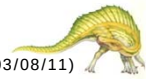
Chapter 15: Security

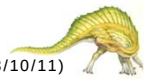
- The Security Problem
- Program Threats
- System and Network Threats
- Cryptography as a Security Tool
- Authentication
- Implementing Security Defenses
- Firewalling to Protect Systems and Networks
- Computer-Security Classifications
- An Example: Windows XP



(slides improved by R. Doemer, 03/08/11)

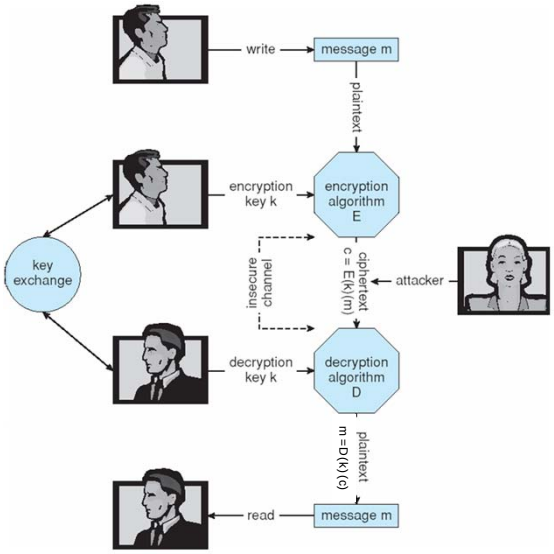
Cryptography as a Security Tool

- **Cyptography**
  - Broadest **security tool** available
    - Source and destination of **messages** cannot be trusted without cryptography
    - Means to constrain potential senders (*sources*) and/or receivers (*destinations*) of *messages*
  - Based on **secrets** (**keys**)
    - Messages are **encrypted** using secret keys
- **Cryptography Basics** (see notes on white board)
  - Requirements
    - Encryption and decryption functions: *efficiently computable*
    - Encryption and decryption functions: *collision-resistant*
    - Encryption function: *one-way function*

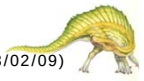


(slide improved by R. Doemer, 03/10/11)

Secure Communication over Insecure Medium



(slide fixed by R. Doemer, 03/02/09)



Encryption

- **Encryption algorithm** consists of
  - Set of *K* **keys**
  - Set of *M* **messages**
  - Set of *C* **ciphertexts** (encrypted messages)
  - An **encryption function** $E : K \rightarrow (M \rightarrow C)$.
    - That is, for each $k \in K$, $E(k)$ is a function for generating ciphertexts from messages.
    - Both E and $E(k)$ for any k should be *efficiently computable* functions.
  - A **decryption function** $D : K \rightarrow (C \rightarrow M)$.
    - That is, for each $k \in K$, $D(k)$ is a function for generating messages from ciphertexts.
    - Both D and $D(k)$ for any k should be *efficiently computable* functions.

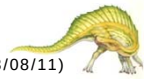
(slide improved by R. Doemer, 03/08/11)

Encryption

- **Encryption algorithm** must provide this **essential property**:

  Given a ciphertext $c \in C$, a computer can compute m such that $E(k)(m) = c$ *only if* it possesses $D(k)$.

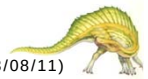
  - A computer holding $D(k)$ can **decrypt** ciphertexts to the plaintexts used to produce them,
  - A computer *not* holding $D(k)$ ***cannot*** **decrypt** ciphertexts.
  - Since ciphertexts are generally exposed (e.g. sent on the network), it is important that it be ***infeasible*** to derive $D(k)$ from the ciphertexts.



(slide improved by R. Doemer, 03/08/11)

Symmetric Encryption

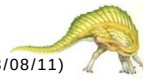
- **Same key** used to encrypt and decrypt
  - $E(k)$ can be derived from $D(k)$, and vice versa
- Examples
  - **DES** is most commonly used symmetric block-encryption algorithm (created by US Government)
    - Encrypts a block of data at a time
  - **Triple-DES** considered more secure
  - Advanced Encryption Standard (**AES**), **twofish** up and coming
  - **RC4** is most common symmetric stream cipher, but known to have vulnerabilities
    - Encrypts/decrypts a stream of bytes (i.e. wireless transmission)
    - Key is a input to pseudo-random bit generator, generates an infinite **keystream**



(slide improved by R. Doemer, 03/08/11)

Asymmetric Encryption

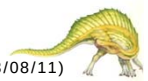
- **Public key** encryption based on each user having two keys:
  - *public key* – published key used to encrypt data
  - *private key* – key known only to individual user used to decrypt data
- Must be an encryption scheme that can be made public without making it easy to figure out the decryption scheme
- Example: **RSA block cipher**
  - Most common is public key encryption standard
  - Named after inventors Rivest, Shamir, Adleman
    - **Efficient algorithm** exists for testing whether or not a number is prime
    - **No efficient algorithm** is known for finding the prime factors of a number



(slide improved by R. Doemer, 03/08/11)

Asymmetric Encryption (Cont.)

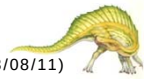
- **RSA algorithm**
- Formally, it is computationally infeasible to derive $D(k_d, N)$ from $E(k_e, N)$, and so $E(k_e, N)$ need not be kept secret and can be widely disseminated
  - $E(k_e, N)$ (or just k_e) is the **public key**
  - $D(k_d, N)$ (or just k_d) is the **private key**
  - N is the product of two large, randomly chosen **prime numbers** p and q (for example, p and q are 512 bits each)
  - Encryption algorithm is
    - $E(k_e, N)(m) = m^{k_e} \bmod N$, where k_e satisfies $k_e k_d \bmod (p-1)(q-1) = 1$
  - The decryption algorithm is then
    - $D(k_d, N)(c) = c^{k_d} \bmod N$



(slide improved by R. Doemer, 03/08/11)

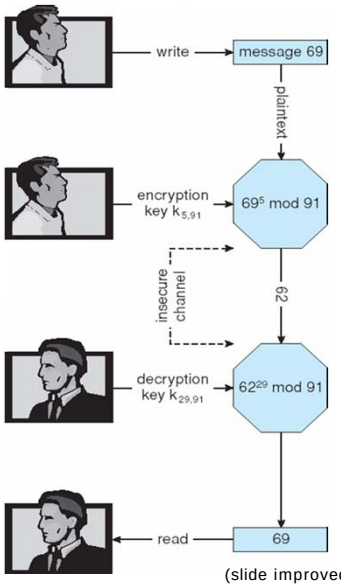
RSA Encryption Example

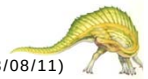
- For example, make p = 7and q = 13
- We then calculate $N = 7*13 = 91$ and $(p-1)(q-1) = 72$
- We next select k_e relatively prime to 72 and < 72, yielding 5
- Finally, we calculate k_d such that $k_e k_d \bmod 72 = 1$, yielding 29
- We now have our keys
  - Public key, k_e, N = 5, 91
  - Private key, k_d, N = 29, 91
- Encrypting the message 69 with the public key results in the cyphertext 62
- Cyphertext can be decoded with the private key
  - Public key can be distributed in clear text to anyone who wants to communicate with holder of private key



(slide improved by R. Doemer, 03/08/11)

RSA Encryption Example

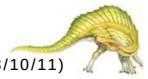




(slide improved by R. Doemer, 03/08/11)

Symmetric/Asymmetric Cryptography

- Symmetric cryptography is based on transformations
  - Simple, very efficient
  - Often used for bulk data encryption
- Asymmetric cryptography is based on mathematical functions
  - Very computation intensive
  - Often used only for initiating a secure communication
    - Authentication
    - Key exchange



(slide improved by R. Doemer, 03/10/11)

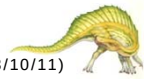
User Authentication

- Crucial to identify user correctly, as protection systems depend on user ID
- User identity most often established through *passwords*, can be considered a special case of either keys or capabilities
  - Also can include something user has and/or a user attribute
- Passwords must be kept secret
  - Frequent change of passwords
  - Use of “non-guessable” passwords
  - Log all invalid access attempts
- Passwords may also either be encrypted or allowed to be used only once

(slide fixed by R. Doemer, 03/02/09)

Network Authentication

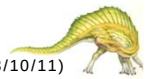
- **Authentication**
  - Constraining set of potential senders of a message
    - Complementary and sometimes redundant to encryption
    - Subset of encryption!
  - Can prove that a message is from the correct sender
  - Can prove that a message is unmodified
- Why authentication, if it is a subset of encryption?
  - Fewer computations (except for RSA digital signatures)
  - Authenticator usually shorter than message
  - Sometimes want authentication but not confidentiality



(slide improved by R. Doemer, 03/10/11)

Authentication

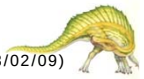
- **Authentication Algorithm** components
  - A set K of keys
  - A set M of messages
  - A set A of authenticators
  - A function $S : K \rightarrow (M \rightarrow A)$
    - That is, for each $k \in K$, $S(k)$ is a function for generating authenticators from messages
    - Both S and $S(k)$ for any k should be *efficiently computable* functions
  - A function $V : K \rightarrow (M \times A \rightarrow \{\text{true, false}\})$.
    - That is, for each $k \in K$, $V(k)$ is a function for verifying authenticators on messages
    - Both V and $V(k)$ for any k should be *efficiently computable* functions



(slide improved by R. Doemer, 03/10/11)

Authentication

- For a message m, a computer can generate an authenticator $a \in A$ such that $V(k)(m, a) = \texttt{true}$ only if it possesses $S(k)$
- Thus, a computer holding $S(k)$ can generate authenticators on messages so that any other computer possessing $V(k)$ can verify them
- Computer not holding $S(k)$ cannot generate authenticators on messages that can be verified using $V(k)$
- Since authenticators are generally exposed (for example, they are sent on the network with the messages themselves), it must not be feasible to derive $S(k)$ from the authenticators



(slide fixed by R. Doemer, 03/02/09)

Encryption Example - SSL

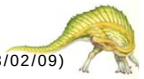
- Insertion of cryptography at one layer of the ISO network model (the transport layer)
- SSL – Secure Socket Layer (also called TLS)
- Cryptographic protocol that limits two computers to only exchange messages with each other
  - Very complicated, with many variations
- Used between web servers and browsers for secure communication (credit card numbers)
- The server is verified with a **certificate** assuring client is talking to correct server
- Asymmetric cryptography used to establish a secure **session key** (symmetric encryption) for bulk of communication during session
- Communication between each computer then uses symmetric key cryptography

(slide fixed by R. Doemer, 03/02/09)

Digital Certificates

- Proof of who or what owns a public key
- Public key digitally signed by a trusted party
- Trusted party receives proof of identification from entity and certifies that public key belongs to entity
- Certificate authority is trusted party
  - Their public keys are included with web browser distributions
  - They vouch for other authorities via digitally signing their keys

(slide fixed by R. Doemer, 03/02/09)



End of Chapter 15