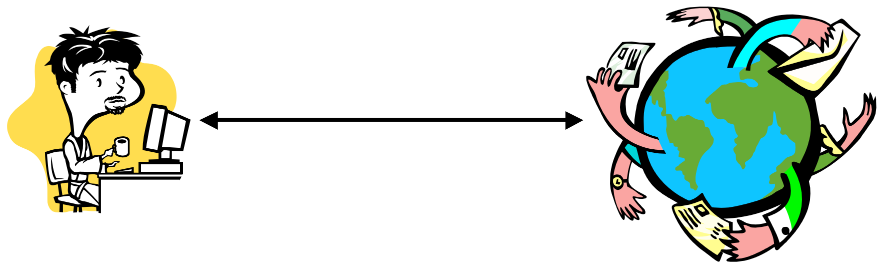
CPSC410/611: Security

- Security
  - Security Attacks
  - Security Threats
  - Crypto
  - Authentication
- Examples
  - SSL

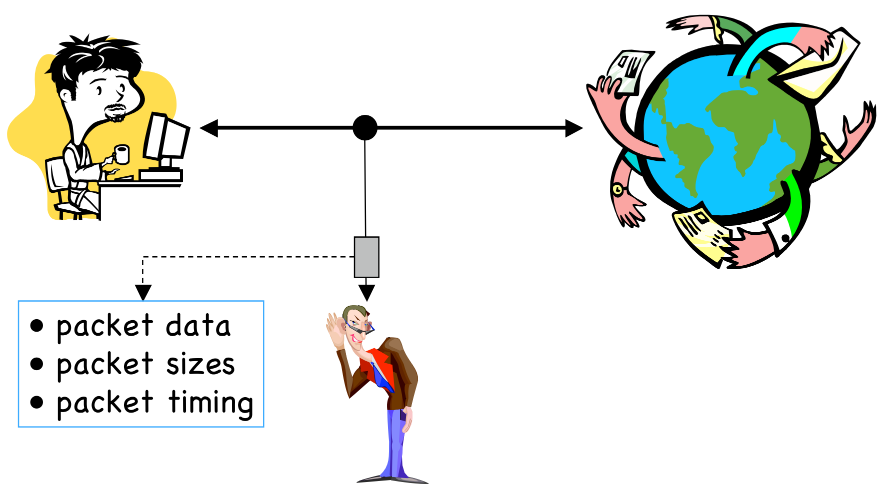
Security Threats

- Breach of confidentiality
  - unauthorized access to and/or dissemination of information
  - result of theft or illegal action of who has access to information
- Breach of integrity
  - unauthorized modification of data
- Information destruction:
  - loss of internal data structures
  - loss of stored information
  - information may be destroyed without being disclosed
- Unauthorized use of service:
  - bypass system accounting policies
  - unauthorized use of some proprietary services
  - obtain "free computing time"
- Denial of service:
  - prevent an authorized user from utilizing the system's services in a timely manner

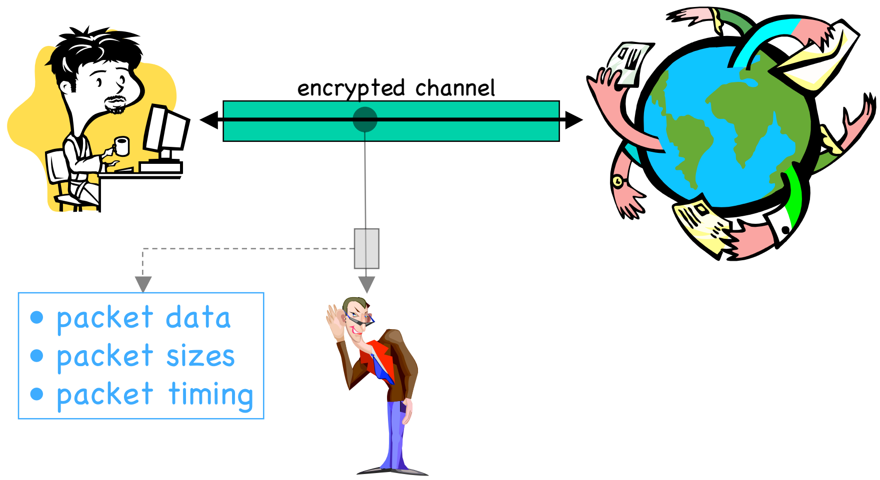
Typical Attacks



Typical Attacks: Breach of Confidentiality

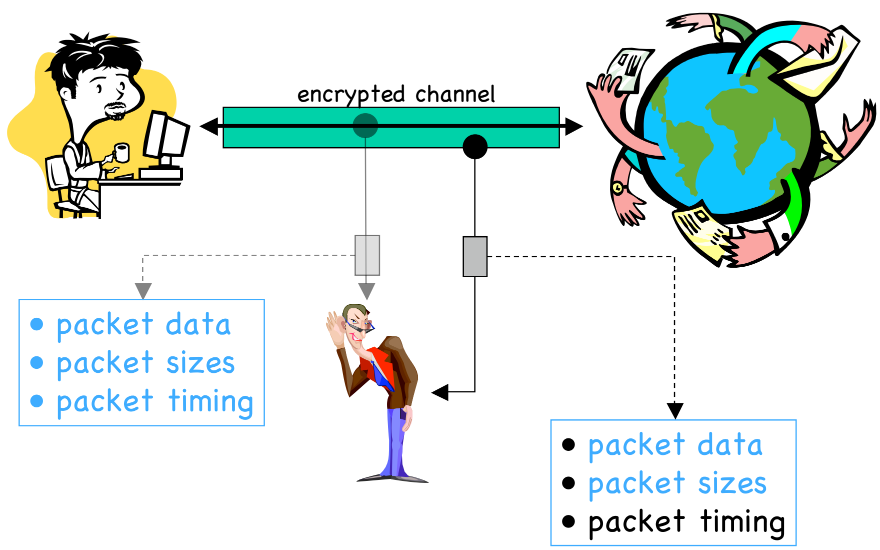


Countermeasure: Encryption

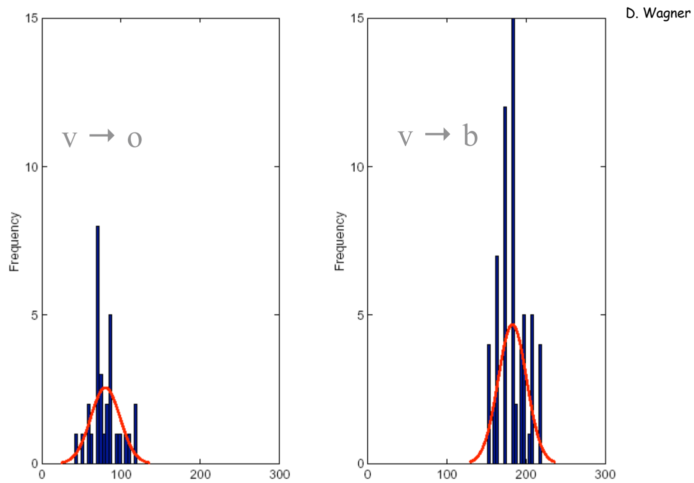


Countermeasure: Encryption Sufficient?

Example: Keystroke Analysis. [D. Wagner et al. "Timing Analysis of Keystrokes and Timing Attacks on SSH", Usenix'01]

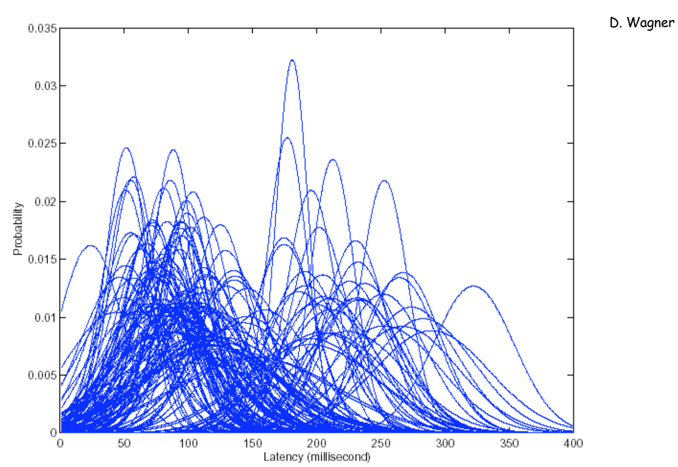


Character-Pair Delays



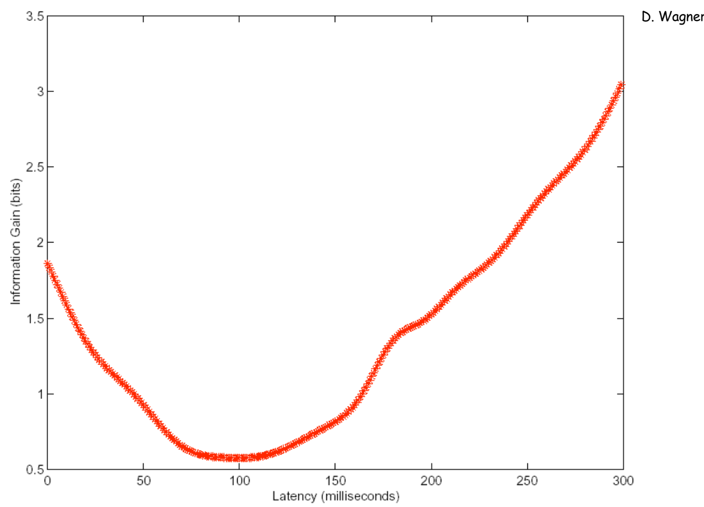
Measured delay between characters.

Character-Pair Delay Distributions



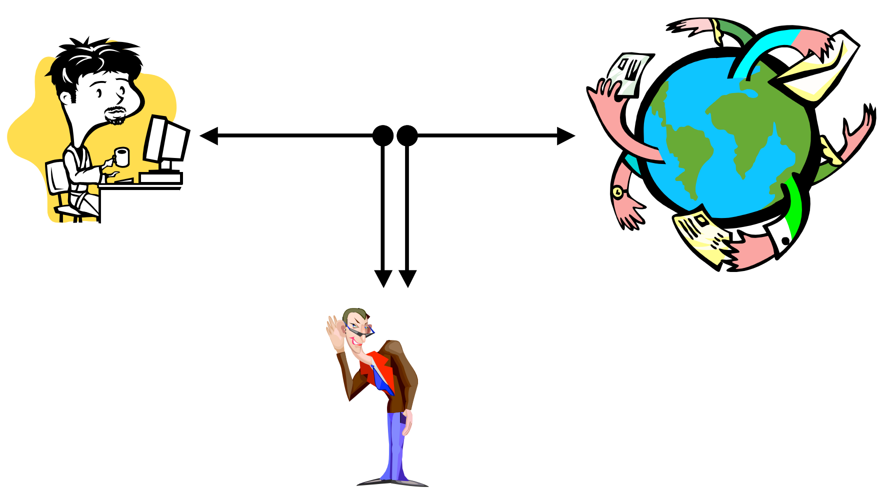
Estimated Gaussian delay distributions of character pairs collected from a user.

Information Content of Keystroke Data

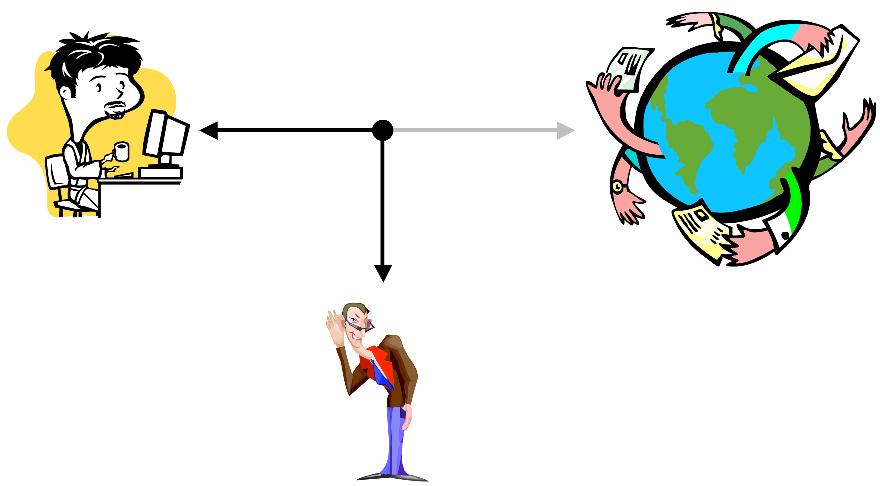


Information Gain

Typical Attacks: Man-In-The-Middle

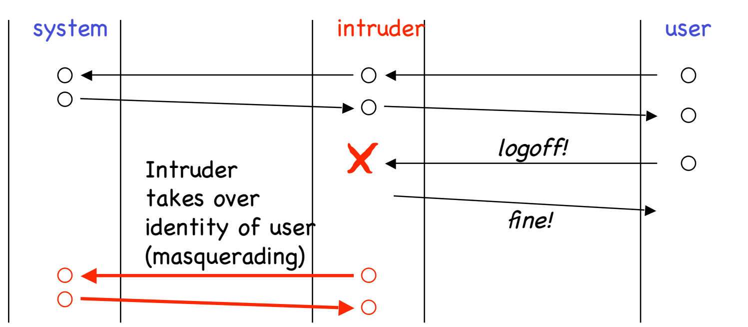


Typical Attacks: Masquerading



Man-In-The-Middle: Example

- Passive tapping
  - Listen to communication without altering contents.
- Active wire tapping
  - Modify data being transmitted
  - Example:



Typical Attacks: Penetration Attempts

- Two basic forms:
  - completely bypass authentication mechanism
  - obtain information or alter the system so as to enter system as authorized user
- Attempts:
  - Wire tapping (active *vs.* passive)
  - Trial and error
  - Browsing
    - Search storage (in particular previously allocated, but now available) for unauthorized information.
  - Trap doors
    - Unspecified and undocumented features of the system that may be exploited to perform unauthorized actions.
  - Trojan horse
  - Searching of waste

Prototypical Security Attacks (Tanenbaum)

- Request memory or disk space and simply read it.
- Try illegal system calls, and/or with illegal parameters
- Start logging in and try to abort login sequence.
- Modify OS structures kept in user space.
- Look for "Do not do X". Try as many variations of X as you can think of.
- Trojan horses
- Trapdoors
- Bribe personnel

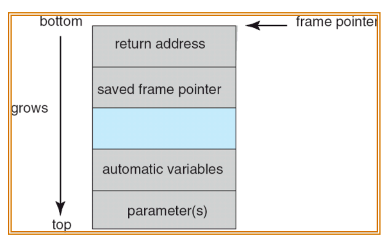
Famous (fixed) Security Flaws (Tanenbaum)

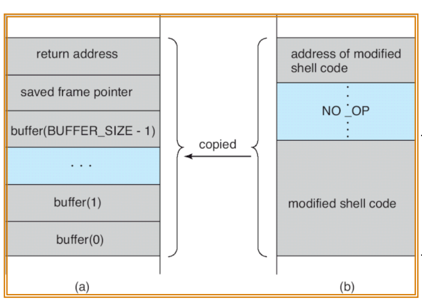
- Unix: `lpr` has option to delete file after is printed. So, print and remove password file.
- Unix: Link file called `core` to password file. Force core dump in program running with root privileges.
- Unix: The `mkdir` command runs with root privileges, creating i-node with system call `mknod`, then changes owner of directory with `chown` system call.
- TENEX: The "aligned password" trick.
- OS/360: To open file, OS verified password first. Then went to fetch filename. In the meantime, the filename could be overwritten by a DMA operation.

Buffer Overrun Attacks (Silberschatz et al)

[Example and illustrations from Silberschatz et al. "Operating Systems Concepts" Ch. 15]

```
#include <stdio.h>
#define BUFFER SIZE 256
int main(int argc, char *argv[])
    {
    char buffer[BUFFER SIZE];
    if (argc < 2)
           return -1;
    else {
           strcpy(buffer,argv[1]);
           return 0;
    }
}
```



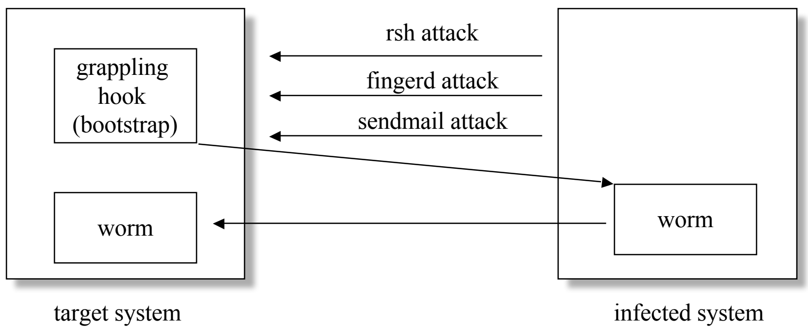



```
#include <stdio.h>
int main(int argc, char *argv[])
{
    execvp(''\bin\sh'',''\bin \sh'', NULL);
    return 0;
}
```

The Morris Worm (Nov 2nd, 1988)

[Example and illustrations from Silberschatz et al. "Operating Systems Concepts" Ch. 15]

- Worm: A process that replicates itself and uses up system resources (tape worm) (*The Shockwave Rider*, J. Brunner 1975)
- Virus: Piece of code that adds itself to other programs. Cannot execute independently (*When Charlie Was One*, D. Gerrold 1972)
- Morris Worm: first grand-scale attack on Internet.



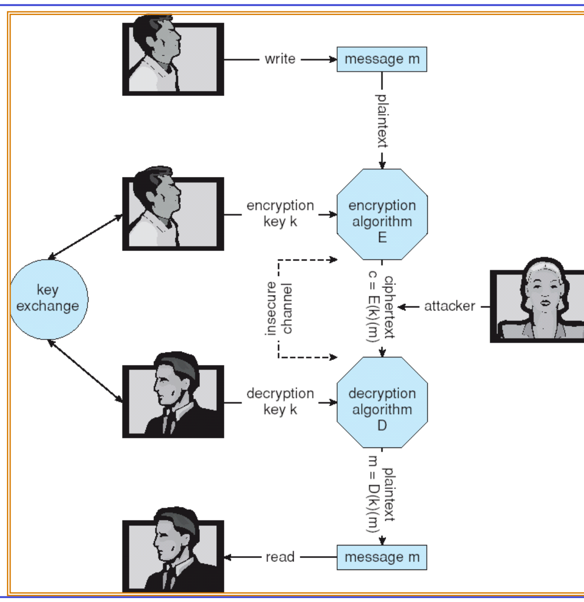
Safeguards

- External safeguards:
  - control physical access to computing facility
  - badges, locks, sign-in procedures, ...
  - administrative mechanisms:
    - audit trails
    - threat monitoring
- Internal safeguards:
  - Verification of user identity (Authentication)
  - Access control (e.g. at file-system level)
  - Information flow control:
    - It is not always necessary to *access* an object to get information. Sometimes information can be transferred or inferred.
  - Encryption

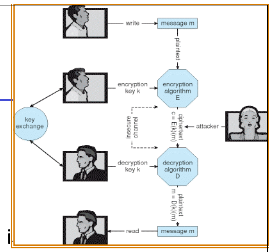
CPSC410/611: Security

- Security
  - Security Attacks
  - Security Threats
  - Crypto
  - Authentication
- Examples
  - SSL

Secure Communication over Insecure Medium



Encryption



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

- An encryption algorithm must provide this essential property:

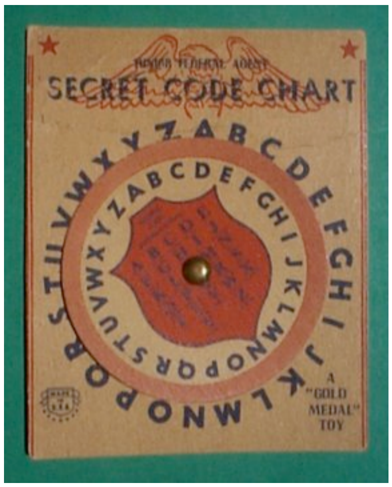
  Given a ciphertext $c \in C$, a computer can compute m
  such that $E(k)(m) = c$
  only if it possesses $D(k)$.

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

Symmetric Encryption

- Same key used to encrypt and decrypt
  - $E(k)$ can be derived from $D(k)$, and vice versa
- Data Encryption Standard (**DES**) is most commonly used symmetric block-encryption algorithm (created by US Govt)
- **Triple-DES** considered more secure
- Advanced Encryption Standard (**AES**), **twofish** up and coming

Symmetric Encryption: Caesar Cipher



MERRY CHRISTMAS

3

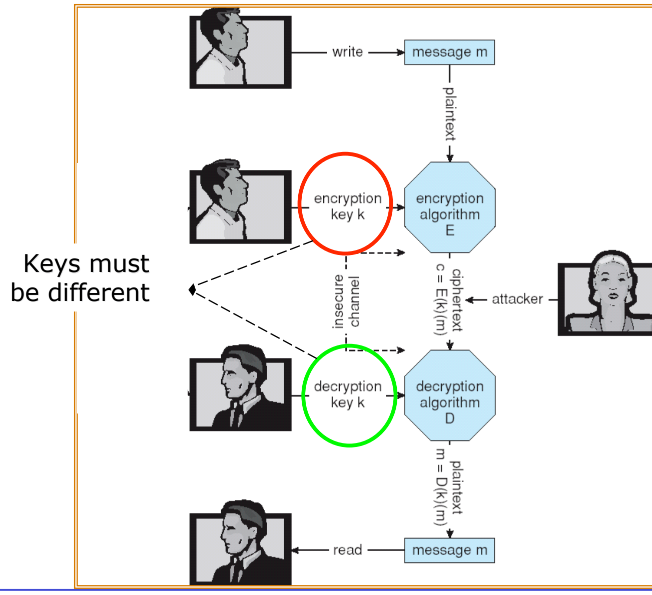
PHUUB FKULVWPDV

Symmetric Encryption: Jefferson's Wheel Cipher

Monticello Web Site: www.monticello.org/reports/interests/wheel_cipher.html

- Sender:
  - assemble wheels in some (secret) order.
  - Align message on one line.
  - Choose any of the other lines as ciphertext.
- Receive:
  - Assemble wheels in same secret order.
  - Align cipertext on one line.
  - Look for meaningful message on other lines.

Asymmetric Encryption



Asymmetric Encryption (cont.)

- 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
  - Most common is RSA block cipher
  - Efficient algorithm for testing whether or not a number is prime
  - No efficient algorithm is know for finding the prime factors of a number

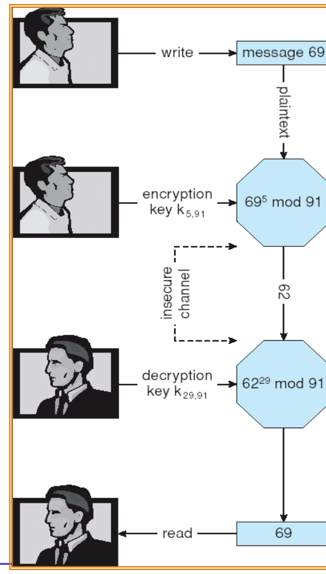
Asymmetric Encryption (Cont.)

- 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
  - $E(k_e , N)$ is the **public key**
  - $D(k_d , N)$ 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 $D(k_d , N)(c) = c^{k_d} \bmod N$

An Example

- For example. make $p = 7$ and $q = 13$
- 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 how 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 ciphertext 62
  - $69^5 \bmod 91 = 62$
- Ciphertext can be decoded with the private key
  - $62^{29} \bmod 91 = 69$
- Public key can be distributed in clear text to anyone who wants to communicate with holder of public key

Encryption and Decryption using Asymmetric Cryptography



Symmetric vs. Asymmetric

- Symmetric cryptography based on transformations
- Asymmetric based on mathematical functions
  - Asymmetric much more compute intensive
  - Typically not used for bulk data encryption
  - Used, instead, for short plaintexts, for example symmetric keys.

Authentication

- Constraining set of potential senders of a message
  - Also can prove message unmodified
- 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 $V : K \rightarrow (M \times A \rightarrow \{true, false\})$. 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

Authentication (Cont.)

- For a message m, a computer can generate an authenticator $a \in A$ such that $V(k)(m, a)$ = `true` only if it possesses $S(k)$
- Thus, 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

Authentication – Digital Signature

- Based on asymmetric keys and digital signature algorithm
- Authenticators produced are **digital signatures**
- In a digital-signature algorithm, computationally infeasible to derive $S(k_s)$ from $V(k_v)$
  - V is a one-way function
  - Thus, k_v is the public key and k_s is the private key
- Consider the RSA digital-signature algorithm
  - Similar to the RSA encryption algorithm, but the key use is reversed
  - Digital signature of message $S(k_s)(m) = H(m)^{k_s} \bmod N$
  - The key k_s again is a pair (d, N), where N is the product of two large, randomly chosen prime numbers p and q
  - Verification algorithm is $V(k_v)(m, a) \equiv (a^{k_v} \bmod N = H(m))$
    - Where k_v satisfies $k_v k_s \bmod (p-1)(q-1) = 1$

SSL

- Applications: HTTP, IMAP, FTP, etc…
- Client and server negotiate symmetric key that they will use for the length of the data session.
- Two phases in SSL:
  - Connection Establishment
  - Data Transfer

SSL: Connection Establishment

- Step 1: Client sends request to server, containing
  - SSL version; connection preferences; nonce (i.e. some random number)
- Step 2: Server chooses among preferences, and sends reply, containing
  - Chosen preferences; nonce; public-key certificate
  - Public-key certificate is a public key that has been digitally signed by a trusted authority.
- Step 3: Client can use certification authority's public key to check authenticity of server's public key.
- Step 4: Server can request public key of client and verify it similarly (optional)
- Step 5: Client chooses random number (premaster secret), encrypts it with server's public key, and sends it to server.
- Step 6: Both parties compute session key (used during data transfer) based on premaster secret and the two nonces.
  - Note: At no point is the session key transferred between client and server.

SSL: Data Transfer

- Messages are fragmented into 16kB portions.
- Each portion is optionally compressed.
- A Message Authentication Code (MAC) is appended
  - MAC is a hash derived from plaintext, two nonces, and pre-master secret
- Plaintext and MAC are encrypted using the symmetric key constructed during connection establishment.