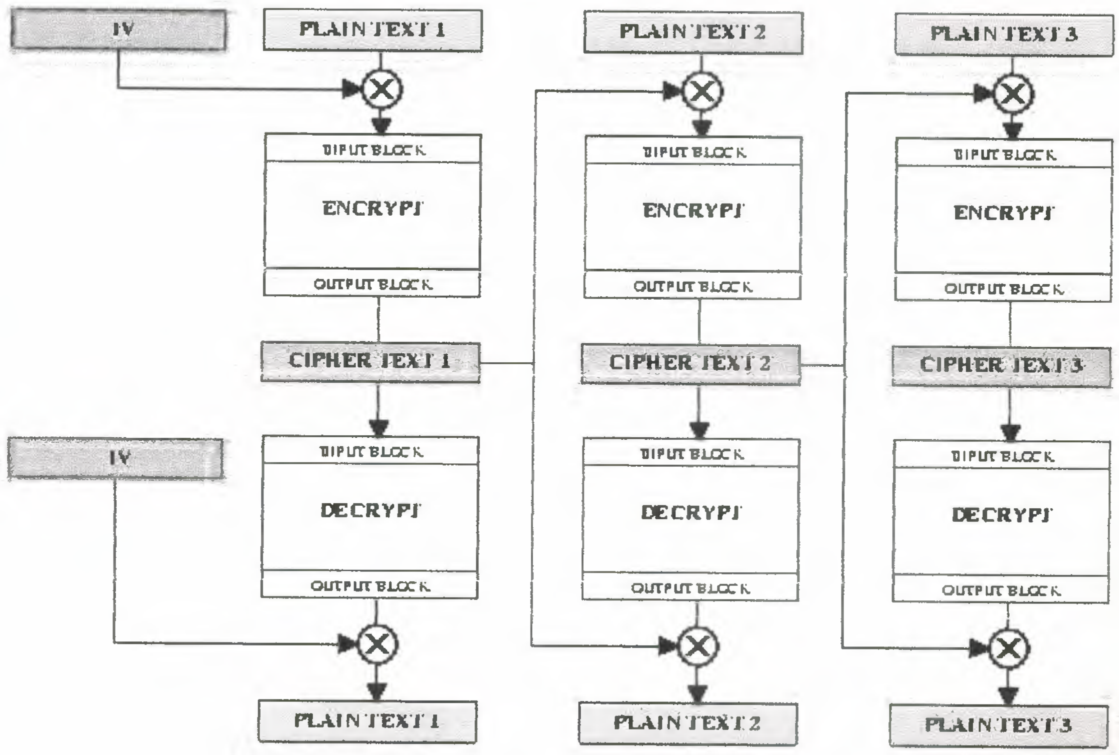


Figure 2.2: Shows a CBC Encryption/Decryption Model

2.2.4 Feistel Ciphers

The figure shows the general design of a Feistel cipher, a scheme used by almost all modern block ciphers. The input is broken into two equal size blocks, generally called left (L) and right (R), which are then repeatedly cycled through the algorithm. At each cycle, a hash function (f) is applied to the right block and the key, and the result of the hash is XOR-ed into the left block. The blocks are then swapped. The XOR-ed result becomes the new right block and the unaltered right block becomes the left block. The process is then repeated a number of times.

The hash function is just a bit scrambler. The correct operation of the algorithm is not based on any property of the hash function, other than it is completely deterministic; i.e. if it's run again with the exact same inputs, identical output will be produced. To decrypt, the ciphertext is broken into L and R blocks, and the key and the R block are run through the hash function to get the same hash result used in the last cycle of encryption; notice that the R block was unchanged in the last encryption cycle. The hash is then XOR'ed into the L block to reverse the last encryption cycle, and the process is repeated until all the encryption cycles have been backed out. The security of a Feistel cipher depends primarily on the key size and the irreversibility of the hash function. Ideally, the output of the hash function should appear to be random bits from which nothing can be determined about the input(s).

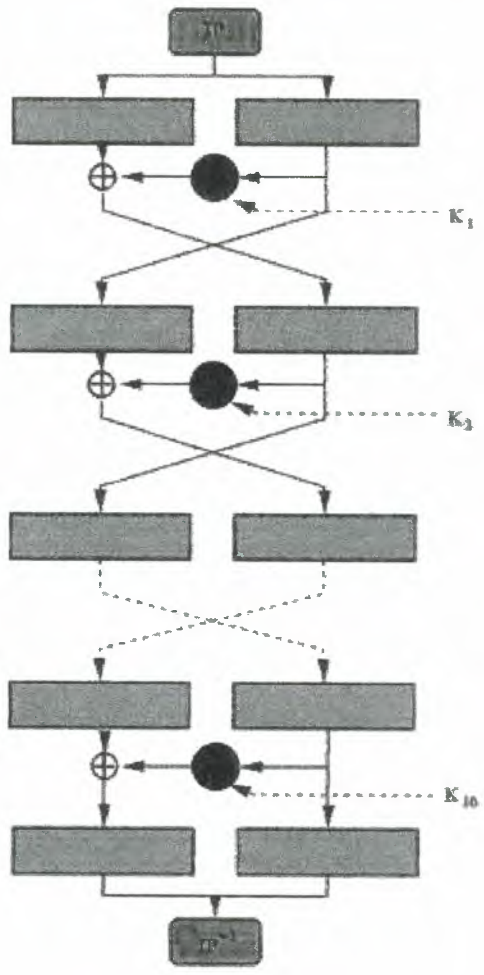


Figure 2.3: Shows a Feistel Model

2.2.5 Data Encryption Standard (DES)

DES is a Feistel-type Substitution-Permutation Network (SPN) cipher. DES uses a 56-bit key which can be broken using brute-force methods, and is now considered obsolete. A 16 cycle Feistel system is used, with an overall 56-bit key permuted into 16 48-bit subkeys, one for each cycle. To decrypt, the identical algorithm is used, but the order of subkeys is reversed. The L and R blocks are 32 bits each, yielding an overall block size of 64 bits. The hash function "*f*", specified by the standard using the so-called "S-boxes", takes a 32-bit data block and one of the 48-bit subkeys as input and produces 32 bits of output. Sometimes DES is said to use a 64-bit key, but 8 of the 64 bits are used only for parity checking, so the effective key size is 56 bits.

2.2.5.1 Triple DES

Triple DES was developed to address the obvious flaws in DES without designing a whole new cryptosystem. Triple DES simply extends the key size of DES by applying the algorithm three times in succession with three different keys. The combined key size is thus 168 bits (3 times 56), beyond the reach of brute-force techniques such as those used by the EFF DES Cracker. Triple DES has always been regarded with some suspicion, since the original algorithm was never designed to be used in this way, but no serious flaws have been uncovered in its design, and it is today a viable cryptosystem used in a number of Internet protocols.

2.3 Stream Ciphers

A stream cipher is a symmetric encryption algorithm. Stream ciphers can be designed to be exceptionally fast, much faster in fact than any block cipher. While block ciphers operate on large blocks of data, stream ciphers typically operate on smaller units of plaintext, usually bits. The encryption of any particular plaintext with a block cipher will result in the same ciphertext when the same key is used. With a stream cipher, the transformation of these smaller plaintext units will vary, depending on when they are encountered during the encryption process.

A stream cipher generates what is called a keystream and encryption is provided by combining the keystream with the plaintext, usually with the bitwise XOR operation. The generation of the keystream can be independent of the plaintext and ciphertext or it can depend on the data and its encryption.

Current stream ciphers are most commonly attributed to the appealing of theoretical properties of the one-time pad, but there have been no attempts to standardize on any particular stream cipher proposal as has been the case with block ciphers. Interestingly, certain modes of operation of a block cipher effectively transform it into a keystream generator and in this way; any block cipher can be used as a stream cipher. However, stream ciphers with a dedicated design are likely to be much faster.

2.3.1 Linear Feedback Shift Register

A Linear Feedback Shift Register (LFSR) is a mechanism for generating a sequence of binary bits. The register consists of a series of cells that are set by an initialization vector that is, most often, the secret key. The behavior of the register is regulated by a clock and at each clocking instant, the contents of the cells of the register are shifted right by one position, and the XOR of a subset of the cell contents is placed in the leftmost cell. One bit of output is usually derived during this update procedure.

LFSRs are fast and easy to implement in both hardware and software. With a sensible choice of feedback taps the sequences that are generated can have a good statistical appearance. However, the sequences generated by single LFSRs are not secure because a powerful mathematical framework has been developed over the years which allows for their straightforward analysis. However, LFSRs are useful as building blocks in more secure systems.

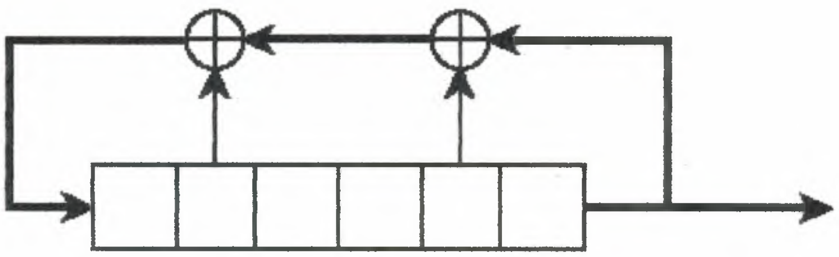


Figure 2.1: Shows a Linear Feed Back Register Model

2.3.1.1 Shift Register Cascades

A shift register cascade is a set of LFSRs connected together in such a way that the behavior of one particular LFSR depends on the behavior of the previous LFSRs in the cascade. This dependent behavior is usually achieved by using one LFSR to control the clock of the following LFSR. For instance one register might be advanced by one step if the preceding register output is 1 and advanced by two steps otherwise. Many different configurations are possible and certain parameter choices appear to offer very good security.

2.3.1.2 Shrinking and Self-Shrinking Generators

It is a stream cipher based on the simple interaction between the outputs from two LFSRs. The bits of one output are used to determine whether the corresponding bits of the second output will be used as part of the overall keystream. The shrinking generator is simple and scaleable, and has good security properties. One drawback of the shrinking generator is that the output rate of the keystream will not be constant unless precautions are taken. A variant of the shrinking generator is the self-shrinking generator, where instead of using one output from one LFSR to "shrink" the output of another, the output of a single LFSR is used to extract bits from the same output.

2.3.2 Other Stream Ciphers

There are a vast number of alternative stream ciphers that have been proposed in cryptographic literature as well as an equally vast number that appear in implementations and products world-wide. Many are based on the use of LFSRs since such ciphers tend to be more amenable to analysis and it is easier to assess the security that they offer.

There are essentially four distinct approaches to stream cipher design. The first is termed the information-theoretic approach explained in one-time pad. The second approach is that of system-theoretic design. In essence, the cryptographer designs the cipher along established guidelines which ensure that the cipher is resistant to all known attacks. While there is, of course, no substantial guarantee that future cryptanalysis will be unsuccessful, it is this design approach that is perhaps the most common in cipher design. The third approach is to attempt to relate the difficulty of breaking the stream cipher to solving some difficult problem. This complexity-theoretic approach is very appealing, but in practice the ciphers that have been developed tend to be rather slow and impractical. The final approach is that of designing a randomized cipher. Here the aim is to ensure that the cipher is resistant to any practical amount of cryptanalytic work rather than being secure against an unlimited amount of work.

2.3.2.1 One-time Pad

A one-time pad, sometimes called the Vernam cipher, uses a string of bits that is generated completely at random. The keystream is the same length as the plaintext message and the random string is combined using bitwise XOR with the plaintext to produce the ciphertext. Since the entire keystream is random, an opponent with infinite computational resources can only guess the plaintext if he sees the ciphertext. Such a cipher is said to offer perfect secrecy and the analysis of the one-time pad is seen as one of the cornerstones of modern cryptography.

2.4 Hash Functions

Hash Functions take a block of data as input, and produce a hash or message digest as output. The usual intent is that the hash can act as a signature for the original data, without revealing its contents. Therefore, it's important that the hash function be irreversible - not only should it be nearly impossible to retrieve the original data, it must also be unfeasible to construct a data block that matches some given hash value. Randomness, however, has no place in a hash function, which should completely deterministic. Given the exact same input twice, the hash function should always produce the same output. Even a single bit changed in the input, though, should produce a different hash value. The hash value should be small enough to be manageable in further manipulations, yet large enough to prevent an attacker from randomly finding a block of data that produces the same hash.

MD5, documented in RFC 1321, is perhaps the most widely used hash function at this time. It takes an arbitrarily sized block of data as input and produces a 128-bit (16-byte) hash. It uses bitwise operations, addition, and a table of values based on the sine function to process the data in 64-byte blocks. RFC 1810 discusses the performance of MD5, and presents some speed measurements for various architectures.

Hash functions can't be used directly for encryption, but are very useful for authentication. One of the simplest uses of a hash function is to protect passwords. UNIX systems, in particular, will apply a hash function to a user's password and store the hash value, not the password itself. To authenticate the user, a password is requested, and the response runs through the hash function. If the resulting hash value is the same as the one stored, then the user must have supplied the correct password, and is authenticated. Since the hash function is irreversible, obtaining the hash values doesn't reveal the passwords to an attacker. In practice, though, people will often use guessable passwords, so obtaining the hashes might reveal passwords to an attacker who, for example, hashes all the words in the dictionary and compares the results to the password hashes.

Another use of hash functions is for interactive authentication over the network. Transmitting a hash instead of an actual password has the advantage of not revealing the password to anyone sniffing on the network traffic. If the password is combined with some changing value, then the hashes will be different every time, preventing an attacker from using an old hash to authenticate again. The server sends a random challenge to the client, which combines the challenge with the password, computes the hash value, and sends it back to the server. The server, possessing both the stored secret password and the random challenge, performs the same hash computation, and checks its result against the reply from the client. If they match, then the client must know the password to have correctly computed the hash value. Since the next authentication would involve a different random challenge, the expected hash value would be different, preventing an attacker from using a replay attack. Thus, hash functions, though not encryption algorithms in their own right, can be used to provide significant security services, mainly identity authentication.

2.4.1 Hash functions for hash table lookup

A hash function for hash table lookup should be fast, and it should cause as few collisions as possible. If you know the keys you will be hashing before you choose the hash function, it is possible to get zero collisions -- this is called perfect hashing. Otherwise, the best you can do is to map an equal number of keys to each possible hash value and make sure that similar keys are not unusually likely to map to the same value. Unfortunately, that hash is only average. The problem is the per-character mixing: it only rotates bits, it doesn't really mix them. Every input bit affects only 1 bit of hash until the final %. If two input bits land on the same hash bit, they cancel each other out. Also, % can be extremely slow.

2.5 Attacks on Ciphers

Here the different kinds of possible attacks what have been observed so for and can be expected are explained in detail.

2.5.1 Exhaustive Key Search

Exhaustive key search, or brute-force search, is the basic technique of trying every possible key in turn until the correct key is identified. To identify the correct key it may be necessary to possess a plaintext and its corresponding ciphertext, or if the plaintext has some recognizable characteristic, ciphertext alone might suffice. Exhaustive key search can be mounted on any cipher and sometimes a weakness in the key schedule of the cipher can help improve the efficiency of an exhaustive key search attack. Advances in technology and computing performance will always make exhaustive key search an increasingly practical attack against keys of a fixed length. When DES was designed, it was generally considered secure against exhaustive key search without a vast financial investment in hardware. Over the years, this line of attack will become increasingly attractive to a potential adversary.

While the 56-bit key in DES now only offers a few hours of protection against exhaustive search by a modern dedicated machine, the current rate of increase in computing power is such that 80-bit key can be expected to offer the same level of protection against exhaustive key search in 18 years time as DES does today.

2.5.2 Differential Cryptanalysis

Differential cryptanalysis is a type of attack that can be mounted on iterative block ciphers. Differential cryptanalysis is basically a chosen plaintext attack and relies on an analysis of the evolution of the differences between two related plaintexts as they are encrypted under the same key. By careful analysis of the available data, probabilities can be assigned to each of the possible keys and eventually the most probable key is identified as the correct one.

Differential cryptanalysis has been used against a great many ciphers with varying degrees of success. In attacks against DES, its effectiveness is limited by what was very careful design of the S-boxes during the design of DES. Differential cryptanalysis has also been useful in attacking other cryptographic algorithms such as hash functions.

2.5.3 Linear Cryptanalysis

Linear cryptanalysis is a known plaintext attack and uses a linear approximation to describe the behavior of the block cipher. Given sufficient pairs of plaintext and corresponding ciphertext, bits of information about the key can be obtained and increased amounts of data will usually give a higher probability of success. There have been a variety of enhancements and improvements to the basic attack. Differential-linear cryptanalysis is an attack which combines elements of differential cryptanalysis with those of linear cryptanalysis. A linear cryptanalytic attack using multiple approximations might allow for a reduction in the amount of data required for a successful attack.

2.5.4 Weak Key for a Block Cipher

Weak keys are secret keys with a certain value for which the block cipher in question will exhibit certain regularities in encryption or, in other cases, a poor level of encryption. For instance, with DES there are four keys for which encryption is exactly the same as decryption. This means that if one were to encrypt twice with one of these weak keys, then the original plaintext would be recovered. For IDEA there is a class of keys for which cryptanalysis is greatly facilitated and the key can be recovered. However, in both these cases, the number of weak keys is such a small fraction of all possible keys that the chance of picking one at random is exceptionally slight. In such cases, they pose no significant threat to the security of the block cipher when used for encryption.

Of course for other block ciphers, there might well be a large set of weak keys (perhaps even with the weakness exhibiting itself in a different way) for which the chance of picking a weak key is too large for comfort. In such a case, the presence of weak keys would have an obvious impact on the security of the block cipher.

2.5.5 Algebraic Attacks

Algebraic attacks are a class of techniques which rely for their success on some block cipher exhibiting a high degree of mathematical structure. For instance, it is conceivable that a block cipher might exhibit what is termed a group structure. If this were the case, then encrypting a plaintext under one key and then encrypting the result under another key would always be equivalent to single encryption under some other single key. If so, then the block cipher would be considerably weaker, and the use of multiple encryptions would offer no additional security over single encryption. For most block ciphers, the question of whether they form a group is still open. For DES, however, it is known that the cipher is not a group. There are a variety of other concerns with regards to algebraic attacks.

2.5.6 Data Compression Used With Encryption

Data compression removes redundant character strings in a file. This means that the compressed file has a more uniform distribution of characters. In addition to providing shorter plaintext and ciphertext, which reduces the amount of time needed to encrypt, decrypt and transmit a file, the reduced redundancy in the plaintext can potentially hinder certain cryptanalytic attacks.

By contrast, compressing a file after encryption is inefficient. The ciphertext produced by a good encryption algorithm should have an almost statistically uniform distribution of characters. As a consequence, a compression algorithm should be unable to find redundant patterns in such text and there will be little, if any, data compression. In fact, if a data compression algorithm is able to significantly compress encrypted text, then this indicates a high level of redundancy in the ciphertext which, in turn, is evidence of poor encryption.

2.6 When an Attack Become Practical

There is no easy answer to this question since it depends on many distinct factors. Not only must the work and computational resources required by the cryptanalyst be reasonable, but the amount and type of data required for the attack to be successful must also be taken into account. One classification distinguishes among cryptanalytic attacks according to the data they require in the following way: chosen plaintext or chosen ciphertext, known plaintext, and ciphertext-only. This classification is not particular to secret-key ciphers and can be applied to cryptanalytic attacks on any cryptographic function. A chosen plaintext or chosen ciphertext attack gives the cryptanalyst the greatest freedom in analyzing a cipher. The cryptanalyst chooses the plaintext to be encrypted and analyzes the plaintext together with the resultant ciphertext to derive the secret key. Such attacks will, in many circumstances, be difficult to mount but they should not be discounted. A known plaintext attack is more useful to the cryptanalyst than a chosen plaintext attack (with the same amount of data) since the cryptanalyst now requires a certain numbers of plaintexts and their corresponding ciphertexts without specifying the values of the plaintexts. This type of information is presumably easier to collect. The most practical attack, but perhaps the most difficult to actually discover, is a ciphertext-only attack. In such an attack, the cryptanalyst merely intercepts a number of encrypted messages and subsequent analysis somehow reveals the key used for encryption. Note that some knowledge of the statistical distribution of the plaintext is required for a ciphertext-only attack to succeed.

An added level of sophistication to the chosen text attacks is to make them adaptive. By this we mean that the cryptanalyst has the additional power to choose the text that is to be encrypted or decrypted after seeing the results of previous requests. The computational effort and resources together with the amount and type of data required are all important features in assessing the practicality of some attack.

2.7 Strong Password-Only Authenticated Key Exchange

A new simple password exponential key exchange method (SPEKE) is described. It belongs to an exclusive class of methods which provide authentication and key establishment over an insecure channel using only a small password, without risk of off-line dictionary attack. SPEKE and the closely-related Diffie-Hellman Encrypted Key Exchange (DH-EKE) are examined in light of both known and new attacks, along with sufficient preventive constraints. Although SPEKE and DH-EKE are similar, the constraints are different. The class of strong password-only methods is compared to other authentication schemes. Benefits, limitations, and tradeoffs between efficiency and security are discussed. These methods are important for several uses, including replacement of obsolete systems, and building hybrid two-factor systems where independent password-only and key-based methods can survive a single event of either key theft or password compromise.

It seems paradoxical that small passwords are important for strong authentication. Clearly, cryptographically large passwords would be better, if only ordinary people could remember them. Password verification over an insecure network has been a particularly tough problem, in light of the ever-present threat of dictionary attack. Password problems have been around so long that many have assumed that strong remote authentication using only a small password is impossible. In fact, it can be done. In this paper we outline the problem, and describe a new simple password exponential key exchange, SPEKE, which performs strong authentication, over an insecure channel, using only a small password. That a small password can accomplish this alone goes against common wisdom. This is not your grandmother's network login. We compare SPEKE to the closely-related Diffie-Hellman Encrypted Key Exchange, and review the potential threats and countermeasures in some detail. We show that previously-known and new attacks against both methods are dissatisfied when proper constraints are applied. These methods are broadly useful for authentication in many applications: bootstrapping new system installations, cellular phones or other keypad systems, diskless workstations, user-to-user applications, multi-factor password + key systems, and for upgrading obsolete password systems. More generally, they are needed anywhere that prolonged key storage is risky or impractical, and where the communication channel may be insecure.

2.7.1 The Remote Password Problem

Ordinary people seem to have a fundamental inability to remember anything larger than a small secret. Yet most methods of remote secret-based authentication presume the secret to be large. We really want to use an easily memorized small secret password, and not are susceptible to dictionary attack. We make a clear distinction between passwords and keys: Passwords must be memorized, and are thus small, while keys can be recorded, and can be much larger. The problem is that most methods need keys that are too large to be easily remembered. User-selected passwords are often confined to a very small, easily searchable space, and attempts to increase the size of the space just make them hard to remember. Bank-card PIN codes use only 4-digits to remove even the temptation to write them down. A ten-digit phone number has about 30 bits, which compels many people to record them. Meanwhile, strong symmetric keys need 60 bits or more, and nobody talks about memorizing public-keys. It is also fair to assume that a memorizable password belongs to a brute-force searchable space. With ever-increasing computer power, there is a growing gap between the size of the smallest safe key and the size of the largest easily remembered password.

The problem is compounded by the need to memorize multiple passwords for different purposes. One example of a small-password-space attack is the verifiable plain-text dictionary attack against login. A general failure of many obsolete password methods is due to presuming passwords to be large. We assume that any password belongs to a cryptographically-small space, which is also brute-force searchable with a modest effort. Large passwords are arguably weaker since they can't be memorized.

So why do we bother with passwords? A pragmatic reason is that they are less expensive and more convenient than smart-cards and other alternatives. A stronger reason is that, in a well-designed and managed system, passwords are more resistant to theft than persistent stored keys or carry-around tokens. More generally, passwords represent something you know, one of the "big three" categories of factors in authentication.

2.7.2 Characteristics of Strong Password-only Methods

We now define exactly what we mean by strong password-only remote authentication. We first list the desired characteristics for these methods, focusing on the case of user-to-host authentication. Both SPEKE and DH-EKE have these distinguishing characteristics.

1. Prevent off-line dictionary attack on small passwords.
2. Survive on-line dictionary attack.
3. Provide mutual authentication.
4. Integrated key exchange.
5. User needs no persistent recorded

   (a) Secret data, or

   (b) Sensitive host-specific data.

Since we assume that all passwords are vulnerable to dictionary attack, given the opportunity, we need to remove the opportunities. On-line dictionary attacks can be easily detected, and thwarted, by counting access failures. But off-line dictionary attack presents a more complex threat. These attacks can be made by someone posing as a legitimate party to gather information, or by one who monitors the messages between two parties during a legitimate valid exchange. Even tiny amounts of information "leaked" during an exchange can be exploited. The method must be immune to such off-line attack, even for tiny passwords. This is where SPEKE and DH-EKE excel.

2.7.2.1 SPEKE

The simple password exponential key exchange (SPEKE) has two stages. The first stage uses a DH exchange to establish a shared key K, but instead of the commonly used fixed primitive base g, a function f converts the password S into a base for exponentiation. The rest of the first stage is pure Diffie-Hellman, where Alice and Bob start out by choosing two random numbers R_A and R_B:

Table 2.1: Shows First Stages of SPEKE

S1.	Alice computes:	$Q_A = f(S)^{R_A} \bmod p$,	$A \rightarrow B$: Q_A.
S2.	Bob computes:	$Q_B = f(S)^{R_B} \bmod p$,	$B \rightarrow A$: Q_B.
S3.	Alice computes:	$K = h(Q_B^{R_A} \bmod p)$	
S4.	Bob computes:	$K = h(Q_A^{R_B} \bmod p)$	

In the second stage of SPEKE, both Alice and Bob confirm each other's knowledge of K before proceeding to use it as a session key. One way is:

Table 2.2: Shows Second Stage of SPEKE

S5.	Alice chooses random C_A,	$A \rightarrow B$: $E_K (C_A)$.
S6.	Bob chooses random C_B,	$B \rightarrow A$: $E_K (C_B, C_A)$.
S7.	Alice verifies that C_A is correct,	$A \rightarrow B$: $E_K (C_B)$.
S8.	Bob verifies that C_B is correct.	

To prevent discrete log computations, which can result in the attacks the value of p^{-1} must have a large prime factor q. The function *f* is chosen in SPEKE to create a base of large prime order. This is different than the commonly used primitive base for DH. The use of a prime-order group may also be of theoretical importance.

Other variations of the verification stage are possible. This stage is identical to that of the verification stage of DH-EKE. More generally, verification of K can use any classical method, since K is cryptographically large. This example repeatedly uses a one-way hash function:

Table 2.3: Shows Verification Stage of SPEKE

S5.	Alice sends proof of K:	$A \rightarrow B$: h(h(K))
S6.	Bob verifies h(h(K)) is correct,	$B \rightarrow A$: h(K)
S7.	Alice verifies h (K)) is correct.	

This approach uses K in place of explicit random numbers, which is possible since K was built with random information from both sides.

2.7.2.2 DH-EKE

DH-EKE (Diffie-Hellman Encrypted Key Exchange) are the simplest of a number of methods. The method can also be divided into two stages. The first stage uses a DH exchange to establish a shared key K, where one or both parties encrypts the exponential using the password S. With knowledge of S, they can each decrypt the other's message using E_S^{-1} and compute the same key K.

Table 2.4: Shows First Stage of DH-EKE

D1.	Alice computes:	$Q_A = g^{R}{}_{A} \bmod p,$	$A \rightarrow B$: $E_S (Q_A)$.
D2.	Bob computes:	$Q_B = g^{R}{}_{B} \bmod p,$	$B \rightarrow A$: $E_S (Q_B)$.
D3.	Alice computes:	$K = h(Q_B{}^{R}{}_{A} \bmod p)$	
D4.	Bob computes:	$K = h(Q_A{}^{R}{}_{B} \bmod p)$	

It is widely suggested that at least one of the encryption steps can be omitted, but this may leave the method open to various types of attacks. The values of p and g, and the symmetric encryption function E_S must be chosen carefully to preserve the security of DH-

EKE. In the second stage of DH-EKE, both Alice and Bob confirm each other's knowledge of K before proceeding to use it as a session key. However, with DH-EKE the order of the verification messages can also be significant.

2.8 Different kinds of Security Attacks

Here different kinds of attacks on the security in authentication which have been observed so for and which are expected are explained in detail.

2.8.1 Discrete Log Attack

As the security of these schemes rests primarily on exponentiation being a one-way function, there is a general threat of an attacker computing the discrete logarithms on the exponentials. Known methods of discrete log require a massive pre-computation for each specific modulus. Modulus size is a primary concern. No method is currently known that could ever compute the discrete log for a safe modulus greater than a couple thousand bits; however a concerted attack on a 512 bit modulus may be soon feasible with considerable expense. Somewhere in between is an ideal size balancing speed against the need for security, in a given application.

It is noted that if we assume that a discrete log pre-computation has been made for the modulus, a password attack must also compute the specific log for each entry in the password dictionary (until a match is found). It is also noted that for any session established with a modulus vulnerable to log attack, perfect forward secrecy is no longer guaranteed, providing another reason for keeping the discrete log computation out of reach. The feasibility of a pre-computed log table remains a primary concern, and the efficiency of the second phase of the attack is secondary.

2.8.2 Leaking Information

If one is not careful, the exchanged messages Q_x may reveal discernible structure, and can "leak" information about S, enabling a partition attack. This section shows how to prevent these attacks.

2.8.2.1 DH-EKE Partition Attack

In DH-EKE, Alice and Bob use a Diffie-Hellman exponential key exchange in the group Z_p^*, with a huge prime p, where p^{-1} has a huge prime factor q. Then we use the traditional preference for g as a primitive root of p. In fact, g must be primitive to prevent a partition attack by an observer. A third party can do trial decryptions of E_S ($g^R{}_X$ mod p) using a dictionary of S_i. If g is not primitive, a bad guess S_i is confirmed by a primitive result. In general, the encrypted exponentials Q_x must contain no predictable structure to prevent this attack against DH-EKE. Constraining g to be primitive insures a random distribution across Z_p^*.

2.8.2.2 SPEKE Partition Attack

Using a primitive base is not required in SPEKE. If the base f(S) is an arbitrary member of Z_p^*, since the exponentials are not encrypted, an observer can test the result for membership in smaller subgroups. When the result is a primitive root of p, he knows that the base also is primitive. For a safe prime p, this case reveals 1 bit of information about S. When p varies, as has been recommended when using a reduced modulus size, new information from runs with different p allow a partition attack to reduce a dictionary of possible S_i. When, for any S, the base f(S) is a generator of a particular large prime subgroup, and then no information is leaked through the exponential result. Suitable functions for f(S) create a result of known large order. We assume the use of a large prime-order base in SPEKE for the rest of the discussion. Because SPEKE does not encrypt the exponentials, a formal analysis of security may be simpler to achieve for SPEKE than for DH-EKE. The prime-order subgroup is the same as that used in the DSA and Digital signature methods.

2.8.3 Stolen Session Key Attack

In an analysis of several flavors of EKE, where a stolen session key K is used to mount a dictionary attack on the password. The attack on the public-key flavor of EKE is also noted which correctly points out that DH-EKE resists this attack (as does SPEKE). Resistance to this attack is closely related to perfect forward secrecy, which also isolates one kind of sensitive data from threats to another. We note that, in DH-EKE, a stolen value of R_A in addition to K permits a dictionary attack against the password S. For each trial password S_i, the attacker computes:

$$K' = (E_{Si}^{-1}(E_S\,(g^R{}_B)))^{RA}$$

When K' equals K, he knows that S_i equals S. SPEKE is also vulnerable to an attack using R_A to find S. These concerns highlight the need to promptly destroy ephemeral sensitive data, such as R_A and R_B. It also notes a threat when the long-term session key K is used in an extra stage of authentication of the extended A-EKE method; a dictionary attack is possible using the extra messages. To counter this threat, one can use K for the extra stage, set K' = h (K) using a strong one-way function, and promptly discard K.

2.8.4 Verification Stage Attacks

The verification stage of either DH-EKE or SPEKE is where both parties prove to each other knowledge of the shared key K. Because K is cryptographically large, the second stage is presumed to be immune to brute-force attack, and thus verifying K can be done by traditional means. However, the order of verification may be important to resist the protocol attack against DH-EKE.

2.8.5 The "password-in-exponent" Attack

It is generally a good idea for $f(S)$ to create a result of the same known order for all S, so that testing the order of the exponential doesn't reveal information about S. When considering suitable functions, it may be tempting to choose $f(S) = g_c^{h(S)}$ for some fixed prime-order g_c and some well-known hash function h. Unfortunately, while this is a convenient way to convert an arbitrary number into a generator of a prime-order group, it creates an opening for attack. To show the attack, let's assume that $g_c = 2$, and $h(S) = S$, so that $f(S) = 2^S$. Alice's protocol can be rewritten as:

1. Choose a random R_A.
2. Compute $Q_A = 2^{(S R_A)} \bmod p$.
3. Send Q_A to Bob.
4. Receive Q_B from Bob.
5. Compute $K = Q_B^{R_A} \bmod p$.

Bob should perform his part, sending Q_B to Alice. The problem is that an attacker Barry can perform a dictionary attack off-line after performing a single failed exchange. His initial steps are:

1. Choose a random X.
2. Compute $Q_B = 2^X$.
3. Receive Q_A from Alice
4. Send Q_B to Alice.
5. Receive verification data for K from Alice.

Barry then goes off-line to perform the attack as follows:

For each candidate password S':

Compute $K' = (Q_B^X)^{1/S'} \bmod p$.

Compare Alice's verification message for K to K', when they match he knows that S' = S.

This attack works because:

$$K' = Q_A^{(X/S')} \bmod p$$

$$= 2^{(S R_A)(X/S')} \bmod p$$

$$= 2^{(X R_A S/S')} \bmod p$$

$$= Q_B^{(R_A^{S/S'})} \bmod p$$

$$= K^{(S/S')} \bmod p$$

Thus, when S' = S, K' = K. More generally, the attack works because the dictionary of passwords $\{S_1, S_2... S_n\}$ is equivalent to a dictionary of exponents $E = \{e_1, e_2... e_n\}$, such that for a given fixed generator g_c, the value of f (S_i) for each candidate can be computed as $g_c^{e_i}$. This allows the password to be effectively removed from the DH computation.

In general, we must insure that no such dictionary E is available to an attacker. We should note that while it is true that for any function f there will always be some fixed g_c and hypothetical dictionary E that corresponds to f(S), for most functions f, computing the value of each e_i requires a discrete log computation. This makes the dictionary E generally unknowable to anyone. As a specific example, for the function $f(S) = S$, the attack is infeasible. The password-in-exponent attack is possible only when f(S) is equivalent to exponentiation (within the group) of some fixed gc to a power which is a known function of S.

2.9 A Logic of Authentication

In computer networks the communicating parties share not only the media, but also the set of rules on how to communicate. These rules, or protocols, have become more and more important in communication networks and distributed computing. However, the increase of the knowledge of the communication protocols has also brought up the question of how to secure the communication against intruders. To solve this, a large number of cryptographic protocols have been produced.

Cryptographic protocols were developed to combat against various attacks of intruders in computer networks. Nowadays, the comprehension is that the security of data should rely on the underlying cryptographic technology, and that the protocols should be open and available. However, many protocols have been found to be vulnerable to attacks that do not require breaking the encryption, but instead manipulate the messages in the protocol to gain some advantage. The advantages range from the compromise of confidentiality to the ability to impersonate another user.

As there are different protocol designs decisions appropriate to different circumstances, there also exists a variety of authentication protocols. Protocols often differ in their final states, and sometimes they even depend on assumptions that one would not care to make. To understand what is really accomplished with such a protocol, a formal description method is needed. The goal of the logic of authentication is to formally describe the knowledge and the beliefs of the parties involved in authentication, the evolution of the knowledge and the beliefs while analyzing the protocol step by step. After the analysis, all the final states of the protocol are set out.

3. ENCRYPTION AND DECRYPTION

3.1 Overview

There two main types of encryption

- Asymmetric encryption, also known as, public -key encryption
- Symmetric encryption

An asymmetric encryption is an encryption system in which the sender and receiver of a message share a single, common key that is used to encrypt and decrypt the message. Symmetric-key systems are simpler and faster, but their main drawback is that the two parties must somehow exchange the key in a secure way. One example of an asymmetric encryption system is the Data Encryption Standard.

Public-key encryption is a cryptographic system that uses two keys opposed to an asymmetric encryption that only uses one common key. The two keys used are -- the public key which is known to everyone and the private or secret key known only to the recipient of the message. An important element to the public key system is that the public and private keys are related in such a way that only the public key can be used to encrypt messages and only the corresponding private key can be used to decrypt them. Moreover, it is virtually impossible to deduce the private key if you know the public key. One example of a public-key encryption system would be Pretty Good Privacy.

3.2 Different types of Cryptosystems/Encryptions:

There are four ways of encryption that we mostly use in network system

- Rivest, Shamir, Adleman, RSA
- Pretty Good Privacy, PGP
- Keyless Encryption, KE
- One-Time Pads, OTP

3.2.1 RSA

RSA stands for the initials of the three men Ron Rivest, Adi Shamir, and Len Adleman. The security behind RSA lies in the difficulty of factoring large numbers into their primes. The process involves selecting two large (hundreds of digits) prime numbers (p and q), and multiplying them together to get the sum, n. These numbers are passed through a mathematical algorithm to determine the public key KU = {e, n} and the private key KR = {d, n}, which are mathematically related (the necessary equations are given at the bottom of the page). It is extremely difficult to determine e and/or d given n, thus the security of the algorithm. Once the keys have been created a message can be encrypted in blocks, and passed though the following equation:

$$C = M^e \bmod n$$

Where C is the ciphertext, M is the plaintext, and e is the recipient's public key. Similarly, the above message could be decrypted by the following equation:

$$M = C^d \bmod n$$

Where d is the recipient's private key. For example: let's assume that our M is 19 (we will use smaller numbers for simplicity, normally theses numbers would be MUCH larger). We will use 7 as p and 17 as q. Thus, n = 7 * 17 = 119. Our e is then calculated to be 5 and d is calculated to be 77. Thus our KU is {5, 119} and our KR is {77, 119}. We can then pass the needed values through equation (1) to compute C. In this case C is 66. We could then decrypt C (66) to get back our original plain text. We pass the needed values through equation (2) and get 19, our original plaintext! Try it yourself with other numbers.

Note: To determine e and d, perform the following:

Calculate $f(n) = (p - 1)(q - 1)$

Choose e to be relatively prime to $f(n)$ and less than f(n).

Determine d such that $de = 1 \bmod f(n)$ and $d < f(n)$.

3.2.2 PGP

Pretty Good Privacy. Up to this point we've talked about private-key cryptography (one key used y both parties). There was one problem with this kind of encryption: If the key was intercepted, a third party could decrypt the messages. So, the ideas of public-key cryptography were developed. Here's how it works...

Everyone has two keys: a public and a private key. When someone wants to send something to a recipient, they (the sender) encrypt it with the recipient's public key. Then the only way to decrypt it is with the recipient's private key. One of the other benefits to PGP is that it allows the sender to "sign" their messages. This proves that the message came from the sender and has not been altered in transport. Based on this theory, PGP allows everyone to publicize their public keys, while keeping their private keys secret. The result is that anyone can encrypt a message to someone else, as long as they have that person's public key.

In actuality, PGP uses a series of private key, public key and one-way hash functions to encrypt a message. A one-way hash function takes some plaintext and translates it into a specific hash. The hash is unique to the message (like a fingerprint is to a person). The hash is also non-reversible, hence the name one-way. Let's run through an example of what PGP does to encrypt and decrypt an e-mail message. Our sender will be Chris and our receiver will be Brian.

- Chris writes his message.
- Chris uses a one-way hash function (such as MD5) to create a hash for the message.
- Chris, via RSA or some other digital signature algorithm, signs the hash with his private key.
- Chris merges the message and the signature, resulting in a new signed message.
- A random encryption key is generated, the session key.
- Chris uses the session key to encrypt the message, using DES or some other private key method.
- Chris gets Brian's public key.
- Chris then encrypts the key with Brian's public key, via RSA or some other public key method.
- Chris merges the encrypted message and the encrypted key and mails it to Brian.

Once Brian receives the message he can have PGP decrypt it. Here's what it would do:

- Brian separates the encrypted message and the encrypted session key.
- Using RSA, Brian decrypts the session key.
- Using DES, Brian decrypts the message with the decrypted session key.
- Brian then separates the message and the signature.
- Using MD5, Brian calculates the hash value of the message.
- Brian gets Chris' public key.
- Via RSA, and Chris' public key, Brian decrypts the signature.
- Brian then compares the hash value and the decrypted signature. If they are the same, Brian knows that the message is authentic and has not been altered since Chris signed it.

3.3 Public Key Cryptography

The basis for the security associated with most public key cryptosystems is the difficulty of factoring large integers. This difficulty is not a proven mathematical assumption, however. Because there is a mathematical relation between the two keys, one could theoretically calculate the private key from the public key. The difficulty and thus the time associated with solving these problems grow exponentially with the size of the key. For example, a computer capable of performing 10^10 computations per second would factor a 100 digit number in 10^40 seconds. Considering that 10^17 is the estimated age of the universe, this is a pretty long time to wait. The likelihood of someone creating an algorithm that would make this anything less than an extremely difficult problem is very low. It would seem reasonable then to come to the conclusion that these public key cryptosystems will remain secure. This may not be the case, however, if one considers non-mainstream forms of computing, particularly quantum computing.

Quantum computing can be viewed as massive parallel computing, but instead of having many processors working simultaneously, one quantum processor does the work. The theoretical power of this type of computing would decrease the time to solve one of these problems to a fraction of a second for any length of key. This would shatter the security public key algorithms. The question is whether it will ever be practical to build a machine to accomplish this, or will it remain theory. Those in the quantum research community claim that it can be done, but there are many that view quantum computing as an insurmountable problem in it.

Public-key cryptography and related standards and techniques underlie security features of many Netscape products, including signed and encrypted email, form signing, object signing, single sign-on, and the Secure Sockets Layer (SSL) protocol. The basic concepts of public-key cryptography are as follows.

- Internet Security Issues
- Encryption and Decryption
- Digital Signatures
- Certificates and Authentication
- Managing Certificates

3.4 Internet Security Issues

All communication over the Internet uses the Transmission Control Protocol/Internet Protocol (TCP/IP). TCP/IP allows information to be sent from one computer to another through a variety of intermediate computers and separate networks before it reaches its destination.

The great flexibility of TCP/IP has led to its worldwide acceptance as the basic Internet and intranet communications protocol. At the same time, the fact that TCP/IP allows information to pass through intermediate computers makes it possible for a third party to interfere with communications in the following ways:

- **Eavesdropping.** Information remains intact, but its privacy is compromised. For example, someone could learn your credit card number, record a sensitive conversation, or intercept classified information.
- **Tampering.** Information in transit is changed or replaced and then sent on to the recipient. For example, someone could alter an order for goods or change a person's resume.
- **Impersonation.** Information passes to a person who poses as the intended recipient. Impersonation can take two forms:
    - **Spoofing.** A person can pretend to be someone else. For example, a person can pretend to have the email address jdoe@mozilla.com, or a computer can identify itself as a site called www.mozilla.com when it is not. This type of impersonation is known as spoofing.
    - **Misrepresentation.** A person or organization can misrepresent itself. For example, suppose the site www.mozilla.com pretends to be a furniture store when it is really just a site that takes credit-card payments but never sends any goods.

Normally, users of the many cooperating computers that make up the Internet or other networks don't monitor or interfere with the network traffic that continuously passes through their machines. However, many sensitive personal and business communications over the Internet require precautions that address the threats listed above. Fortunately, a set of well-established techniques and standards known as public-key cryptography make it relatively easy to take such precautions.

Public-key cryptography facilitates the following tasks:

- Encryption and decryption allow two communicating parties to disguise information they send to each other. The sender encrypts, or scrambles, information before sending it. The receiver decrypts, or unscrambles, the information after receiving it. While in transit, the encrypted information is unintelligible to an intruder.
- Tamper detection allows the recipient of information to verify that it has not been modified in transit. Any attempt to modify data or substitute a false message for a legitimate one will be detected.
- Authentication allows the recipient of information to determine its origin--that is, to confirm the sender's identity.
- No repudiation prevents the sender of information from claiming at a later date that the information was never sent.

3.5 Encryption and Decryption

Encryption is the process of transforming information so it is unintelligible to anyone but the intended recipient. Decryption is the process of transforming encrypted information so that it is intelligible again. A cryptographic algorithm, also called a cipher, is a mathematical function used for encryption or decryption. In most cases, two related functions are employed, one for encryption and the other for decryption.

With most modern cryptography, the ability to keep encrypted information secret is based not on the cryptographic algorithm, which is widely known, but on a number called a key that must be used with the algorithm to produce an encrypted result or to decrypt previously encrypted information. Decryption with the correct key is simple. Decryption without the correct key is very difficult, and in some cases impossible for all practical purposes.

The sections that follow introduce the use of keys for encryption and decryption.

- Symmetric-Key Encryption
- Public-Key Encryption
- Key Length and Encryption Strength

3.5.1 Symmetric-Key Encryption

With symmetric-key encryption, the encryption key can be calculated from the decryption key and vice versa. With most symmetric algorithms, the same key is used for both encryption and decryption, as shown in Figure 3.1.

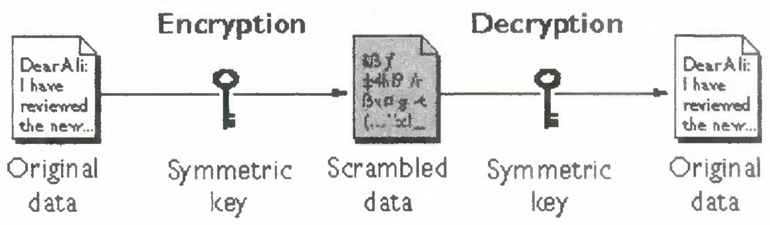


Figure 3.1: Shows Symmetric-key encryption

Implementations of symmetric-key encryption can be highly efficient, so that users do not experience any significant time delay as a result of the encryption and decryption. Symmetric-key encryption also provides a degree of authentication, since information encrypted with one symmetric key cannot be decrypted with any other symmetric key. Thus, as long as the symmetric key is kept secret by the two parties using it to encrypt communications, each party can be sure that it is communicating with the other as long as the decrypted messages continue to make sense.

Symmetric-key encryption is effective only if the symmetric key is kept secret by the two parties involved. If anyone else discovers the key, it affects both confidentiality and authentication. A person with an unauthorized symmetric key not only can decrypt messages sent with that key, but can encrypt new messages and send them as if they came from one of the two parties who were originally using the key.

Symmetric-key encryption plays an important role in the SSL protocol, which is widely used for authentication, tamper detection, and encryption over TCP/IP networks. SSL also uses techniques of public-key encryption, which is described in the next section.

3.5.2 Public-Key Encryption

The most commonly used implementations of public-key encryption are based on algorithms patented by RSA Data Security. Therefore, this section describes the RSA approach to public-key encryption.

Public-key encryption (also called asymmetric encryption) involves a pair of keys--a public key and a private key--associated with an entity that needs to authenticate its identity electronically or to sign or encrypt data. Each public key is published, and the corresponding private key is kept secret. Data encrypted with your public key can be decrypted only with your private key. Figure 3.2 shows a simplified view of the way public-key encryption works.

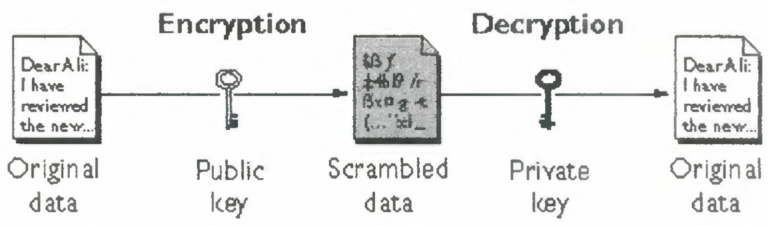


Figure 3.2: Shows Public-key encryption

The scheme shown in Figure 3.2 lets you freely distribute a public key, and only you will be able to read data encrypted using this key. In general, to send encrypted data to someone, you encrypt the data with that person's public key, and the person receiving the encrypted data decrypts it with the corresponding private key.

Compared with symmetric-key encryption, public-key encryption requires more computation and is therefore not always appropriate for large amounts of data. However, it's possible to use public-key encryption to send a symmetric key, which can then be used to encrypt additional data. This is the approach used by the SSL protocol.

As it happens, the reverse of the scheme shown in Figure 3.2 also works: data encrypted with your private key can be decrypted only with your public key. This would not be a desirable way to encrypt sensitive data, however, because it means that anyone with your public key, which is by definition published, could decrypt the data.

Nevertheless, private-key encryption is useful, because it means you can use your private key to sign data with your digital signature--an important requirement for electronic commerce and other commercial applications of cryptography. Client software such as Communicator can then use your public key to confirm that the message was signed with your private key and that it hasn't been tampered with since being signed. Digital Signatures and subsequent sections describe how this confirmation process works.

3.5.3 Key Length and Encryption Strength

In general, the strength of encryption is related to the difficulty of discovering the key, which in turn depends on both the cipher used and the length of the key. For example, the difficulty of discovering the key for the RSA cipher most commonly used for public-key encryption depends on the difficulty of factoring large numbers, a well-known mathematical problem.

Encryption strength is often described in terms of the size of the keys used to perform the encryption: in general, longer keys provide stronger encryption. Key length is measured in bits. For example, 128-bit keys for use with the RC4 symmetric-key cipher supported by SSL provide significantly better cryptographic protection than 40-bit keys for use with the same cipher. Roughly speaking, 128-bit RC4 encryption is 3×10^{26} times stronger than 40-bit RC4 encryption.

Different ciphers may require different key lengths to achieve the same level of encryption strength. The RSA cipher used for public-key encryption, for example, can use only a subset of all possible values for a key of a given length, due to the nature of the mathematical problem on which it is based. Other ciphers, such as those used for symmetric key encryption, can use all possible values for a key of a given length, rather than a subset of those values. Thus a 128-bit key for use with a symmetric-key encryption cipher would provide stronger encryption than a 128-bit key for use with the RSA public-key encryption cipher.

This difference explains why the RSA public-key encryption cipher must use a 512-bit key (or longer) to be considered cryptographically strong, whereas symmetric key ciphers can achieve approximately the same level of strength with a 64-bit key. Even this level of strength may be vulnerable to attacks in the near future.

3.6 Digital Signatures

Encryption and decryption address the problem of eavesdropping, one of the three Internet security issues mentioned at the beginning of this document. But encryption and decryption, by themselves, do not address the other two problems mentioned in Internet Security Issues: tampering and impersonation.

This section describes how public-key cryptography addresses the problem of tampering. The sections that follow describe how it addresses the problem of impersonation. Tamper detection and related authentication techniques rely on a mathematical function called a one-way hash (also called a message digest). A one-way hash is a number of fixed lengths with the following characteristics:

- The value of the hash is unique for the hashed data. Any change in the data, even deleting or altering a single character, results in a different value.
- The content of the hashed data cannot, for all practical purposes, be deduced from the hash--which is why it is called "one-way."

As mentioned in Public-Key Encryption, it's possible to use your private key for encryption and your public key for decryption. Although this is not desirable when you are encrypting sensitive information, it is a crucial part of digitally signing any data. Instead of encrypting the data itself, the signing software creates a one-way hash of the data, and then uses your private key to encrypt the hash. The encrypted hash, along with other information, such as the hashing algorithm, is known as a digital signature.
Figure 3.3 shows a simplified view of the way a digital signature can be used to validate the integrity of signed data.

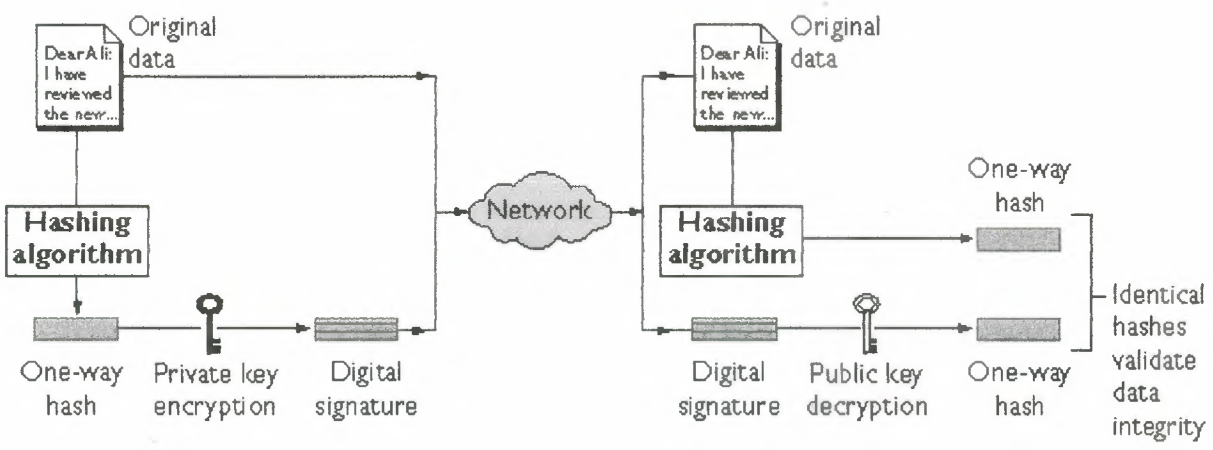


Figure 3.3: Shows Validate Data Integrity Using Digital Signature

Figure 3.3 shows two items transferred to the recipient of some signed data: the original data and the digital signature, which is basically a one-way hash (of the original data) that has been encrypted with the signer's private key. To validate the integrity of the data, the receiving software first uses the signer's public key to decrypt the hash. It then uses the same hashing algorithm that generated the original hash to generate a new one-way hash of the same data. (Information about the hashing algorithm used is sent with the digital signature, although this isn't shown in the figure 3.3.) Finally, the receiving software compares the new hash against the original hash. If the two hashes match, the data has not changed since it was signed. If they don't match, the data may have been tampered with since it was signed, or the signature may have been created with a private key that doesn't correspond to the public key presented by the signer.

If the two hashes match, the recipient can be certain that the public key used to decrypt the digital signature corresponds to the private key used to create the digital signature. Confirming the identity of the signer, however, also requires some way of confirming that the public key really belongs to a particular person or other entity. For a discussion of the way this works.

The significance of a digital signature is comparable to the significance of a handwritten signature. Once you have signed some data, it is difficult to deny doing so later--assuming that the private key has not been compromised or out of the owner's control. This quality of digital signatures provides a high degree of non-repudiation--that is, digital signatures make it difficult for the signer to deny having signed the data. In some situations, a digital signature may be as legally binding as a handwritten signature.

3.7 Certificates and Authentication

Certificates and authentication means that the right is person is logon on or it is original person who must get the data or not. There are five types we can have which are as follows:

- A Certificate Identifies Someone or Something
- Authentication Confirms an Identity
- How Certificates Are Used
- Contents of a Certificate
- How CA Certificates Are Used to Establish Trust

3.7.1 A Certificate Identifies Someone or Something

A certificate is an electronic document used to identify an individual, a server, a company, or some other entity and to associate that identity with a public key. Like a driver's license, a passport, or other commonly used personal IDs, a certificate provides generally recognized proof of a person's identity. Public-key cryptography uses certificates to address the problem of impersonation.

To get a driver's license, you typically apply to a government agency, such as the Department of Motor Vehicles, which verifies your identity, your ability to drive, your address, and other information before issuing the license. To get a student ID, you apply to a school or college, which performs different checks (such as whether you have paid your tuition) before issuing the ID. To get a library card, you may need to provide only your name and a utility bill with your address on it.

Certificates work much the same way as any of these familiar forms of identification. Certificate authorities (CAs) are entities that validate identities and issue certificates. They can be either independent third parties or organizations running their own certificate-issuing server software (such as Netscape Certificate Server). The methods used to validate an identity vary depending on the policies of a given CA--just as the methods to validate other forms of identification vary depending on who is issuing the ID and the purpose for which it will be used. In general, before issuing a certificate, the CA must use its published verification procedures for that type of certificate to ensure that an entity requesting a certificate is in fact who it claims to be.

The certificate issued by the CA binds a particular public key to the name of the entity the certificate identifies (such as the name of an employee or a server). Certificates help prevent the use of fake public keys for impersonation. Only the public key certified by the certificate will work with the corresponding private key possessed by the entity identified by the certificate.

In addition to a public key, a certificate always includes the name of the entity it identifies, an expiration date, the name of the CA that issued the certificate, a serial number, and other information. Most importantly, a certificate always includes the digital signature of the issuing CA. The CA's digital signature allows the certificate to function as a "letter of introduction" for users who know and trust the CA but don't know the entity identified by the certificate.

3.7.2 Authentication Confirms an Identity

Authentication is the process of confirming an identity. In the context of network interactions, authentication involves the confident identification of one party by another party. Authentication over networks can take many forms. Certificates are one way of supporting authentication.

Network interactions typically take place between a client, such as browser software running on a personal computer, and a server, such as the software and hardware used to host a Web site. Client authentication refers to the confident identification of a client by a server (that is, identification of the person assumed to be using the client software). Server authentication refers to the confident identification of a server by a client (that is, identification of the organization assumed to be responsible for the server at a particular network address).

Client and server authentication are not the only forms of authentication that certificates support. For example, the digital signature on an email message, combined with the certificate that identifies the sender, provide strong evidence that the person identified by that certificate did indeed send that message. Similarly, a digital signature on an HTML form, combined with a certificate that identifies the signer, can provide evidence, after the fact, that the person identified by that certificate did agree to the contents of the form. In addition to authentication, the digital signature in both cases ensures a degree of non-repudiation--that is, a digital signature makes it difficult for the signer to claim later not to have sent the email or the form.

Client authentication is an essential element of network security within most intranets or extranets. The sections that follow contrast two forms of client authentication:

- Password-Based Authentication. Almost all server software permits client authentication by means of a name and password. For example, a server might require a user to type a name and password before granting access to the server. The server maintains a list of names and passwords; if a particular name is on the list, and if the user types the correct password, the server grants access.
- Certificate-Based Authentication. Client authentication based on certificates is part of the SSL protocol. The client digitally signs a randomly generated piece of data and sends both the certificate and the signed data across the network. The server uses techniques of public-key cryptography to validate the signature and confirm the validity of the certificate.

3.7.2.1 Password-Based Authentication

Figure 3.4 shows the basic steps involved in authenticating a client by means of a name and password. Figure 3.4 assumes the following:

- The user has already decided to trust the server, either without authentication or on the basis of server authentication via SSL.
- The user has requested a resource controlled by the server.
- The server requires client authentication before permitting access to the requested resource.

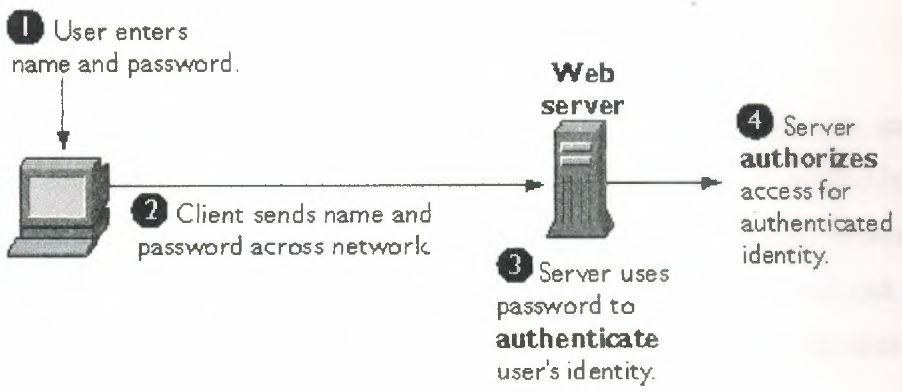

1 User enters name and password.
Web server
2 Client sends name and password across network
3 Server uses password to authenticate user's identity.
4 Server authorizes access for authenticated identity.

Figure 3.4: Shows Password Authenticate of a Client to a Server

These are the steps shown in Figure 3.4:

1. In response to an authentication request from the server, the client displays a dialog box requesting the user's name and password for that server. The user must supply a name and password separately for each new server the user wishes to use during a work session.
2. The client sends the name and password across the network, either in the clear or over an encrypted SSL connection.
3. The server looks up the name and password in its local password database and, if they match, accepts them as evidence authenticating the user's identity.
4. The server determines whether the identified user is permitted to access the requested resource, and if so allows the client to access it.

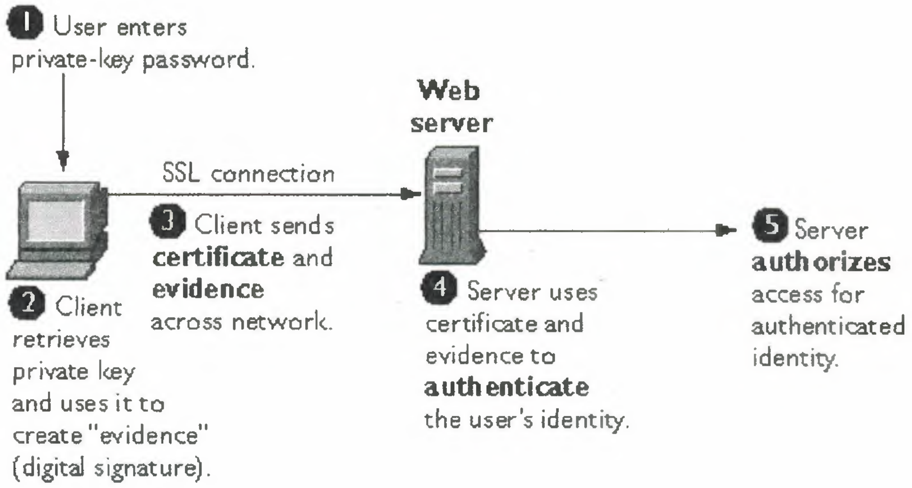
With this arrangement, the user must supply a new password for each server, and the administrator must keep track of the name and password for each user, typically on separate servers.

As shown in the next section, one of the advantages of certificate-based authentication is that it can be used to replace the first three steps in Figure 3.2 with a mechanism that allows the user to supply just one password (which is not sent across the network) and allows the administrator to control user authentication centrally.

3.7.2.2 Certificate-Based Authentication

Figure 3.5 shows how client authentication works using certificates and the SSL Protocol. To authenticate a user to a server, a client digitally signs a randomly generated piece of data and sends both the certificate and the signed data across the network. For the purposes of this discussion, the digital signature associated with some data can be thought of as evidence provided by the client to the server. The server authenticates the user's identity on the strength of this evidence.

Like Figure 3.4, Figure 3.5 assumes that the user has already decided to trust the server and has requested a resource, and that the server has requested client authentication in the process of evaluating whether to grant access to the requested resource.



1 User enters private-key password.
Web server
SSL connection
3 Client sends certificate and evidence across network.
2 Client retrieves private key and uses it to create "evidence" (digital signature).
4 Server uses certificate and evidence to authenticate the user's identity.
5 Server authorizes access for authenticated identity.

Figure 3.5: Shows Certificate Authentication of client to a server

Unlike the process shown in Figure 3.4, the process shown in Figure 3.5 requires the use of SSL. Figure 3.5 also assumes that the client has a valid certificate that can be used to identify the client to the server. Certificate-based authentication is generally considered preferable to password-based authentication because it is based on what the user has (the private key) as well as what the user knows (the password that protects the private key). However, it's important to note that these two assumptions are true only if unauthorized personnel have not gained access to the user's machine or password, the password for the client software's private key database has been set, and the software is set up to request the password at reasonably frequent intervals.

Important neither password-based authentication nor certificate-based authentication address security issues related to physical access to individual machines or passwords. Public- key cryptography can only verify that a private key used to sign some data corresponds to the public key in a certificate. It is the user's responsibility to protect a machine's physical security and to keep the private-key password secret.

These are the steps shown in Figure 3.3:

1. The client software, such as Communicator, maintains a database of the private keys that correspond to the public keys published in any certificates issued for that client. The client asks for the password to this database the first time the client needs to access it during a given session--for example, the first time the user attempts to access an SSL-enabled server that requires certificate-based client authentication. After entering this password once, the user doesn't need to enter it again for the rest of the session, even when accessing other SSL-enabled servers.
2. The client unlocks the private-key database, retrieves the private key for the user's certificate, and uses that private key to digitally sign some data that has been randomly generated for this purpose on the basis of input from both the client and the server. This data and the digital signature constitute "evidence" of the private key's validity. The digital signature can be created only with that private key and can be validated with the corresponding public key against the signed data, which is unique to the SSL session.
3. The client sends both the user's certificate and the evidence (the randomly generated piece of data that has been digitally signed) across the network.
4. The server uses the certificate and the evidence to authenticate the user's identity.
5. At this point the server may optionally perform other authentication tasks, such as checking that the certificate presented by the client is stored in the user's entry in an LDAP directory. The server then continues to evaluate whether the identified user is permitted to access the requested resource. This evaluation process can employ a variety of standard authorization mechanisms, potentially using additional information in an LDAP directory, company databases, and so on. If the result of

the evaluation is positive, the server allows the client to access the requested resource.

As you can see by comparing Figure 3.5 to Figure 3.4, certificates replace the authentication portion of the interaction between the client and the server. Instead of requiring a user to send passwords across the network throughout the day, single sign-on requires the user to enter the private-key database password just once, without sending it across the network. For the rest of the session, the client presents the user's certificate to authenticate the user to each new server it encounters. Existing authorization mechanisms based on the authenticated user identity are not affected.

3.7.3 How Certificates Are Used

- Types of Certificates
- SSL Protocol
- Signed and Encrypted Email
- Form Signing
- Single Sign-On
- Object Signing

3.7.3.1 Types of Certificates

Five kinds of certificates are commonly used with Netscape products:

- Client SSL certificates. Used to identify clients to servers via SSL (client authentication). Typically, the identity of the client is assumed to be the same as the identity of a human being, such as an employee in an enterprise. See Certificate-Based Authentication for a description of the way client SSL certificates are used for client authentication. Client SSL certificates can also be used for Form Signing and as part of a Single Sign-On solution.

  Examples: A bank gives a customer a client SSL certificate that allows the bank's servers to identify that customer and authorize access to the customer's accounts. A company might give a new employee a client SSL certificate that allows the company's servers to identify that employee and authorize access to the company's servers.

- Server SSL certificates. Used to identify servers to clients via SSL (server authentication). Server authentication may be used with or without client authentication. Server authentication is a requirement for an encrypted SSL session. See SSL Protocol.

  Example: Internet sites that engage in electronic commerce (commonly known as e-commerce) usually support certificate-based server authentication, at a minimum, to establish an encrypted SSL session and to assure customers that they are dealing with a web site identified with a particular company. The encrypted SSL session ensures that personal information sent over the network, such as credit card numbers, cannot easily be intercepted.

- S/MIME certificates. Used for signed and encrypted email. As with client SSL certificates, the identity of the client is typically assumed to be the same as the identity of a human being, such as an employee in an enterprise. A single certificate may be used as both an S/MIME certificate and an SSL certificate. See Signed and Encrypted Email. S/MIME certificates can also be used for Form Signing and as part of a Single Sign-On solution.

  Examples: A company deploys combined S/MIME and SSL certificates solely for the purpose of authenticating employee identities, thus permitting signed email and client SSL authentication but not encrypted email. Another company issues S/MIME certificates solely for the purpose of both signing and encrypting email that deals with sensitive financial or legal matters.

- Object-signing certificates. Used to identify signers of Java code, JavaScript scripts, or other signed files. See Object Signing.

Example: Software company signs software distributed over the Internet to provide users with some assurance that the software is a legitimate product of that company. Using certificates and digital signatures in this manner can also make it possible for users to identify and control the kind of access downloaded software has to their computers.

- CA certificates. Used to identify CAs. Client and server software use CA certificates to determine what other certificates can be trusted. See How CA Certificates Are Used to Establish Trust.

  Example: The CA certificates stored in Communicator determine what other certificates that copy of Communicator can authenticate. An administrator can implement some aspects of corporate security policies by controlling the CA certificates stored in each user's copy of Communicator.

The sections that follow describe how certificates are used by Netscape products.

3.7.3.2 SSL Protocol

The Secure Sockets Layer (SSL) protocol, which was originally developed by Netscape, is a set of rules governing server authentication, client authentication, and encrypted communication between servers and clients. SSL is widely used on the Internet, especially for interactions that involve exchanging confidential information such as credit card numbers.

SSL requires a server SSL certificate, at a minimum. As part of the initial "handshake" process, the server presents its certificate to the client to authenticate the server's identity. The authentication process uses Public-Key Encryption and Digital Signatures to confirm that the server is in fact the server it claims to be. Once the server has been authenticated, the client and server use techniques of Symmetric-Key Encryption, which is very fast, to encrypt all the information they exchange for the remainder of the session and to detect any tampering that may have occurred.

Servers may optionally be configured to require client authentication as well as server authentication. In this case, after server authentication is successfully completed, the client must also present its certificate to the server to authenticate the client's identity before the encrypted SSL session can be established.

3.7.3.3 Signed and Encrypted Email

Some email programs (including Messenger, which is part of Communicator) support digitally signed and encrypted email using a widely accepted protocol known as Secure Multipurpose Internet Mail Extension (S/MIME). Using S/MIME to sign or encrypt email messages requires the sender of the message to have an S/MIME certificate.

An email message that includes a digital signature provides some assurance that it was in fact sent by the person whose name appears in the message header, thus providing authentication of the sender. If the digital signature cannot be validated by the email software on the receiving end, the user will be alerted.

The digital signature is unique to the message it accompanies. If the message received differs in any way from the message that was sent--even by the addition or deletion of a comma--the digital signature cannot be validated. Therefore, signed email also provides some assurance that the email has not been tampered with. As discussed at the beginning of this document, this kind of assurance is known as non-repudiation. In other words, signed email makes it very difficult for the sender to deny having sent the message. This is important for many forms of business communication. S/MIME also makes it possible to encrypt email messages. This is also important for some business users. However, using encryption for email requires careful planning. If the recipient of encrypted email messages loses his or her private key and does not have access to a backup copy of the key, for example, the encrypted messages can never be decrypted.

3.7.3.4 Single Sign-On

Network users are frequently required to remember multiple passwords for the various services they use. For example, a user might have to type a different password to log into the network, collect email, use directory services, use the corporate calendar program, and access various servers. Multiple passwords are an ongoing headache for both users and system administrators. Users have difficulty keeping track of different passwords, tend to choose poor ones, and tend to write them down in obvious places. Administrators must keep track of a separate password database on each server and deal with potential security problems related to the fact that passwords are sent over the network routinely and frequently.

Solving this problem requires some way for a user to log in once, using a single password, and get authenticated access to all network resources that user is authorized to use--without sending any passwords over the network. This capability is known as single sign-on.

Both client SSL certificates and S/MIME certificates can play a significant role in a comprehensive single sign-on solution. For example, one form of single sign-on supported by Netscape products relies on SSL client. A user can log in once, using a single password to the local client's private-key database, and get authenticated access to all SSL-enabled servers that user is authorized to use--without sending any passwords over the network. This approach simplifies access for users, because they don't need to enter passwords for each new server. It also simplifies network management, since administrators can control access by controlling lists of certificate authorities (CAs) rather than much longer lists of users and passwords.

In addition to using certificates, a complete single-sign on solution must address the need to interoperate with enterprise systems, such as the underlying operating system, that rely on passwords or other forms of authentication.

3.7.3.5 Form Signing

Many kinds of e-commerce require the ability to provide persistent proof that someone has authorized a transaction. Although SSL provides transient client authentication for the duration of an SSL connection, it does not provide persistent authentication for transactions that may occur during that connection. S/MIME provides persistent authentication for email, but e-commerce often involves filling in a form on a web page rather than sending an email.

The Netscape technology known as form signing addresses the need for persistent authentication of financial transactions. Form signing allows a user to associate a digital signature with web-based data generated as the result of a transaction, such as a purchase order or other financial document. The private key associated with either a client SSL certificate or an S/MIME certificate may be used for this purpose.

When a user clicks the Submit button on a web-based form that supports form signing, a dialog box appears that displays the exact text to be signed. The form designer can either specify the certificate that should be used or allow the user to select a certificate from among the client SSL and S/MIME certificates that are installed in Communicator. When the user clicks OK, the text is signed, and both the text and the digital signature are submitted to the server. The server can then use a Netscape utility called the Signature Verification Tool to validate the digital signature.

3.7.3.6 Object Signing

Communicator and other Netscape products support a set of tools and technologies called object signing. Object signing uses standard techniques of public-key cryptography to let users get reliable information about code they download in much the same way they can get reliable information about shrink-wrapped software.
Most importantly, object signing helps users and network administrators implement decisions about software distributed over intranets or the Internet--for example, whether to allow Java applets signed by a given entity to use specific computer capabilities on specific users' machines.

The "objects" signed with object signing technology can be applets or other Java code, JavaScript scripts, plug-ins, or any kind of file. The "signature" is a digital signature. Signed objects and their signatures are typically stored in a special file called a JAR file.

Software developers and others who wish to sign files using object-signing technology must first obtain an object-signing certificate.

3.7.4 Contents of a Certificate

The contents of certificates supported by Netscape and many other software companies are organized according to the X.509 v3 certificate specification. Users don't usually need to be concerned about the exact contents of a certificate. However, system administrators working with certificates may need some familiarity with the information provided here.

3.7.4.1 Distinguished Names

An X.509 v3 certificate binds a distinguished name (DN) to a public key. A DN is a series of name-value pairs, such as uid=doe, that uniquely identify an entity--that is, the certificate subject.

For example, this might be a typical DN for an employee of Netscape Communications Corporation:

uid=doe,e=doe@netscape.com,cn=John Doe, o=Netscape Communications Corp., c=US

The abbreviations before each equal sign in this example have these meanings:

- uid: user ID
- e: email address
- cn: the user's common name
- o: organization
- c: country

DNs may include a variety of other name-value pairs. They are used to identify both certificate subjects and entries in directories that support the Lightweight Directory Access Protocol (LDAP).

The rules governing the construction of DNs can be quite complex and are beyond the scope of this document.

3.7.5 How CA Certificates Are Used to Establish Trust

Certificate authorities (CAs) are entities that validate identities and issue certificates. They can be either independent third parties or organizations running their own certificate-issuing server software (such as the Netscape Certificate Server). Any client or server software that supports certificates maintains a collection of trusted CA certificates. These CA certificates determine which other certificates the software can validate--in other words, which issuers of certificates the software can trust. In the simplest case, the software can validate only certificates issued by one of the CAs for which it has a certificate. It's also possible for a trusted CA certificate to be part of a chain of CA certificates, each issued by the CA above it in a certificate hierarchy.

The sections that follow explains how certificate hierarchies and certificate chains determine what certificates software can trust.

- CA Hierarchies
- Certificate Chains
- Verifying a Certificate Chain

3.7.5.1 CA Hierarchies

In large organizations, it may be appropriate to delegate the responsibility for issuing certificates to several different certificate authorities. For example, the number of certificates required may be too large for a single CA to maintain; different organizational units may have different policy requirements; or it may be important for a CA to be physically located in the same geographic area as the people to whom it is issuing certificates.

It's possible to delegate certificate-issuing responsibilities to subordinate CAs. The X.509 standard includes a model for setting up a hierarchy of CAs like that shown in Figure 3.6.

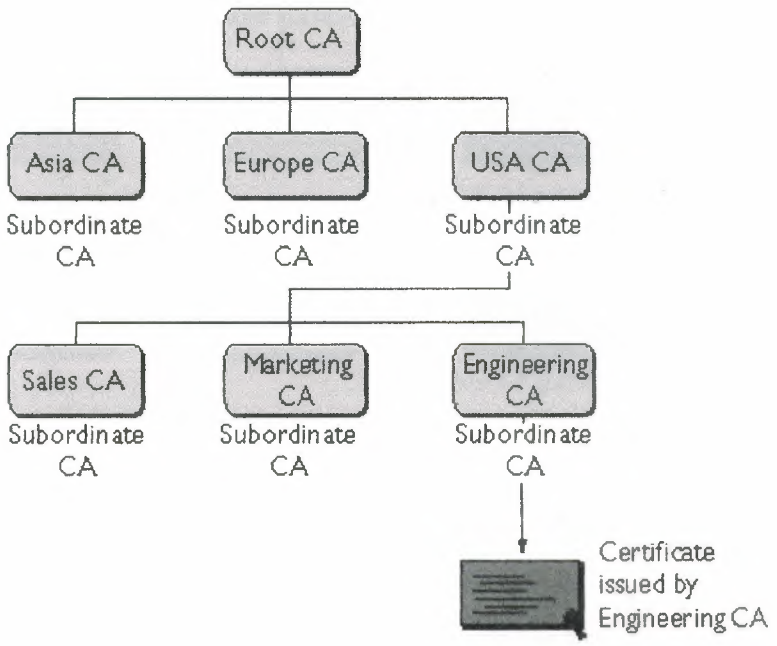


Figure 3.6: Shows an Example of a Hierarchy of Certificate Authorities

In this model, the root CA is at the top of the hierarchy. The root CA's certificate is a self-signed certificate: that is, the certificate is digitally signed by the same entity--the root CA--which the certificate identifies. The CAs that are directly subordinate to the root CA have CA certificates signed by the root CA. CAs under the subordinate CAs in the hierarchy have their CA certificates signed by the higher-level subordinate CAs.

Organizations have a great deal of flexibility in terms of the way they set up their CA hierarchies. Figure 3.6 shows just one example; many other arrangements are possible.

3.7.5.2 Certificate Chains

CA hierarchies are reflected in certificate chains. A certificate chain is series of certificates issued by successive CAs. Figure 3.7 shows a certificate chain leading from a certificate that identifies some entity through two subordinate CA certificates to the CA certificate for the root CA (based on the CA hierarchy shown in Figure 3.6).

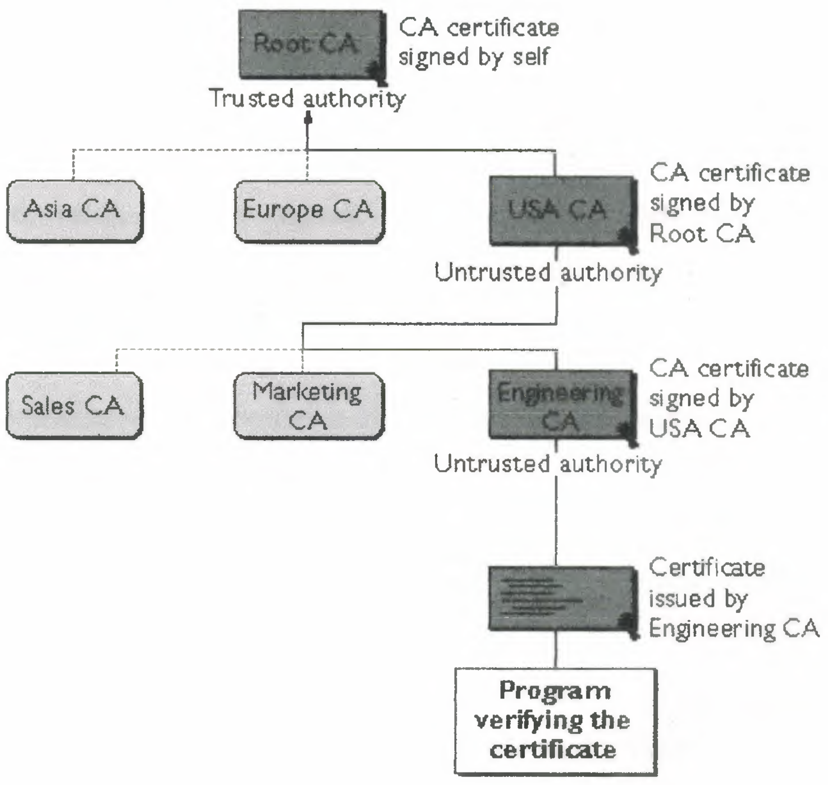

Root CA
CA certificate signed by self
Trusted authority
Asia CA
Europe CA
USA CA
CA certificate signed by Root CA
Untrusted authority
Sales CA
Marketing CA
Engineering CA
CA certificate signed by USA CA
Untrusted authority
Certificate issued by Engineering CA
Program verifying the certificate

Figure 3.7: Shows an Example of a certificate chain

A certificate chain traces a path of certificates from a branch in the hierarchy to the root of the hierarchy. In a certificate chain, the following occur:

- Each certificate is followed by the certificate of its issuer.
- Each certificate contains the name (DN) of that certificate's issuer, which is the same as the subject name of the next certificate in the chain.

  In Figure 3.7, the Engineering CA certificate contains the DN of the CA (that is, USA CA), that issued that certificate. USA CA's DN is also the subject name of the next certificate in the chain.

- Each certificate is signed with the private key of its issuer. The signature can be verified with the public key in the issuer's certificate, which is the next certificate in the chain.

  In Figure 3.7, the public key in the certificate for the USA CA can be used to verify the USA CA's digital signature on the certificate for the Engineering CA.

3.7.5.3 Verifying a Certificate Chain

Certificate chain verification is the process of making sure a given certificate chain is well-formed, valid, properly signed, and trustworthy. Netscape software uses the following procedure for forming and verifying a certificate chain, starting with the certificate being presented for authentication:

1. The certificate validity period is checked against the current time provided by the verifier's system clock.
2. The issuer's certificate is located. The source can be either the verifier's local certificate database (on that client or server) or the certificate chain provided by the subject (for example, over an SSL connection).
3. The certificate signature is verified using the public key in the issuer's certificate.
4. If the issuer's certificate is trusted by the verifier in the verifier's certificate database, verification stops successfully here. Otherwise, the issuer's certificate is checked to make sure it contains the appropriate subordinate CA indication in the Netscape

certificate type extension, and chain verification returns to step 1 to start again, but with this new certificate. Figure 3.8 presents an example of this process.

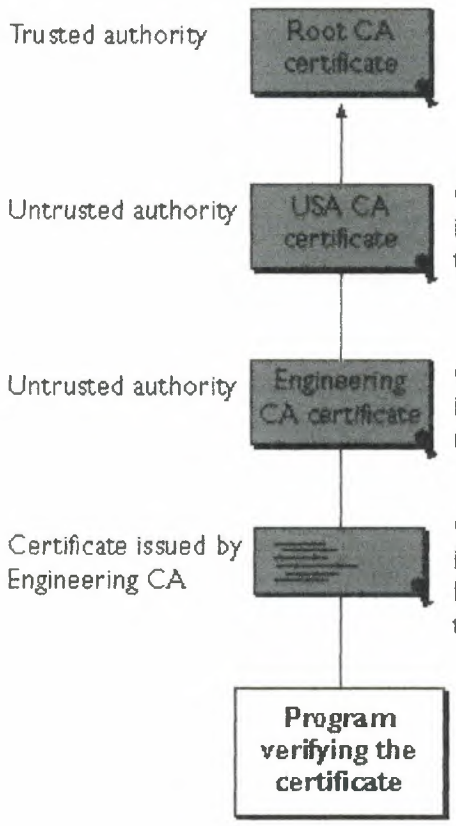

Trusted authority
Root CA certificate
Untrusted authority
USA CA certificate
Check validity period and verify that this is signed by Root CA. Since Root CA is trusted, verification stops here.
Untrusted authority
Engineering CA certificate
Check validity period and verify that this is signed by USA CA. Since USA CA is not trusted, check the next certificate.
Certificate issued by Engineering CA
Check validity period and verify that this is signed by Engineering CA. Since Engineering CA is not trusted, check the next certificate.
Program verifying the certificate

Figure 3.8: Verifying a Certificate Chain all the way to the Root CA

Figure 3.8 shows what happens when only Root CA is included in the verifier's local database. If a certificate for one of the intermediate CAs shown in Figure 3.8, such as Engineering CA, is found in the verifier's local database, verification stops with that certificate, as shown in Figure 3.9.

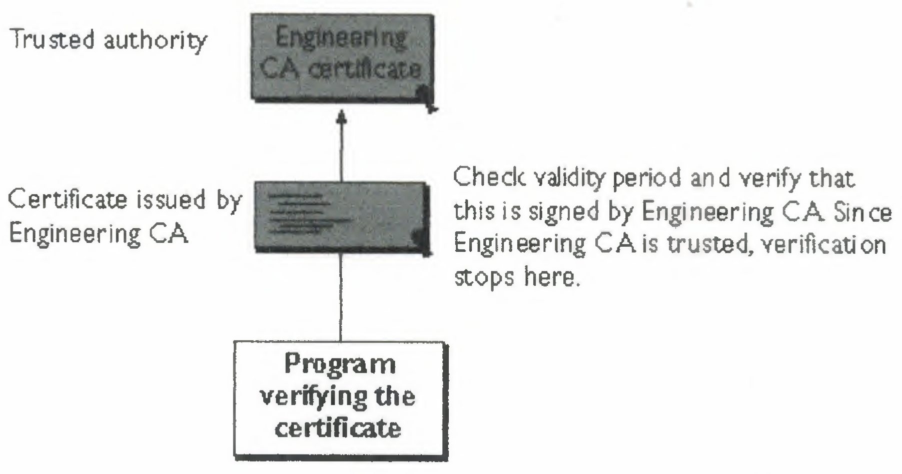

Trusted authority
Engineering CA certificate
Certificate issued by Engineering CA
Check validity period and verify that this is signed by Engineering CA. Since Engineering CA is trusted, verification stops here.
Program verifying the certificate

Figure 3.9: Verifying a Certificate Chain to an Intermediate CA

Expired validity dates, an invalid signature, or the absence of a certificate for the issuing CA at any point in the certificate chain causes authentication to fail. For example, Figure 3.10 shows how verification fails if neither the Root CA certificate nor any of the intermediate CA certificates are included in the verifier's local database.

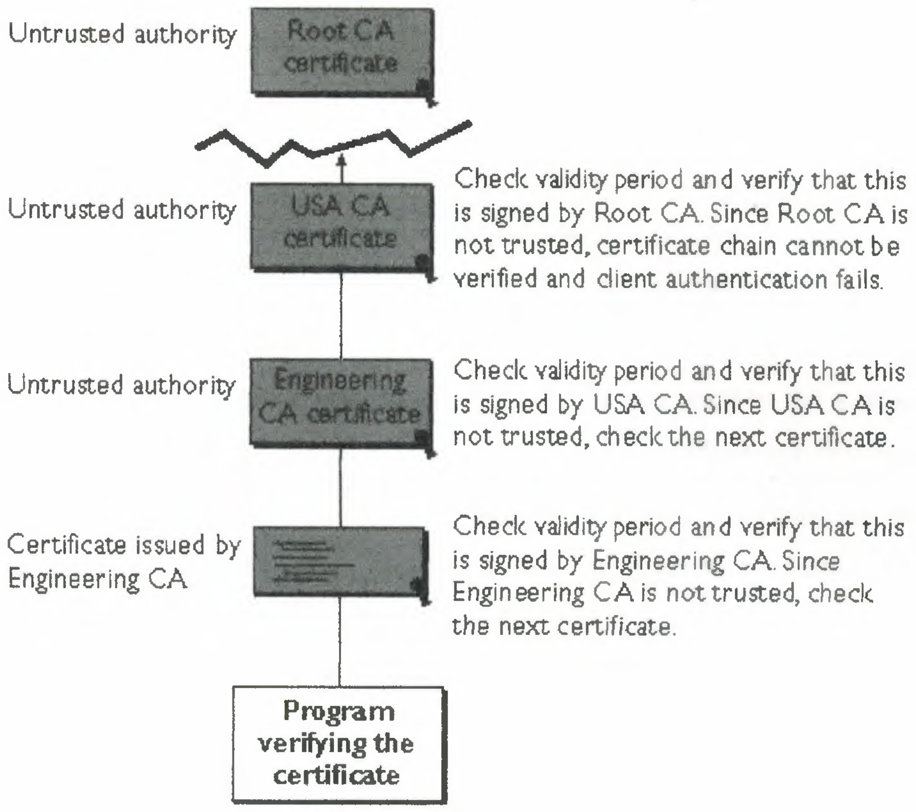

Untrusted authority
Root CA certificate
Untrusted authority
USA CA certificate
Check validity period and verify that this is signed by Root CA. Since Root CA is not trusted, certificate chain cannot be verified and client authentication fails.
Untrusted authority
Engineering CA certificate
Check validity period and verify that this is signed by USA CA. Since USA CA is not trusted, check the next certificate.
Certificate issued by Engineering CA
Check validity period and verify that this is signed by Engineering CA. Since Engineering CA is not trusted, check the next certificate.
Program verifying the certificate

Figure 3.10: Shows a Certificate Chain that can't be verified

3.8 Managing Certificates

The set of standards and services that facilitate the use of public-key cryptography and X.509 v3 certificates in a networked environment is called the public key infrastructure (PKI). PKI management is complex topic beyond the scope of this document. The sections that follow introduce some of the specific certificate management issues addressed by Netscape products.

- Issuing Certificates
- Certificates and the LDAP Directory
- Key Management
- Renewing and Revoking Certificates
- Registration Authorities

3.8.1 Issuing Certificates

The process for issuing a certificate depends on the certificate authority that issues it and the purpose for which it will be used. The process for issuing non-digital forms of identification varies in similar ways. For example, if you want to get a generic ID card (not a driver's license) from the Department of Motor Vehicles in California, the requirements are straightforward: you need to present some evidence of your identity, such as a utility bill with your address on it and a student identity card. If you want to get a regular driving license, you also need to take a test--a driving test when you first get the license, and a written test when you renew it. If you want to get a commercial license for an eighteen-wheeler, the requirements are much more stringent. If you live in some other state or country, the requirements for various kinds of licenses will differ.

Similarly, different CAs has different procedures for issuing different kinds of certificates. In some cases the only requirement may be your email address. In other cases, your UNIX or NT login and password may be sufficient. At the other end of the scale, for certificates that identify people who can authorize large expenditures or make other sensitive decisions, the issuing process may require notarized documents, a background check, and a personal interview.

Depending on an organization's policies, the process of issuing certificates can range from being completely transparent for the user to requiring significant user participation and complex procedures. In general, processes for issuing certificates should be highly flexible, so organizations can tailor them to their changing needs.
The Netscape Certificate Server, part of the Mission Control family of products, allows an organization to set up its own certificate authority and issue certificates.
Issuing certificates is one of several management's tasks that can be handled by separate Registration Authorities.

3.8.2 Certificates and the LDAP Directory

The Lightweight Directory Access Protocol (LDAP) for accessing directory services supports great flexibility in the management of certificates within an organization. System administrators can store much of the information required to manage certificates in an LDAP-compliant directory. For example, a CA can use information in a directory to repopulate a certificate with a new employee's legal name and other information. The CA can leverage directory information in other ways to issue certificates one at a time or in bulk, using a range of different identification techniques depending on the security policies of a given organization. Other routine management tasks, such as Key Management and Renewing and Revoking Certificates, can be partially or fully automated with the aid of the directory.

Information stored in the directory can also be used with certificates to control access to various network resources by different users or groups. Issuing certificates and other certificate management tasks can thus be an integral part of user and group management.

In general, high-performance directory services are an essential ingredient of any certificate management strategy. The Netscape Directory Server, part of the Mission Control family of products, is fully integrated with the Netscape Certificate Server to provide a comprehensive certificate management solution.

3.8.3 Key Management

Before a certificate can be issued, the public key it contains and the corresponding private key must be generated. Sometimes it may be useful to issue a single person one certificate and key pair for signing operations, and another certificate and key pair for encryption operations. Separate signing and encryption certificates make it possible to keep the private signing key on the local machine only, thus providing maximum non-repudiation, and to back up the private encryption key in some central location where it can be retrieved in case the user loses the original key or leaves the company.

Keys can be generated by client software or generated centrally by the CA and distributed to users via an LDAP directory. There are trade-offs involved in choosing between local and centralized key generation. For example, local key generation provides maximum non-repudiation, but may involve more participation by the user in the issuing process. Flexible key management capabilities are essential for most organizations.

Key recovery, or the ability to retrieve backups of encryption keys under carefully defined conditions, can be a crucial part of certificate management (depending on how an organization uses certificates). Key recovery schemes usually involve an m of n mechanism: for example, m of n managers within an organization might have to agree, and each contribute a special code or key of their own, before a particular person's encryption key can be recovered. This kind of mechanism ensures that several authorized personnel must agree before an encryption key can be recovered.

3.8.4 Renewing and Revoking Certificates

Like a driver's license, a certificate specifies a period of time during which it is valid. Attempts to use a certificate for authentication before or after its validity period will fail. Therefore, mechanisms for managing certificate renewal are essential for any certificate management strategy. For example, an administrator may wish to be notified automatically when a certificate is about to expire, so that an appropriate renewal process can be completed in plenty of time without causing the certificate's subject any inconvenience. The renewal process may involve reusing the same public-private key pair or issuing a new one.

A driver's license can be suspended even if it has not expired--for example, as punishment for a serious driving offense. Similarly, it's sometimes necessary to revoke a certificate before it has expired--for example, if an employee leaves a company or moves to a new job within the company.

Certificate revocation can be handled in several different ways. For some organizations, it may be sufficient to set up servers so that the authentication process includes checking the directory for the presence of the certificate being presented. When an administrator revokes a certificate, the certificate can be automatically removed from the directory, and subsequent authentication attempts with that certificate will fail even though the certificate remains valid in every other respect. Another approach involves publishing a certificate revocation list (CRL)--that is, a list of revoked certificates--to the directory at regular intervals and checking the list as part of the authentication process. For some organizations, it may be preferable to check directly with the issuing CA each time a certificate is presented for authentication. This procedure is sometimes

3.8.5 Registration Authorities called real-time status checking.

Interactions between entities identified by certificates (sometimes called end entities) and CAs are an essential part of certificate management. These interactions include operations such as registration for certification, certificate retrieval, certificate renewal, certificate revocation, and key backup and recovery. In general, a CA must be able to authenticate the identities of end entities before responding to the requests. In addition, some requests need to be approved by authorized administrators or managers before being services.

As previously discussed, the means used by different CAs to verify an identity before issuing a certificate can vary widely, depending on the organization and the purpose for which the certificate will be used. To provide maximum operational flexibility, interactions with end entities can be separated from the other functions of a CA and handled by a separate service called a Registration Authority (RA).

An RA acts as a front end to a CA by receiving end entity requests, authenticating them, and forwarding them to the CA. After receiving a response from the CA, the RA notifies the end entity of the results. RAs can be helpful in scaling an PKI across different departments, geographical areas, or other operational units with varying policies and authentication requirements.

3.9 Symmetric Encryption Keys:

In order to see how encryption might work, let us consider a very simple example as follows:

First generate the key for example a large "pseudo random" number say

7 3 9 2 6 5 1 0 7 0 3 8 2 4 1 7 4 9

Given a message

H I H O W A R E Y O U

we can assign numbers to the letters according to their position in the alphabet A=1 B=2 C=3 and so forth.

And we can encode our message by using the random number to "shift" the letter to another by adding the random amount to the position number of the letter thus:

Table 3.1: Shows Coding of Message using Random Numbers

H	I		H	O	W		A	R	E		Y	O	U
8	9	0	8	15	23	0	1	18	5	0	25	15	21
7	3	9	2	6	5	1	0	7	0	3	8	2	4
15	12	9	10	21	1	1	1	25	5	3	5	17	25
O	L	I	J	U	A	A	A	Y	E	C	E	Q	Y

As the result.

The key can be used to encrypt the message and also to decrypt it merely by subtraction rather than addition. Thus we call this key a symmetric key.

3.9.1 Diffie and Hellman's Contrtibution:

The problem with symmetric keys is that because they can be used both to encrypt and to decrypt, they must be kept very secret. Before any messages are sent, the sender and the receiver must communicate the key very secretly. If the key is found by anyone, they can use it to snoop on the messages. But this limitation is a severe one. If I want to send sensitive information to someone I've never met, perhaps my credit card number to purchase an item, must I first meet with him to set up a secure key? Clearly this is not ideal.

Diffie and Hellman solved this problem by devising a coding scheme called public key cryptography. Actually there are two keys, one public the other private. The public key is used for encoding messages and the private one for decrypting them. It's like a strong box which uses one key to lock up the information and another key to open it.

If I wish to use such a system, I can generate my two keys and give everyone my public key for them to use to encrypt messages they wish to send to me. Only I can decrypt them with my private key. Any one, who wishes to receive encoded messages from me, can do likewise. That is they can generate two keys and send me their public key for encoding messages to them.

3.9.2 The Problem:

Up until now, the government has had a monopoly on encryption, and decryption of messages. They have put a tremendous investment of time, money, research, and special purpose coding machines into becoming the world's experts in making and breaking codes. But the encryption schemes of Diffie and Hellman (D-H), and now also Rivest, Sharmir, Adleman (RSA) have, with the use of personal computers, become so powerful that the NSA cannot break them. And they don't like that, because they can no longer carryout their mission to the government. Both the NSA (mostly dealing with international matters) and the FBI (which deals mostly with internal crimes) want to have the encoding schemes broken in some way. They want the user's private key placed where they can get at it if necessary. If necessary, they say. According to their plan these private keys would be held in escrow generally for ever, but if the agencies found it necessary, under a court order they could obtain permission to retrieve the key. Of course, what is necessary is probably a matter of debate. We'll discuss this more in the future.

3.9.3 Trap Door Functions:

If we think of encrypting a message as applying a function to the message which results in the encrypted message, then decrypting the coded message is the inverse function. In our example we used addition to encrypt the message and the inverse subtraction was used to decrypt the message. Many functions like addition have any easy inverse. Trap door functions, do not, they are functions which can easily be carried out in one direction, say to encrypt a message, but cannot easily be inverted to decrypt the message with out some special knowledge (the private key). The notion of a trap door is that the door can easily be opened in one direction letting the passer through, but without special knowledge cannot be opened from the other side thus trapping him.

In the terminology we developed last time. One function is computationally easy (requiring polynomial time dependence on N), the inverse is computationally hard (requiring exponential time dependence on N).

Such an example is the function used in the RSA encryption algorithm. Consider a number N which is a product of two large primes' p & q. N = p * q. (A prime number has no factors which divide it evenly except 1 and the number itself.) Examples are: 2, 3, 5, 7, 11, 13, 17... Etc. Determining the factors of large numbers is a computationally hard problem as we can see from the notes of last time. Thus we have a trap door function. As example it is relatively hard to determine that 2,229,013 has the factors 1499 and 1487, which themselves are both prime numbers. But verifying that 1499 and 1487 are the prime factors of 2,229,013 only requires their multiplication.

4. NETWORK SECURITY

4.1 Overview

A basic understanding of computer networks is requisite in order to understand the principles of network security. In this section, we'll cover some of the foundations of computer networking, then move on to an overview of some popular networks. Following that, we'll take a more in-depth look at TCP/IP, the network protocol suite that is used to run the Internet and many intranets. Once we've covered this, we'll go back and discuss some of the threats that managers and administrators of computer networks need to confront, and then some tools that can be used to reduce the exposure to the risks of network computing.

4.2 What is a Network?

A set of interlinking lines resembling a net, a network of roads || an interconnected system, a network of alliances." This definition suits our purpose well: a computer network is simply a system of interconnected computers. How they're connected is irrelevant, and as we'll soon see, there are a number of ways to do this.

4.3 The ISO/OSI Reference Model

The International Standards Organization (ISO) Open Systems Interconnect (OSI) Reference Model defines seven layers of communications types, and the interfaces among them. See Figure 4.1. Each layer depends on the services provided by the layer below it, all the way down to the physical network hardware, such as the computer's network interface card, and the wires that connect the cards together.

An easy way to look at this is to compare this model with something we use daily: the telephone. In order for you and me to talk when we're out of earshot, we need a device like a telephone. (In the ISO/OSI model, this is at the application layer.) The telephones, of course, are useless unless they have the ability to translate the sound into electronic pulses that can be transferred over wire and back again. (These functions are provided in layers below the application layer.) Finally, we get down to the physical connection: both must be plugged into an outlet that is connected to a switch that's part of the telephone system's network of switches.

If I place a call to you, I pick up the receiver, and dial your number. This number specifies which central office to which to send my request, and then which phone from that central office to ring. Once you answer the phone, we begin talking, and our session has begun. Conceptually, computer networks function exactly the same way.

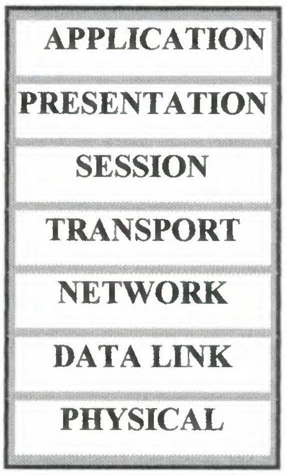


Figure 4.1: The ISO/OSI Reference Model

4.4 Some Popular Networks

Over the last 25 years or so, a number of networks and network protocols have been defined and used. We're going to look at two of these networks, both of which are "public" networks. Anyone can connect to either of these networks, or they can use types of networks to connect their own hosts (computers) together, without connecting to the public networks. Each type takes a very different approach to providing network services.

4.4.1 UUCP

UUCP (Unix-to-Unix CoPy) was originally developed to connect UNIX (surprise!) hosts together. UUCP has since been ported to much different architecture, including PCs, Macs, Amigas, Apple IIs, VMS hosts, everything else you can name, and even some things you can't. Additionally, a number of systems have been developed around the same principles as UUCP.

4.4.1.1 Batch-Oriented Processing

UUCP and similar systems are batch-oriented systems: everything that they have to do is added to a queue, and then at some specified time, everything in the queue is processed.

4.4.1.2 Implementation Environment

UUCP networks are commonly built using dial-up (modem) connections. This doesn't have to be the case though: UUCP can be used over any sort of connection between two computers, including an Internet connection.

Building a UUCP network is a simple matter of configuring two hosts to recognize each other, and know how to get in touch with each other. Adding on to the network is simple; if hosts called A and B have a UUCP network between them, and C would like to join the network, then it must be configured to talk to A and/or B. Naturally, anything that C talks to must be made aware of C's existence before any connections will work. Now, to connect D to the network, a connection must be established with at least one of the hosts on the network, and so on. Figure 4.2 shows a sample UUCP network.

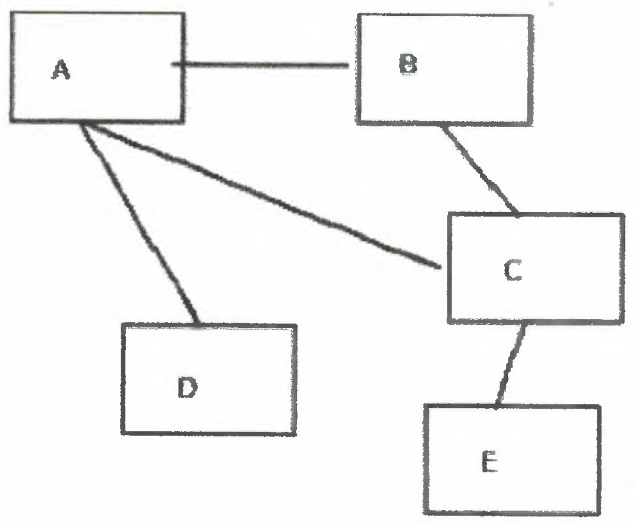


Figure 4.2: A Sample UUCP Network

In a UUCP network, users are identified in the format host!userid. The ``!" character (pronounced ``bang" in networking circles) is used to separate hosts and users. A bangpath is a string of host(s) and a userid like A!cmcurtin or C!B!A!cmcurtin. If I am a user on host A and you are a user on host E, I might be known as A!cmcurtin and you as E!you. Because there is no direct link between your host (E) and mine (A), in order for us to communicate, we need to do so through a host (or hosts!) that has connectivity to both E and A. In our sample network, C has the connectivity we need. So, to send me a file, or piece of email, you would address it to C!A!cmcurtin. Or, if you feel like taking the long way around, you can address me as C!B!A!cmcurtin. The "public" UUCP network is simply a huge worldwide network of hosts connected to each other.

4.4.1.3 Popularity.

The public UUCP network has been shrinking in size over the years, with the rise of the availability of inexpensive Internet connections. Additionally, since UUCP connections are typically made hourly, daily, or weekly, there is a fair bit of delay in getting data from one user on a UUCP network to a user on the other end of the network. UUCP isn't very flexible, as it's used for simply copying files (which can be netnews, email, documents, etc.) Interactive protocols (that make applications such as the World Wide Web possible) have become much more the norm, and are preferred in most cases.

However, there are still many people whose needs for email and netnews are served quite well by UUCP, and its integration into the Internet has greatly reduced the amount of cumbersome addressing that had to be accomplished in times past.

4.4.1.4 Security.

UUCP, like any other application, has security tradeoffs. Some strong points for its security is that it is fairly limited in what it can do, and it's therefore more difficult to trick into doing something it shouldn't; it's been around a long time, and most its bugs have been discovered, analyzed, and fixed; and because UUCP networks are made up of occasional connections to other hosts, it isn't possible for someone on host E to directly make contact with host B, and take advantage of that connection to do something naughty.

On the other hand, UUCP typically works by having a system-wide UUCP user account and password. Any system that has a UUCP connection with another must know the appropriate password for the uucp or nuucp account. Identifying a host beyond that point has traditionally been little more than a matter of trusting that the host is who it claims to be, and that a connection is allowed at that time. More recently, there has been an additional layer of authentication, whereby both hosts must have the same sequence number that is a number that is incremented each time a connection is made.

Hence, if I run host B, I know the uucp password on host A. If, though, I want to impersonate host C, I'll need to connect, identify myself as C, hope that I've done so at a time that A will allow it, and try to guess the correct sequence number for the session. While this might not be a trivial attack, it isn't considered very secure.

4.4.2 The Internet

Internet: This is a word that I've heard way too often in the last few years. Movies, books, newspapers, magazines, television programs, and practically every other sort of media imaginable has dealt with the Internet recently.

4.4.2.1 What is the Internet?

The Internet is the world's largest network of networks. When you want to access the resources offered by the Internet, you don't really connect to the Internet; you connect to a network that is eventually connected to the Internet backbone, a network of extremely fast (and incredibly overloaded!) network components. This is an important point: the Internet is a network of networks -- not a network of hosts.

A simple network can be constructed using the same protocols and such that the Internet uses without actually connecting it to anything else. Such a basic network is shown in Figure 4.3.

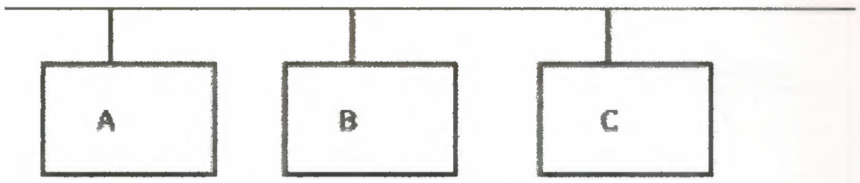


Figure 4.3: A Simple Local Area Network

I might be allowed to put one of my hosts on one of my employer's networks. We have a number of networks, which are all connected together on a backbone that is a network of our networks. Our backbone is then connected to other networks, one of which is to an Internet Service Provider (ISP) whose backbone is connected to other networks, one of which is the Internet backbone.

If you have a connection ``to the Internet" through a local ISP, you are actually connecting your computer to one of their networks, which is connected to another, and so on. To use a service from my host, such as a web server, you would tell your web browser to connect to my host. Underlying services and protocols would send packets (small datagrams) with your query to your ISP's network, and then a network they're connected to, and so on, until it found a path to my employer's backbone, and to the exact network my host is on. My host would then respond appropriately, and the same would happen in reverse: packets would traverse all of the connections until they found their way back to your computer, and you were looking at my web page.

In Figure 4.4, the network shown in Figure 4.3 is designated ``LAN 1" and shown in the bottom-right of the picture. This shows how the hosts on that network are provided connectivity to other hosts on the same LAN, within the same company, outside of the company, but in the same ISP cloud , and then from another ISP somewhere on the Internet.

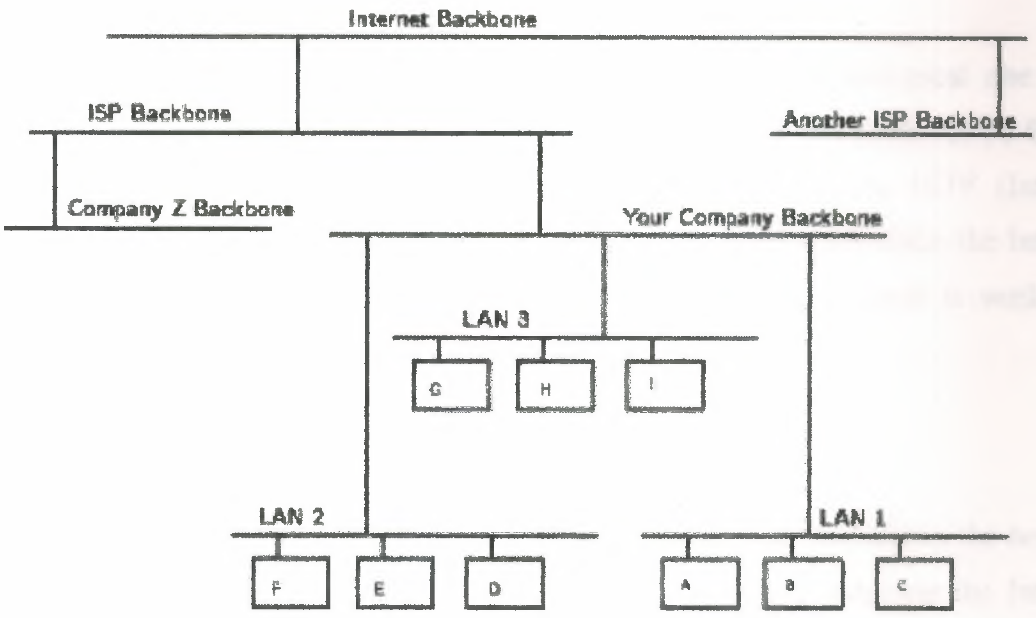


Figure 4.4: A Wider View of Internet-connected Networks

The Internet is made up of a wide variety of hosts, from supercomputers to personal computers, including every imaginable type of hardware and software. How do all of these computers understand each other and work together?

4.5 TCP/IP: The Language of the Internet

TCP/IP (Transport Control Protocol/Internet Protocol) is the ``language" of the Internet. Anything that can learn to ``speak TCP/IP" can play on the Internet. This is functionality that occurs at the Network (IP) and Transport (TCP) layers in the ISO/OSI Reference Model. Consequently, a host that has TCP/IP functionality (such as UNIX, OS/2, MacOS, or Windows NT) can easily support applications (such as Netscape's Navigator) that use the network.

4.5.1 Open Design

One of the most important features of TCP/IP isn't a technological one: The protocol is an ``open" protocol and anyone who wishes to implement it may do so freely. Engineers and scientists from all over the world participate in the IETF (Internet Engineering Task Force) working groups that design the protocols that make the Internet work. Their time is typically donated by their companies, and the result is work that benefits everyone.

4.5.2 IP

As noted, IP is a ``network layer" protocol. This is the layer that allows the hosts to actually ``talk" to each other. Such things as carrying datagrams, mapping the Internet address (such as 10.2.3.4) to a physical network address (such as 08:00:69:0a:ca:8f), and routing, which takes care of making sure that all of the devices that have Internet connectivity can find the way to each other.

4.5.2.1 Understanding IP

IP has a number of very important features which make it an extremely robust and flexible protocol. For our purposes, though, we're going to focus on the security of IP, or more specifically, the lack thereof.

4.5.2.2 Attacks against IP

A number of attacks against IP are possible. Typically, these exploit the fact that IP does not perform a robust mechanism for authentication, which is proving that a packet came from where it claims it did. A packet simply claims to originate from a given address, and there isn't a way to be sure that the host that sent the packet is telling the truth. This isn't necessarily a weakness, per se, but it is an important point, because it means that the facility of host authentication has to be provided at a higher layer on the ISO/OSI Reference Model. Today, applications that require strong host authentication (such as cryptographic applications) do this at the application layer.

a) IP Spoofing--This is where one host claims to have the IP address of another. Since many systems (such as router access control lists) define which packets may and which packets may not pass based on the sender's IP address, this is a useful technique to an attacker: he can send packets to a host, perhaps causing it to take some sort of action. Additionally, some applications allow login based on the IP address of the person making the request. These are both good examples how trusting un-trustable layers can provide security that is -- at best -- weak.

b) Session Hijacking--This is a relatively sophisticated attack, first described by Steve. This is very dangerous, however, because there are now toolkits available in the underground community that allow otherwise unskilled bad-guy-wannabes to perpetrate this attack. IP Session Hijacking is an attack whereby a user's session is taken over, being in the control of the attacker. If the user was in the middle of email, the attacker is looking at the email, and then can execute any commands he wishes as the attacked user. The attacked user simply sees his session dropped, and may simply login again, perhaps not even noticing that the attacker is still logged in and doing things. For the description of the attack, let's return to our large network of networks in Figure 4.4. In this attack, a user on host A is carrying on a session with host G. Perhaps this is a telnet session, where the user is reading his email, or using a UNIX shell account from home. Somewhere in the network between A and B sits host H which is run by a naughty person. The naughty person on host H watches the traffic between A and G, and runs a tool which starts to impersonate A to G, and at the same time tells A to shut up, perhaps trying to convince it that G is no longer on the net (which might happen in the event of a crash, or major network outage). After a few seconds of this, if the attack is successful, naughty person has ``hijacked" the session of our user. Anything that the user can do legitimately can now be done by the attacker, illegitimately. As far as G knows, nothing has happened. This can be solved by replacing standard telnet-type applications with encrypted versions of the same thing. In this case, the attacker can still take over the session, but he'll see only ``gibberish" because the session is encrypted. The attacker will not have the needed cryptographic key(s) to decrypt the data stream from G, and will, therefore, be unable to do anything with the session.

4.5.3 TCP

TCP is a transport-layer protocol. It needs to sit on top of a network-layer protocol, and was designed to ride atop IP. (Just as IP was designed to carry, among other things, TCP packets.) Because TCP and IP were designed together and wherever you have one, you typically have the other, the entire suite of Internet protocols are known collectively as ``TCP/IP." TCP itself has a number of important features that we'll cover briefly.

4.5.3.1 Guaranteed Packet Delivery

Probably the most important is guaranteed packet delivery. Host A sending packets to host B expects to get acknowledgments back for each packet. If B does not send an acknowledgment within a specified amount of time, A will resend the packet.

Applications on host B will expect a data stream from a TCP session to be complete, and in order. As noted, if a packet is missing, it will be resent by A, and if packets arrive out of order, B will arrange them in proper order before passing the data to the requesting application.

This is suited well toward a number of applications, such as a telnet session. A user wants to be sure every keystroke is received by the remote host, and that it gets every packet sent back, even if this means occasional slight delays in responsiveness while a lost packet is resent, or while out-of-order packets are rearranged.

It is not suited well toward other applications, such as streaming audio or video, however. In these, it doesn't really matter if a packet is lost (a lost packet in a stream of 100 won't be distinguishable) but it does matter if they arrive late (i.e., because of a host resending a packet presumed lost), since the data stream will be paused while the lost packet is being resent. Once the lost packet is received, it will be put in the proper slot in the data stream, and then passed up to the application.

4.5.4 UDP

UDP (User Datagram Protocol) is a simple transport-layer protocol. It does not provide the same features as TCP, and is thus considered ``unreliable.'' Again, although this is unsuitable for some applications, it does have much more applicability in other applications than the more reliable and robust TCP.

4.5.4.1 Lower Overhead than TCP

One of the things that make UDP nice is its simplicity. Because it doesn't need to keep track of the sequence of packets, whether they ever made it to their destination, etc., it has lower overhead than TCP. This is another reason why it's more suited to streaming-data applications: there's less screwing around that needs to be done with making sure all the packets are there, in the right order, and that sort of thing.

4.6 Risk Management: The Game of Security

It's very important to understand that in security, one simply cannot say ``what's the best firewall?'' There are two extremes: absolute security and absolute access. The closest we can get to an absolutely secure machine is one unplugged from the network, power supply, locked in a safe, and thrown at the bottom of the ocean. Unfortunately, it isn't terribly useful in this state. A machine with absolute access is extremely convenient to use: it's simply there, and will do whatever you tell it, without questions, authorization, passwords, or any other mechanism. Unfortunately, this isn't terribly practical, either: the Internet is a bad neighborhood now, and it isn't long before some bonehead will tell the computer to do something like self-destruct, after which, it isn't terribly useful to you.

This is no different from our daily lives. We constantly make decisions about what risks we're willing to accept. When we get in a car and drive to work, there's a certain risk that we're taking. It's possible that something completely out of control will cause us to become part of an accident on the highway. When we get on an airplane, we're accepting the level of risk involved as the price of convenience. However, most people have a mental picture of what an acceptable risk is, and won't go beyond that in most circumstances. If I happen to be upstairs at home, and want to leave for work, I'm not going to jump out the window. Yes, it would be more convenient, but the risk of injury outweighs the advantage of convenience.

Every organization needs to decide for itself where between the two extremes of total security and total access they need to be. A policy needs to articulate this, and then define how that will be enforced with practices and such. Everything that is done in the name of security, then, must enforce that policy uniformly.

4.7 Types and Sources of Network Threats

Now, we've covered enough background information on networking that we can actually get into the security aspects of all of this. First of all, we'll get into the types of threats there are against networked computers, and then some things that can be done to protect you against various threats.

4.7.1 Denial-of-Service

DoS (Denial-of-Service) attacks are probably the nastiest, and most difficult to address. These are the nastiest, because they're very easy to launch, difficult (sometimes impossible) to track, and it isn't easy to refuse the requests of the attacker, without also refusing legitimate requests for service.

The premise of a DoS attack is simple: send more requests to the machine than it can handle. There are toolkits available in the underground community that make this a simple matter of running a program and telling it which host to blast with requests. The attacker's program simply makes a connection on some service port, perhaps forging the packet's header information that says where the packet came from, and then dropping the connection. If the host is able to answer 20 requests per second, and the attacker is sending 50 per second, obviously the host will be unable to service all of the attacker's requests, much less any legitimate requests (hits on the web site running there, for example).

Some things that can be done to reduce the risk of being stung by a denial of service attack include

- Not running your visible-to-the-world servers at a level too close to capacity
- Using packet filtering to prevent obviously forged packets from entering into your network address space.

  Obviously forged packets would include those that claim to come from your own hosts, addresses reserved for private networks as defined in RFC 1918 [4], and the loop back network (127.0.0.0).

- Keeping up-to-date on security-related patches for your hosts' operating systems.

4.7.2 Unauthorized Access

``Unauthorized access'' is a very high-level term that can refer to a number of different sorts of attacks. The goal of these attacks is to access some resource that your machine should not provide the attacker. For example, a host might be a web server, and should provide anyone with requested web pages. However, that host should not provide command shell access without being sure that the person making such a request is someone who should get it, such as a local administrator.

4.7.2.1 Executing Commands Illicitly

It's obviously undesirable for an unknown and un-trusted person to be able to execute commands on your server machines. There are two main classifications of the severity of this problem: normal user access, and administrator access. A normal user can do a number of things on a system (such as read files, mail them to other people, etc.) that an attacker should not be able to do. This might, then, be all the access that an attacker needs. On the other hand, an attacker might wish to make configuration changes to a host (perhaps changing its IP address, putting a start-up script in place to cause the machine to shut down every time it's started or something similar). In this case, the attacker will need to gain administrator privileges on the host.

4.7.2.2 Confidentiality Breaches

We need to examine the threat model: what is it that you're trying to protect yourself against? There is certain information that could be quite damaging if it fell into the hands of a competitor, an enemy, or the public. In these cases, it's possible that compromise of a normal user's account on the machine can be enough to cause damage (perhaps in the form of PR, or obtaining information that can be used against the company, etc.)

While many of the perpetrators of these sorts of break-ins are merely thrill-seekers interested in nothing more than to see a shell prompt for your computer on their screen, there are those who are more malicious, as we'll consider next. (Additionally, keep in mind that it's possible that someone who is normally interested in nothing more than the thrill could be persuaded to do more: perhaps an unscrupulous competitor is willing to hire such a person to hurt you.)

4.7.2.3 Destructive Behavior

Among the destructive sorts of break-ins and attacks, there are two major categories.

Data Diddling--The data diddler is likely the worst sort, since the fact of a break-in might not be immediately obvious. Perhaps he's toying with the numbers in your spreadsheets, or changing the dates in your projections and plans. Maybe he's changing the account numbers for the auto-deposit of certain paychecks. In any case, rare is the case when you'll come in to work one day, and simply know that something is wrong. An accounting procedure might turn up a discrepancy in the books three or four months after the fact. Trying to track the problem down will certainly be difficult, and once that problem is discovered, how can any of your numbers from that time period be trusted? How far back do you have to go before you think that your data is safe?

Data Destruction--Some of those perpetrate attacks are simply twisted jerks who like to delete things. In these cases, the impact on your computing capability -- and consequently your business -- can be nothing less than if a fire or other disaster caused your computing equipment to be completely destroyed.

4.7.3 Where Do They Come From?

How, though, does an attacker gain access to your equipment? Through any connection that you have to the outside world. This includes Internet connections, dial-up modems, and even physical access. (How do you know that one of the temps that you've brought in to help with the data entry isn't really a system cracker looking for passwords, data phone numbers, vulnerabilities and anything else that can get him access to your equipment?)

In order to be able to adequately address security, all possible avenues of entry must be identified and evaluated. The security of that entry point must be consistent with your stated policy on acceptable risk levels.

4.7.4 Lessons Learned

From looking at the sorts of attacks that are common, we can divine a relatively short list of high-level practices that can help prevent security disasters, and to help control the damage in the event that preventative measures were unsuccessful in warding off an attack.

4.7.4.1 Hope you have backups

This isn't just a good idea from a security point of view. Operational requirements should dictate the backup policy, and this should be closely coordinated with a disaster recovery plan, such that if an airplane crashes into your building one night, you'll be able to carry on your business from another location. Similarly, these can be useful in recovering your data in the event of an electronic disaster: a hardware failure or a breakin that changes or otherwise damages your data.

4.7.4.2 Don't Put Data where it doesn't need to be

Although this should go without saying, this doesn't occur to lots of folks. As a result, information that doesn't need to be accessible from the outside world sometimes is, and this can needlessly increase the severity of a break-in dramatically.

4.7.4.3 Avoid Systems with Single Points of Failure

Any security system that can be broken by breaking through any one component isn't really very strong. In security, a degree of redundancy is good, and can help you protect your organization from a minor security breach becoming a catastrophe.

4.7.4.4 Stay Current with Relevant Operating System Patches

Be sure that someone who knows what you've got is watching the vendors' security advisories. Exploiting old bugs is still one of the most common (and most effective!) means of breaking into systems.

4.7.4.5 Watch for Relevant Security Advisories

In addition to watching what the vendors are saying, keep a close watch on groups like CERT and CIAC. Make sure that at least one person (preferably more) is subscribed to these mailing lists

4.7.4.6 Have Someone on Staff be Familiar with Security

Having at least one person who is charged with keeping abreast of security developments is a good idea. This need not be a technical wizard, but could be someone who is simply able to read advisories issued by various incident response teams, and keep track of various problems that arise. Such a person would then be a wise one to consult with on security related issues, as he'll be the one who knows if web server software version such-and-such has any known problems, etc.

4.8 Firewalls

As we've seen in our discussion of the Internet and similar networks, connecting an organization to the Internet provides a two-way flow of traffic. This is clearly undesirable in many organizations, as proprietary information is often displayed freely within a corporate intranet (that is, a TCP/IP network, modeled after the Internet that only works within the organization).

In order to provide some level of separation between an organization's intranet and the Internet, firewalls have been employed. A firewall is simply a group of components that collectively form a barrier between two networks.

A number of terms specific to firewalls and networking are going to be used throughout this section, so let's introduce them all together.

Bastion host--A general-purpose computer used to control access between the internal (private) network (intranet) and the Internet (or any other un-trusted network). Typically, these are hosts running a flavor of the UNIX operating system that has been customized in order to reduce its functionality to only what is necessary in order to support its functions. Many of the general-purpose features have been turned off, and in many cases, completely removed, in order to improve the security of the machine.
Router--A special purpose computer for connecting networks together. Routers also handle certain functions, such as routing, or managing the traffic on the networks they connect.
Access Control List (ACL)--Many routers now have the ability to selectively perform their duties, based on a number of facts about a packet that comes to it. This includes things like origination address, destination address, destination service port, and so on. These can be employed to limit the sorts of packets that are allowed to come in and go out of a given network.
Demilitarized Zone (DMZ)--The DMZ is a critical part of a firewall: it is a network that is neither part of the un-trusted network, nor part of the trusted network. But, this is a network that connects the un-trusted to the trusted. The importance of a DMZ is tremendous: someone who breaks into your network from the Internet should have to get through

several layers in order to successfully do so. Those layers are provided by various components within the DMZ.

Proxy--This is the process of having one host act in behalf of another. A host that has the ability to fetch documents from the Internet might be configured as a proxy server, and host on the intranet might be configured to be proxy clients. In this situation, when a host on the intranet wishes to fetch the <http://www.interhack.net/> web page, for example, the browser will make a connection to the proxy server, and request the given URL. The proxy server will fetch the document, and return the result to the client. In this way, all hosts on the intranet are able to access resources on the Internet without having the ability to direct talk to the Internet.

4.8.1 Types of Firewalls

There are three basic types of firewalls, and we'll consider each of them.

4.8.1.1 Application Gateways

The first firewalls were application gateways, and are sometimes known as proxy gateways. These are made up of bastion hosts that run special software to act as a proxy server. This software runs at the Application Layer of our old friend the ISO/OSI Reference Model, hence the name. Clients behind the firewall must be proxitized (that is, must know how to use the proxy, and be configured to do so) in order to use Internet services. Traditionally, these have been the most secure, because they don't allow anything to pass by default, but need to have the programs written and turned on in order to begin passing traffic.

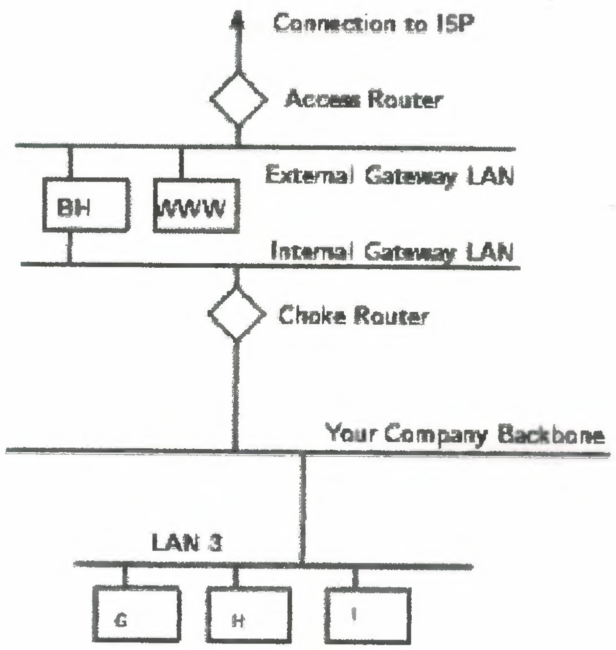


Figure 4.5: A sample application gateway

These are also typically the slowest, because more processes need to be started in order to have a request serviced. Figure 4.5 show an application gateway.

4.8.1.2 Packet Filtering

Packet filtering is a technique whereby routers have ACLs (Access Control Lists) turned on. By default, a router will pass all traffic sent it, and will do so without any sort of restrictions. Employing ACLs is a method for enforcing your security policy with regard to what sorts of access you allow the outside world to have to your internal network, and vice versa.

There is less overhead in packet filtering than with an application gateway, because the feature of access control is performed at a lower ISO/OSI layer (typically, the transport or session layer). Due to the lower overhead and the fact that packet filtering is done with routers, which are specialized computers optimized for tasks related to networking, a packet filtering gateway is often much faster than its application layer cousins. Figure 4.6 show a packet filtering gateway.

Because we're working at a lower level, supporting new applications either comes automatically, or is a simple matter of allowing a specific packet type to pass through the gateway. (Not that the possibility of something automatically makes it a good idea; opening things up this way might very well compromise your level of security below what your policy allows.)

There are problems with this method, though. Remember, TCP/IP has absolutely no means of guaranteeing that the source address is really what it claims to be. As a result, we have to use layers of packet filters in order to localize the traffic. We can't get all the way down to the actual host, but with two layers of packet filters, we can differentiate between a packet that came from the Internet and one that came from our internal network. We can identify which network the packet came from with certainty, but we can't get more specific than that.

4.8.1.3 Hybrid Systems

In an attempt to marry the security of the application layer gateways with the flexibility and speed of packet filtering, some vendors have created systems that use the principles of both.

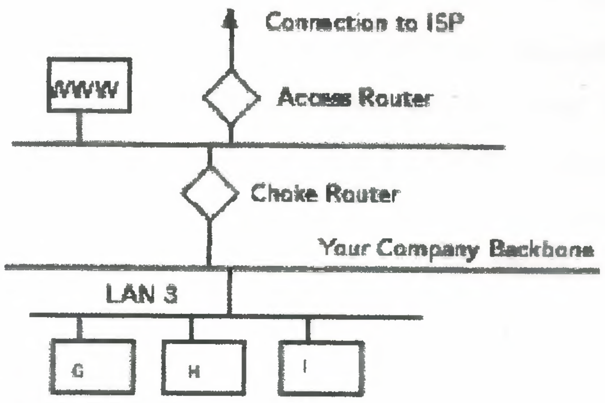


Figure 4.6: A sample packet filtering gateway

In some of these systems, new connections must be authenticated and approved at the application layer. Once this has been done, the remainder of the connection is passed down to the session layer, where packet filters watch the connection to ensure that only packets that are part of an ongoing (already authenticated and approved) conversation are being passed.

Other possibilities include using both packet filtering and application layer proxies. The benefits here include providing a measure of protection against your machines that provide services to the Internet (such as a public web server), as well as provide the security of an application layer gateway to the internal network. Additionally, using this method, an attacker, in order to get to services on the internal network, will have to break through the access router, the bastion host, and the choke router.

4.8.2 So, what's best for me?

Lots of options are available, and it makes sense to spend some time with an expert, either in-house, or an experienced consultant who can take the time to understand your organization's security policy, and can design and build a firewall architecture that best implements that policy. Other issues like services required, convenience, and scalability might factor in to the final design.

4.8.3 Some Words of Caution

The business of building firewalls is in the process of becoming a commodity market. Along with commodity markets come lots of folks who are looking for a way to make a buck without necessarily knowing what they're doing. Additionally, vendors compete with each other to try and claim the greatest security, the easiest to administer, and the least visible to end users. In order to try to quantify the potential security of firewalls, some organizations have taken to firewall certifications. The certification of a firewall means nothing more than the fact that it can be configured in such a way that it can pass a series of tests. Similarly, claims about meeting or exceeding U.S. Department of Defense ``Orange Book" standards, C-2, B-1, and such all simply mean that an organization was able to configure a machine to pass a series of tests. This doesn't mean that it was loaded with the vendor's software at the time, or that the machine was even usable. In fact, one vendor has been claiming their operating system is ``C-2 Certified" didn't make mention of the fact that their operating system only passed the C-2 tests without being connected to any sort of network devices.

Such gauges as market share, certification, and the like are no guarantees of security or quality. Taking a little bit of time to talk to some knowledgeable folks can go a long way in providing you a comfortable level of security between your private network and the big, bad Internet.

Additionally, it's important to note that many consultants these days have become much less the advocate of their clients, and more of an extension of the vendor. Ask any consultants you talk to about their vendor affiliations, certifications, and whatnot. Ask what difference it makes to them whether you choose one product over another, and vice versa. And then ask yourself if a consultant who is certified in technology XYZ is going to provide you with competing technology ABC, even if ABC best fits your needs.

4.8.3.1 Single Points of Failure

Many ``firewalls'' are sold as a single component: a bastion host, or some other black box that you plug your networks into and get a warm-fuzzy, feeling safe and secure. The term ``firewall'' refers to a number of components that collectively provide the security of the system. Any time there is only one component paying attention to what's going on between the internal and external networks, an attacker has only one thing to break (or fool!) in order to gain complete access to your internal networks.

4.9 Secure Network Devices

It's important to remember that the firewall only one entry point to your network. Modems, if you allow them to answer incoming calls, can provide an easy means for an attacker to sneak around (rather than through) your front door (or, firewall). Just as castles weren't built with moats only in the front, your network needs to be protected at all of its entry points.

4.9.1 Secure Modems; Dial-Back Systems

If modem access is to be provided, this should be guarded carefully. The terminal server, or network device that provides dial-up access to your network needs to be actively administered, and its logs need to be examined for strange behavior. Its password need to be strong -- not ones that can be guessed. Accounts that aren't actively used should be disabled. In short, it's the easiest way to get into your network from remote: guard it carefully.

There are some remote access systems that have the feature of a two-part procedure to establish a connection. The first part is the remote user dialing into the system, and providing the correct userid and password. The system will then drop the connection, and call the authenticated user back at a known telephone number. Once the remote user's system answers that call, the connection is established, and the user is on the network. This works well for folks working at home, but can be problematic for users wishing to dial in from hotel rooms and such when on business trips.

Other possibilities include one-time password schemes, where the user enters his userid, and is presented with a ``challenge," a string of between six and eight numbers. He types this challenge into a small device that he carries with him that looks like a calculator. He then presses enter, and a ``response" is displayed on the LCD screen. The user types the response, and if all is correct, he login will proceed. These are useful devices for solving the problem of good passwords, without requiring dial-back access. However, these have their own problems, as they require the user to carry them, and they must be tracked, much like building and office keys.

No doubt many other schemes exist. Take a look at your options, and find out how what the vendors have to offer will help you enforce your security policy effectively.

4.9.2 Crypto-Capable Routers

A feature that is being built into some routers is the ability to session encryption between specified routers. Because traffic traveling across the Internet can be seen by people in the middle who have the resources (and time) to snoop around, these are advantageous for providing connectivity between two sites, such that there can be secure routes.

4.9.3 Virtual Private Networks

Given the ubiquity of the Internet, and the considerable expense in private leased lines, many organizations have been building VPNs (Virtual Private Networks). Traditionally, for an organization to provide connectivity between a main office and a satellite one, an expensive data line had to be leased in order to provide direct connectivity between the two offices. Now, a solution that is often more economical is to provide both offices connectivity to the Internet. Then, using the Internet as the medium, the two offices can communicate.

The danger in doing this, of course, is that there is no privacy on this channel, and it's difficult to provide the other office access to ``internal'' resources without providing those resources to everyone on the Internet.

VPNs provide the ability for two offices to communicate with each other in such a way that it looks like they're directly connected over a private leased line. The session between them, although going over the Internet, is private (because the link is encrypted), and the link is convenient, because each can see each others' internal resources without showing them off to the entire world.

A number of firewall vendors are including the ability to build VPNs in their offerings, either directly with their base product, or as an add-on. If you have needed to connect several offices together, this might very well be the best way to do it.

CONCLUSION

In computer security, cryptography is one of several ways of achieving isolation, and is not one of the more important ways. In network security, cryptography is the main attraction - everything is done via message-passing (ultimately), so the only secure way to achieve confidentiality and the authentication needed for access control is through cryptography. Security is a very difficult topic. Everyone has a different idea of what "security" as there are many ways of encrypt and decrypt data so no body can decide the acceptable levels of risk. The key for building a secure network is depend on the cryptography functions and size of key chosen. Once that has been defined, everything that goes on with the network can be evaluated with respect to that policy. Many people pay great amounts of lip service to security, but do not want to be bothered with it when it gets in their way. It's important to build systems and networks in such a way that the user is not constantly reminded of the security system around him. Users who find security policies and systems too restrictive will find ways around them. It's important to get their feedback to understand what can be improved, and it's important to let them know why what have been done, the sorts of risks that are deemed unacceptable, and what has been done to minimize the organization's exposure to them. Security is everybody's business, and only with everyone's cooperation, an intelligent policy, and consistent practices, will it be achievable.

REFERENCE:

[1] Symmetric cryptographic system for data encryption by C. ADAMS in Apr 1996.

[2] Proceedings of the Internet Society Symposium on Network and Distributed System Security by IDUP and SPKM in1996.

[3] Random sources for cryptographic systems by G.B. AGNEW in 1988.

[4] An implementation for a fast public-key cryptosystem by G.B. AGNEW, R.C. MULLIN, I.M. ONYSZCHUK & S.A. VANSTONE in1991.

[5] A secure key distribution system by N. ALEXANDRIS, M. BURMESTER & V. CHRISSIKOPOULOS.

[6] A Weakness in the 4.2BSD UNIX TCP/IP Software by R.T. Morris in 1985.

[7] Security Problems in the TCP/IP Protocol Suite Vol. 19 by S.M. Bellovin in April 1989.

[8] Handbook of Applied Cryptography by A. Menezes, P. van Oorschot, and S. Vanstone in 1996.