Motivation 1 - Key Distribution Problem

- In a secret key cryptosystem, the secret key must be transmitted via a secure channel
- Inconvenient
  - n parties want to communicate with each other, how many keys total keys are needed and how many other keys must each n store?
  - n entities – There will be n*(n-1)/2 keys total
  - Each entity has to store n-1 keys
- Insecure
  - Is the secure channel really secure?
- Public key cryptosystem solves the problem
  - Public key known by everyone – telephone directory
  - Privacy key is never transmitted

How many Symmetric Keys needed?

<table>
<thead>
<tr>
<th>n</th>
<th>Total Keys</th>
<th>Secure Channel</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>21</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>28</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>36</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>45</td>
<td>8</td>
</tr>
<tr>
<td>15</td>
<td>105</td>
<td>14</td>
</tr>
<tr>
<td>20</td>
<td>210</td>
<td>18</td>
</tr>
<tr>
<td>25</td>
<td>255</td>
<td>20</td>
</tr>
<tr>
<td>30</td>
<td>375</td>
<td>22</td>
</tr>
<tr>
<td>40</td>
<td>760</td>
<td>28</td>
</tr>
<tr>
<td>50</td>
<td>1375</td>
<td>30</td>
</tr>
</tbody>
</table>

Administration Problems:
- Adding new entities
- Removing existing entities
- Changing keys
Motivation 2- Digital Signature

- In a secret key cryptosystem, authentication and non-repudiation may be difficult
- Authentication
  - You must share a secret key with someone in order to verify his signature
- Non-repudiation
  - “I didn’t sign it. You did since you also have the key”
- Public key cryptosystem solves the problem
  - Verification of signature needs only the public key
  - One is solely responsible for his private key

Motivation (cont) - Email Security

- The percentage of email that is actually readable by an attacker, or that can be manipulated while in transit with little chance of discovery is close to 100%.

Is this Key Exchange Secure?

\[
\text{Alice and Bob want to establish a shared secret key.}
\]

- Have agreed on the value \( p = 353 \) (prime) and \( g = 3 \)
- Select the random secret values:
  - Alice chooses \( X_a = 97 \), Bob chooses \( X_b = 233 \)
- Derive the public keys:
  - \( T_a = g^{X_a} \mod n = 3^{97} \mod 353 = 40 \) (Alice’s)
  - \( T_b = g^{X_b} \mod n = 3^{233} \mod 353 = 248 \) (Bob’s)
- Derive the shared secret key:
  - \( K = T_b^{X_a} \mod n = 248^{97} \mod 353 = 160 \) (Alice’s)
  - \( K = T_a^{X_b} \mod n = 40^{233} \mod 353 = 160 \) (Bob’s)

Diffie-Hellman Example
Diffie-Hellman Man-in-the-middle

Public Key Cryptography Brainstorming

• What if Alice’s public key is forged?
  – Everyone (but Alice) somehow get a fake public key (generate by Bob) and they all believe it is Alice’s public key
  – Alice can not communicate with everyone through public key cryptosystem
  – Alice can not decrypt anything encrypted by others using the fake Alice’s public key
  – When Alice signs anything with her private key, everyone else will think it is fake

• How do you solve this problem?

Certificate

• A signed message claiming somebody’s public key is such & such
  – [Alice’s public key is 12345] Bob
• If Carol does not know Bob or Bob’s key, then the certificate signed by Bob doesn’t mean much to Carol
• If Carol knows and trusts David, who can certify Bob’s key, then Carol may have some trust on Bob’s certificate
• There could be a chain of certificate that indirectly certify the authenticity of the public key

Third Party Trust Model
Public Key Infrastructure (PKI)

- What is PKI for?
  - Facilitate secure distribution of every one’s public key
  - Provide some “authenticity” of the public keys distributed
- Basic Components of X509 PKI
  - Certification authorities (CA)
  - Registration Authority (RA)
  - Certificate Distribution System
  - Certificates
  - PKI Enabled Application

PKI Simplified Model

- Not an algorithm
- Uses public key cryptography
- Enables secure data transmission over the Internet
- “A PKI is a set of agreed-upon standards, Certification Authorities (CA), structure between multiple CAs, methods to discover and validate Certification Paths, Operational Protocols, Management Protocols, Interoperable Tools and supporting Legislation”

“Digital Certificates” book – Jalal Feghhi, Jalil Feghhi, Peter Williams
PKI Steps

- Provides Proof of identity
- Verifies subscriber based on class level
- Issues Certificate & posts in Repository
- Can validate subscriber public key

PKI Components: Certificate Authority I

- Basic Responsibilities
  - Key Generation
  - Digital Certificate Generation
  - Certificate Issuance and Distribution
  - Revocation
  - Key Backup and Recovery System
  - Cross-Certification

PKI Components: Certificate Authority II

- Public keys are issued by a certification authority (CA)
- A CA verifies & vouches for the identity information in a Certificate
  - Like a Government for a passport
  - Like a bank for an ATM card.
- A certificate that the CA issues to a company binds a public key to the recipient’s private key

PKI Components: Certificate Authority III

- The CA has a public and private key pair, just like people and devices
- The CA uses its private key to sign the body of the certificate, just as people use personal private keys to sign messages
- To verify, one must use the CA’s public key to decrypt the signature, just as one would verify a personal signature from another user!
PKI Components: Registration Authority (RA)

- Basic Responsibilities
  - Registration of Certificate Information
  - Face-to-Face Registration
  - Remote Registration
  - Automatic Registration
  - Revocation
- Verification level depends on class requested
  - Class 1 can be done from public records
  - Class 3 can be for signing software (more powerful)

PKI Components: Certificate Distribution System

- Provides Repository for:
  - Digital Certificates
  - Certificate Revocation Lists (CRLs)
- Typically
  - Special Purpose Databases
  - LDAP directories

PKI Components: Certificate I

An electronic passport that proves your identity and authenticates you

- Who you are
- What your public key is
- Who issued your certificate

Physical World Analogies
- ATM Card - a Certificate to conduct electronic banking
- Driver’s license - a Certificate to operate a vehicle
- Employee badge - a Certificate to gain facility access
- U.S. Passport - a Certificate telling who you are

PKI Components: Certificate II

- A digital Certificate becomes a passport that proves your identity and authenticates you
  - A US passport is issued by a trusted Government – when another Government sees it, they trust it
- A digital Certificate issued by a trusted CA can also be trusted
PKI Components: Certificate III

- A certificate binds an entity to a key pair
- The public key (embedded in a digital certificate) is in a public directory that is freely accessible
- Now when you download someone’s public key, you know that it belongs to a specific person

PKI Components: Certificate IV

- Issuer
- Subject
- Subject Public Key
- Issuer Signature

Certificate Expiration and Renewal

- A period of validity is assigned to each certificate
  - After that date, the certificate expires
- A certificate can be renewed with a new expiration date assigned
  - If the keys are still valid and remain uncompromised
Certificate Revocation

- Sometimes there is a need to revoke a certificate before it expires:
  - A hardware crash causes a key to be lost
  - A private key is compromised
  - Termination of affiliation with some organization
- **Revocation is permanent**
- Suspension can be lifted

Certificate Revocation List (CRL)

- CRLs are published by CAs at well defined interval of times
- Contains all revoked and suspended certificates
- It is a responsibility of “Users” of certificates to “download” a CRL and verify if a certificate has been revoked
- User application must deal with the revocation processes

CA Technology Evolution

Simple Certification Hierarchy

- X.509 standard is the general model for certification hierarchy.
- If you trust the CA that signed the certificate, you can trust the certificate
- Each entity has its own certificate and may have more than one. The root CA’s certificate is self signed and each sub-CA is signed by its parent CA.
- Each CA may also issue CRLs. In particular the lowest level CAs issue CRLs frequently.
Simple Certification Hierarchy – An Example

Alice trusts the root CA
Bob sends a message to Alice

Alice needs Bob’s certificate, the certificate of the CA that signed Bob's certificate, and so on up to the root CA's self signed certificate.

Alice also needs each CRL for each CA.

Only then can Alice verify that Bob’s certificate is valid and trusted and so verify the Bob’s signature.

Root Certificates built into browser

• If the CA is a widely recognized authority, its certificate (along with its public key) will already be embedded in browsers.

Microsoft Root CA

• You can set up your own Certificate Authority Server
  – Windows Server 2003 or Windows 2000 Server
  – Install the Certificate Services
  – Note that after installing this service the name of the domain or computer cannot change

Microsoft Root CA

• Specify options to generate certificates, including
  – Cryptographic Service Provider
  – Hash algorithm
  – Key length
PKI solving security needs

- How do we solve the 4 security needs?

  - Confidentiality
  - Integrity
  - Non-Repudiation
  - Authentication

Digital Signature, Date and Time

ISE at George Mason University

SSL/TLS/SSH

- SSL/TLS overview and basic features
- SSL Record Protocol
- SSL Handshake Protocol
- Other SSL Protocols
- SSL and TLS differences
- SSL applications
- Comparison of IPsec & SSL

SSL/TLS Overview

- SSL = Secure Sockets Layer.
  - Originally developed to secure http
  - unreleased v1, flawed but useful v2, good v3.
  - TLS1.0 = SSL3.0 with minor tweaks ≈ SSL3.1
  - Defined in RFC 2246.
  - Open-source implementation at http://www.openssl.org/.
- SSL/TLS provides security ‘at TCP layer’.
  - Uses TCP to provide reliable, end-to-end transport.
  - Applications need some modification.
  - In fact, usually a thin layer between TCP and HTTP.
SSL Functionality

- Session-Layer Security:
  - Protection of (bi-directional) transport protocol
- Security Services:
  - Integrity, Authenticity, Confidentiality
- Client Security:
  - Server must be authenticated (using public-key certificates)
- Server Security:
  - Client may be authenticated (using public-key certificates)
- Security Suite:
  - Client and Server negotiate Algorithms and methods

SSL Services

- Peer entity and data authentication
- Data confidentiality
- Data integrity
- Compression/decompression
- Generation/distribution of session keys
- Security parameter negotiation.

SSL/TLS Basic Features

- SSL/TLS widely used in Web browsers and servers to support ‘secure e-commerce’ over HTTP.
  - Built into Microsoft IE, Netscape, Mozilla, Apache, IIS,…
  - The (in)famous browser lock.
- SSL architecture provides two layers:
  - SSL Record Protocol
  - Upper layer carrying:

SSL Protocol Architecture

- A two-layer protocol
SSL Connection and Session

- Each SSL session can be used for multiple SSL connections.
- SSL Session
  - An association between a client and a server.
  - Created by handshake protocol.
  - Is used to avoid negotiation of new security parameters for each connection.
- SSL Connection
  - A connection is a transport that provides a suitable type of service.
  - Peer-to-peer, transient
  - Each connection is associate with one session.

SSL Session

- We can view an SSL session as an SSL security association.
  - Created by handshake protocol
  - Defines set of cryptographic parameters (encryption and hash algorithm, master secret, certificates).
  - Carries multiple connections to avoid repeated use of expensive handshake protocol.
- A SSL session consists of
  - Session ID
  - X.509 public-key certificate of peer (could be null)
  - Compression algorithm
  - Cipher spec:
    - Encryption algorithm, message digest algorithm, etc.
  - Master secret: 48 byte secret shared between the client and server
  - Is reusable

SSL Connection

- SSL Connection concept
  - State defined by nonces, secret keys for encryption, integrity, and IV.
  - Keys for many connections derived from single master secret created during handshake protocol.
- An SSL Connection consists of
  - Server and client random numbers
  - Server write MAC secret
  - Client write MAC secret
  - Server write key
  - Client write key
  - Server IV
  - Client IV
  - Sequence number

SSL Record Protocol

- SSL Record Protocol provides secure, reliable channel to second layer:
  - Data origin authentication and integrity.
    - MAC using algorithm similar to HMAC.
    - Based on MD-5 or SHA-1 hash algorithms.
    - MAC protects 64 bit sequence number for anti-replay.
  - Confidentiality.
    - Bulk encryption using symmetric algorithm.
      - IDEA, RC2-40, DES-40 (exportable), DES, 3DES,...
      - RC4-40 and RC4-128.
  - Carries application data and SSL ‘management’ data.
SSL Record Protocol

• Data from application/upper layer SSL protocol partitioned into fragments (max size $2^{14}$ bytes).
• Optional compression
• MAC
• then pad (if needed)
• finally encrypt.
• Prepend header.
  – Content type, version, length of fragment.
• Submit to TCP.

SSL Record Protocol Operation

SSL Handshake Protocol

• Like IPSec, SSL needs symmetric keys:
  – MAC and encryption at Record Layer.
  – Different keys in each direction.
• These keys are established as part of the SSL Handshake Protocol.
• As with IKE in IPSec, the SSL Handshake Protocol is a complex protocol with many options…

SSL Handshake Protocol Security Goals

• Secure ciphersuite negotiation.
  – Encryption and hash algorithms
  – Authentication and key establishment methods.
• Entity authentication of participating parties.
  – Participants are ‘client’ and ‘server’.
  – Server nearly always authenticated, client more rarely.
  – Appropriate for most e-commerce applications.
• Establishment of a fresh, shared secret.
  – Shared secret used to derive further keys.
  – For confidentiality and authentication in SSL Record Protocol.
SSL Handshake Protocol

- Initially SSL session has null compression and encryption algorithm.
- Both are set by the handshake protocol at the beginning of session.
- Handshake protocol may be repeated during the session.
- Four phases
  - Establish Security Capabilities
  - Server Authentication and Key Exchange
  - Client Authentication and Key Exchange
  - Finish

SSL Connection Steps

- When you hit an SSL encrypted page here is what usually happens:
  - Server sends certificate to client
  - Client checks to see if signing CA is in trusted list in browser
  - Client computes hash of certificate & compares message digest of certificate by decrypting using CA’s public key (CA signed certificate)
  - Client checks validity dates of certificate
  - Client checks URL in certificate to and verifies it matches the current URL
  - Client extracts server’s public key from certificate
  - Client creates a session key (symmetric)
  - Client encrypts session key with server’s public key and sends it over
  - Server decrypts using private key

Simplified SSLv3/TLS

Client

- Hello, ciphers I support, Rclient
- Certificate, ciphers I chose, Rserver

Server

- [keyed hash of all handshake msgs]
- [keyed hash of K & handshake msgs]
- {keyed hash of all handshake msgs}
- Data protected with keys derived from K
- K=f(S, Rclient, RServer)

Phase 1. Establish Security Capabilities

Client

- Client_hello*

Server

- Server_hello*

Message marked by * are mandatory; Other messages are optional.
Phase 1 (Cont’d)

• **Client_hello**
  – Version: The highest SSL version understood by the client
  – Random: 4-byte timestamp + 28-byte random number.
  – Session ID: zero for new session, non-zero for a previous session
  – CipherSuite: list of supported algorithms
  – Compression Method: list of supported compression methods

• **Server_hello**
  – Version: min (client_hello version, highest version supported by the server)
  – Random: 4-byte timestamp + 28-byte random number.
    • Generated by the server
  – Session ID:
  – CipherSuite: selected from the client’s list by the server
  – Compression method: selected from the client’s list by the server

Phase 2: Server Authentication and Key Exchange

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Certificate</td>
<td>Server_key_exchange</td>
</tr>
<tr>
<td>Server_key_exchange</td>
<td>Certificate_request</td>
</tr>
<tr>
<td>Certificate_request</td>
<td>Server_done*</td>
</tr>
</tbody>
</table>

A Certificate is almost always used.

Server_key_exchange message

• Not required if
  – The server has sent a certificate with fixed D-H parameters, or
  – RSA key exchange is to be used.

• Needed for
  – Anonymous D-H
  – RSA key exchange, in which the server is using RSA but has a signature-only RSA key.
Certificate_request message

- Request a certificate from the client
- Two parameters
  - Certificate_type
    - RSA, signature only
    - DSS, signature only
    - ...
  - CertificateAuthorities

Server_done message

- Indicate the end of server hello and associated messages.

Phase 3. Client Authentication and Key Exchange

- Certificate
  - One or a chain of certificates.
- Client_key_exchange
  - RSA: encrypted pre-master secret with the server’s public key.
  - D-H: client’s public key.
- Certificate_verify
  - Only sent following any client certificate that has signing capability
  - Proves the client is the valid owner of the certificate.
SSL Handshake Protocol – Key Exchange

- SSL supports several key establishment mechanisms.
- Most common is RSA encryption
  - Client chooses `pre_master_secret`, encrypts using public RSA key of server, sends to server.
- Can also create `pre_master_secret` from:
  - Fixed Diffie-Hellman
    - Server (and possibly Client) certificate contains DH parameters.
  - Anonymous Diffie-Hellman
    - Each side sends Diffie-Hellman values, but no authentication.
    - Vulnerable to man-in-middle attacks.

SSL Handshake Protocol – Entity Authentication

- SSL supports several different entity authentication mechanisms.
- Most common based on RSA.
  - Ability to decrypt `pre_master_secret` and generate correct MAC in `finished` message using keys derived from `pre_master_secret` authenticates server to client
  - DSS or RSA signatures on nonces (and other fields, e.g. Diffie-Hellman values).

SSL Key Derivation

- Keys used for MAC and encryption in Record Layer derived from `pre_master_secret`:
  - Derive `master_secret` from `pre_master_secret` and client/server nonces using MD5 and SHA-1 hash functions.
  - Derive key material from `master_secret` and client/server nonces, by repeated use of hash functions.
  - Split key material up into MAC and encryption keys for Record Protocol as needed.

SSL Handshake Protocol Run

- An illustrative protocol run follows.
- We choose the most common use of SSL.
  - No client authentication.
  - Client sends `pre_master_secret` using Server’s public encryption key from Server certificate.
  - Server authenticated by ability to decrypt to obtain `pre_master_secret`, and construct correct `finished` message.
- Other protocol runs are similar.
SSL Handshake Protocol Run

M1: C → S: ClientHello

- Client initiates connection.
- Sends client version number.
  - 3.1 for TLS.
- Sends ClientNonce.
  - 28 random bytes plus 4 bytes of time.
- Offers list of cipher suites.
  - Key exchange and authentication options, encryption algorithms, hash functions.
  - E.g. TLS_RSA_WITH_3DES_EDE_CBC_SHA.

SSL Handshake Protocol Run

M2: S → C: ServerHello, ServerCertChain, ServerHelloDone

- Sends server version number.
- Sends ServerNonce and SessionID.
- Selects single cipher suite from list offered by client.
  - E.g. TLS_RSA_WITH_3DES_EDE_CBC_SHA.
- Sends ServerCertChain message.
  - Allows client to validate server’s public key back to acceptable root of trust.
- (optional) CertRequest message.
  - Omitted in this protocol run – no client authentication.
- Finally, ServerHelloDone.

SSL Handshake Protocol Run

M3: C → S: ClientKeyExchange,
  ChangeCipherSpec, ClientFinished

- ClientKeyExchange contains encryption of pre_master_secret under server’s public key.
- ChangeCipherSpec indicates that client is updating cipher suite to be used on this session.
  - Only when client is authenticated.
- Finally, ClientFinished message.
  - A MAC on all messages sent so far (both sides).
  - MAC computed using master_secret.

SSL Handshake Protocol Run

M4: S → C: ChangeCipherSpec,
  ServerFinished

- ChangeCipherSpec indicates that server is updating cipher suite to be used on this session.
- Finally, ServerFinished message.
  - A MAC on all messages sent so far (both sides).
  - MAC computed using master_secret.
  - Server can only compute MAC if it can decrypt pre_master_secret in M3.
SSL Handshake Protocol Run

Summary:

M1: C → S: ClientHello
M2: S → C: ServerHello, ServerCertChain, ServerHelloDone
M3: C → S: ClientKeyExchange, ChangeCipherSpec, ClientFinished
M4: S → C: ChangeCipherSpec, ServerFinished

SSL Handshake Protocol Run

1. Is the client authenticated to the server in this protocol run?
2. Can an adversary learn the value of pre_master_secret?
3. Is the server authenticated to the client?
4. Who creates the session key in a way SSL connection?
5. Who creates the session key in a two way SSL connection?

1. No!
2. No! Client has validated server’s public key; Only holder of private key can decrypt ClientKeyExchange to learn pre_master_secret.
3. Yes! ServerFinished includes MAC on nonces computed using key derived from pre_master_secret.
4. The client
5. The client

Other SSL Handshake Protocol Runs

• Many optional/situation-dependent protocol messages:
  – M2 (S → C) can include:
    • ServerKeyExchange (e.g. for DH key exchange).
    • CertRequest (for client authentication).
  – M3 (C → S) can include:
    • ClientCert (for client authentication),
    • ClientCertVerify (for client authentication).
• For details, see RFC 2246 (TLS).

SSL Handshake Protocol – Additional Features

• SSL Handshake Protocol supports session resumption and ciphersuite re-negotiation.
  – Allows authentication and shared secrets to be reused across multiple connections.
    • Eg. next webpage from same website.
  – Allows re-keying of current connection using fresh nonces.
  – Allows change of ciphersuite during session.
    – ClientHello quotes old SessionID.
    – Both sides contribute new nonces, update master_secret and key_block.
  – All protected by existing Record Protocol.
Other SSL Protocols

- **Alert protocol.**
  - Management of SSL session, error messages.
  - Fatal errors and warnings.
- **Change cipher spec (CCS) protocol.**
  - Used to switch to agreed cipher-spec
- **Both protocols run over Record Protocol (so peers of Handshake Protocol).**

Alert Protocol

- **Convey SSL related alerts to the peer.**
- **Compressed and encrypted.**
- **Two types of alerts**
  - Fatal (5)
    - SSL immediately terminates the connection.
    - Examples
      - Unexpected message
      - Bad_record_mac
  - Warning (7)
    - Examples
      - Close_notify
      - No_certificate

Change Cipher Spec Protocol

- **Session State**
  - Current state
    - The session state in effect
  - Pending state
    - The session being negotiated.
- **Change Cipher Spec Protocol**
  - Cause the pending state to be copied into the current state.

Application Ports Used with SSL

- **https** 443
- **smtps** 465
- **nntps** 563
- **ldaps** 636
- **pop3s** 995
- **ftp-data** 889
- **ftps** 990
- **imaps** 991
SSL and TLS

TLS1.0 = SSL3.0 with minor differences.
• TLS signalled by version number 3.1.
• Use of newer HMAC for MAC algorithm.
• Differences in cipher suites
• Additional alert codes.
• More client certificate types.
• Minor changes in some cryptographic computations
• And more ….

SSL/TLS Applications

• Secure e-commerce using SSL/TLS.
  – Client authentication not needed until client decides to buy something.
  – SSL provides secure channel for sending credit card information.
  – Client authenticated using credit card information, merchant bears (most of) risk.
• No guarantees about what happens to client data (including credit card details) after session: may be stored on insecure server.

Some SSL/TLS Security Flaws

• (Historical) flaws in random number generation for SSL.
  – Low quality RNG leads to predictable session keys.
  – Goldberg and Wagner, Dr. Dobb’s Journal, Jan. 1996.
• Flaws in error reporting.
  – (differing response times by server in event of padding failure and MAC failure) + (analysis of padding method for CBC-mode) = recovery of SSL plaintext.
• Timing attacks.
  – analysis of OpenSSL server response times allows attacker in same LAN segment to derive server’s private key.
  – Boneh and Brumley, 12th Usenix Security Symposium.

Comparing IPsec & SSL/TLS I

• Both have initial (authenticated) key establishment then key derivation.
  – IKE in IPsec
  – Handshake Protocol in SSL/TLS
• Both protect ciphersuite negotiation.
• Both use keys established to build a ‘secure channel’.
### Comparing IPsec, SSL/TLS II

- Operate at different network layers.
  - This brings pros and cons for each protocol suite.
  - Recall “Where shall we put security?” discussion.

- All practical, but not simple.
  - Complexity leads to vulnerabilities.
  - Complexity makes configuration and management harder.
  - Complexity can create computational bottlenecks.
  - Complexity necessary to give both flexibility and security.

### Comparing IPsec & SSL/TLS III

Security of both undermined by:

- Implementation weaknesses.
- Weak server platform security.
  - Worms, malicious code, rootkits,…
- Weak user platform security.
  - Keystroke loggers, malware,…
- Limited deployment of certificates and infrastructure to support them.
  - Especially client certificates.
- Lack of user awareness and education.
  - Users click-through on certificate warnings.
  - Users fail to check URLs.
  - Users send sensitive account details to bogus websites (“phishing”) in response to official-looking e-mail.