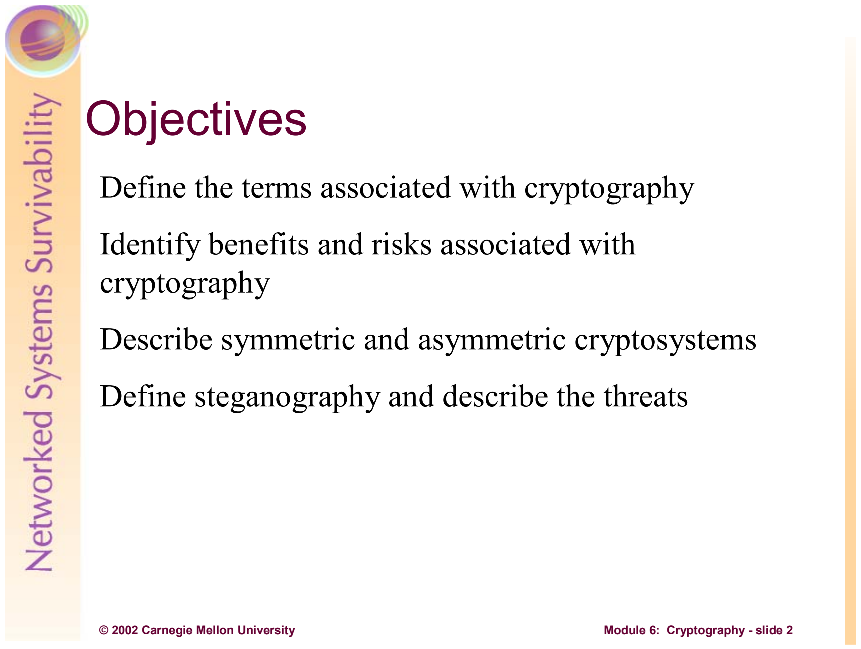
Information Security for Technical Staff

Module 6:

Cryptography

Networked Systems Survivability

CERT® Coordination Center
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

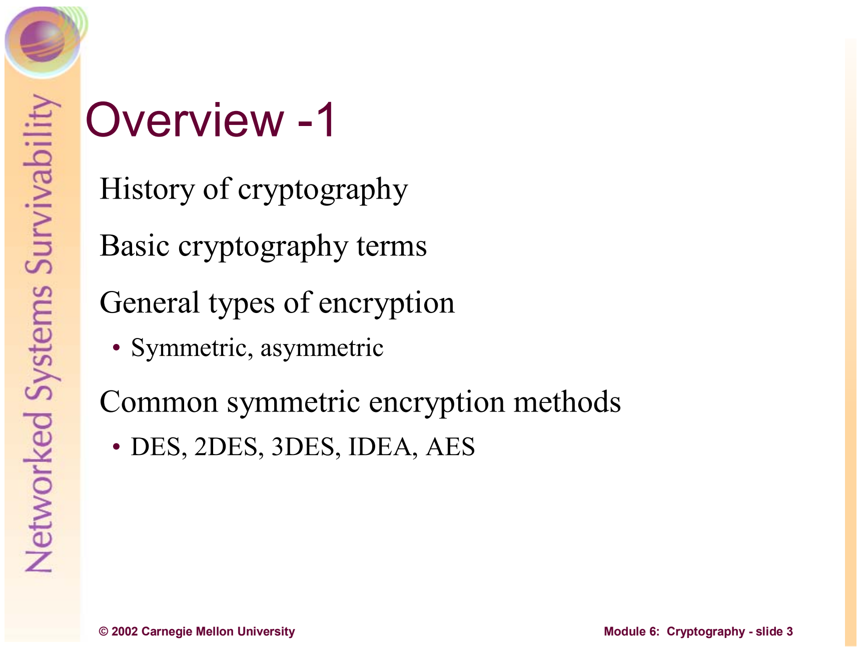


Networked Systems Survivability

Objectives

- Define the terms associated with cryptography
- Identify benefits and risks associated with cryptography
- Describe symmetric and asymmetric cryptosystems
- Define steganography and describe the threats

© 2002 Carnegie Mellon University Module 6: Cryptography - slide 2



Overview -1

History of cryptography

Basic cryptography terms

General types of encryption

- Symmetric, asymmetric

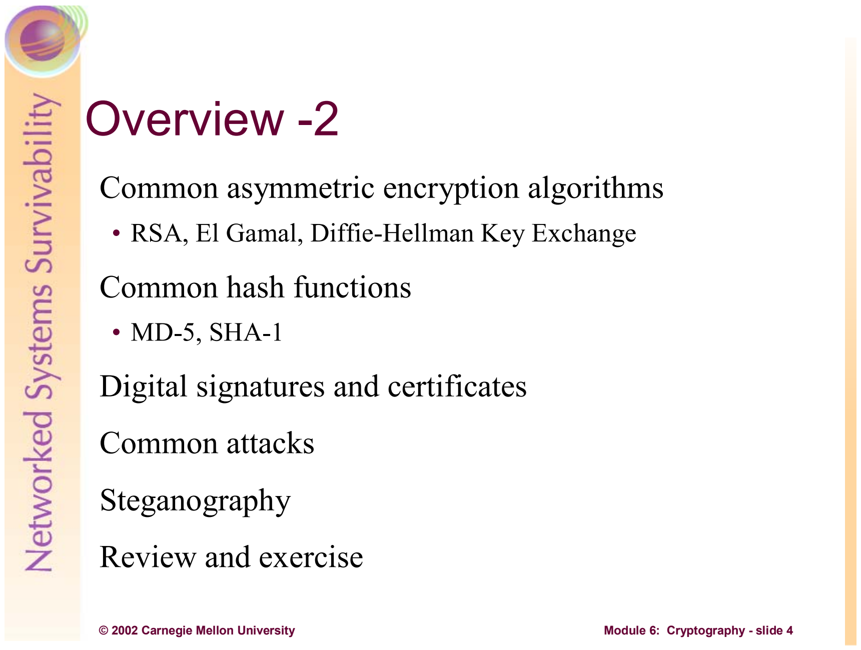
Common symmetric encryption methods

- DES, 2DES, 3DES, IDEA, AES

Networked Systems Survivability

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 3



Networked Systems Survivability

Overview -2

Common asymmetric encryption algorithms

- RSA, El Gamal, Diffie-Hellman Key Exchange

Common hash functions

- MD-5, SHA-1

Digital signatures and certificates

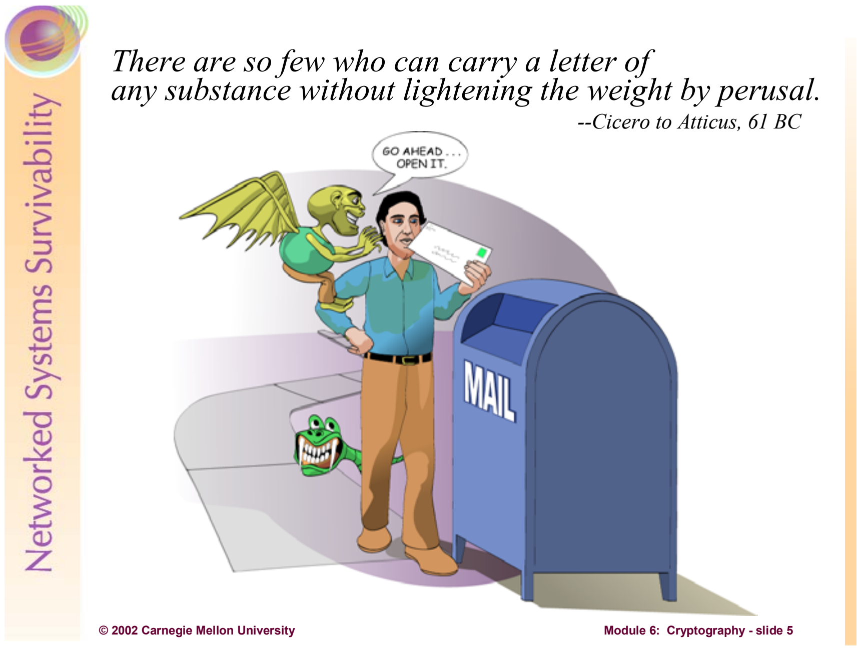
Common attacks

Steganography

Review and exercise

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 4



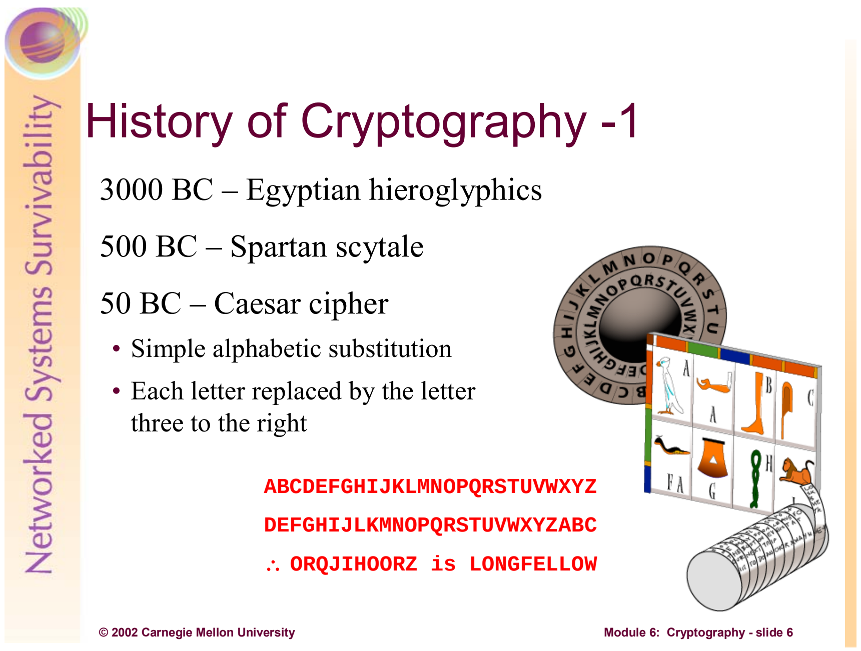
There are so few who can carry a letter of any substance without lightening the weight by perusal.

--Cicero to Atticus, 61 BC

Networked Systems Survivability
GO AHEAD... OPEN IT.
MAIL
© 2002 Carnegie Mellon University
Module 6: Cryptography - slide 5

Given the public nature of the networks that carry a vast majority of the traffic of the Internet, and given the sensitive nature of some of the information that we must pass, it is imperative that we consider ways to conceal the traffic we will be passing. Cryptography can do this task quite well for us if we implement it correctly and manage it properly.

This module will describe some of the basics of cryptography and give you a baseline understanding of common methods of protecting information through encryption.



Networked Systems Survivability

History of Cryptography -1

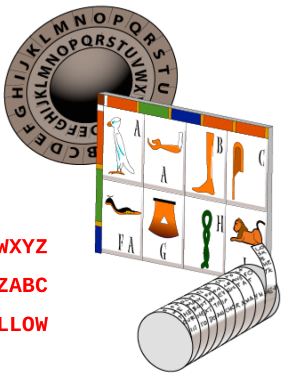
3000 BC – Egyptian hieroglyphics

500 BC – Spartan scytale

50 BC – Caesar cipher

- Simple alphabetic substitution
- Each letter replaced by the letter three to the right

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
DEFGHIJLKMNOPQRSTUVWXYZABC
∴ ORQJIHOORZ is LONGFELLOW
```

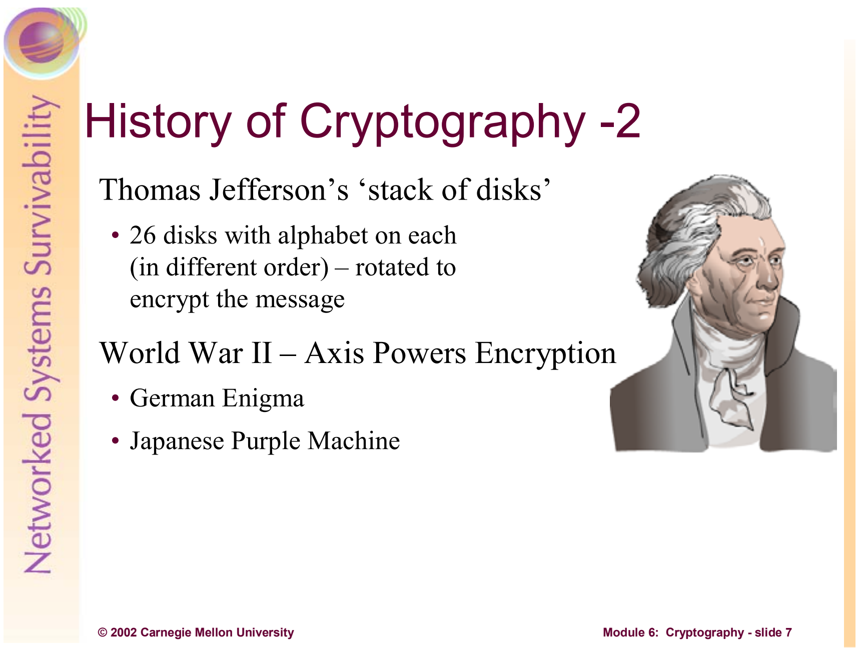


© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 6

The art and science of cryptography has a long and distinguished heritage. Not long after people started writing, they wanted a way to keep their information secret from others. Ancient Egyptians employed hieroglyphics to conceal writings from unintended recipients as early as 3000 B.C. Another of the oldest known examples is the Spartan scytale. Over 2500 years ago, Spartan government sent secret messages to its generals in the following clever way. Sender and recipient each had a cylinder (called a *scytale*) of exactly the same radius. The sender wound a narrow ribbon of parchment around his cylinder, and then wrote on it lengthwise. After the ribbon is unwound, only a person who had a cylinder of exactly the same circumference could read the writing.

Julius Caesar created one of the first encryption systems in Rome – later referred to as the Caesar Cipher. Caesar's cipher was a simple substitution cipher – where each letter in the message was replaced with the letter three to the right of that letter. Therefore, 'A' would be replaced with 'D,' 'M' with 'P,' and 'X' with 'A.' This simple alphabetic substitution scheme can be used effectively by using an alphabet shifted from 2 to 25 spaces. There is even a simple alphabetic substitution program built into UNIX known as ROT13, which uses an alphabet shifted 13 to the right. While not an effective encryption scheme for data security, it does hide messages from offensive language filters applied to many newsgroups, or can also be used to hide answers to puzzles or riddles.



Networked Systems Survivability

History of Cryptography -2

Thomas Jefferson's 'stack of disks'

- 26 disks with alphabet on each (in different order) – rotated to encrypt the message

World War II – Axis Powers Encryption

- German Enigma
- Japanese Purple Machine

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 7

The interest in strong cryptography, especially for those not in the military or other governmental bodies, was very limited for many centuries. Thomas Jefferson worked on an encryption scheme whereby a stack of 26 disks, each with the alphabet in a different order, would be rotated based on a certain 'key' – which would be changed for each message.

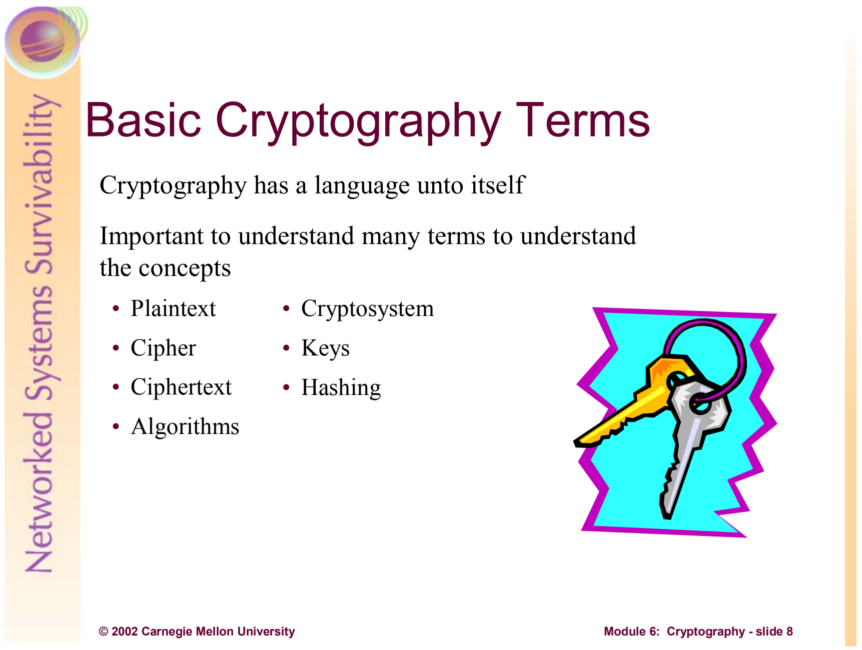
Jefferson's device did not come into wide use until it was reinvented by Brazilian Commandant Etienne Bazeries over a century later, and became known as the "Bazeries cylinder". A Bazeries cylinder consisted of a set of roughly 20 to 30 numbered disks, with a different cipher alphabet on the edge of each disk, and a hole in the center of the disks to allow them to be stacked on an axle. The disks are not fixed in place – they can be removed and can be mounted on the axle in different orders. The order of the disks can be considered the cipher key for the Bazeries cylinder, with both the sender and the receiver arranging the disks in the same predefined order.

The Enigma cipher is most well known for its contributions to World War II on the Germans' side. They developed what came to be known as The Enigma Machine. The machine was based on a system of three rotors that substituted cipher text letters for plain text letters. The rotors would spin in conjunction with each other, thus performing varying substitutions much like the Caeser Shift. When a letter was typed on the keyboard of the machine, it was first sent through the first rotor, which would shift the letter according to its present setting. The new letter would then pass through the second rotor, where it would be replaced by a substitution according to the present setting of the second rotor. This new letter would in turn pass through the third rotor, again being substituted accordingly. Next, this new letter would be bounced off of a reflector, and back through the three rotors in reverse order. The trick that made Enigma so powerful for its time though, was the spinning of the rotors.

As the plain text letter passed through the first rotor, the first rotor would rotate one position. The other two rotors would remain stationary until the first rotor had rotated 26 times (the number of letters in the alphabet and therefore, one full rotation). Then the second rotor would rotate one position. After the second rotor had rotated 26 times (26X26 letters, since the first rotor has to rotate 26 times for every time the second rotor rotates), the third rotor would rotate one position. The cycle would continue like this for the entire length of the message. The result was a shifting shift. In other words, an s could be encoded as a b in the first part of the message, and then as an m later in the message. This principle of the shifting rotors allowed for 26X26X26 = 17576 possible positions of the rotors. In order for the recipient to decode the message, they would need to know the initial settings of the rotors, and then put the cipher text through the machine to find the plain text. The Germans devised a system by which all of the recipients would set their rotors to predetermined settings according to the date. Each clerk had a book detailing the settings for each day. This presented a major weakness in the system though. Obviously, if anyone could figure out what the settings of the rotors were for a particular day, they would be able to decode that day's messages, assuming they had an Enigma Machine themselves.

Purple was the U.S. codename for a Japanese cipher machine used during World War II. The U.S. called it Purple because the first Japanese code, called orange had increased in difficulty and went on to the color red until the most difficult, the Purple, was reached. This was the code constructed on a machine unlike other previous codes which consisted of pen and paper. The Purple machine was made up of two electric typewriters, separated by a plugboard and a box of cipher wheels. The enciphered message was typed on one of the typewriters and the ciphered text was printed out of the other typewriter.

The enciphering process starts on one of the typewriters. Each key on the typewriter's keyboard connects to a wire which links to a switchboard. Thus, it has twenty-six holes lettered A to Z and a wired plug in each of them. As electric impulses are received from pressing the typewriter keys they are re-routed and sent on to different destinations by rearranging the plugs in the plugboard. So if an A plug is placed in the hole marked D, the letter A pressed on the typewriter will be changed into a D by the plugboard. This is a simple substitution, but the Japanese had a plan to switch their plugs around each day and confused the enemy by constantly changing the plugboard's alphabet. The letter A already changed into a D by the plugboard travels along the D wire into the box of cipher wheels. Each wheel represents a different letter and each disc has twenty-six letters of the alphabet in scrambled order written around the edges. The original A is now a D, after the plugboard travels to the wheel standing in the D position. It can represent another letter, for example the letter S. If S is represented, then S will be printed by the second typewriter. After S has been typed, the cogs (which are attached to the wheels) turn a certain number of spaces and the next time A is typed it may come out as S, T, or M. This results in a word being coded in many thousands of ways. It is hard to do a frequency count on the Purple machine since it is constructed to almost never to repeat itself. With Purple in the American hands, they could decipher the Japanese diplomatic messages concerning the Japanese attack on Pearl Harbor. This Purple machine gave the only clue to Japanese intentions during the war.



Basic Cryptography Terms

Cryptography has a language unto itself

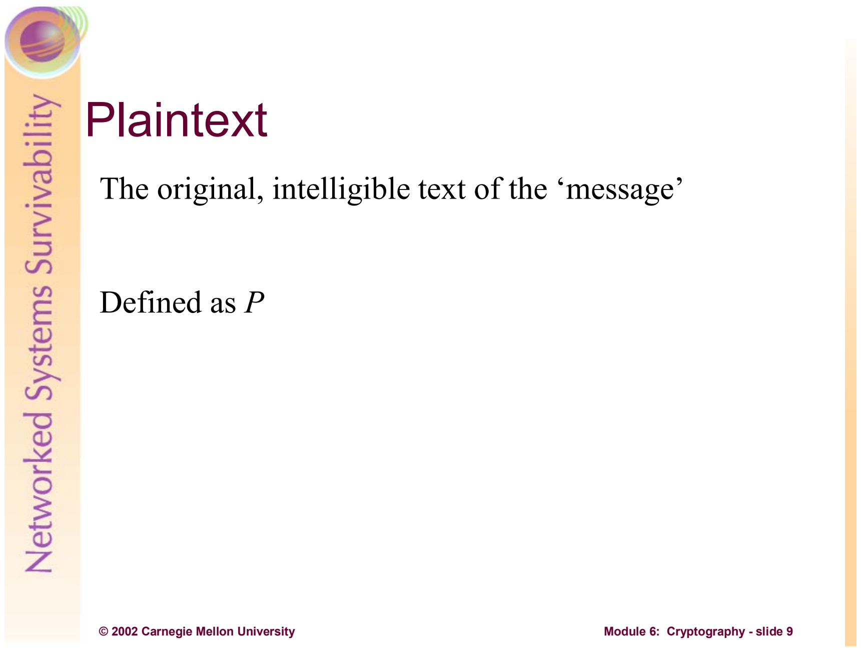
Important to understand many terms to understand the concepts

- Plaintext
- Cipher
- Ciphertext
- Algorithms
- Cryptosystem
- Keys
- Hashing

Networked Systems Survivability

Module 6: Cryptography - slide 8

A list of the terms that will be defined.


Networked Systems Survivability

Plaintext

The original, intelligible text of the 'message'

Defined as *P*

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 9

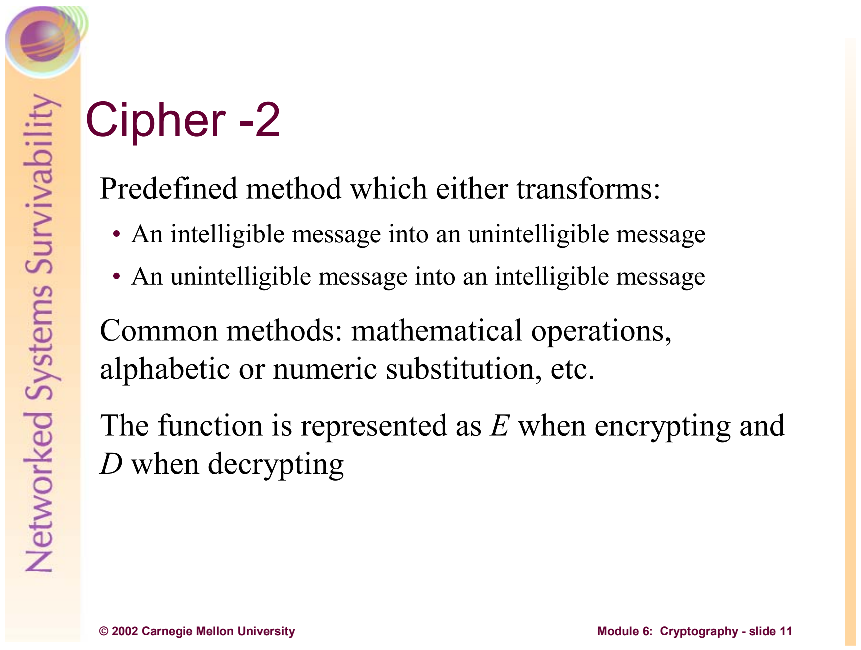
Plaintext, quite simply, is what you want to encrypt. It is the war plans. It is your prize-winning recipe for watermelon cookies. It is your confession that you shot J.R. It is anything to which you don't want the world to have access. And it is the information that will be put through the encryption process and will therefore be protected.

Networked Systems Survivability

Cipher -1

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 10



Cipher -2

Predefined method which either transforms:

- An intelligible message into an unintelligible message
- An unintelligible message into an intelligible message

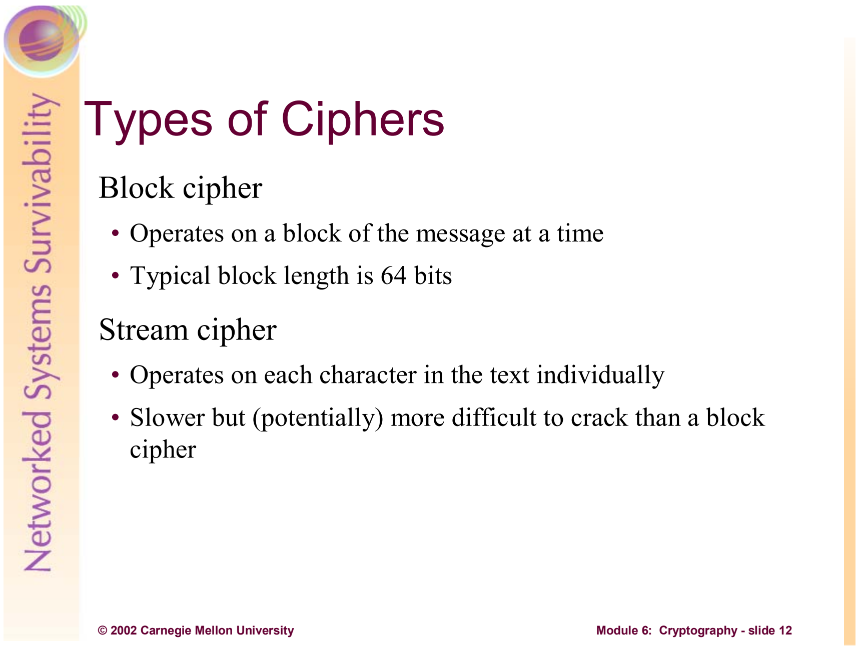
Common methods: mathematical operations, alphabetic or numeric substitution, etc.

The function is represented as *E* when encrypting and *D* when decrypting

Networked Systems Survivability

Module 6: Cryptography - slide 11

Cryptography is impossible without a good method of changing your plaintext into unintelligible text. The cipher is the process for this transformation. It can be as simple as the steps taken in the Caesar Cipher – substitute each letter with the letter three spaces to the right. Or, in the age of digital computers, it can be a very complex set of mathematical operations that take a very detailed understanding of prime numbers and factoring to comprehend. But either way, the cipher is the tool that operates on your plaintext to make it unintelligible. In computer systems, it is a mathematical function operating on the bits of data. Often in the past the cipher itself had to be kept secret. This is called a 'restricted algorithm,' and is still used occasionally today. The security of the encrypted text is dependant solely on the secrecy of the cipher. It is virtually impossible to use a restricted algorithm in a large group, or a group that changes even occasionally – as this would require constant changing of the cipher itself. So most of today's algorithms are commonly known and survive the test of public scrutiny.



Types of Ciphers

Block cipher

- Operates on a block of the message at a time
- Typical block length is 64 bits

Stream cipher

- Operates on each character in the text individually
- Slower but (potentially) more difficult to crack than a block cipher

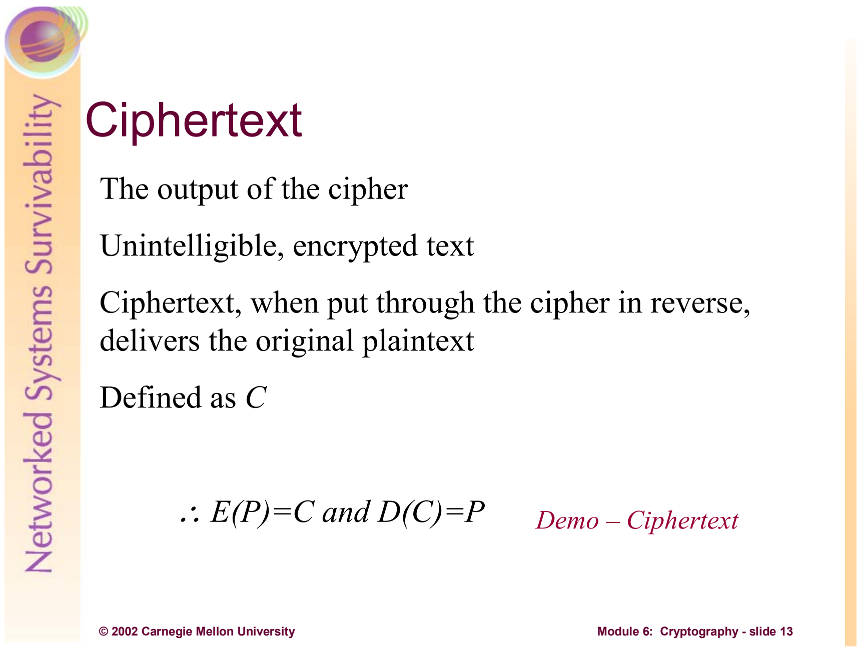
Networked Systems Survivability
© 2002 Carnegie Mellon University
Module 6: Cryptography - slide 12

There are two main types of ciphers – *block ciphers* and *stream ciphers*. These operate on the plaintext in two distinct ways.

Block cipher – operates on a set number of bits at the same time. For example, if you have a message that is 1024 bits long, and your block size is 64 bits, the cipher will take the first 64 bits of data and operate on them. Once that block is finished, it moves on to the next 64-bit block and operates on it, and so on until it has operated on all 1024 bits in the plaintext. Likewise, to decrypt the data, the cipher will operate on 64 bit blocks, turning the encrypted text back to plaintext and then move on to the next 64-bit block.

Stream cipher – operates on one byte at a time. Stream ciphers do all of their calculations and operations on every single byte. In our 1024 byte example message, the cipher would have to do the encryption operations 1024 times – once for each plaintext bit.

Block ciphers are generally faster than stream ciphers, as they can operate on larger numbers of bits, and do a smaller number of operations on plaintext than the stream cipher would.



Ciphertext

The output of the cipher

Unintelligible, encrypted text

Ciphertext, when put through the cipher in reverse, delivers the original plaintext

Defined as *C*

$$\therefore E(P)=C \text{ and } D(C)=P$$

Demo – Ciphertext

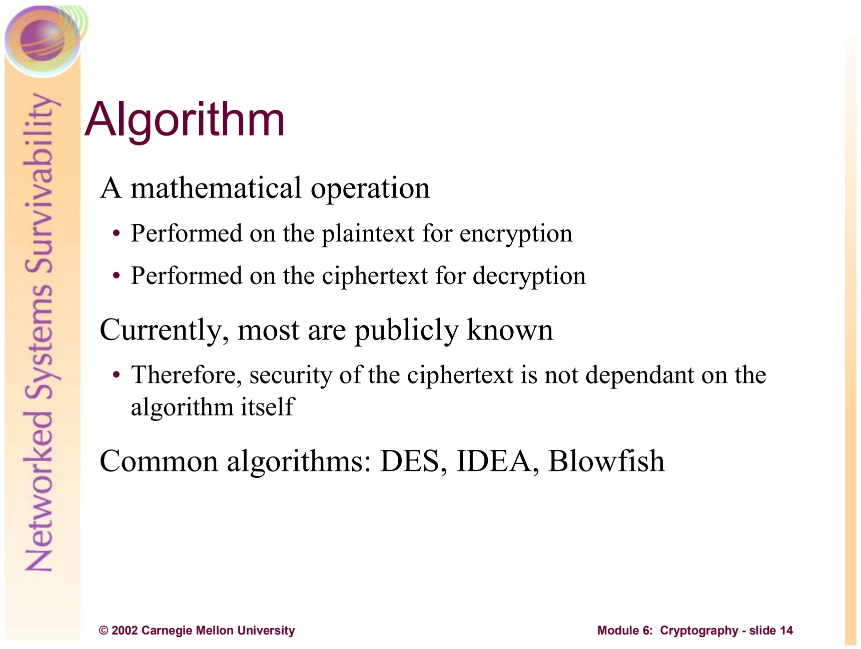
Networked Systems Survivability

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 13

Once you've run your plaintext message through your cipher, the output you get is called ciphertext. Ciphertext is unintelligible. For example, taking the string "I shot J.R." and running it through a strong encryption cipher, the output you get is analogous to "#@^<<:*&*&". As you can tell, it is not easily read by humans, and not easily decrypted without knowing the cipher and some other information (called a key). Your data is then safe from the prying eyes – whether it is stored on your computer or transmitted across the network or Internet.

Ciphertext stays ciphertext until decrypted by the decryption algorithm.



Networked Systems Survivability

Algorithm

A mathematical operation

- Performed on the plaintext for encryption
- Performed on the ciphertext for decryption

Currently, most are publicly known

- Therefore, security of the ciphertext is not dependant on the algorithm itself

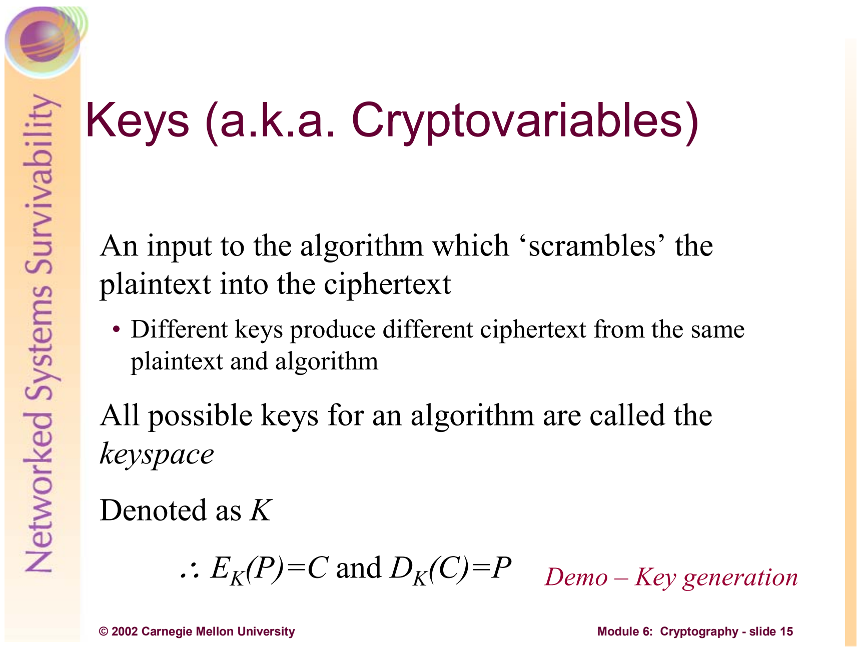
Common algorithms: DES, IDEA, Blowfish

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 14

An algorithm is a detailed sequence of actions performed to accomplish some task. In cryptography, algorithms normally refer to the actual set of mathematical operations that the cipher performs on the plaintext to encrypt, and the ciphertext to decrypt. People will sometimes use cipher and algorithm interchangeably.

There are many common algorithms now that have stood the test of public scrutiny. They include: The Digital Encryption Standard (DES), the International Digital Encryption Algorithm (IDEA), the Advanced Encryption Standard (AES), Blowfish, Twofish, CAST, RSA, and many more. Some of these will be discussed later in this module.



Networked Systems Survivability

Keys (a.k.a. Cryptovariables)

An input to the algorithm which 'scrambles' the plaintext into the ciphertext

- Different keys produce different ciphertext from the same plaintext and algorithm

All possible keys for an algorithm are called the *keyspace*

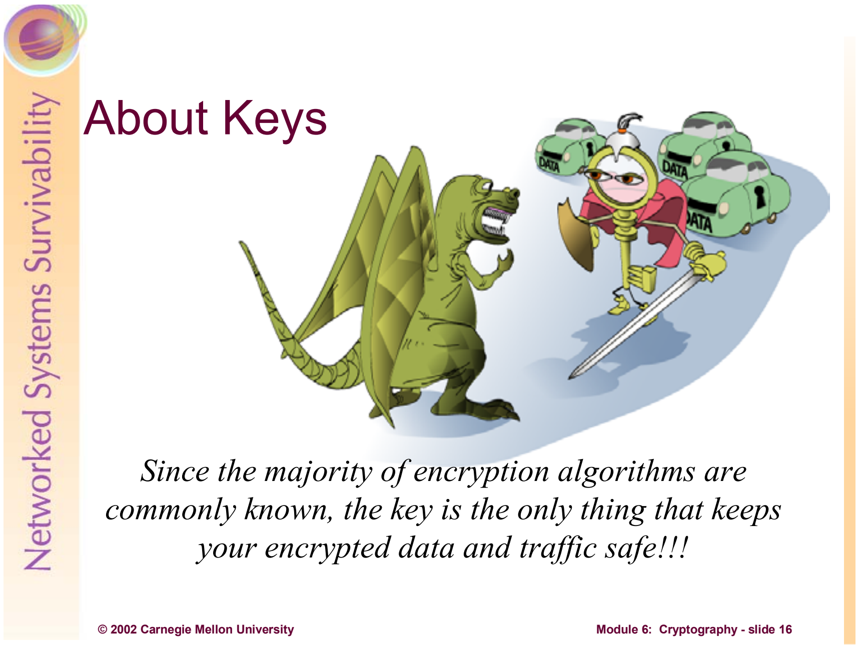
Denoted as *K*

$\therefore E_K(P)=C$ and $D_K(C)=P$ *Demo – Key generation*

© 2002 Carnegie Mellon University Module 6: Cryptography - slide 15

Since the algorithms today are well known, and perform the same basic operations on our plaintext, there must be a way for us to uniquely encrypt our plaintext that is not dependent on our encryption algorithm alone. Keys, which are more accurately called 'cryptovariables,' are the answer to this problem. For today's algorithms, this key is usually a very large series of bits – ranging in size from 56 bits for the Data Encryption Standard to 3092 bits (and possibly more) for RSA. The key and the plaintext are the inputs into the cipher, and the key is used to operate on each of the blocks in a block cipher and on each bit in a stream cipher.

Some encryption schemes use the same (secret) key to encrypt and decrypt the message. Public key encryption uses two different keys – a public key which is known by everyone, and a private key which must be kept secret.



About Keys

Since the majority of encryption algorithms are commonly known, the key is the only thing that keeps your encrypted data and traffic safe!!!

Networked Systems Survivability

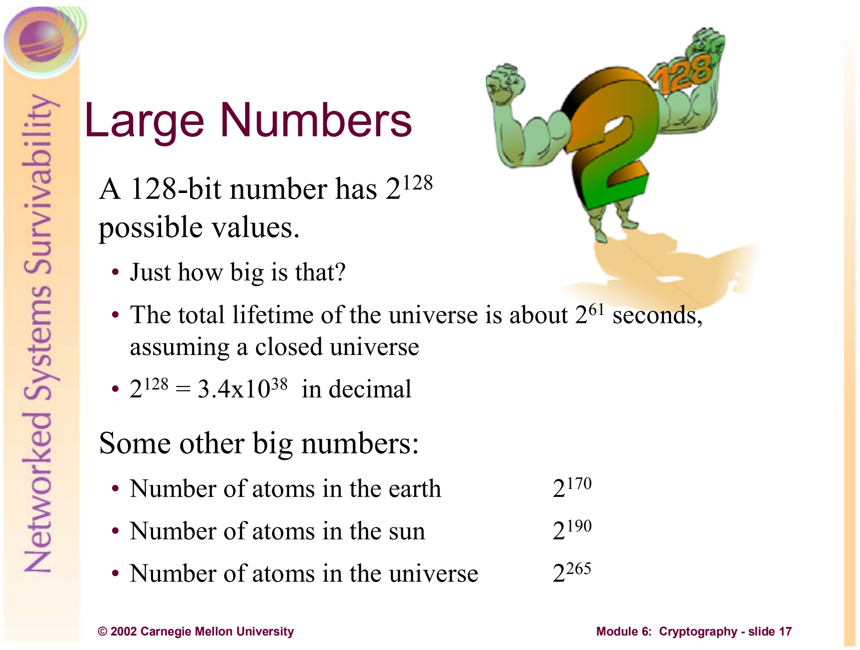
© 2002 Carnegie Mellon University   Module 6: Cryptography - slide 16

Keeping your keys safe is a requirement for strong security. You have the ability to control this.

Keys that are generated based on the programming of the algorithm or other non-random factors will be easier to guess and therefore are considered weak.

If an attacker can predict the key or even narrow down the number of keys they must try, the cryptosystem can be broken much more easily. Another difficulty of strong key creation is this: random numbers are hard to create in a computer – they are better produced in nature. On the computer, encryption software has to be satisfied with a pseudo-random number generator (PRNG) to produce 'random' numbers. This starts with a 'seed' number that is as random as it can be made (usually based on events going on around the computer – movement of the mouse, speed of typing keys on the keyboard, etc.). The PRNG then expands the seed into a longer, random-looking stream of binary bits. Hence the name pseudo – it's not completely random, but it appears to be random. Since the seed (or seeds) feed the PRNG – which in turn feeds the key generation algorithm, they need to be quite random. If they are not, an attacker can guess (or narrow down) which pseudo-random streams of bits would have been used in making the session key – and that's not a good thing.

The security of SSL version (which will be covered in more detail in Module 12 – Securing Remote Access) depends on the unpredictability of the session key. In it's first release in 1995, it turned out that Netscape's SSL 1.0 seed depended on three relatively non-random items: the time of day, the process ID, and the parent process ID. All three can be found or guessed (rather accurately). A UNIX user using the Netscape browser could get the process ID and the parent process ID simply by using a command that lists the process IDs of *all* processes on the system. The other missing information is then the time of day when the session key was generated. And this is not terribly difficult to guess if the attacker has some transmission information about the encrypted session. Packet capturing programs (like sniffer or tcpdump) will record the exact time they see each packet go by. That means an attacker can guess the time to within a second. And that in turn means the attacker would only need to try a million seeds in order to get the microseconds input to the PRNG. Testing all one million possibilities takes only a few seconds on a reasonably equipped PC.


Networked Systems Survivability
128

Large Numbers

A 128-bit number has 2^{128} possible values.

- Just how big is that?
- The total lifetime of the universe is about 2^{61} seconds, assuming a closed universe
- $2^{128} = 3.4 \times 10^{38}$ in decimal

Some other big numbers:

• Number of atoms in the earth	2^{170}
• Number of atoms in the sun	2^{190}
• Number of atoms in the universe	2^{265}

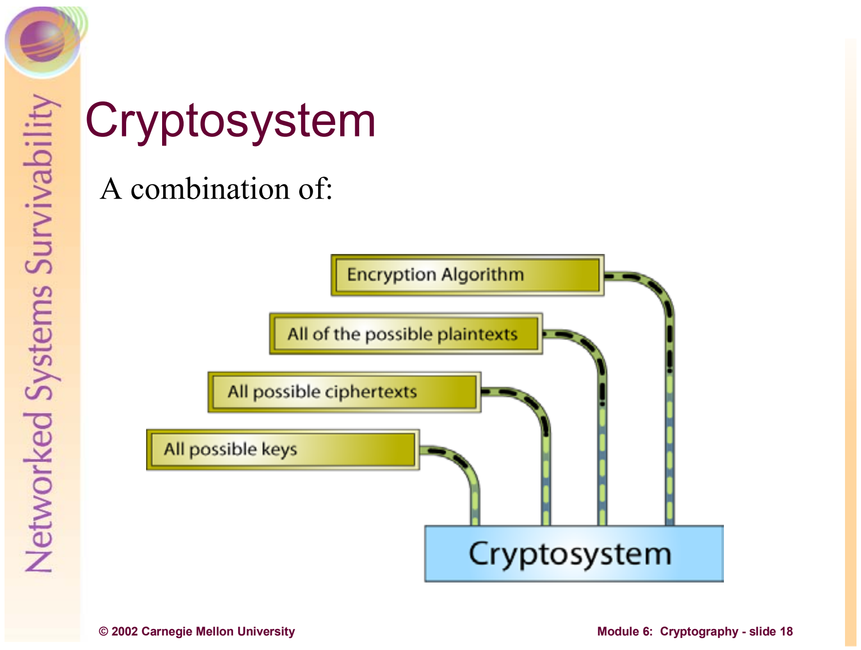
© 2002 Carnegie Mellon University Module 6: Cryptography - slide 17

Just how big are cryptographic keys? To crack the RSA algorithm (which will be discussed later), one must factor large prime numbers. The difficulty of factoring a large number of a given size is known, and the time it would take any given computer to do that can be estimated.

Bits in the public key	Mips-years needed to factor
512	30,000
2048	3×10^{20}

A million instructions per second for a year = 1 mips-year. And to give the public-key sizes a familiar scale, 2048 bits is about 616 decimal digits (10^{616}). A 1 Ghz Pentium 4 is approximately a 1000 mips machine. This means that it would take such a system 30 years to factor a 512-bit key, and 3 x 10^{17} years to factor a 2048 bit key. Another example is Pittsburgh Supercomputer Center's new TeraScale system of 64 interconnected Compaq ES40 Alpha servers, which can perform 342000mips. At this rate, it would take this system about 46 minutes to factor the 512-bit key, but would still take 8.7 x 10^{10} years to factor the 2048-bit key.

So, large keys (when used in conjunction with scrutinized algorithms) are safe against today's computing power. However, the largest key must be protected properly (via passwords, access controls, etc) or it is useless. Therefore, key management is the challenge to making your encrypted data survivable.



Cryptosystem

A combination of:

Encryption Algorithm
All of the possible plaintexts
All possible ciphertexts
All possible keys
Cryptosystem
Networked Systems Survivability

© 2002 Carnegie Mellon University

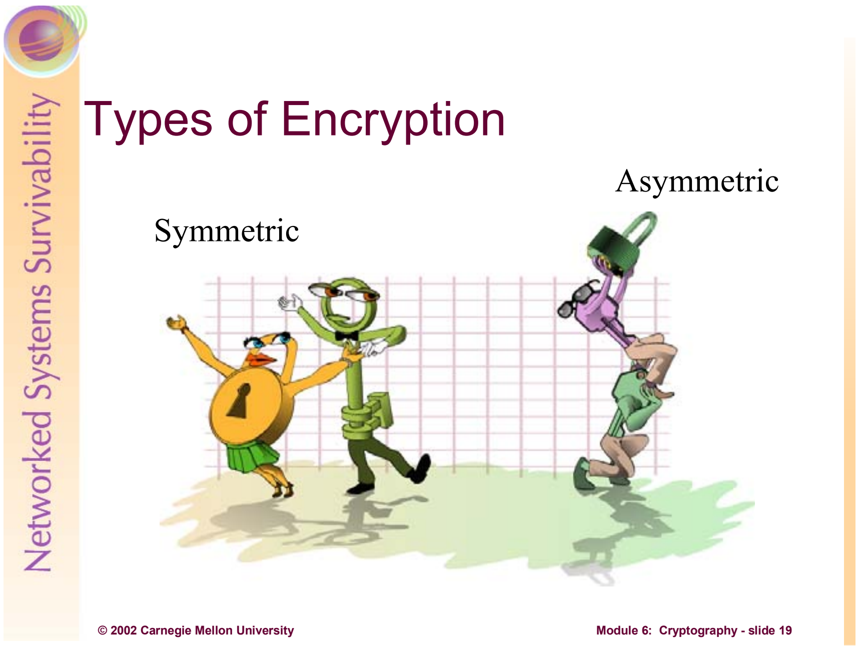
Module 6: Cryptography - slide 18

A cryptosystem is a set of transformations from a message space to a ciphertext space. A cryptosystem is composed of the combination of the algorithms used for encryption/decryption, and all of the possible plaintexts, ciphertexts, and keys. The strength of a cryptosystem depends primarily on two things – the strength of the algorithm and the number of possible keys. One can represent a cryptosystem as follows:

$E(M,K)=C$

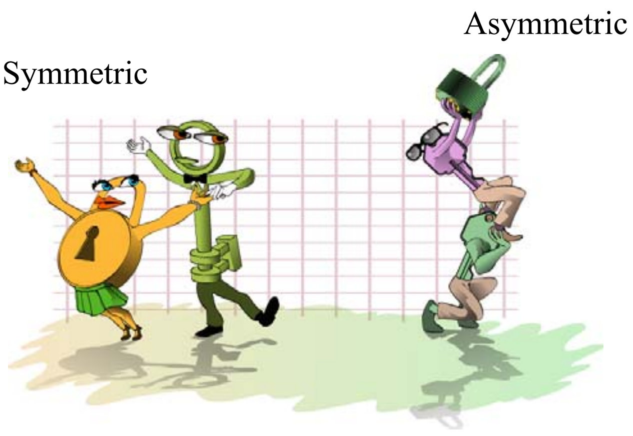
$D(C,K)=D[E(M,K),K]=M$

where M = Plaintext, C = Ciphertext, E = the encryption transformation, D = the decryption transformation, and K = the cryptovariable or key.



Networked Systems Survivability

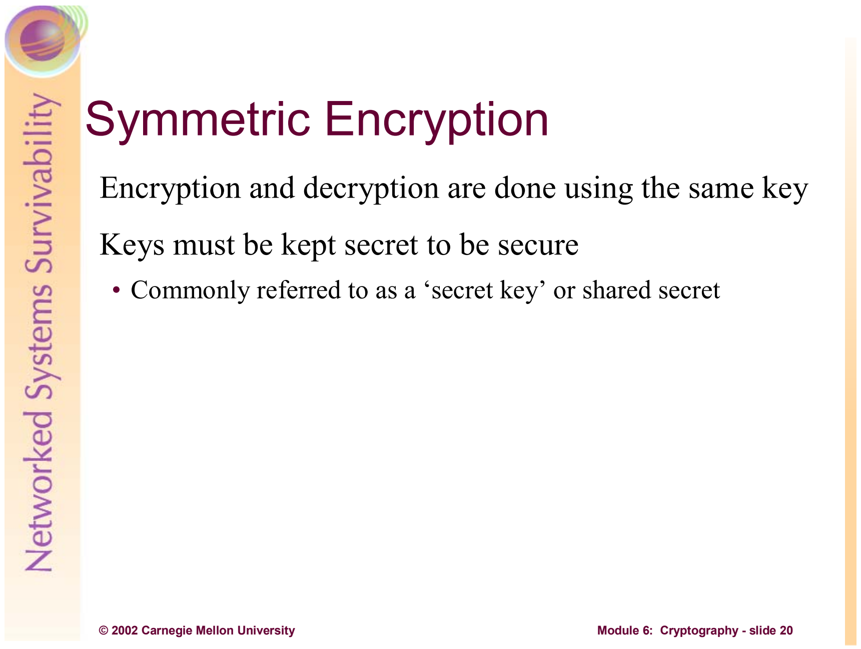
Types of Encryption



© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 19

We will review two main types of encryption – symmetric encryption (aka secret key), and asymmetric encryption (aka public/private key encryption).



Symmetric Encryption

Encryption and decryption are done using the same key

Keys must be kept secret to be secure

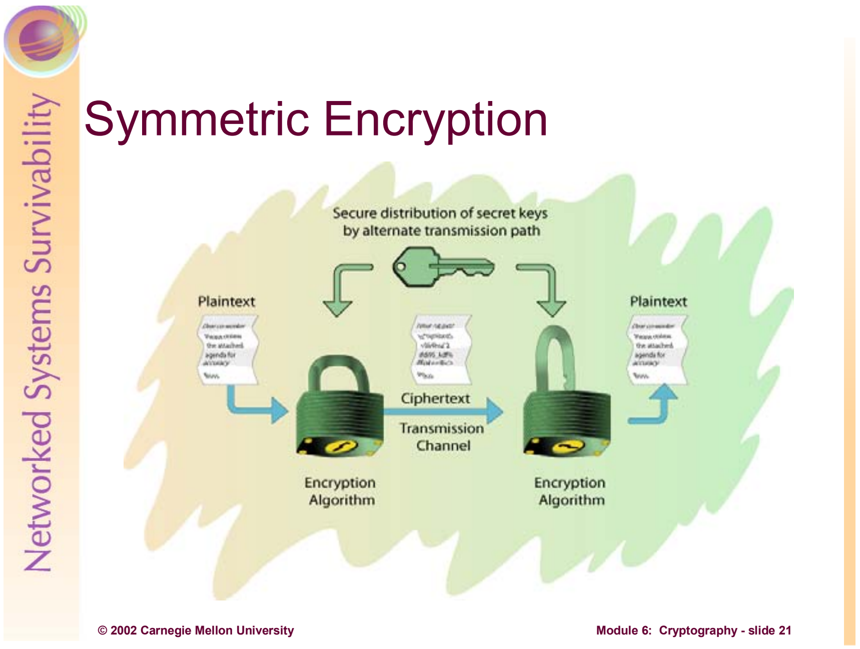
- Commonly referred to as a 'secret key' or shared secret

Networked Systems Survivability
© 2002 Carnegie Mellon University
Module 6: Cryptography - slide 20

Symmetric encryption algorithms work on one simple principle – the same exact key that is used to encrypt the plaintext is used to decrypt the ciphertext. For example, in a DES implementation, the same key must be used on the sender side to encrypt the message as will be used on the receiver's side to decrypt the message. Because this type of encryption scheme requires that the key be kept completely secret, this type of key is often referred to as a 'secret key.'

The implementation of secret key algorithms makes for some very difficult obstacles. For example, how can two individuals who have no means of sharing their secret key (i.e. geographically separated) agree on a secret key? They cannot use an insecure communications link, as that would allow eavesdroppers to learn the secret key while it is being shared and thereby use it to decrypt the messages. Could they use a trusted courier? That will be more secure, but introduces a great deal of wait-time. Now let's imagine that your organization needs to communicate securely with thousands – or hundreds of thousands – of people or other organizations. What are the issues involved in managing a large number of secret keys? Storage, destruction, renewing, etc., are all very difficult problems.

Strict symmetric key algorithms work well for some purposes, but have a number of logistical issues that prevent flexible use, and present scalability issues.

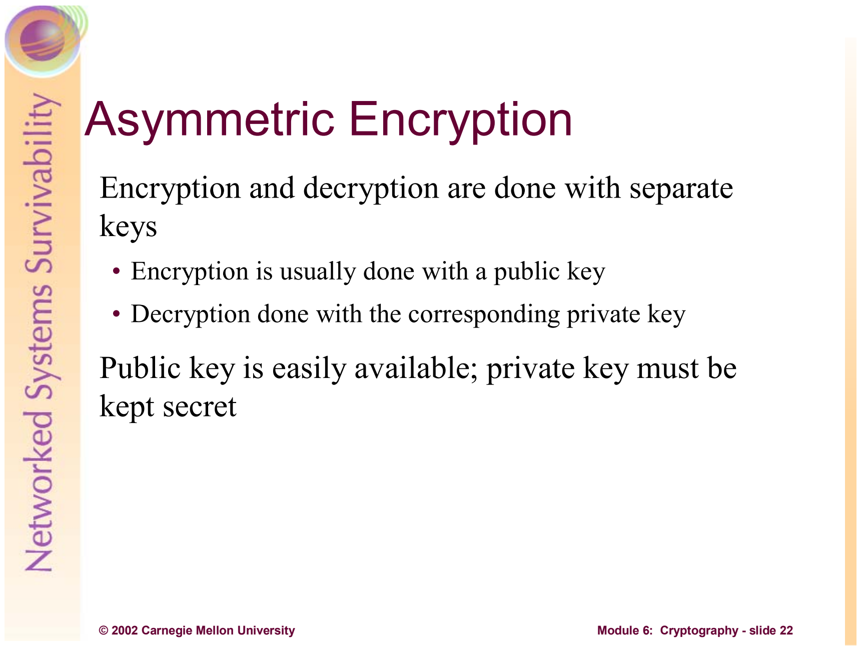

Networked Systems Survivability

Symmetric Encryption

Secure distribution of secret keys by alternate transmission path
Plaintext
Ciphertext
Transmission Channel
Encryption Algorithm
Encryption Algorithm
Plaintext

Module 6: Cryptography - slide 21

In a symmetric encryption scheme, the sender encrypts the message into ciphertext using a secret key. The message is then sent to the receiver, who must have the identical secret key. The receiver uses this secret key to decrypt the ciphertext back to the original message.



Asymmetric Encryption

Encryption and decryption are done with separate keys

- Encryption is usually done with a public key
- Decryption done with the corresponding private key

Public key is easily available; private key must be kept secret

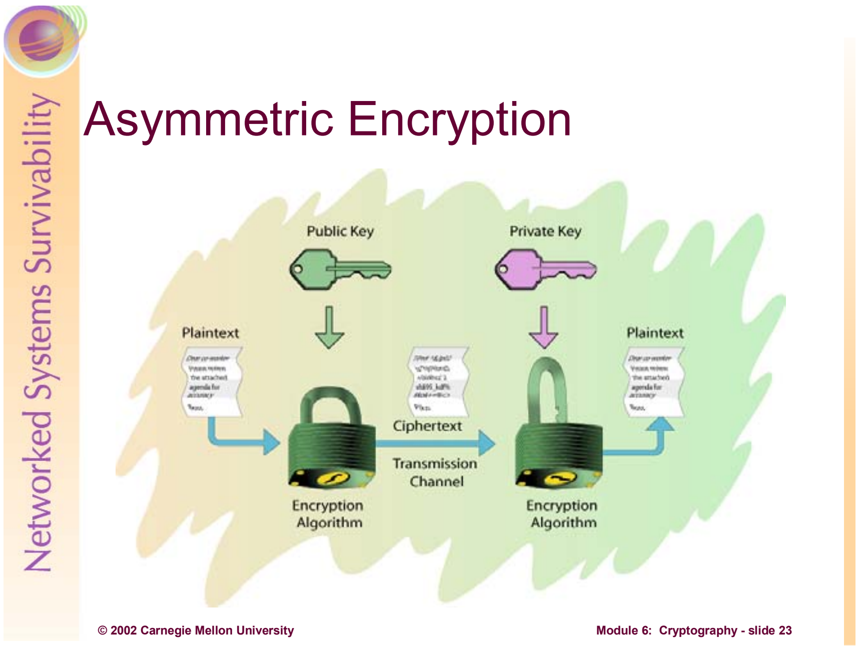
Networked Systems Survivability

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 22

In the 1970s, a few organizations began developing a new type of encryption scheme – which deviated totally from the old method of symmetric encryption. It was theorized, and later proven, that a message could be encrypted with one key and decrypted with another. Better yet – having one of these two keys did not provide a means for mathematically computing the other key. Therefore, Alice could encrypt a plaintext with one key and send it to Bob – who would use a completely different key to decrypt it. This two key encryption algorithm idea became the foundation for public key cryptography. Since the keys were not mathematically related, Alice could publish her 'public key' out to the Internet or anywhere else she wanted. Bob, or anyone else, could use Alice's public key to encrypt a message for her. And only Alice could decrypt it – with her 'private key.' This could be used to address many of the problems of symmetric encryption algorithms. Alice and Bob don't have to meet in private to agree on a secret key if they have to send encrypted information back and forth. And the organization that needs to connect securely to hundreds of thousands of people doesn't have to manage hundreds of thousands of keys now – they can simply publish their public key and allow their customers or partners access to that key, with which they can initiate a secure communications channel.

Asymmetric cryptosystems require the same care, however, that symmetric systems do when it comes to the private key. As the key in a symmetric system must be kept completely secret, so must be the private key in asymmetric cryptosystems. Imagine that Alice believes she is encrypting something that only Bob can read because she has encrypted it with Bob's public key, but Eve has somehow stolen Bob's private key. Eve would then be able to read all of those messages.

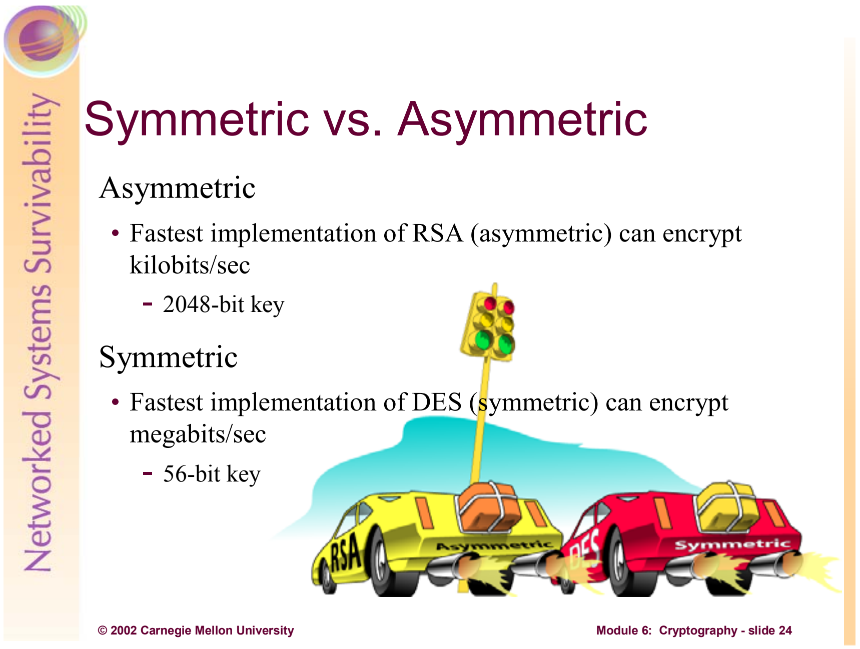


Asymmetric Encryption

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 23

The plaintext is encrypted with a public key. Not just any public key, but the public key of the person to whom the message is destined. Because of the asymmetric nature of the encryption algorithm, only the person with the corresponding private key (of which there is only one) can decrypt the message. In fact, the person who encrypted the message cannot decrypt it, as they will not have the private key of the receiver!



Symmetric vs. Asymmetric

Asymmetric

- Fastest implementation of RSA (asymmetric) can encrypt kilobits/sec
  - 2048-bit key

Symmetric

- Fastest implementation of DES (symmetric) can encrypt megabits/sec
  - 56-bit key

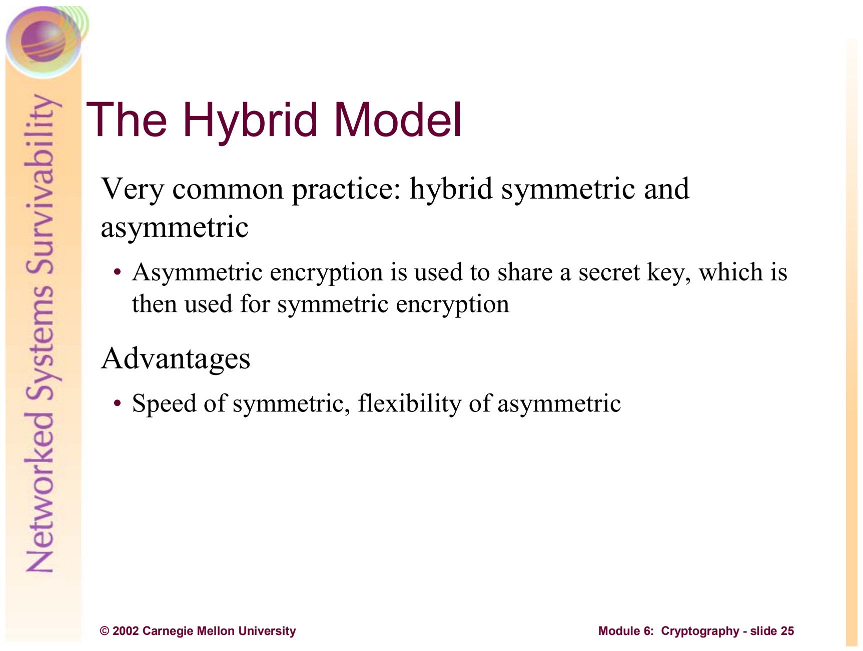
Networked Systems Survivability
RSA
Asymmetric
DES
Symmetric

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 24

Naturally, there are some trade-offs when comparing symmetric and asymmetric cryptosystems. The only major advantage of symmetric over asymmetric cryptosystems is in the speed of processing.

Why is this so? The primary reason is the size of the key for the average symmetric implementation versus the size of the average asymmetric key. Symmetric keys are significantly smaller. For example, a DES key is 56 bits, while most public and private keys are 1024 bits or more. Today, it is possible to get a 4096-bit public key. Remember, the entire key is used in the operation of the cipher on the plaintext. Therefore, large (asymmetric) key systems simply must take more time and processing power to encipher and decipher than smaller (symmetric) ones.



The Hybrid Model

Very common practice: hybrid symmetric and asymmetric

- Asymmetric encryption is used to share a secret key, which is then used for symmetric encryption

Advantages

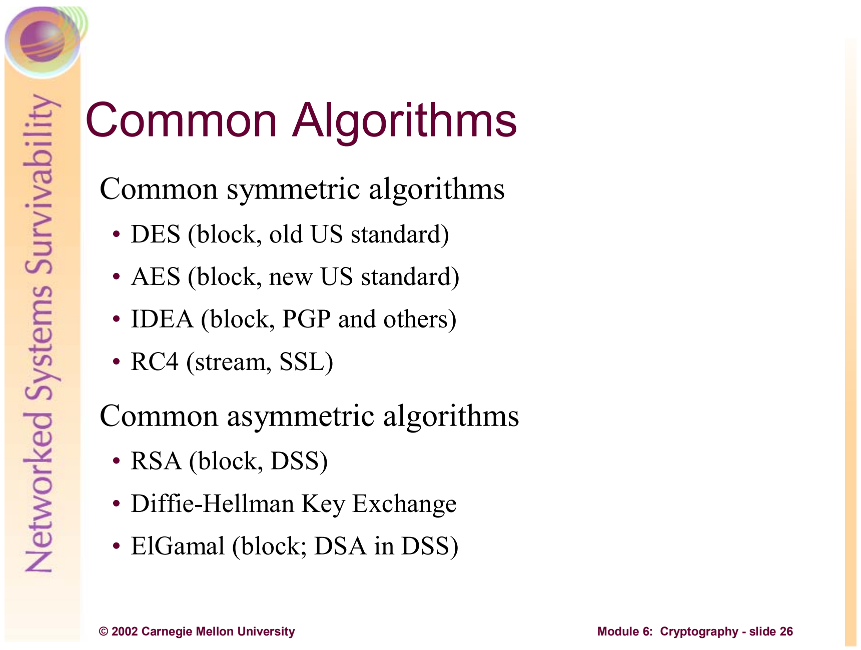
- Speed of symmetric, flexibility of asymmetric

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 25

Networked Systems Survivability

And currently, as speed is always an issue, a 'hybrid' system is very common. The asymmetric cryptosystem is used to share a symmetric (secret) key, which is then used in a symmetric encryption method. This is the best of both worlds – the ease of management and flexibility of an asymmetric cryptosystem and the speed of a symmetric cryptosystem.



Networked Systems Survivability

Common Algorithms

Common symmetric algorithms

- DES (block, old US standard)
- AES (block, new US standard)
- IDEA (block, PGP and others)
- RC4 (stream, SSL)

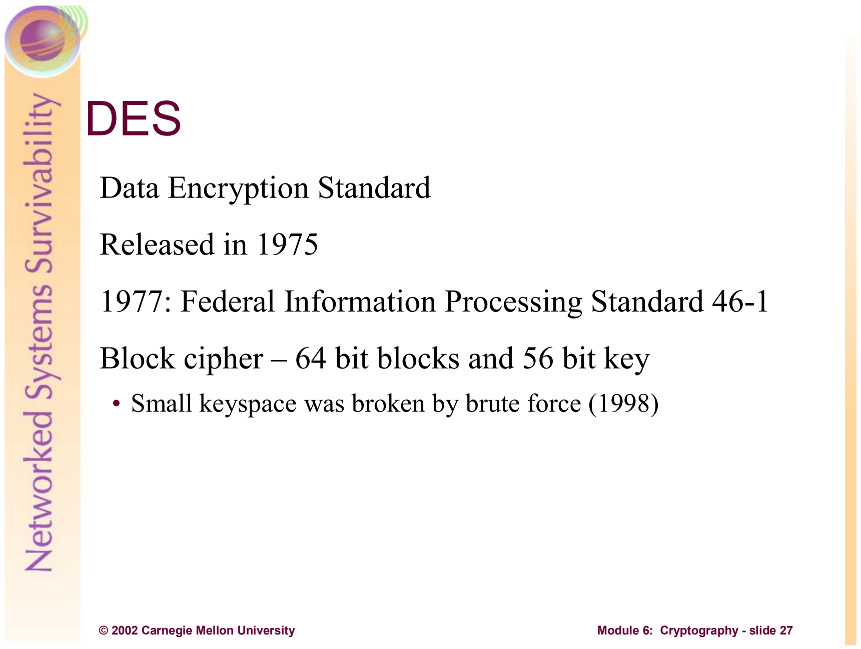
Common asymmetric algorithms

- RSA (block, DSS)
- Diffie-Hellman Key Exchange
- ElGamal (block; DSA in DSS)

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 26

We will discuss some of these algorithms in detail.



DES

Data Encryption Standard

Released in 1975

1977: Federal Information Processing Standard 46-1

Block cipher – 64 bit blocks and 56 bit key

- Small keyspace was broken by brute force (1998)

© 2002 Carnegie Mellon University Module 6: Cryptography - slide 27

The relatively small keyspace (all possible key values) as specified for DES, has been a concern for many years. It was first broken by brute force in 1998, and has since been broken in less than 24 hours through globally distributed computing strategies[1]. Because of this, other DES implementations have been utilized as well as other algorithms.

An excerpt from Federal Information Processing Standards Publication 46-2, which re-certified DES as the standard in 1993:

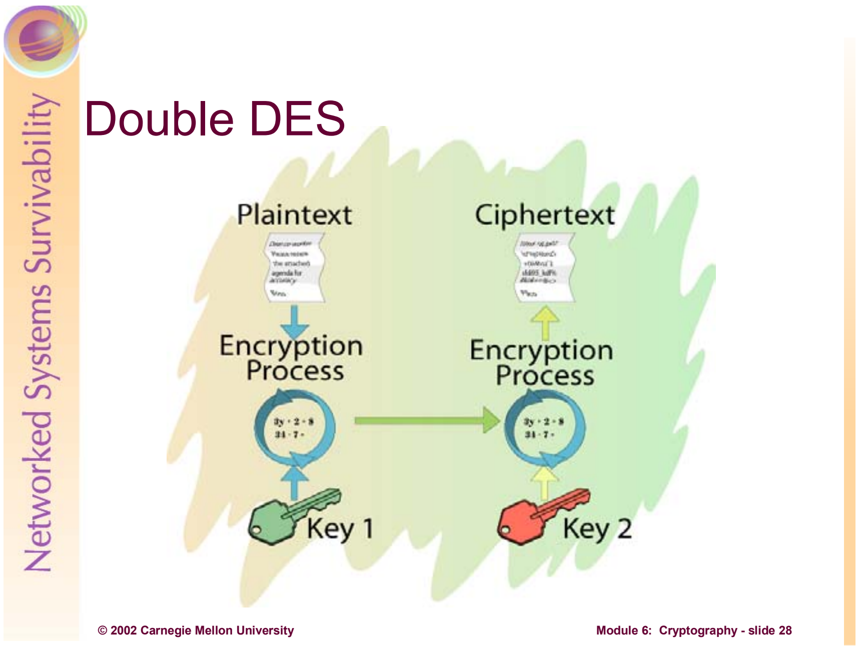
The Data Encryption Standard (DES) specifies a Federal Information Processing Standards (FIPS) approved cryptographic algorithm as required by FIPS 140-1. This publication provides a complete description of a mathematical algorithm for encrypting (enciphering) and decrypting (deciphering) binary coded information. The algorithm described in this standard specifies both enciphering and deciphering operations which are based on a binary number called a key.

A key consists of 64 binary digits ("0"s or "1"s) of which 56 bits are randomly generated and used directly by the algorithm. The other 8 bits, which are not used by the algorithm, are used for error detection. The 8 error detecting bits are set to make the parity of each 8-bit byte of the key odd, i.e., there is an odd number of "1"s in each 8-bit byte. Authorized users of encrypted computer data must have the key that was used to encipher the data in order to decrypt it. The encryption algorithm specified in this standard is commonly known among those using the standard. The unique key chosen for use in a particular application makes the results of encrypting data using the algorithm unique. Selection of a different key causes the cipher that is produced for any given set of inputs to be different. The cryptographic security of the data depends on the security provided for the key used to encipher and decipher the data.

Data can be recovered from cipher only by using exactly the same key used to encipher it. Unauthorized recipients of the cipher who know the algorithm but do not have the correct key cannot derive the original data algorithmically. However, anyone who does have the key and the algorithm can easily decipher the cipher and obtain the original data. A standard algorithm based on a secure key thus provides a basis for exchanging encrypted computer data by issuing the key used to encipher it to those authorized to have the data.

[1] http://www.distributed.net/des/

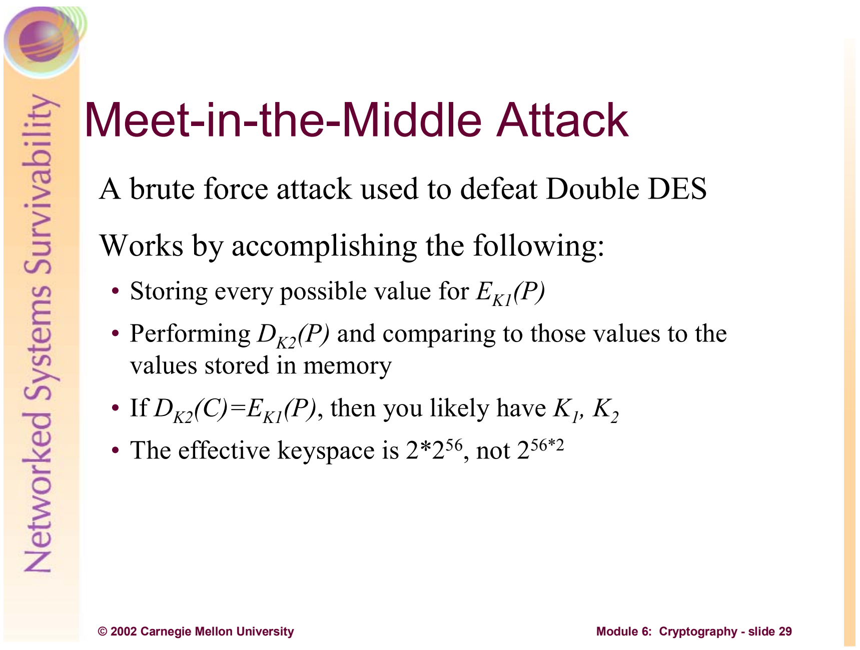
Data that is considered sensitive by the responsible authority, data that has a high value, or data that represents a high value should be cryptographically protected if it is vulnerable to unauthorized disclosure or undetected modification during transmission or while in storage. A risk analysis should be performed under the direction of a responsible authority to determine potential threats. The costs of providing cryptographic protection using this standard, as well as alternative methods of providing this protection and their respective costs, should be projected. A responsible authority then should make a decision, based on these analyses, whether or not to use cryptographic protection and this standard.


Networked Systems Survivability

Double DES

Plaintext
Ciphertext
Encryption Process
Encryption Process
Key 1
Key 2
© 2002 Carnegie Mellon University
Module 6: Cryptography - slide 28

So DES has been a worry for quite some time for its potential keyspace problem. Therefore, it became necessary as computing power increased to consider a replacement to DES with a larger keyspace. The idea of using DES with two distinct enciphering operations and two different 56-bit keys was considered, as this would yield a theoretical 112-bit keyspace.



Meet-in-the-Middle Attack

A brute force attack used to defeat Double DES

Works by accomplishing the following:

- Storing every possible value for $E_{K1}(P)$
- Performing $D_{K2}(P)$ and comparing to those values to the values stored in memory
- If $D_{K2}(C)=E_{K1}(P)$, then you likely have K_1, K_2
- The effective keyspace is $2*2^{56}$, not 2^{56*2}

Networked Systems Survivability

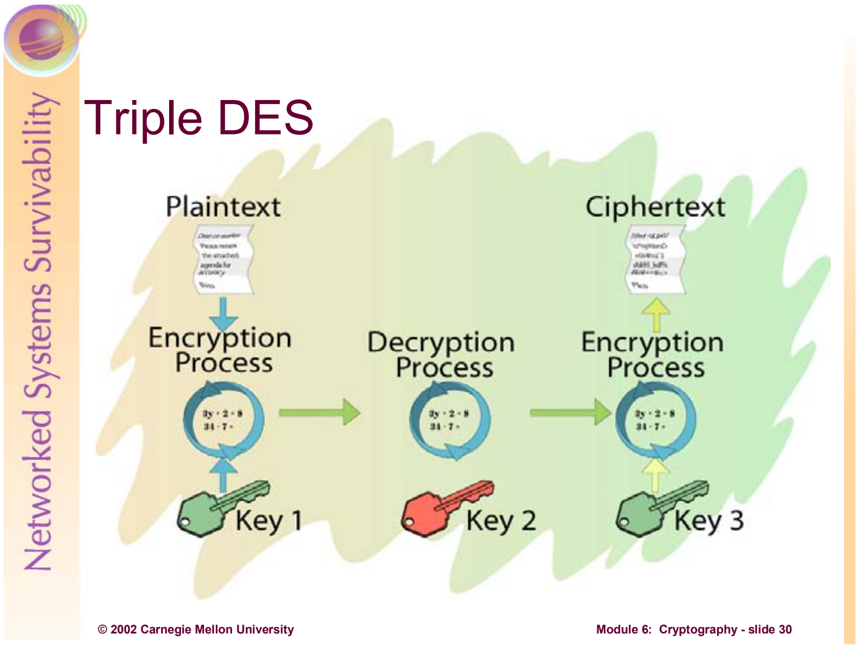
© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 29

But it's not always that easy. For an attacker to defeat this algorithm, they need only a known plaintext and the related ciphertext. Cryptanalysts proved that, through some long mathematical proofs, if someone could take a known plaintext and encrypt it with every possible 56-bit key and store that into memory (a task which is not yet easily accomplished), and subsequently take the known ciphertext from that plaintext and decrypt it with every possible key. If the decrypted text matches any of the text in memory, then there is a very good chance that the keys used for the operation are the correct keys. This is called a 'meet in the middle' attack.

So what does this do to the effective keyspace? Instead of the virtual 112-bit key, we are now left with a 57-bit keyspace – as it should only take twice the number of operations to brute force this cryptosystem than it would to do the same to the 56-bit DES cryptosystem.

So a larger keyspace does not always give stronger cryptosystems.

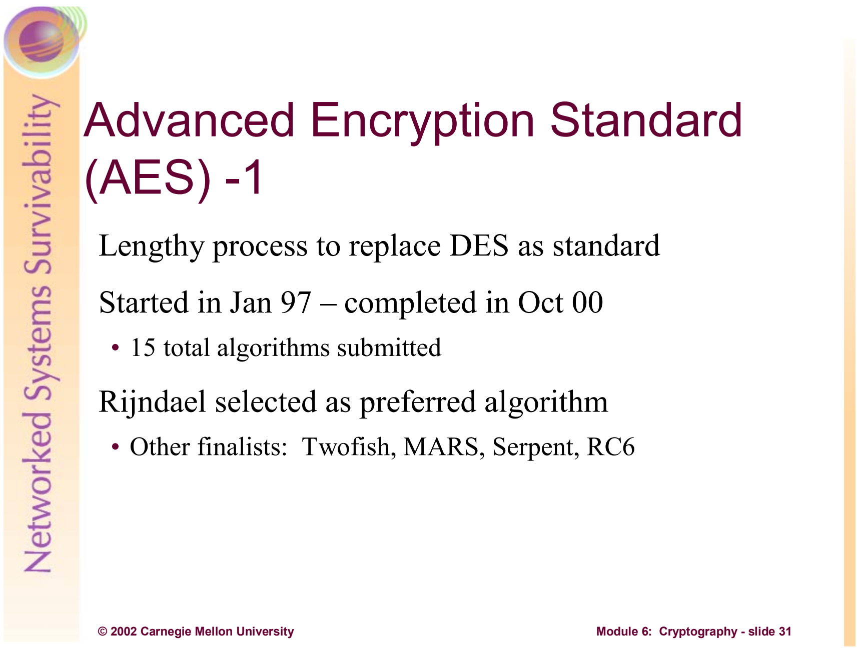


The newest implementation of DES involves three separate cipher operations – and has become known as Triple DES (or 3DES). Triple DES is not susceptible to the meet in the middle attack to which Double DES falls victim. This results from using an additional cipher process. Triple DES uses two or three different DES (56-bit) keys. In a two-key system, the plaintext is encrypted with the first key, then 'decrypted' with the second key, and encrypted again with the first key. For a three-key Triple DES system, the plaintext is encrypted with the first, decrypted with the second key, and encrypted with the third key.

It is important to understand what 'decrypting' with the second key in a Triple DES cryptosystem means. For a symmetric cryptosystem, the exact same key is needed to encrypt the plaintext as is required to decrypt the ciphertext to produce the plaintext again. In Triple DES, the ciphertext from the first encryption operation (encrypting with the first key) is run through the decryption algorithm with the second key. What is the result? Certainly not the plaintext – because, if the two keys are different, it is not possible to get the plaintext from 2 different keys. So what do we get? Junk. But that's not a bad thing. We then run that junk through the DES encryption algorithm again with the first key (or the third key – depending on our implementation) – and we get our final ciphertext. So the 'decryption' step in a Triple DES implementation does not produce the plaintext – only an interim ciphertext that will be further encrypted by the third enciphering process.

And the best part of Triple DES is the fact that it is not a 'group.' If Triple DES were a group, it would mean that, when we run the plaintext through the three enciphering processes and get our subsequent ciphertext out, we would have ciphertext that could be produced by another single key in a DES implementation. In other words, we could use a brute-force DES attack (with only 56-bit keys) and produce the plaintext with a single key. The fact that Triple DES is not a group means that the ciphertext it produces cannot be cracked this way – which further strengthens the algorithm.

The effective keyspace of a Triple DES implementation with two keys is 2^{112}, while with three keys it is 2^{168}.


Networked Systems Survivability

Advanced Encryption Standard (AES) -1

Lengthy process to replace DES as standard

Started in Jan 97 – completed in Oct 00

- 15 total algorithms submitted

Rijndael selected as preferred algorithm

- Other finalists: Twofish, MARS, Serpent, RC6

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 31

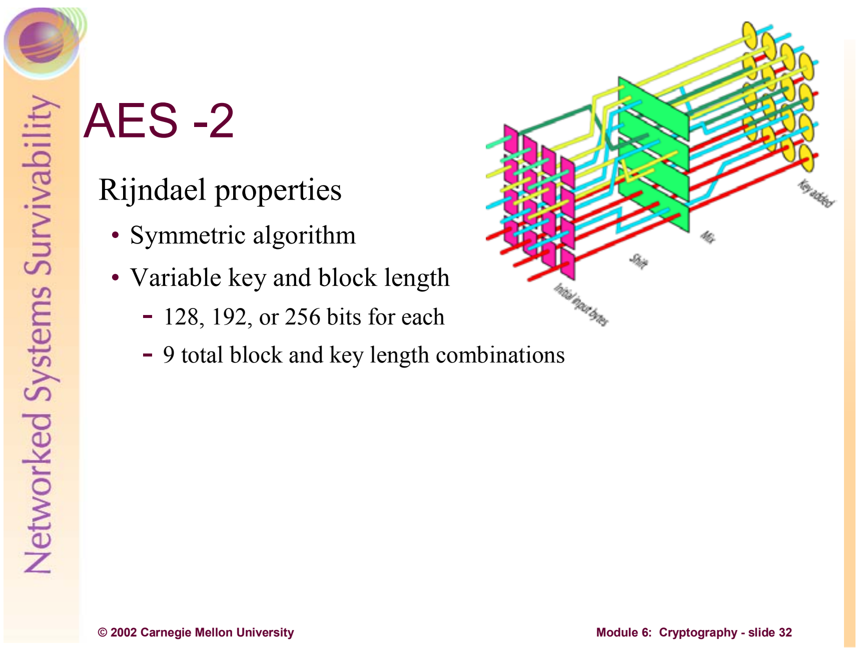
Rijndael is pronouonced 'rhine doll.'

From the draft of FIPS 140-2: SECURITY REQUIREMENTS FOR CRYPTOGRAPHIC MODULES, which identified the new encryption standard for the US Government:

1. **Name of Standard.** Advanced Encryption Standard (AES) (FIPS PUB ZZZ).
2. **Category of Standard.** Computer Security Standard, Cryptography.
3. **Explanation.** The Advanced Encryption Standard (AES) specifies a FIPS-approved cryptographic algorithm that can be used to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. Encryption converts data to an unintelligible form called ciphertext; decrypting the ciphertext converts the data back into its original form, called plaintext. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits.
4. **Approving Authority.** Secretary of Commerce.
5. **Maintenance Agency.** Department of Commerce, National Institute of Standards and Technology, Information Technology Laboratory (ITL).
6. **Applicability.** This standard may be used by Federal departments and agencies when the following conditions apply:

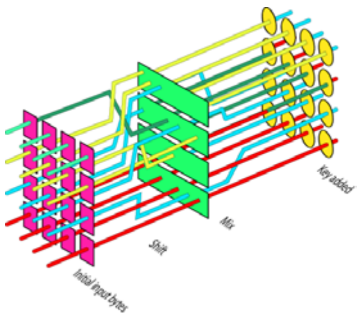
   1. An authorized official or manager responsible for data security or the security of any computer system decides that cryptographic protection is required; and
   2. The data is not classified according to the National Security Act of 1947, as amended, or the Atomic Energy Act of 1954, as amended. Federal agencies or departments that use cryptographic devices for protecting data classified according to either of these acts can use those devices for protecting sensitive data in lieu of this standard.

Other FIPS-approved cryptographic algorithms may be used in addition to, or in lieu of, this standard when implemented in accordance with FIPS 140-2, *Security Requirements for Cryptographic Modules*. In addition, this standard may be adopted and used by non-Federal Government organizations. Such use is encouraged when it provides the desired security for commercial and private organizations.



Networked Systems Survivability

AES -2



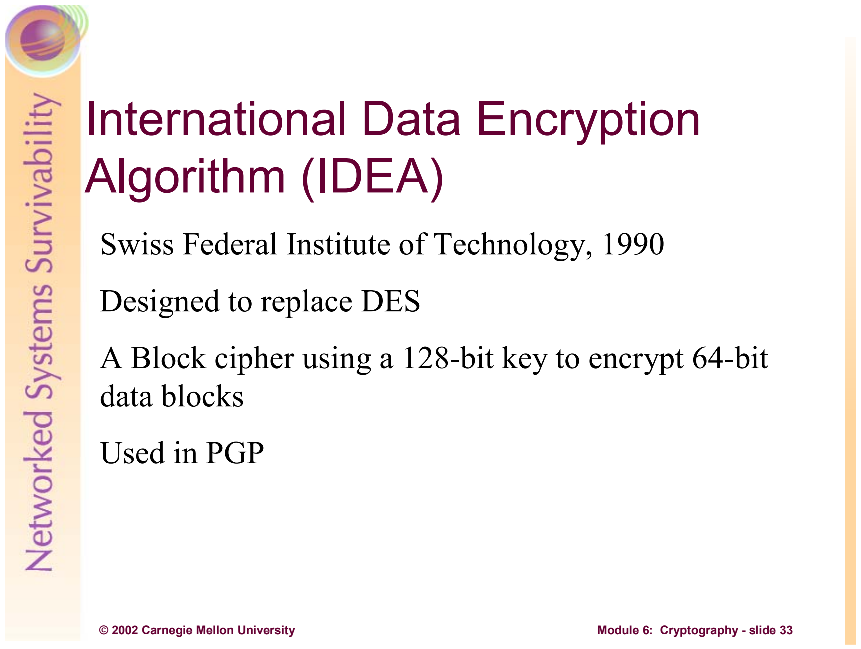
Rijndael properties

- Symmetric algorithm
- Variable key and block length
  - 128, 192, or 256 bits for each
  - 9 total block and key length combinations

Module 6: Cryptography - slide 32

This standard specifies the Rijndael algorithm, a symmetric block cipher that can process data blocks of 128 bits, using cipher keys with lengths of 128, 192, and 256 bits. While the Rijndael algorithm is capable of variable block lengths, this is not currently being included in the FIPS implementation. Rijndael was designed to handle additional block sizes and key lengths, however they are not adopted in this standard. The algorithm may be used with the three different key lengths indicated above, and therefore these different "flavors" may be referred to as "AES-128", "AES-192", and "AES-256."

http://csrc.nist.gov/encryption/aes/



International Data Encryption Algorithm (IDEA)

Swiss Federal Institute of Technology, 1990

Designed to replace DES

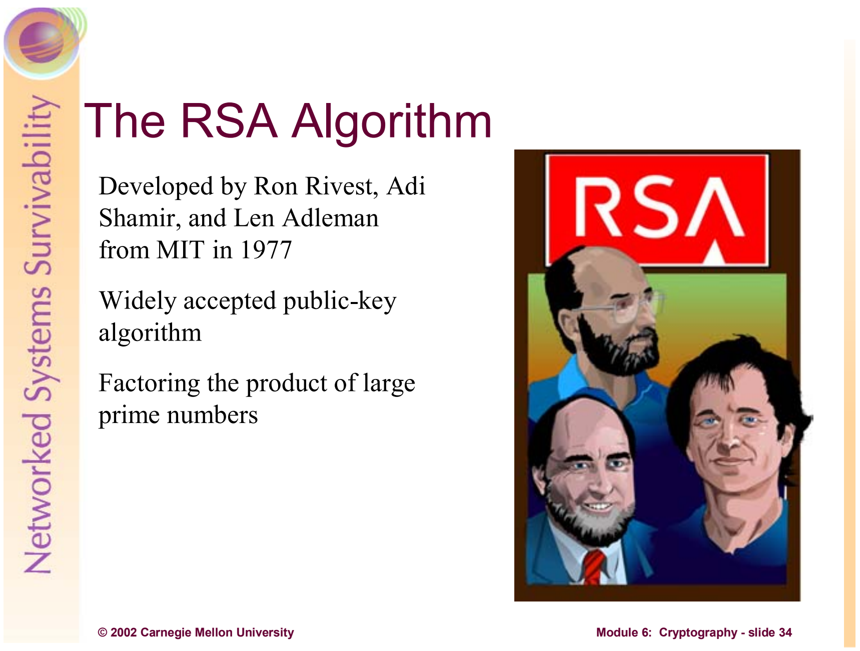
A Block cipher using a 128-bit key to encrypt 64-bit data blocks

Used in PGP

Networked Systems Survivability
© 2002 Carnegie Mellon University
Module 6: Cryptography - slide 33

IDEA is best known as a component of PGP (Pretty Good Privacy application). It is a block cipher that uses a 128-bit length key to encrypt successive 64-bit blocks of plaintext. The procedure is quite complicated using subkeys generated from the key to carry out a series of modular arithmetic and XOR operations on segments of the 64-bit plaintext block. The encryption scheme uses a total of fifty-two 16-bit subkeys. These are generated from the 128-bit subkey as follows:

- The 128-bit key is split into eight 16-bit keys which are the first eight subkeys.
- The digits of the 128-bit key are shifted 25 bits to the left to make a new key which is split into the next eight 16-bit subkeys.



Networked Systems Survivability

The RSA Algorithm

Developed by Ron Rivest, Adi Shamir, and Len Adleman from MIT in 1977

Widely accepted public-key algorithm

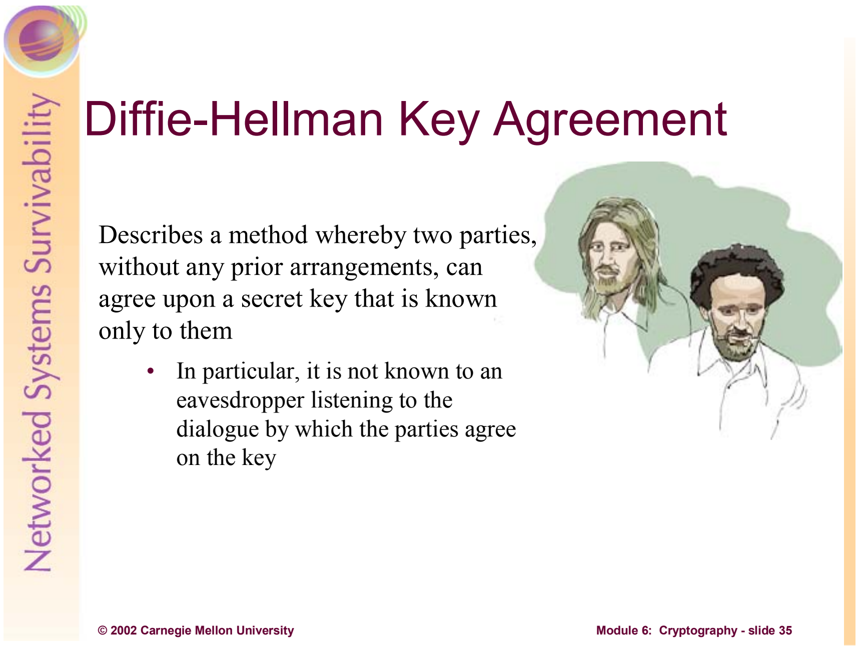
Factoring the product of large prime numbers

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 34

The challenge of public-key cryptography is developing a system in which it is impossible to determine the private key based solely on the knowledge of the public key. This can be accomplished through the use of a one-way mathematical function. With a one-way function, it is relatively easy to compute a result given some input values; however, it is extremely difficult, if not impossible, to determine the original values if you start with the result. In mathematical terms, given x, computing f(x) is easy, but given f(x), computing x is nearly impossible. The one-way function used in RSA is multiplication of prime numbers. It is easy to multiply two big prime numbers, but for most very large primes, it is extremely time-consuming to factor them (and when we say big prime numbers, we mean *big* prime numbers – with 100 to 200 digits). Some forms of public-key cryptography, including RSA, use this function by building a cryptosystem that uses two large primes to build the private key and the product of those primes to build the public key.

As a result of the expiration of their patent on 20 September 2000, RSA® Security Inc. released the RSA public key encryption algorithm into the public domain, allowing anyone to create products that incorporate their own implementation of the algorithm. See http://www.rsasecurity.com/news/pr/000906-1.html.



Diffie-Hellman Key Agreement

Describes a method whereby two parties, without any prior arrangements, can agree upon a secret key that is known only to them

- In particular, it is not known to an eavesdropper listening to the dialogue by which the parties agree on the key

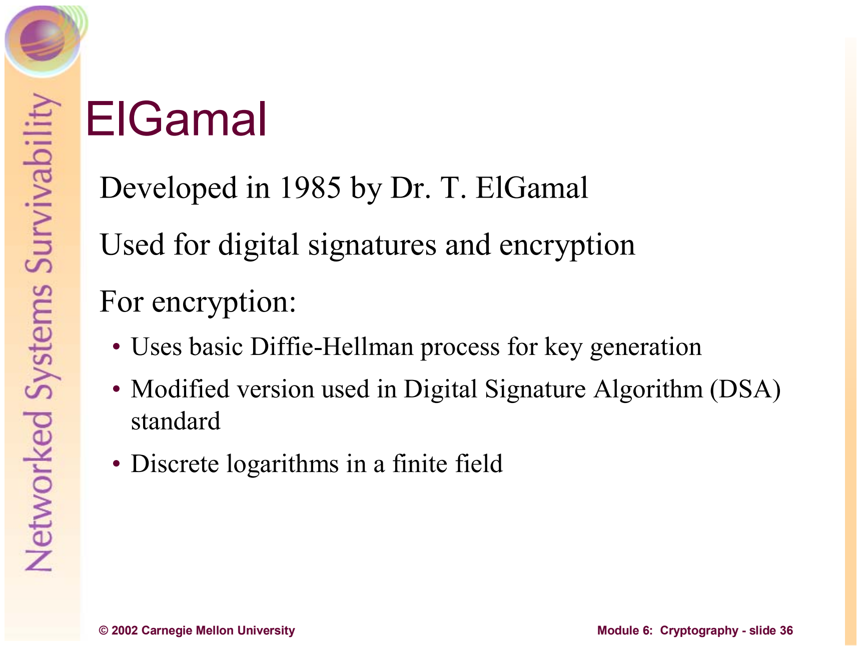
Module 6: Cryptography - slide 35

Excerpt from RFC 2631:

Diffie-Hellman key agreement requires that both the sender and recipient of a message have key pairs. By combining one's private key and the other party's public key, both parties can compute the same shared secret number. This number can then be converted into cryptographic keying material. That keying material is typically used as a key-encryption key (KEK) to encrypt (wrap) a content-encryption key (CEK) which is in turn used to encrypt the message data.

2.1. Key Agreement

The first stage of the key agreement process is to compute a shared secret number, called ZZ. When the same originator and recipient public/private key pairs are used, the same ZZ value will result. The ZZ value is then converted into a shared symmetric cryptographic key. When the originator employs a static private/public key pair, the introduction of a public random value ensures that the resulting symmetric key will be different for each key agreement.



ElGamal

Developed in 1985 by Dr. T. ElGamal

Used for digital signatures and encryption

For encryption:

- Uses basic Diffie-Hellman process for key generation
- Modified version used in Digital Signature Algorithm (DSA) standard
- Discrete logarithms in a finite field

Networked Systems Survivability

© 2002 Carnegie Mellon University Module 6: Cryptography - slide 36

ElGamal is an algorithm that can be used for either encryption of messages or for digital signatures. Dr. T. ElGamal developed the ElGamal algorithm. Much like RSA uses mathematical problems that are difficult in one direction but rather simple in the other, ElGamal does also. It is based on the discrete logarithm problem.

The ElGamal system is a public-key cryptosystem based on the discrete logarithm problem. It consists of both encryption and signature algorithms. The encryption algorithm is similar in nature to the Diffie-Hellman key agreement protocol.

The system parameters consist of a prime p and an integer g, whose powers modulo p generate a large number of elements, as in Diffie-Hellman. Alice has a private key a and a public key y, where $y=g^a$ (mod p). Suppose Bob wishes to send a message m to Alice. Bob first generates a random number k less than p. He then computes

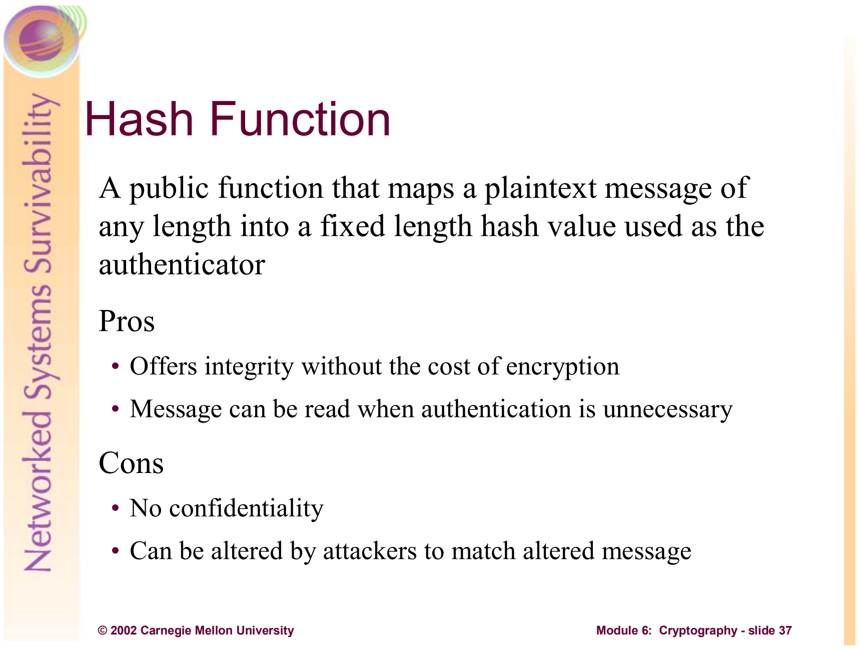
$$y_1=g^k(\text{mod } p) \text{ and } y_2=m(\text{mod } y^k)$$

Bob sends (y_1, y_2) to Alice. Upon receiving the ciphertext, Alice computes

$$m = (y_1^a \text{ mod } p) \text{ mod } y_2$$

The ElGamal signature algorithm is similar to the encryption algorithm in that the public key and private key have the same form; however, encryption is not the same as signature verification, nor is decryption the same as signature creation as in RSA. DSA is based in part on the ElGamal signature algorithm.

Analysis based on the best available algorithms for both factoring and discrete logarithms shows that RSA and ElGamal have similar security for equivalent key lengths. The main disadvantage of ElGamal is the need for randomness, and its slower speed (especially for signing). Another potential disadvantage of the ElGamal system is that message expansion by a factor of two takes place during encryption. However, such message expansion is negligible if the cryptosystem is used only for exchange of secret keys.


Networked Systems Survivability

Hash Function

A public function that maps a plaintext message of any length into a fixed length hash value used as the authenticator

Pros

- Offers integrity without the cost of encryption
- Message can be read when authentication is unnecessary

Cons

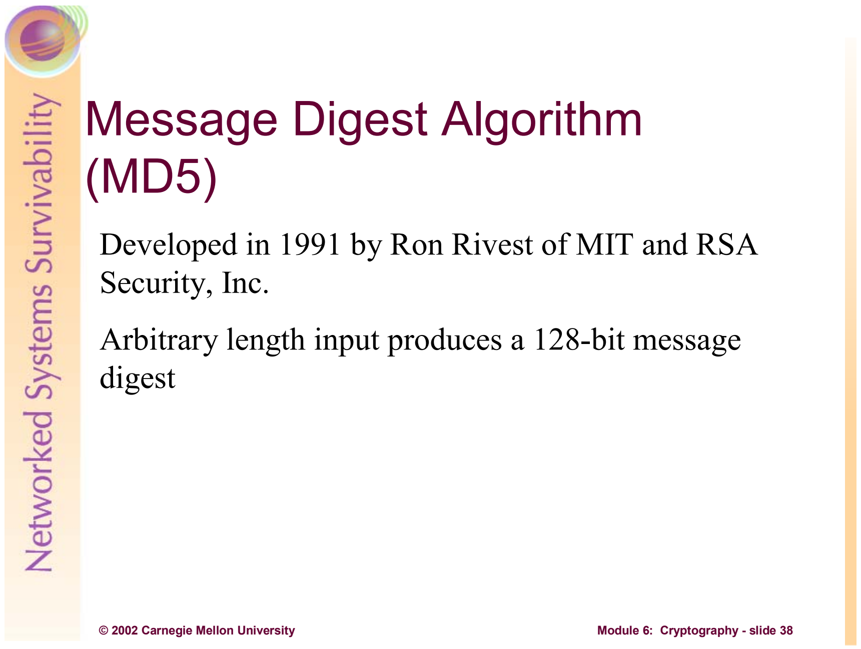
- No confidentiality
- Can be altered by attackers to match altered message

© 2002 Carnegie Mellon University Module 6: Cryptography - slide 37

A one-way hash function has many names. Among them are message digest, fingerprint, and compression function. A hash function is an algorithm that takes a variable-length string as the input and produces a fixed-length binary value (hash) as the output. The tricky part is to make this process irreversible. A good analogy would be a blender. It works well one way (blending), but it is not possible to turn the blender in reverse and recreate what was just blended. That is, knowing a hash and searching for a string that produces that hash should be very, very hard (hence the word "one-way"). It should also be very, very hard to find two arbitrary strings that produce the same hash value.

Many hash algorithms have been produced, but few have met these cryptographic requirements. Among them are MD4, MD5 and SHA-1. Ron Rivest invented MD4 and MD5 for RSA Security, Inc. These produce 128-bit hash values. SHA-1 (also known as simply SHA) was designed by NIST and NSA and produces 160-bit hash values. SHA-1 is generally considered more secure than MD4 and MD5 due to its longer hash value. To crack a 160-bit hash, you would need to try about 2^{160} (or 1.5 x 10^{48} in decimal) various strings – an enormous number by any method.

However, no matter how difficult it is to 'crack' a hash, it is relatively easy by comparison to alter an unencrypted hash to match an altered message. For example, if a message containing "Hello. How are you?" was sent with a corresponding (unencrypted) hash, it would be easy for an attacker to change the message to "Hi. How're ya?", take a hash of the new message and attach that hash to the new altered message. Therefore, while hashes are useful, they do not guarantee authenticity of transmitted message by themselves.



Message Digest Algorithm (MD5)

Developed in 1991 by Ron Rivest of MIT and RSA Security, Inc.

Arbitrary length input produces a 128-bit message digest

Networked Systems Survivability
© 2002 Carnegie Mellon University
Module 6: Cryptography - slide 38

Excerpt from RFC 1321 (which identified MD5 to the Internet community):

The algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest. The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA. The MD5 algorithm is designed to be quite fast on 32-bit machines. In addition, the MD5 algorithm does not require any large substitution tables; the algorithm can be coded quite compactly.

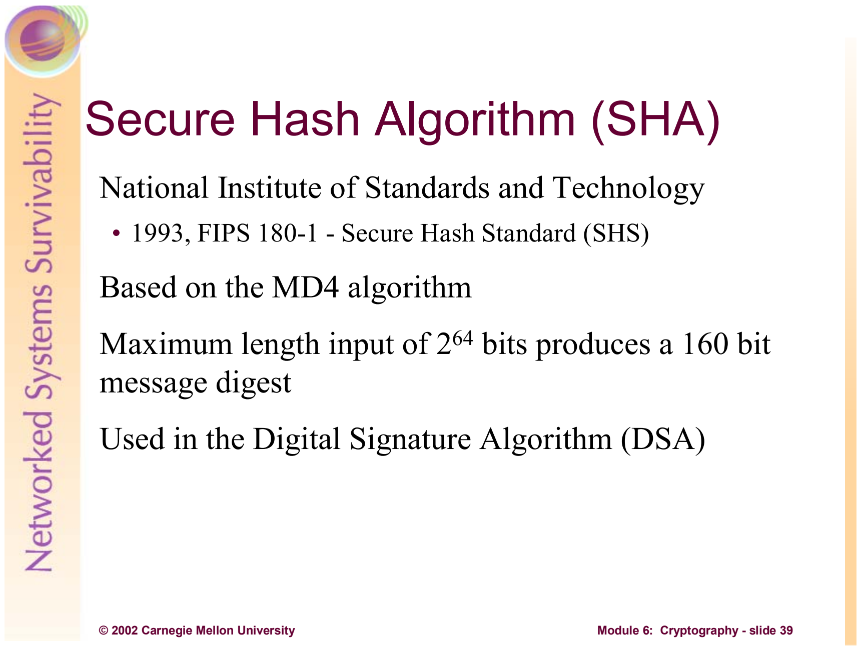
The MD5 algorithm is an extension of the MD4 message-digest algorithm. MD5 is slightly slower than MD4, but is more "conservative" in design. MD5 was designed because it was felt that MD4 was perhaps being adopted for use more quickly than justified by the existing critical review; because MD4 was designed to be exceptionally fast, it is "at the edge" in terms of risking successful cryptanalytic attack. MD5 backs off a bit, giving up a little in speed for a much greater likelihood of ultimate security.

Running an MD5 hash on the word "data" will result in a unique 128 bit message digest. Surprising as it may be, performing an MD5 hash on the entire Bible will also produce a 128-bit fingerprint.

The MD5 algorithm, which can be implemented in as little as 8 lines of PERL code, is a very simple and compact algorithm. And, because of the size of the hash, it is very secure and difficult to 'crack.' In addition, because of the complex nature of the algorithm, and because every bit of hash is dependent on every bit of the message being hashed, it is not susceptible to attacks.

Here are the 8 lines of PERL that perform an MD5 hash:

```
#!/bin/perl -iH9T4C`>_-JXF8NMS^$#)4=@<,$18%"0X4!`L0%P8*#Q4``04``04#!P`` ~JLA
@A=unpack N4C24,unpack u,$^I;@K=map{int abs 2**32*sin$_}1..64;sub L{($x=pop)
<<($n=pop)|2**$n-1&$x>>32-$n}sub M{($x=pop)-($m=1+~0)*int$x/$m}do{$l+=$r=read
STDIN,$_,64;$r++,$_.="\x80"if$r<64&&!$p++;@W=unpack V16,$_."\0"x7;$W[14]=$l*8
if$r<57;($a,$b,$c,$d)=@A;for(0..63){$a=M$b+L$A[4+4*($_>>4)+$_%4],M&{(sub{$b&$c
|$d&~$b},sub{$b&$d|$c&~$d},sub{$b^$c^$d},sub{$c^($b|~$d)})[$z=$_/16]}+$W[($A[
20+$z]+$A[24+$z]*($_%16))%16]+$K[$_]+$a;($a,$b,$c,$d)=($d,$a,$b,$c)}$v=a;for(
@A[0..3]){$_=M$_+${$v++}}}while$r>56;print unpack H32,pack V4,@A # RSA's MD5
```


Networked Systems Survivability

Secure Hash Algorithm (SHA)

National Institute of Standards and Technology
- 1993, FIPS 180-1 - Secure Hash Standard (SHS)

Based on the MD4 algorithm

Maximum length input of 2^{64} bits produces a 160 bit message digest

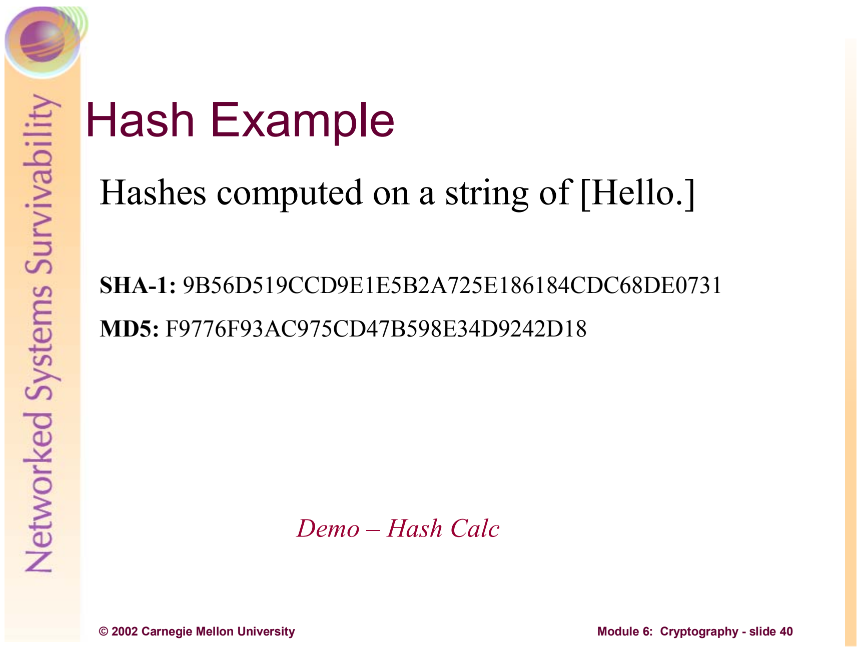
Used in the Digital Signature Algorithm (DSA)

© 2002 Carnegie Mellon University Module 6: Cryptography - slide 39

Excerpt from FIPS 180-1:

Explanation: This Standard specifies a Secure Hash Algorithm, SHA-1, for computing a condensed representation of a message or a data file. When a message of any length $< 2^{64}$ bits is input, the SHA-1 produces a 160-bit output called a message digest. The message digest can then be input to the Digital Signature Algorithm (DSA) which generates or verifies the signature for the message. Signing the message digest rather than the message often improves the efficiency of the process because the message digest is usually much smaller in size than the message. The same hash algorithm must be used by the verifier of a digital signature as was used by the creator of the digital signature.

The SHA-1 is called secure because it is computationally infeasible to find a message which corresponds to a given message digest, or to find two different messages which produce the same message digest. Any change to a message in transit will, with very high probability, result in a different message digest, and the signature will fail to verify. SHA-1 is a technical revision of SHA (FIPS 180). A circular left shift operation has been added to the specifications in section 7, line b, page 9 of FIPS 180 and its equivalent in section 8, line c, page 10 of FIPS 180. This revision improves the security provided by this standard. The SHA-1 is based on principles similar to those used by Professor Ronald L. Rivest of MIT when designing the MD4 message digest algorithm ("The MD4 Message Digest Algorithm," Advances in Cryptology - CRYPTO '90 Proceedings, Springer-Verlag, 1991, pp. 303-311), and is closely modeled after that algorithm.



Networked Systems Survivability

Hash Example

Hashes computed on a string of [Hello.]

SHA-1: 9B56D519CCD9E1E5B2A725E186184CDC68DE0731

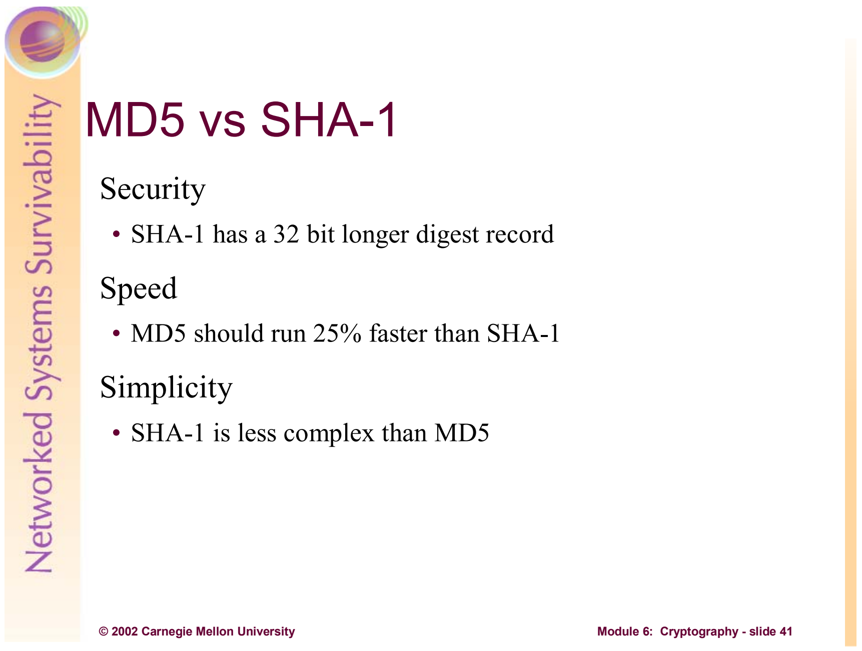
MD5: F9776F93AC975CD47B598E34D9242D18

Demo – Hash Calc

© 2002 Carnegie Mellon University Module 6: Cryptography - slide 40

Here is an example of hashing the string "Hello." using both SHA-1 and MD5. The output is represented in hexadecimal here. The SHA-1 is longer (160 bits) than the MD5 (128 bits).

The tool used to calculate this is available at:
http://www.deathstar.ch/security/downloads/files/HashCalc.exe



MD5 vs SHA-1

Security

- SHA-1 has a 32 bit longer digest record

Speed

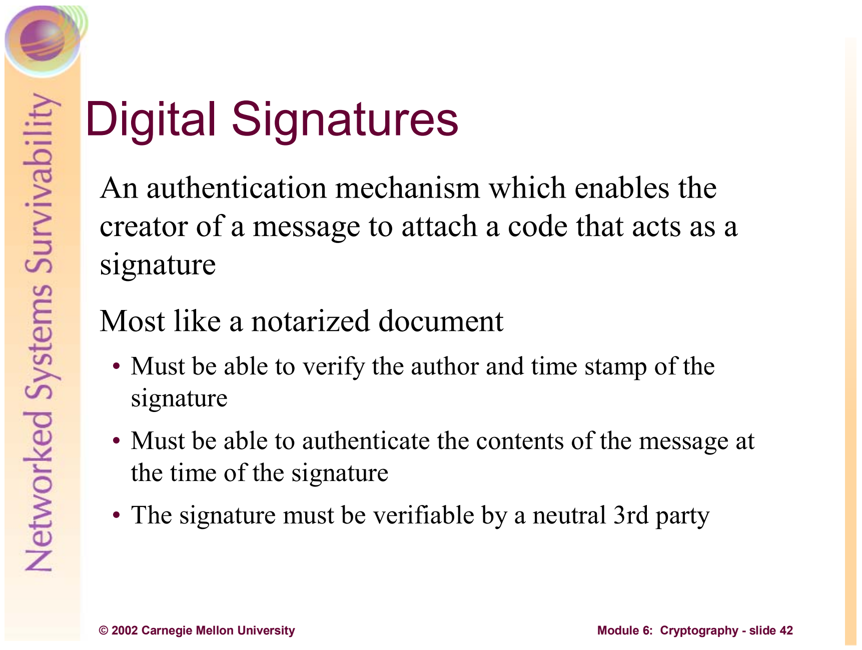
- MD5 should run 25% faster than SHA-1

Simplicity

- SHA-1 is less complex than MD5

Networked Systems Survivability
© 2002 Carnegie Mellon University
Module 6: Cryptography - slide 41

So which is the best way to hash your data? MD5 is shorter by 32 bits than SHA-1, and thus should run 25% faster. However, SHA-1 is a less complex function than is MD5. Therefore, the MD5 hash will run somewhat faster – but is *slightly* more susceptible to attack because it is shorter. Currently, both are considered excellent methods of creating unique strings to positively identify the associated messages from which the strings were 'hashed.'



Digital Signatures

An authentication mechanism which enables the creator of a message to attach a code that acts as a signature

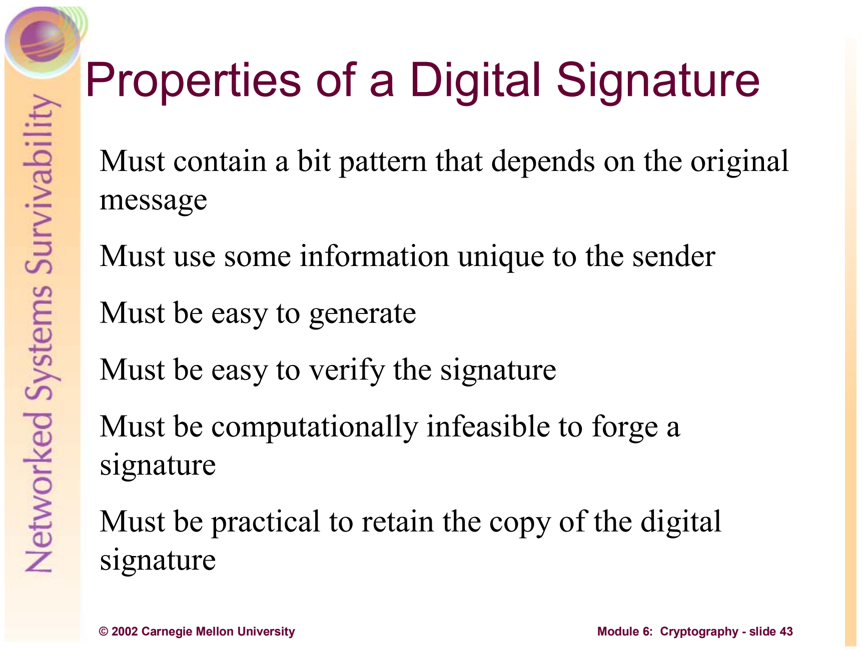
Most like a notarized document

- Must be able to verify the author and time stamp of the signature
- Must be able to authenticate the contents of the message at the time of the signature
- The signature must be verifiable by a neutral 3rd party

Networked Systems Survivability
© 2002 Carnegie Mellon University
Module 6: Cryptography - slide 42

Digital signatures provide integrity, signature assurance and non-repudiation of data. A sender of a document can digitally sign the document before transmitting it to a partner. The receiver of the document can verify who signed the document and determine whether it was modified or not since the signature generation. Digital signatures are a key technology for doing e-business. Properties like integrity of data, proof of data origin and signer authentication are very important for business documents that represent commitments such as contracts, price lists, and manifests.

In the past, business information had primarily been transmitted over private networks. Today, more and more business information exchange is done through the open Internet. As a result, the security problems associated with the Internet can affect the business data exchange. Therefore, there must be a clear and easy way to guarantee the identity and authenticity of the data that is passed for business purposes.



Properties of a Digital Signature

Must contain a bit pattern that depends on the original message

Must use some information unique to the sender

Must be easy to generate

Must be easy to verify the signature

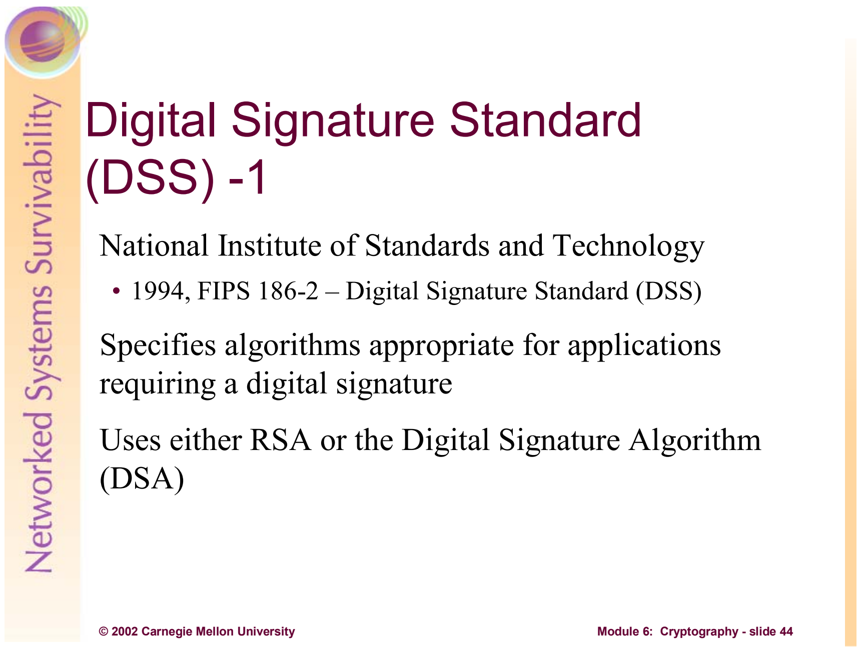
Must be computationally infeasible to forge a signature

Must be practical to retain the copy of the digital signature

Networked Systems Survivability
© 2002 Carnegie Mellon University
Module 6: Cryptography - slide 43

A Digital Signature embodies the following principles:

- Depends on the original message. This is a result of the hash function.
- Uses information unique to the sender. Accomplished through public key cryptographic means.
- Easy to generate. Hashing coupled with the encryption of the hash with a private key is a relatively easy task to accomplish.
- Easy to verify the signature. Since the public key (to decrypt the encrypted hash) and the hashing algorithm are publicly available, it is easy to perform the necessary computations to test the validity of the signature.
- Computationally infeasible to forge a signature. The private key encryption portion of a signature handles this requirement – only the sender can encrypt the hash with their private key.
- Practical to retain a copy of the signature. Since the signature is small (128 or 160 bits) it is easy to retain it with the original message.



Digital Signature Standard (DSS) -1

National Institute of Standards and Technology

- 1994, FIPS 186-2 – Digital Signature Standard (DSS)

Specifies algorithms appropriate for applications requiring a digital signature

Uses either RSA or the Digital Signature Algorithm (DSA)

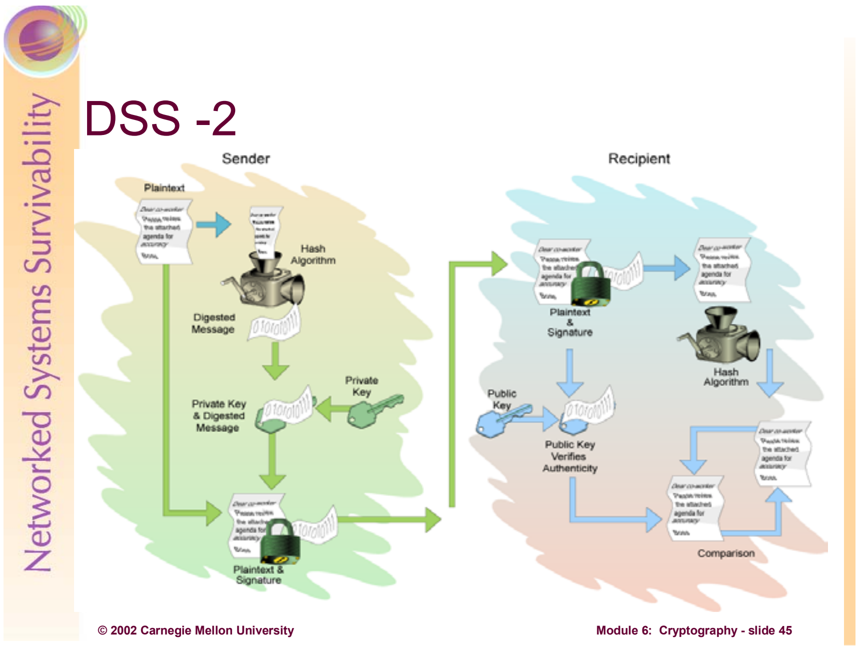
Networked Systems Survivability

Module 6: Cryptography - slide 44

Excerpt from 1994 FIPS 186-2:

Explanation: This Standard specifies a Digital Signature Algorithm (DSA) appropriate for applications requiring a digital rather than written signature. The DSA digital signature is a pair of large numbers represented in a computer as strings of binary digits. The digital signature is computed using a set of rules (i.e., the DSA) and a set of parameters such that the identity of the signatory and integrity of the data can be verified. The DSA provides the capability to generate and verify signatures. Signature generation makes use of a private key to generate a digital signature. Signature verification makes use of a public key which corresponds to, but is not the same as, the private key. Each user possesses a private and public key pair. Public keys are assumed to be known to the public in general. Private keys are never shared. Anyone can verify the signature of a user by employing that user's public key. Signature generation can be performed only by the possessor of the user's private key.

A hash function is used in the signature generation process to obtain a condensed version of data, called a message digest. The message digest is then input to the Digital Signature Algorithm (DSA) or RSA to generate the digital signature. The DSA is based on a modification of the ElGamal algorithm by Claus Schnorr in 1990. The digital signature is sent to the intended verifier along with the signed data (often called the message). The verifier of the message and signature verifies the signature by using the sender's public key. The same hash function must also be used in the verification process. The hash function is specified in a separate standard, the Secure Hash Standard (SHS), FIPS 180. Similar procedures may be used to generate and verify signatures for stored as well as transmitted data.



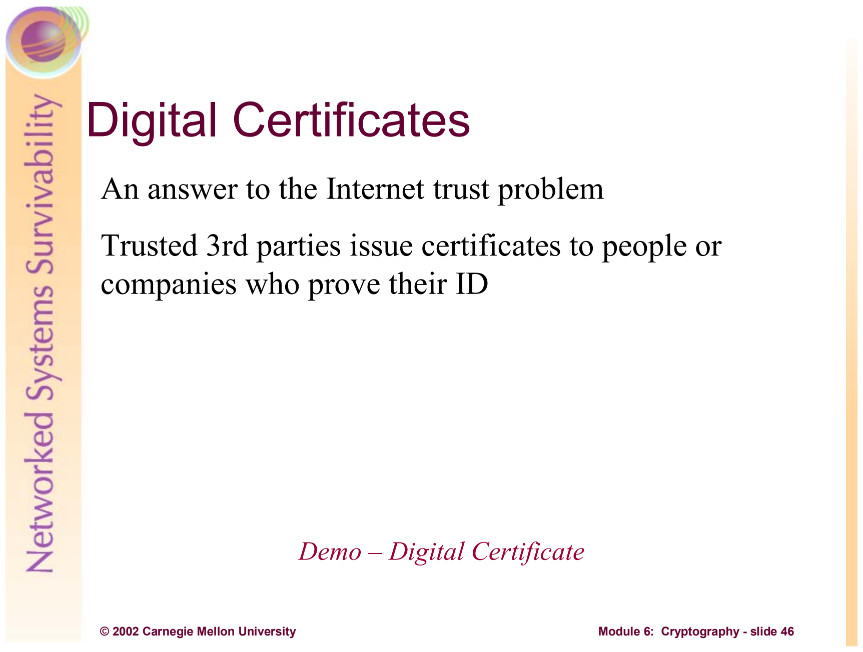
Networked Systems Survivability

DSS -2

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 45

Graphical representation of the DSS.



Digital Certificates

An answer to the Internet trust problem

Trusted 3rd parties issue certificates to people or companies who prove their ID

Demo – Digital Certificate

Networked Systems Survivability

© 2002 Carnegie Mellon University

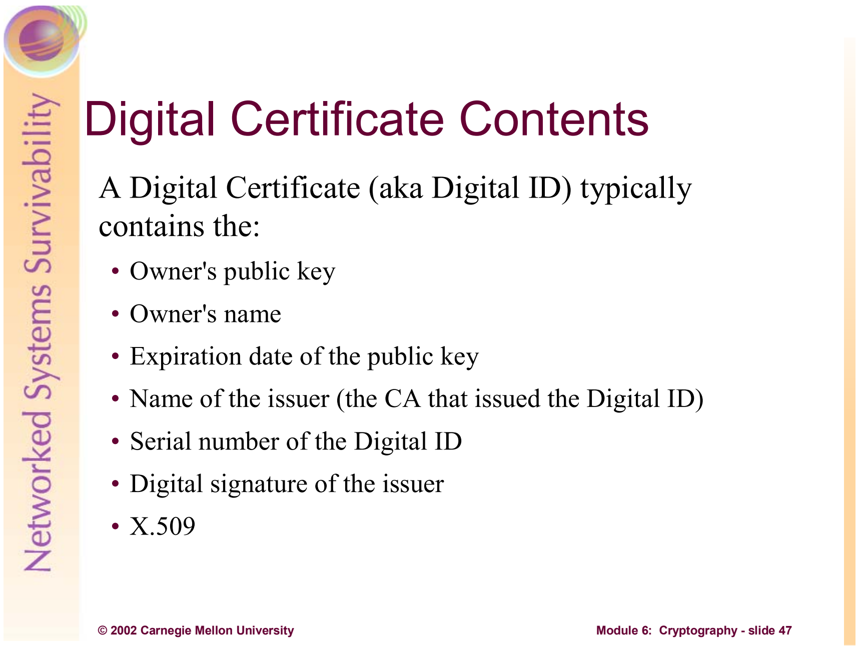
Module 6: Cryptography - slide 46

A digital certificate is an assurance provided by a third party (called a Certification Authority) that a public key does indeed belong to the purported owner. Thus it binds an identity to the public key. The identity, or subject name, may be that of a person, corporation, or some other entity such as a web server.

Digital Certificates are also called Digital IDs. Digital IDs are the electronic counterparts to driver licenses, passports, and membership cards. You can present a Digital ID electronically to prove your identity or to validate your rights to access information or services online.

Digital IDs bind an identity to a pair of electronic keys that can be used to encrypt and sign digital information. A Digital ID makes it possible to verify someone's claim that they have the right to use a given key, helping to prevent people from using phony keys to impersonate other users. Used in conjunction with encryption, Digital IDs provide a more complete security solution, assuring the identity of all parties involved in a transaction.

A Digital ID is issued by a Certification Authority (CA) and signed with the CA's private key.



Digital Certificate Contents

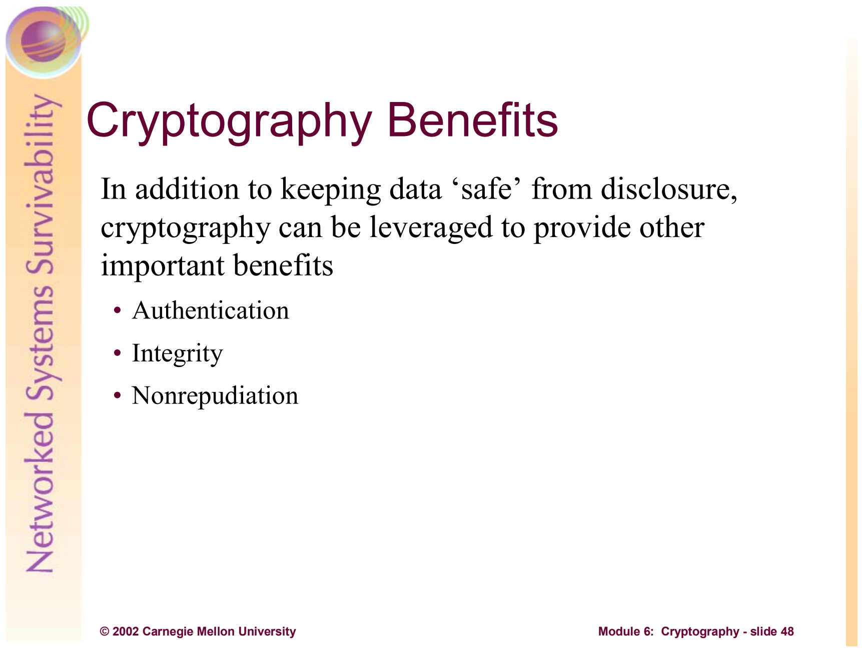
A Digital Certificate (aka Digital ID) typically contains the:

- Owner's public key
- Owner's name
- Expiration date of the public key
- Name of the issuer (the CA that issued the Digital ID)
- Serial number of the Digital ID
- Digital signature of the issuer
- X.509

Networked Systems Survivability
© 2002 Carnegie Mellon University
Module 6: Cryptography - slide 47

The certificate contains, among other fields, a serial number, the subject name, the subject's public key, and the issuer's name. The issuer, or Certificate Authority, digitally signs the certificate to provide integrity protection and assurance that the certificate is authentic.

X.509 is the most widely used standard for defining digital certificates. X.509 is actually an ITU Recommendation, which means that has not yet been officially defined or approved. As a result, companies that issue digital certificates, as well as companies that create software that use digital certificates, have implemented the standard in different ways. For example, Netscape and Microsoft both use X.509 certificates in support of Secure Socket Layer (SSL) in their Web servers and browsers. But an X.509 Certificate generated by Netscape may not be readable by Microsoft products, and vice versa.


Networked Systems Survivability

Cryptography Benefits

In addition to keeping data 'safe' from disclosure, cryptography can be leveraged to provide other important benefits

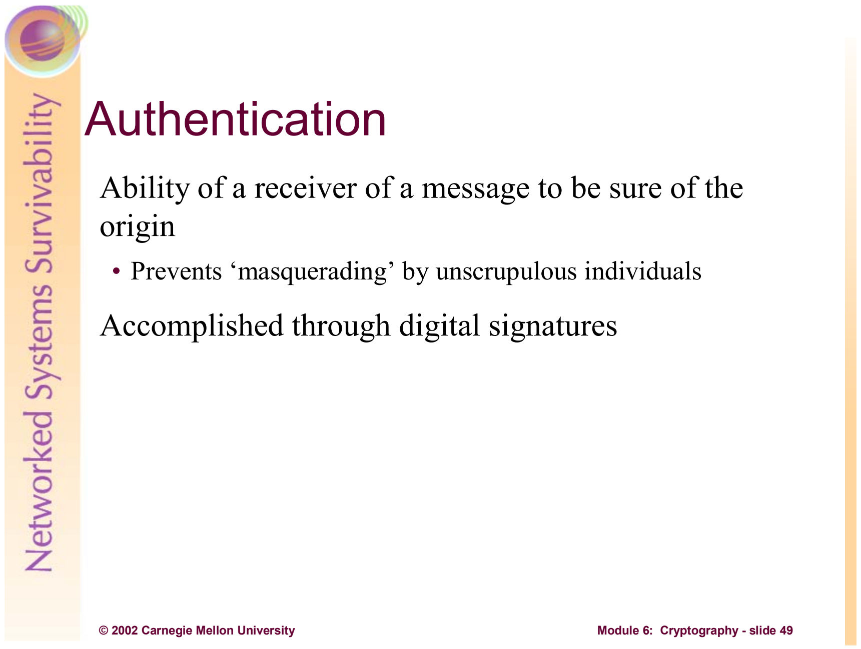
- Authentication
- Integrity
- Nonrepudiation

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 48

So cryptography can keep our data safe (provided, of course, that we keep our keys safe). But what else can it do for us? Given some of the tools that have been mentioned already, we can use cryptography and cryptographic operations to provide three of the tenants of information security:

- Authentication
- Integrity
- Nonrepudiation



Networked Systems Survivability

Authentication

Ability of a receiver of a message to be sure of the origin

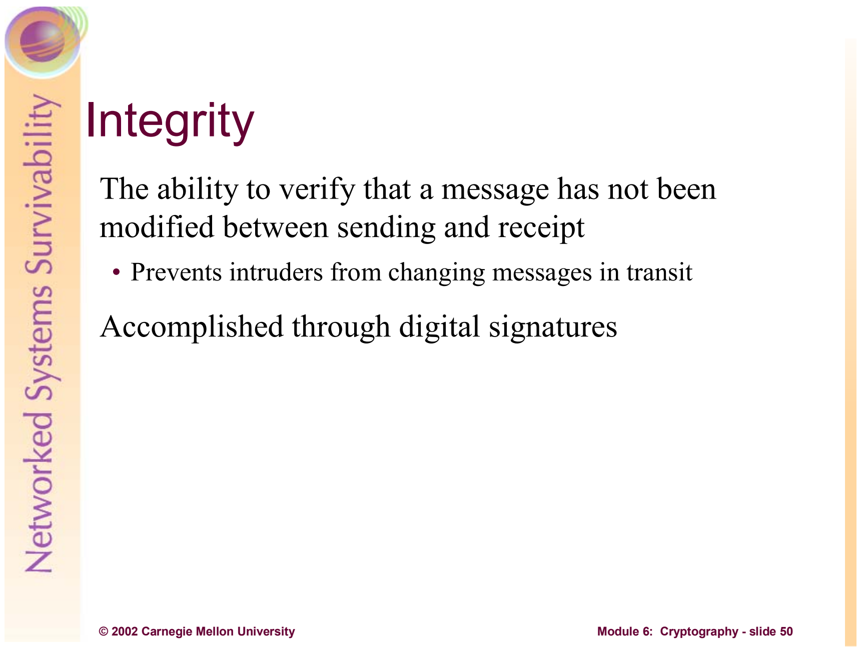
- Prevents 'masquerading' by unscrupulous individuals

Accomplished through digital signatures

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 49

Authentication is the process of determining whether someone or something is, in fact, who or what it is declared to be. The use of digital certificates, which have a digital signature attached to them and are issued and verified by a Certificate Authority (CA) as part of a public key infrastructure is considered likely to become the standard way to perform authentication on the Internet. The digital certificates are our means to provide strong authentication – even across the untrusted Internet.



Networked Systems Survivability

Integrity

The ability to verify that a message has not been modified between sending and receipt

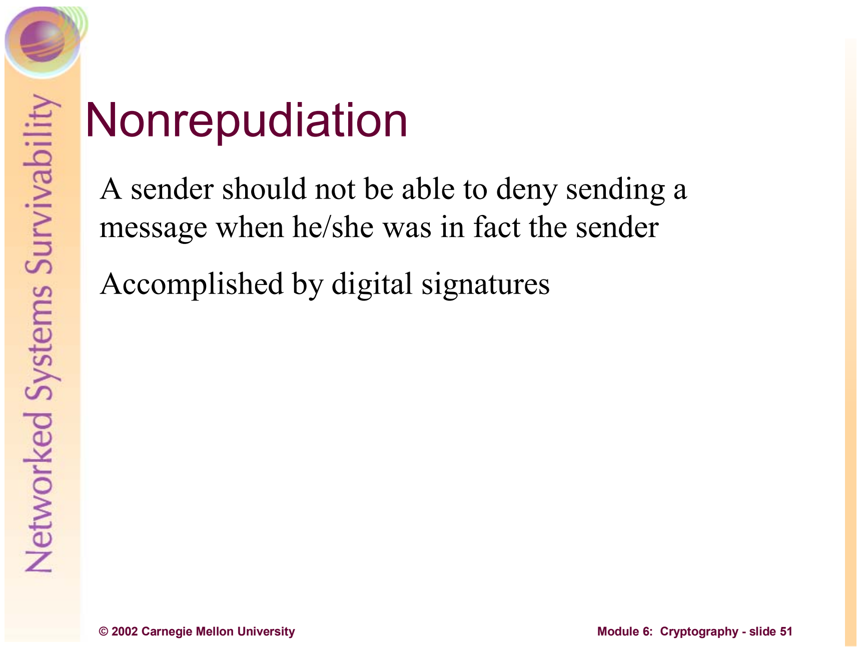
- Prevents intruders from changing messages in transit

Accomplished through digital signatures

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 50

Integrity, in terms of data and network security, is the assurance that information can only be accessed or modified by those authorized to do so. To ensure data integrity, we can use some of the principles of cryptography. We can use digital signatures, which we will discuss later, to verify that messages have not been tampered with in transit.


Networked Systems Survivability

Nonrepudiation

A sender should not be able to deny sending a message when he/she was in fact the sender

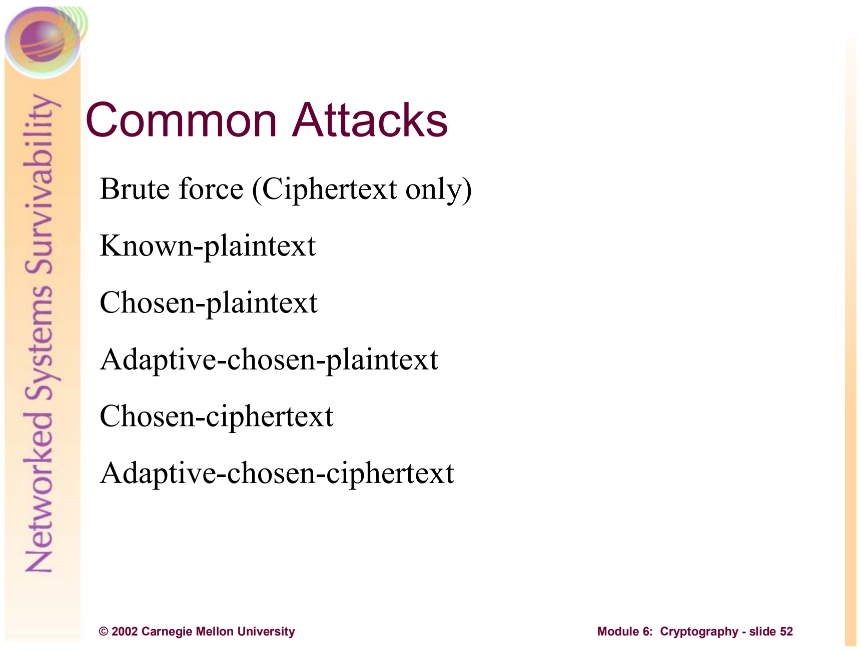
Accomplished by digital signatures

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 51

In general, nonrepudiation is the ability to ensure that a party to a contract or a communication cannot deny the authenticity of their signature on a document or the sending of a message that they originated. On the Internet, the **digital signature** is used not only to ensure that a message or document has been electronically signed by the person that purported to sign the document, but also, since a digital signature can only be created by one person, to ensure that a person cannot later deny that they furnished the signature.

Since no security technology is absolutely foolproof, some experts warn that the digital signature alone may not always guarantee nonrepudiation. It is suggested that multiple approaches be used, such as capturing unique biometric information and other data about the sender or signer that collectively would be difficult to repudiate.


Networked Systems Survivability

Common Attacks

- Brute force (Ciphertext only)
- Known-plaintext
- Chosen-plaintext
- Adaptive-chosen-plaintext
- Chosen-ciphertext
- Adaptive-chosen-ciphertext

© 2002 Carnegie Mellon University　　Module 6: Cryptography - slide 52

A **brute force** attack, also known as a **ciphertext-only** attack, is one in which the cryptanalyst obtains a sample of ciphertext, without the plaintext associated with it. This data is relatively easy to obtain in many scenarios. To decrypt this ciphertext, every possible key is tried one by one and checking to see if the plaintext that is returned is meaningful.

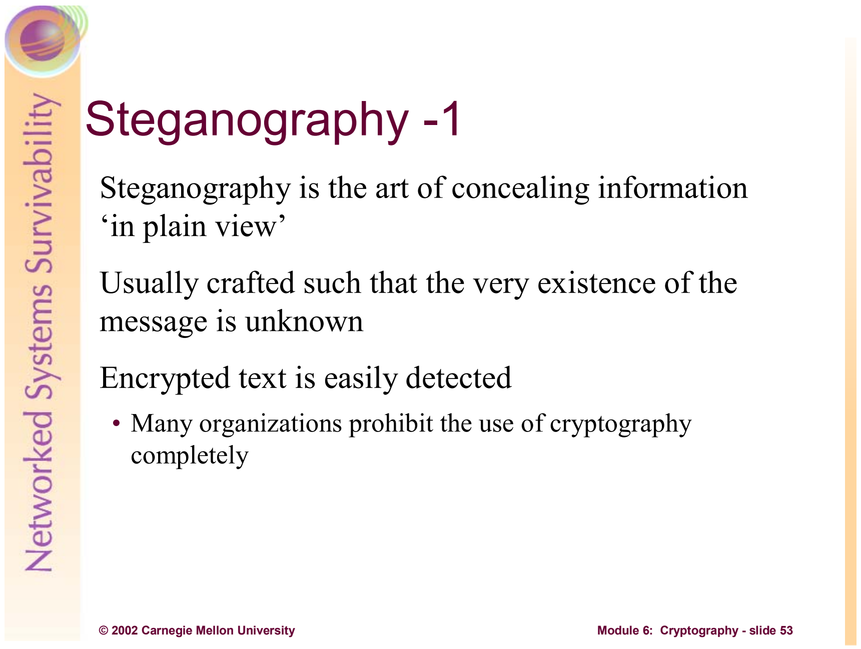
A **known-plaintext** attack is one in which the cryptanalyst obtains a sample of ciphertext and the corresponding plaintext as well.

A **chosen-plaintext** attack is one in which the cryptanalyst is able to choose a quantity of plaintext and then obtain the corresponding encrypted ciphertext.

An **adaptive-chosen-plaintext** attack is a special case of chosen-plaintext attack in which the cryptanalyst is able to choose plaintext samples dynamically, and alter his or her choices based on the results of previous encryptions.

A **chosen-ciphertext** attack is one in which cryptanalyst may choose a piece of ciphertext and attempt to obtain the corresponding decrypted plaintext. This type of attack is generally most applicable to public-key cryptosystems.

An **adaptive-chosen-ciphertext** is the adaptive version of the above attack. A cryptanalyst can mount an attack of this type in a scenario in which he has free use of a piece of decryption hardware, but is unable to extract the decryption key from it.


Networked Systems Survivability

Steganography -1

Steganography is the art of concealing information 'in plain view'

Usually crafted such that the very existence of the message is unknown

Encrypted text is easily detected

- Many organizations prohibit the use of cryptography completely

© 2002 Carnegie Mellon University Module 6: Cryptography - slide 53

Cryptography is certainly not the only way to hide or conceal information. One way that has been growing in popularity lately is steganography.

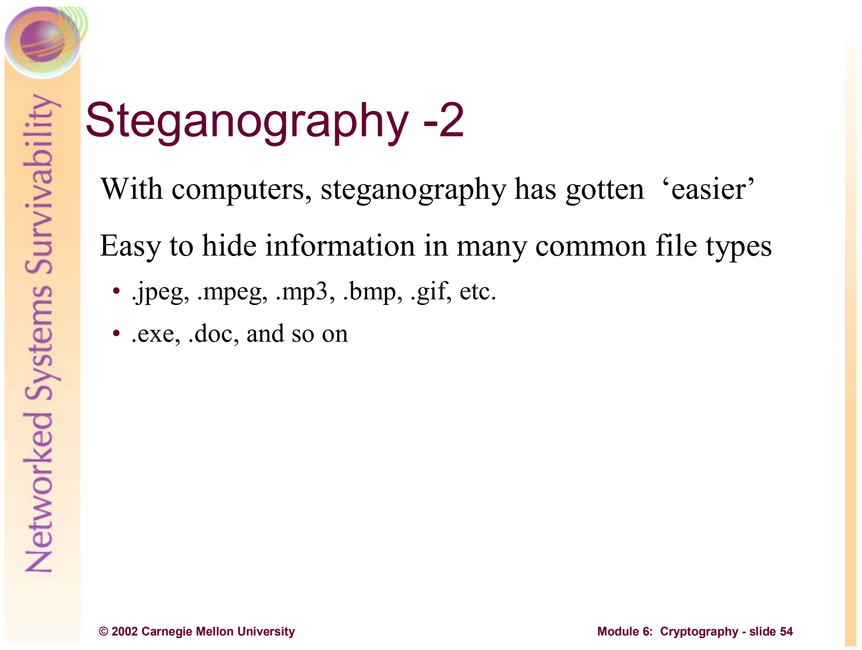
In his history of the Persian Wars, Herodotus tells of a messenger who shaved his head and allowed a secret message to be tattooed on his scalp. He waited until his hair grew back. Then he journeyed to where the recipient awaited him and shaved his head again. The message was revealed. It was history's first use of steganography.

There are many steganographic examples from recent times. Possibly most noticeable was the case of Navy Captain Jeremiah Denton Jr. When captured by enemy forces in Vietnam, Captain Denton was subjected to seven days and nights of interrogation and torture – many times becoming unconscious as a result of the torture. When he was put in front of the camera to make a statement to the media (a Japanese reporter), Captain Denton made the following statement:

"Whatever the position of my government is, I believe in it, yes sir…I am a member of that government and it is my job to support it, and I will as long as I live."

This was reviewed by the Vietnamese, and found to be innocuous enough to pass out of the country and to be given to the international press. What the Vietnamese failed to notice was that Captain Denton blinked Morse code for 'TORTURE.'

So why would we want to use computers to aid us in steganography when cryptography is so prevalent now? Traditional ciphers give themselves away because they appear to be gibberish, and anyone who intercepts them will at least have reason to suspect the sender's motives. But steganography hides the message inside an ordinary-looking object such as a digitized photo or sound file. It is "visible" only to someone who knows it's there. Thus, photo agencies use it to create digital "watermarks" for their pictures. The U.S. Customs Service has warned that some web sites use it to mask pornographic images.



Steganography -2

With computers, steganography has gotten ‘easier’

Easy to hide information in many common file types

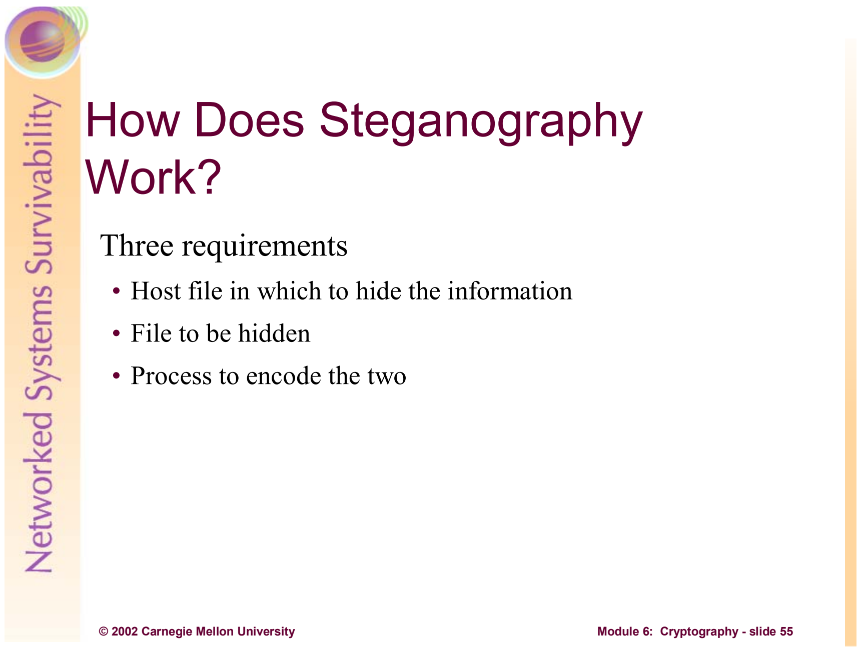
- .jpeg, .mpeg, .mp3, .bmp, .gif, etc.
- .exe, .doc, and so on

Networked Systems Survivability

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 54

Digital images are good vehicles for steganography. All images contain redundant data: information as to color, for example, that is present but unnecessary for the picture to be seen and understood. This enables the senders of a secret message to substitute digitized text for some of the redundant pixels in a photo, for example. Or it could be a verbal message; when transmitting information, computers treat sound, image and text files all the same. They are all parts of bitstreams.



How Does Steganography Work?

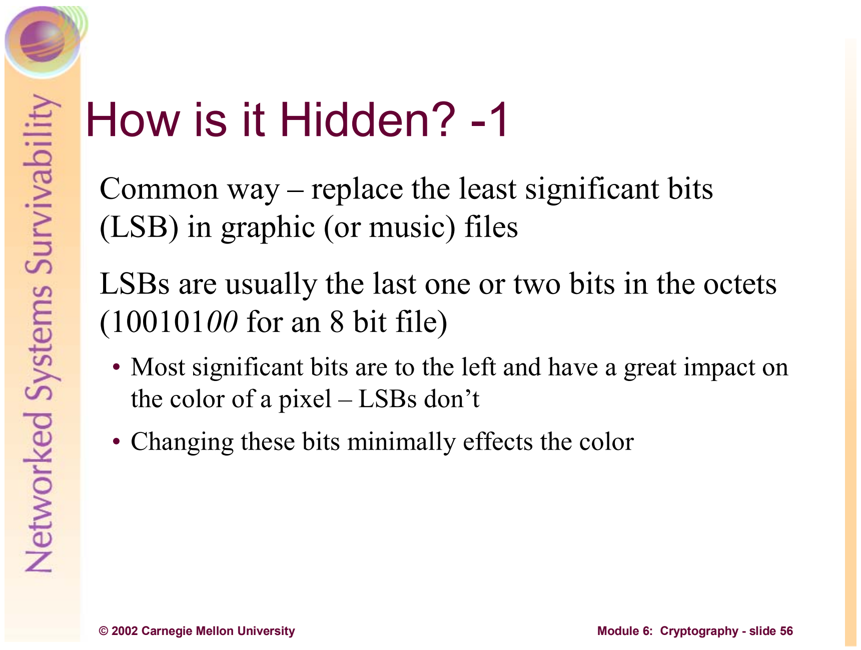
Three requirements

- Host file in which to hide the information
- File to be hidden
- Process to encode the two

Networked Systems Survivability

In order to perform steganography, one must have three requirements on place:

- A host file (in which to hide the information). This host file can be of almost any format, but the most common formats are those that represent digital renderings of analog data. Some examples include picture files, audio files, and video files. These are excellent places to hide information – and will be discussed later.
- The actual file which will be hidden. This can be of any format, since it must be digital (as it is stored in a computer) and can be combined with the host file.
- A process to encode the two files. This will be some algorithm to take the file to be hidden and encode it in the host. There are many ways – the most common being Least Significant Bit (LSB) encoding, which will be covered later.



How is it Hidden? -1

Common way – replace the least significant bits (LSB) in graphic (or music) files

LSBs are usually the last one or two bits in the octets (100101*00* for an 8 bit file)

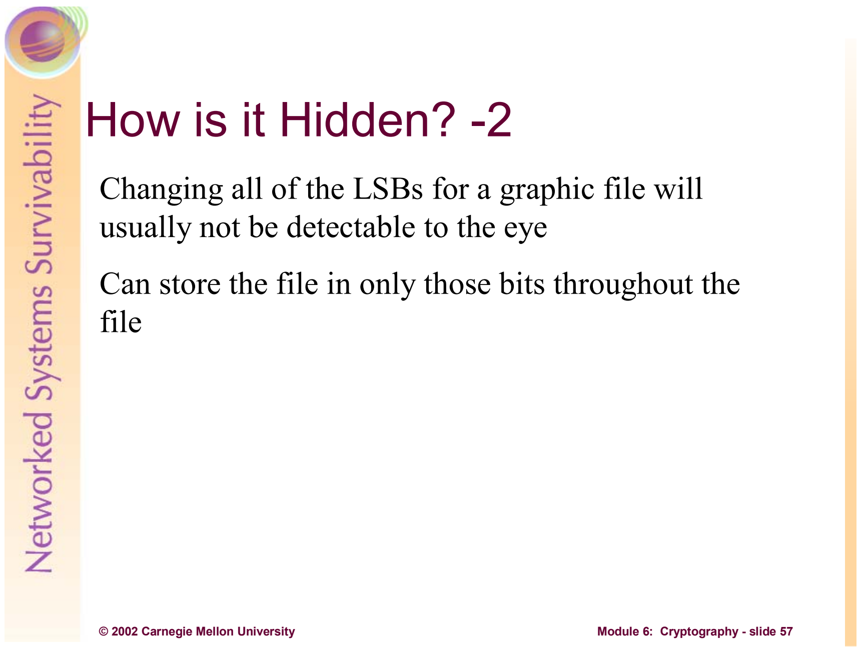
- Most significant bits are to the left and have a great impact on the color of a pixel – LSBs don't
- Changing these bits minimally effects the color

Networked Systems Survivability

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 56

Digital steganography takes a number of forms, but most of them are variations on the most popular technique. It's called "least significant bit" steganography, and to understand how it works, you have to understand the digitizing of analog media work. Consider the pixels on the computer screen that make up a photograph, or the 'pieces' of sound that blend together to form a song as it streams from a CD player. Each pixel and each piece of sound are recorded as a small number of bits – ranging from 18 to 32 (or possibly more). Most of the bits specify important information about the color or sound of the sensory blip they represent, but a few stand for nuances the average eye or ear won't even pick up. These latter, the so-called least significant bits (LSBs), are effectively indistinguishable from noise. And since encrypted data is sufficiently random, it is also indistinguishable from noise. Therefore, when the LSBs of a favorite picture or sound file have been replaced with encrypted data, it may be very difficult to distinguish that encrypted data from the noise introduced into the file – either through transmission or when the analog item (photo or sound) was converted to digital.



How is it Hidden? -2

Changing all of the LSBs for a graphic file will usually not be detectable to the eye

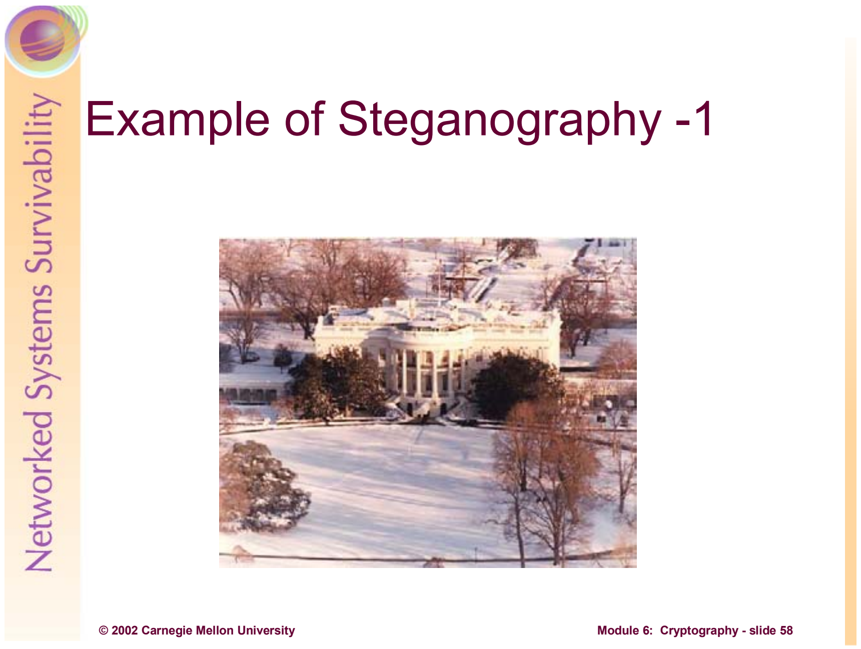
Can store the file in only those bits throughout the file

Networked Systems Survivability

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 57

So let's consider the .jpeg example. Jpeg pictures are 24 bit files – which means that each pixel is defined by a 24-bit word. If we can remove one or two of the LSBs, and leave the 22 other bits in tact, we can implant our data in place of the 1s and 0s we stripped out. We can further obscure our information by passing it through an encryption algorithm prior to sticking the bits in the place of the LSBs we stripped out. There are many tools that allow us to do exactly this – with jpeg, mp3, mpeg, and other common file formats.



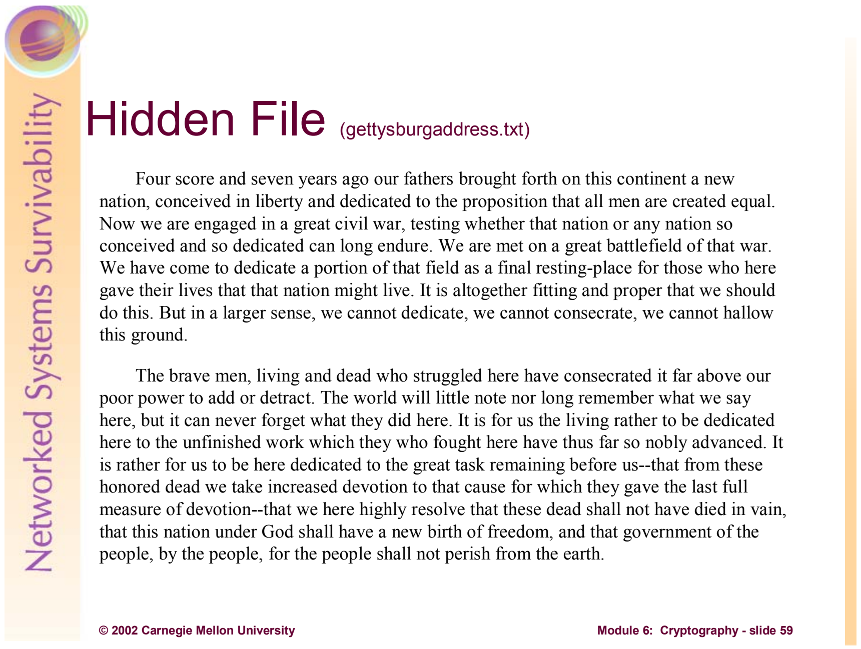
Networked Systems Survivability

Example of Steganography -1

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 58

So let's look at an example of steganography. This is a picture obtained from www.whitehouse.gov. It is a 60k jpeg file. This is the picture without the message attached.



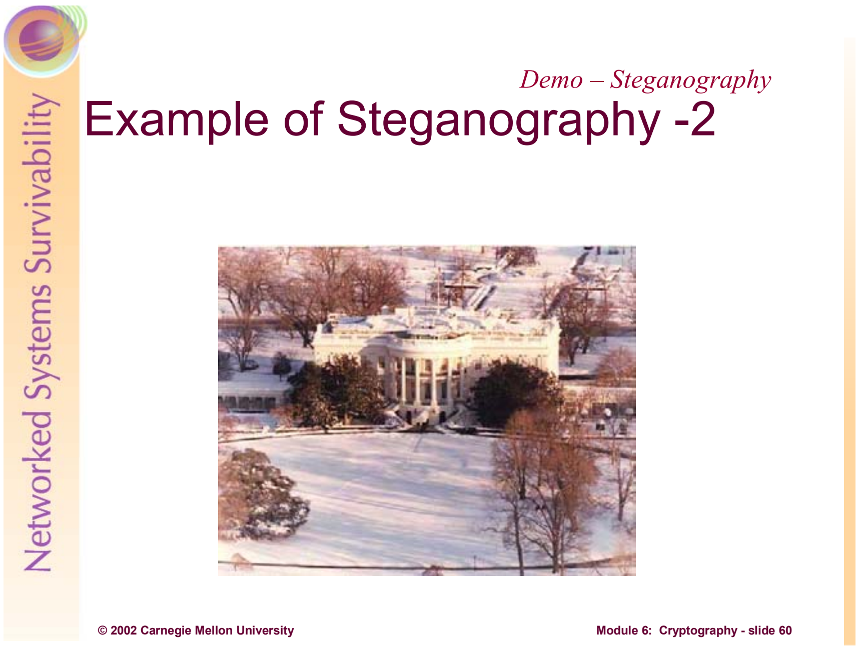
Hidden File (gettysburgaddress.txt)

Four score and seven years ago our fathers brought forth on this continent a new nation, conceived in liberty and dedicated to the proposition that all men are created equal. Now we are engaged in a great civil war, testing whether that nation or any nation so conceived and so dedicated can long endure. We are met on a great battlefield of that war. We have come to dedicate a portion of that field as a final resting-place for those who here gave their lives that that nation might live. It is altogether fitting and proper that we should do this. But in a larger sense, we cannot dedicate, we cannot consecrate, we cannot hallow this ground.

The brave men, living and dead who struggled here have consecrated it far above our poor power to add or detract. The world will little note nor long remember what we say here, but it can never forget what they did here. It is for us the living rather to be dedicated here to the unfinished work which they who fought here have thus far so nobly advanced. It is rather for us to be here dedicated to the great task remaining before us--that from these honored dead we take increased devotion to that cause for which they gave the last full measure of devotion--that we here highly resolve that these dead shall not have died in vain, that this nation under God shall have a new birth of freedom, and that government of the people, by the people, for the people shall not perish from the earth.

Networked Systems Survivability
© 2002 Carnegie Mellon University
Module 6: Cryptography - slide 59

This is the contents of the .txt file which is encrypted and encoded into the picture. It is, of course, Abraham Lincoln's Gettysburg Address.



Networked Systems Survivability

Demo – Steganography

Example of Steganography -2

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 60

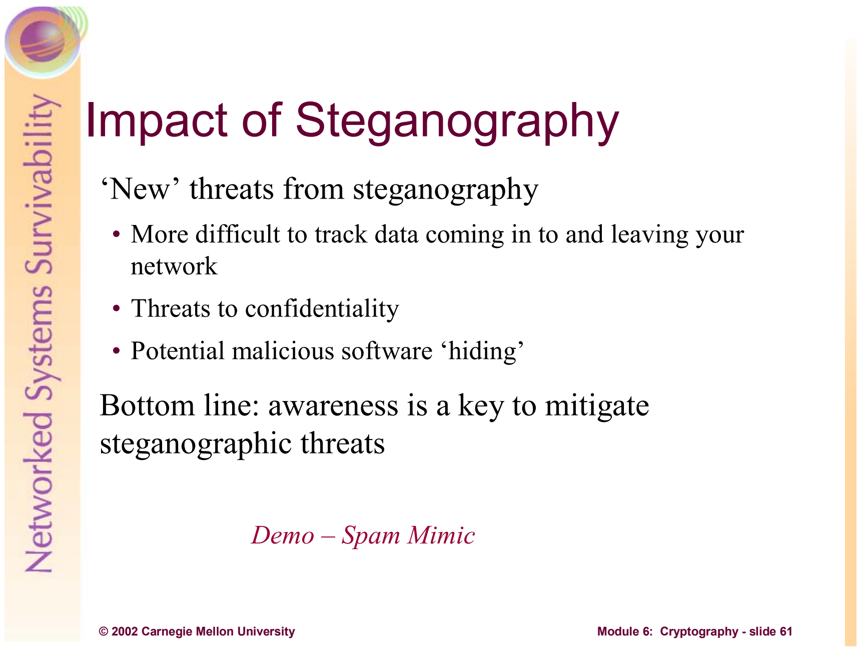
A free tool called ‘JP Hide and JP Seek’ (http://linux01.gwdg.de/~alatham/stego.html) was used to encode a Blowfish encrypted message (with a key of ‘testing’). This is the picture with the blowfish encrypted .txt file attached. Even the most discerning eye won’t be able to distinguish differences from the source picture.

The three requirements for steganography:

Host file: whitehouse.jpeg

File to be hidden: blowfish[gettysburgaddress.txt, testing]

Process to encode the two: JP Hide and JP Seek



Networked Systems Survivability

Impact of Steganography

‘New’ threats from steganography

- More difficult to track data coming in to and leaving your network
- Threats to confidentiality
- Potential malicious software ‘hiding’

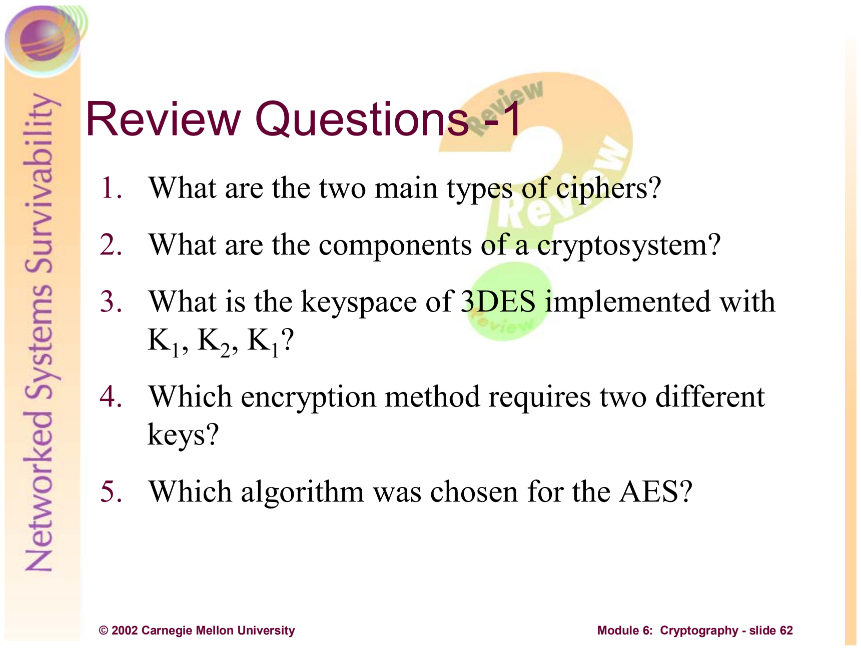
Bottom line: awareness is a key to mitigate steganographic threats

Demo – Spam Mimic

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 61

So why is steganography an issue for security administrators? If you are very aware of what traffic is on your network – what is coming in and going out, that is good. However, if some of the people on your network are able to encode ‘bad messages’ (corporate secrets, etc.) in seemingly innocuous files such as .jpeg, .mp3, and .exe files – how can we be truly certain that we are protecting our information completely and without error? It presents quite a challenge. So steganography simply provides us with a greater incentive for being vigilant. Administrators can review data traffic, and further analyze other access control and data security measures, establishing risk mitigating steps to deal with this threat.

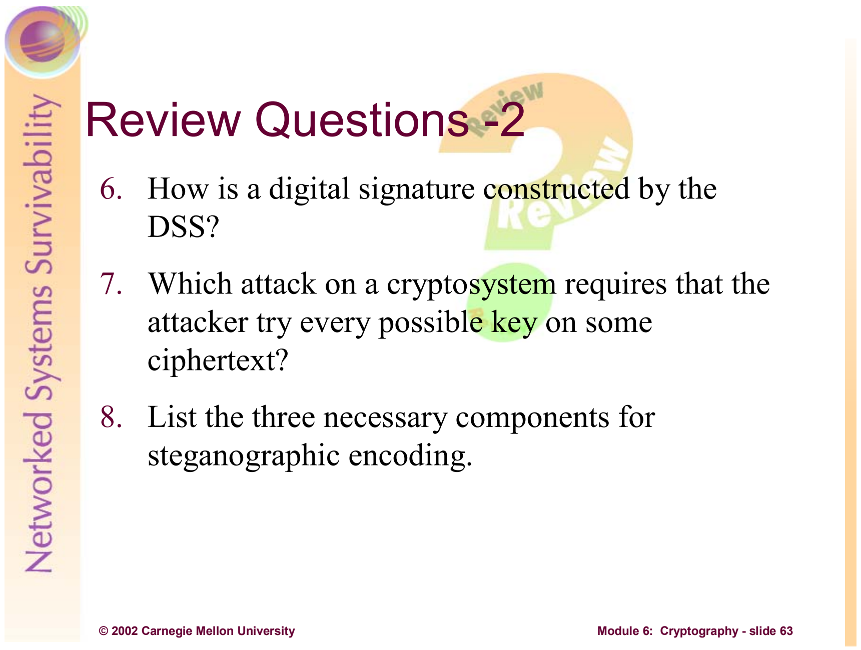


Review Questions -1

1. What are the two main types of ciphers?
2. What are the components of a cryptosystem?
3. What is the keyspace of 3DES implemented with K_1, K_2, K_1?
4. Which encryption method requires two different keys?
5. Which algorithm was chosen for the AES?

Networked Systems Survivability
Review
© 2002 Carnegie Mellon University
Module 6: Cryptography - slide 62

1. What are the main types of ciphers?
   Block and stream ciphers.
2. What are the components of a cryptosystem?
   All possible plaintexts, all possible ciphertexts, the encryption algorithm, and all possible keys
3. What is the keyspace of 3DES implemented with K_1, K_2, K_1?
   2^{56*2} or 2^{112}
4. Which encryption method required two different keys?
   Asymmetric encryption
5. Which algorithm was chosen for AES?
   Rijndael



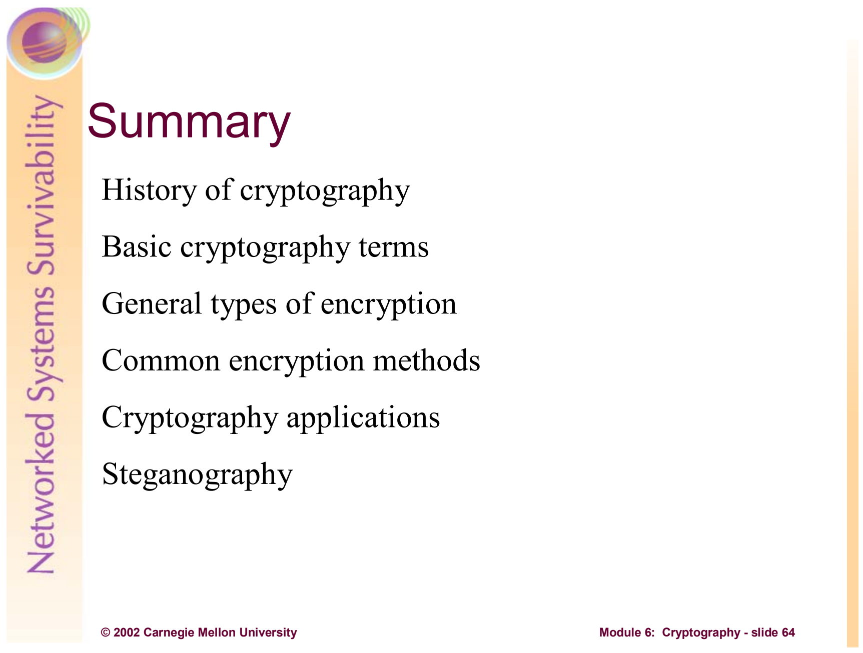
Review Questions -2

6. How is a digital signature constructed by the DSS?
7. Which attack on a cryptosystem requires that the attacker try every possible key on some ciphertext?
8. List the three necessary components for steganographic encoding.

© 2002 Carnegie Mellon University

Module 6: Cryptography - slide 63

6. How is a digital signature constructed by the DSS?
   A plaintext is hashed. The hash is then encrypted with the sender's private key and attached to the plaintext. This is then sent as the message. To verify, the receiver decrypts the encrypted hash with the sender's public key and compares this to a hash that they have computed for the message. If the two hashes match, it is a valid, unaltered message.
7. Which attack on a cryptosystem, requires that the attacker try every possible key on some ciphertext?
   Brute force
8. List the three necessary components for steganographic encoding.
   Host file, file to be hidden, and a process to encode the two



Networked Systems Survivability

Summary

History of cryptography

Basic cryptography terms

General types of encryption

Common encryption methods

Cryptography applications

Steganography