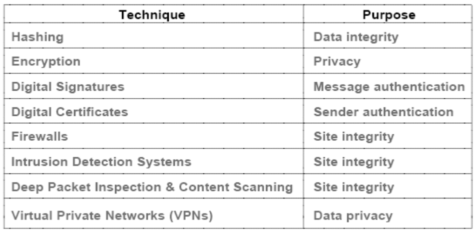
CSIS 4222

Ch 30: Encryption and Digital Certificates

Security Technologies

Many security products exist that perform a variety of functions



Technique	Purpose
Hashing	Data integrity
Encryption	Privacy
Digital Signatures	Message authentication
Digital Certificates	Sender authentication
Firewalls	Site integrity
Intrusion Detection Systems	Site integrity
Deep Packet Inspection & Content Scanning	Site integrity
Virtual Private Networks (VPNs)	Data privacy

Encryption: A Fundamental Security Technique

- A way to ensure the confidentiality of a transmitted message
- Sender applies encryption to scramble the bits
- Someone who intercepts an encrypted message will not be able to extract information

Secure Enhancements for Common Tools

OpenSSH suite – http://www.openssh.org

```
telnet      → ssh
ftp         → sftp
```

Encryption Functions

- Encryption is a function that takes two arguments and produces an cyphertext version of the message:

$$C = encrypt(K_1, M)$$

K_1 – encryption key
M – plaintext message

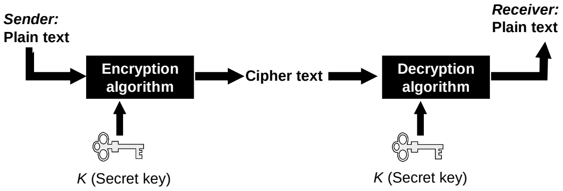
- Decryption is the inverse function that reverses the mapping:

$$M = decrypt(K_2, C)$$

K_2 – decryption key

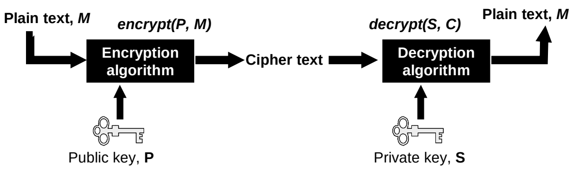
Private Key Encryption

- Uses a single secret key
- Both parties must agree on secret key in advance



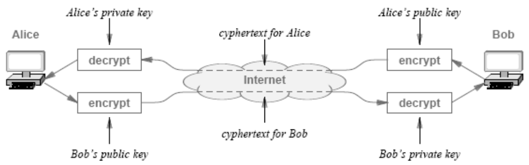
Public Key Encryption

- Each party has private key *S* and public key *P*
- *S* is kept secret, and cannot be deduced from *P*
- *P* is made available to everyone
- Encryption and decryption with *S* and *P* are inverse functions: S(P(m)) = m = P(S(m))



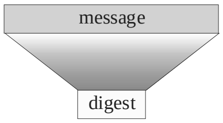
Symmetric and Asymmetric

- Private key systems are *symmetric*
  - each side can send or receive messages using the same key
- Public key systems are *asymmetric*



Message Digest

- Digest function maps an arbitrary length message m to fixed length digest d(m)
- *One-way function*: given d(m), can't find m
- *Collision-free*: infeasible to generate m and m' such that d(m) = d(m')

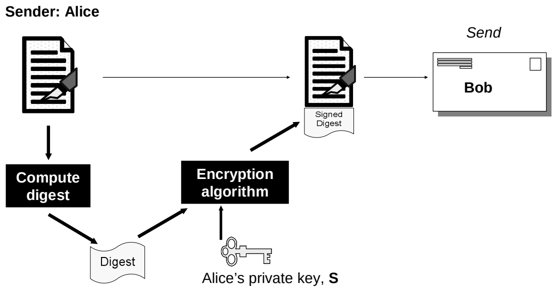


Digital Signature

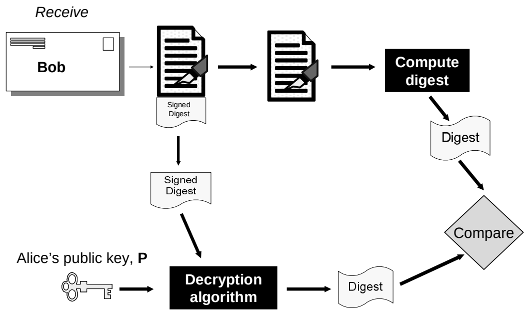
To sign message m,

1. Sender computes digest d(m)
2. Sender computes encrypt(*S*,d(m)) and sends it along with m
3. Receiver computes
   decrypt(*P*, encrypt(*S*, d(m))) = d(m)
4. Receiver computes digest of m and compares with result above – if a match, signature is verified

Digital Signature



Digital Signature



Key Distribution and Certification

- Secret key must be agreed on in advance of communication
  - Must be kept ***secret!***
- Public key does not need this
  - But how to tell it's really someone's true public key?
- Need a **trusted intermediary**
  - Called a **certification authority (CA)** for public keys
  - VeriSign, Thawte

Certification Authority

Validates identities and issues certificates

- Verify that an entity (person, router, etc.) is who it says it is
  - ID checks, etc.
- Once the CA verifies the identity of the entity, it creates a **certificate** that binds the public key to the identity
  - Certificate contains the public key and globally unique identifying information about the owner

Secure Sockets Layer (SSL) and Transport Layer Security (TLS)

SSL

- Originally developed by Netscape, now a standard protocol for data encryption and authentication between a web client and web server
- Between application layer and transport layer
- Begins with a handshake that negotiates an encryption algorithm and keys, and authenticates the server to the client
- After handshake, all data is encrypted using secret session keys

SSL Security

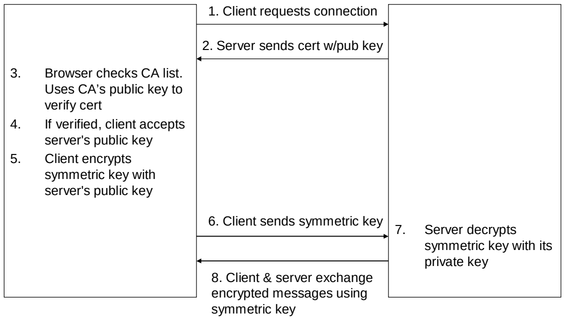
- Used for secure web transactions
- Easily configured into application programs
- Server key verified by trusted 3rd party via signed digital certificate

  No certificate – no SSL
- SSL used in Apache web server on Linux
- Not limited to just HTTP

SSL Handshaking

- Server sends public key (in certificate) to client
- Client verifies certificate's signature
- Client uses public key to send a secret to server
- Both client and server use the secret to generate a symmetric session key, which is used to encrypt the remainder of the transaction
- Has provisions to avoid forgeries and replays

SSL Handshake for HTTPS



1. Client requests connection

2. Server sends cert w/pub key

3. Browser checks CA list. Uses CA's public key to verify cert
4. If verified, client accepts server's public key
5. Client encrypts symmetric key with server's public key

6. Client sends symmetric key

7. Server decrypts symmetric key with its private key

8. Client & server exchange encrypted messages using symmetric key

Digital Certificates

- Contains
  - Name, address, organization…
  - Public key (for encryption, signing)
  - Signature by trusted verifying organization
- Can be self-signed (good for testing)
- Can be your own trusted authority
  - For pretend in the lab
  - Some organizations do it for real, internally
- Generate with openssl (http://openssl.org)

Secure email Sources

- Thawte: http://www.thawte.com
  - Free email certificate (requires SSN)
- Gnu Privacy Guard: http://www.gnupg.org
  - Tools/framework for encryption
  - Free
- Pretty Good Privacy (PGP): http://www.pgp.com
  - Tools/framework for encryption
  - Commercial