Bachelor of Computer Science

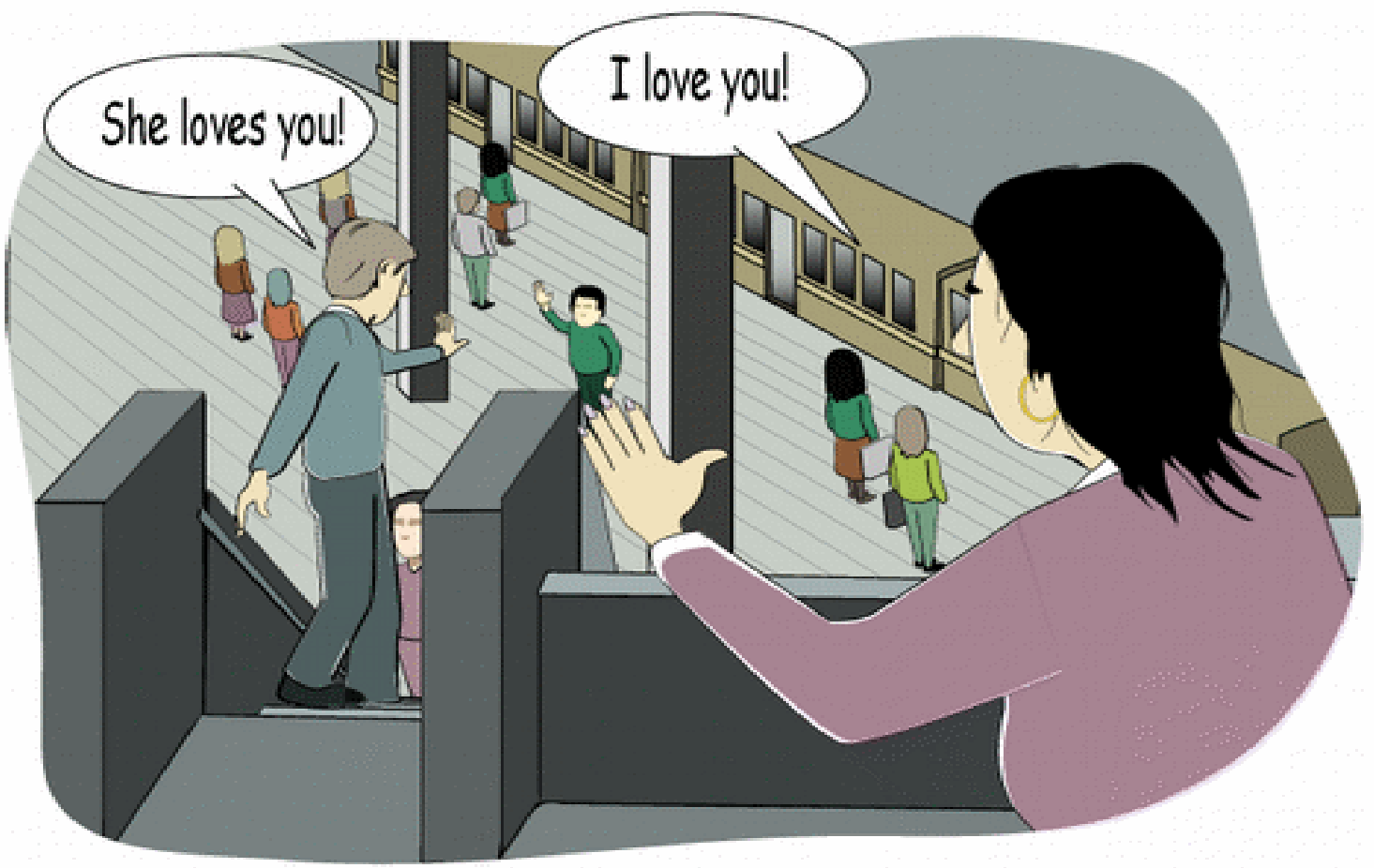
SCS2214 - Information System Security

Handout 7 - Web Security

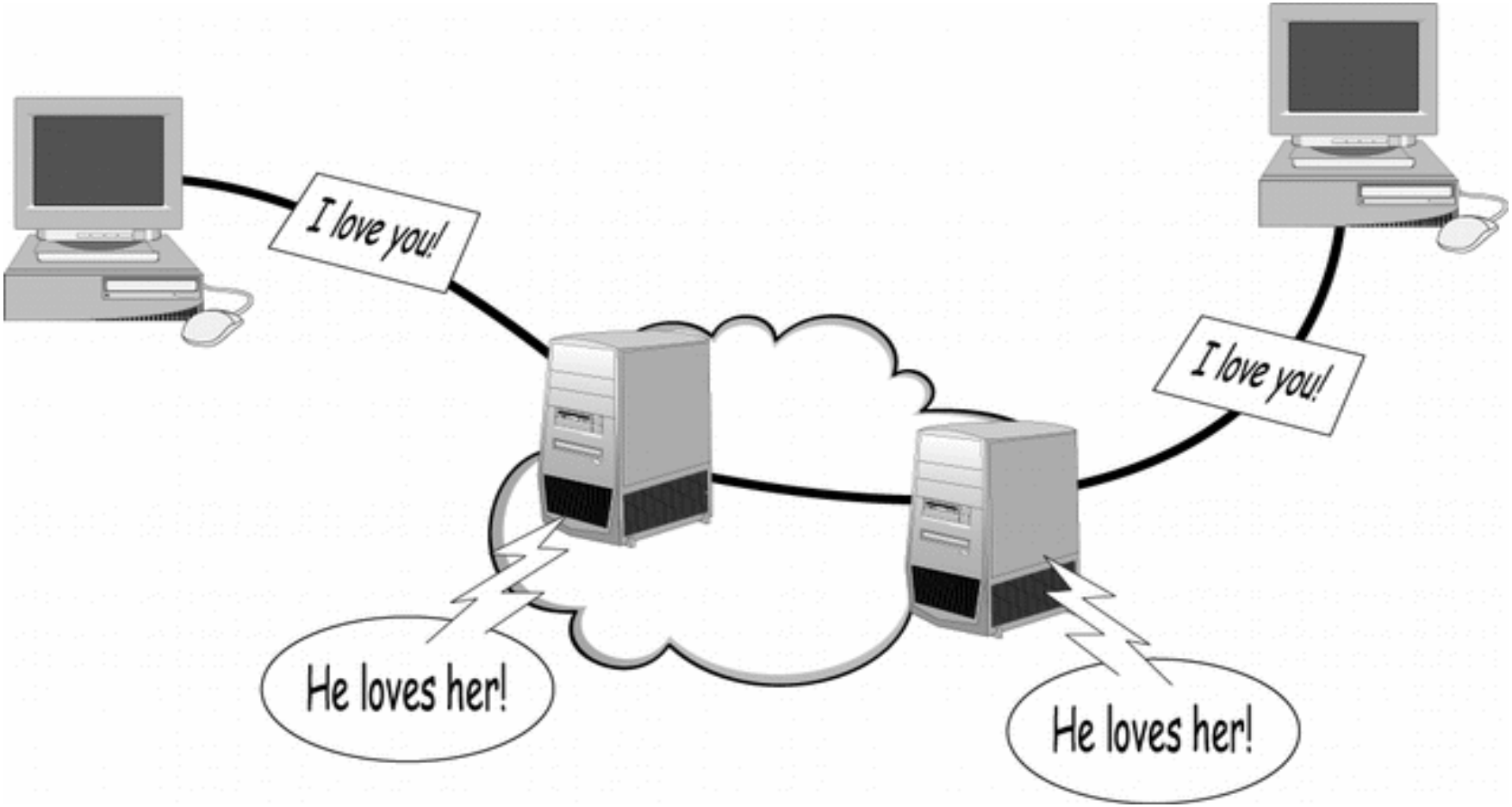
Kasun de Zoysa
kasun@ucsc.cmb.ac.lk

UNIVERSITY OF COLOMBO SCHOOL OF COMPUTING

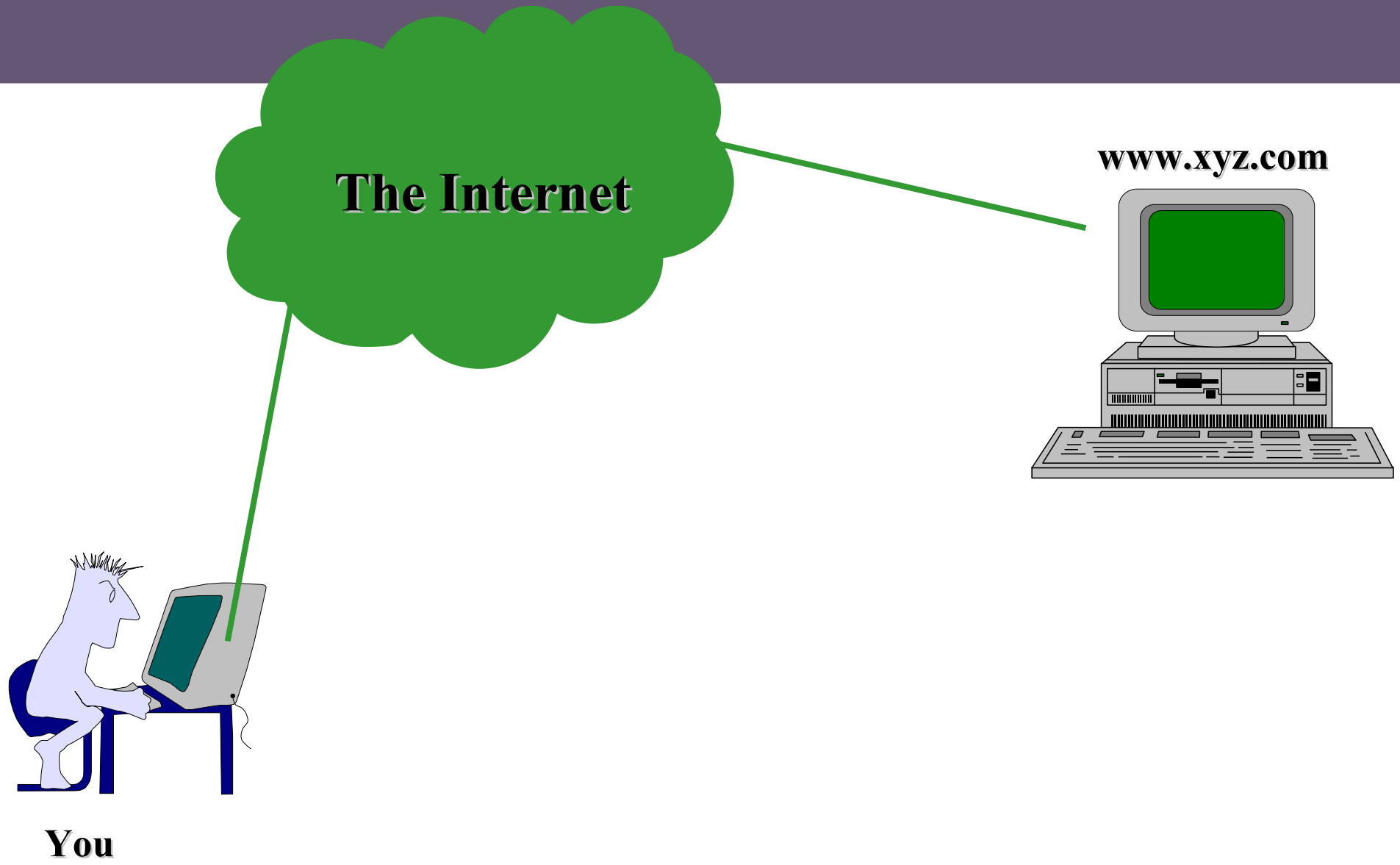
How the Internet Works -1



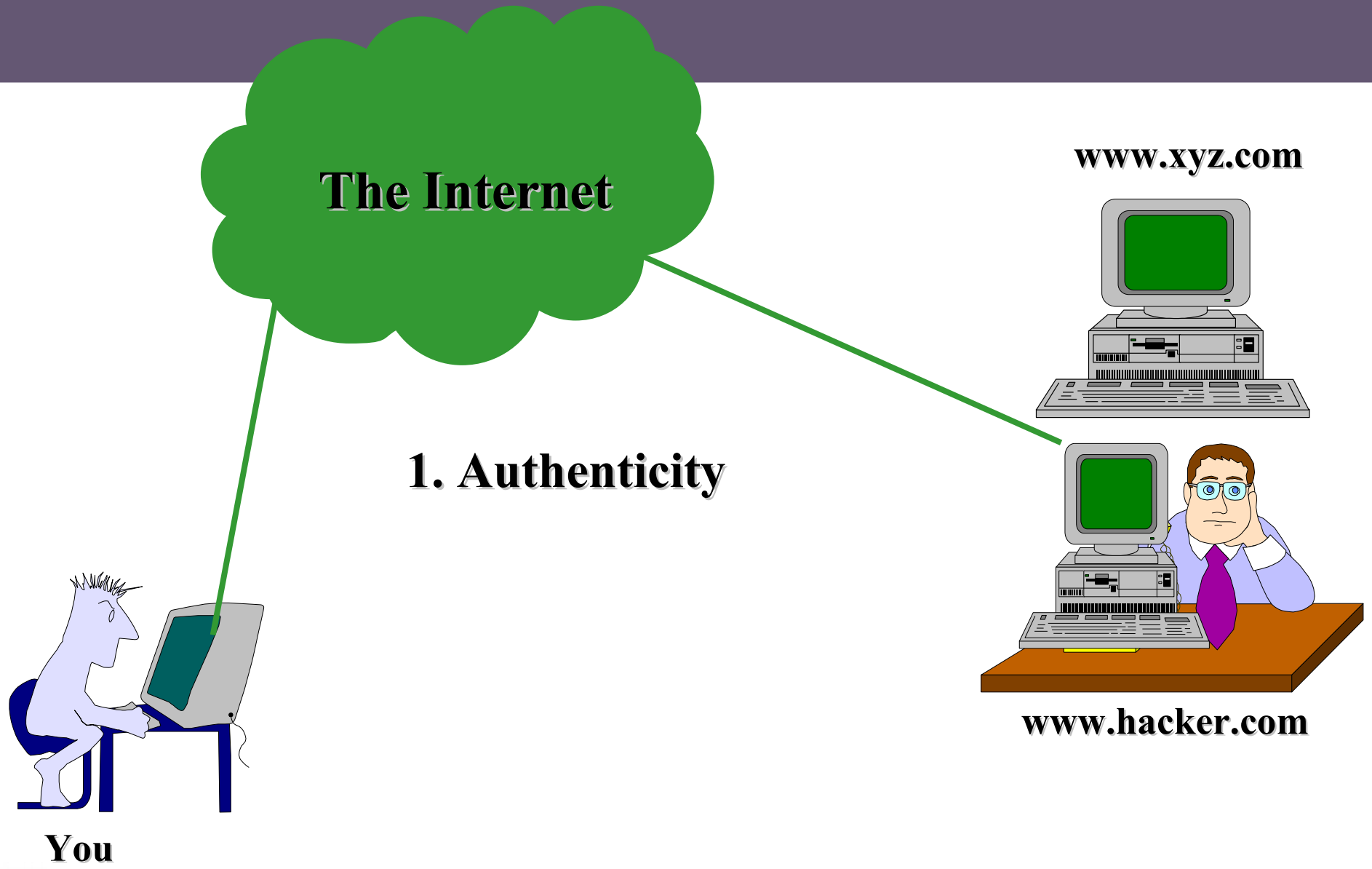
How the Internet Works -2



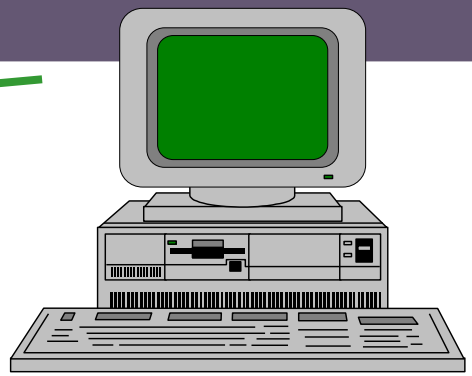
Security Requirements and User Needs



Security Requirements and User Needs

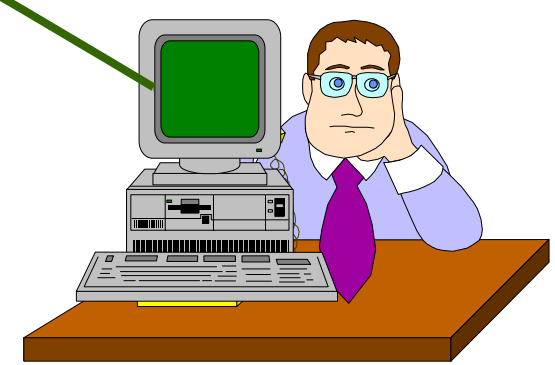


Security Requirements and User Needs

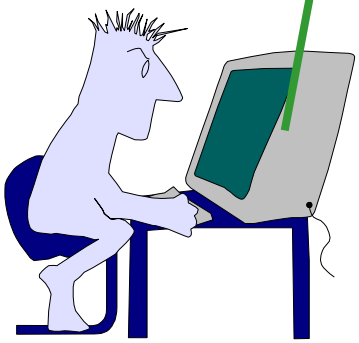


www.xyz.com

1. Authenticity
2. Integrity

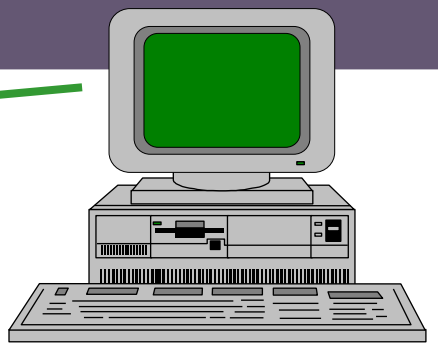


www.hacker.com



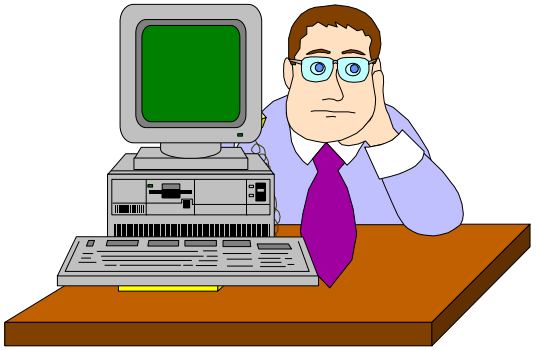
You

Security Requirements and User Needs

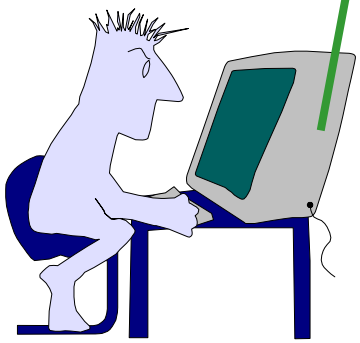


www.xyz.com

1. Authenticity
2. Integrity
3. Confidentiality

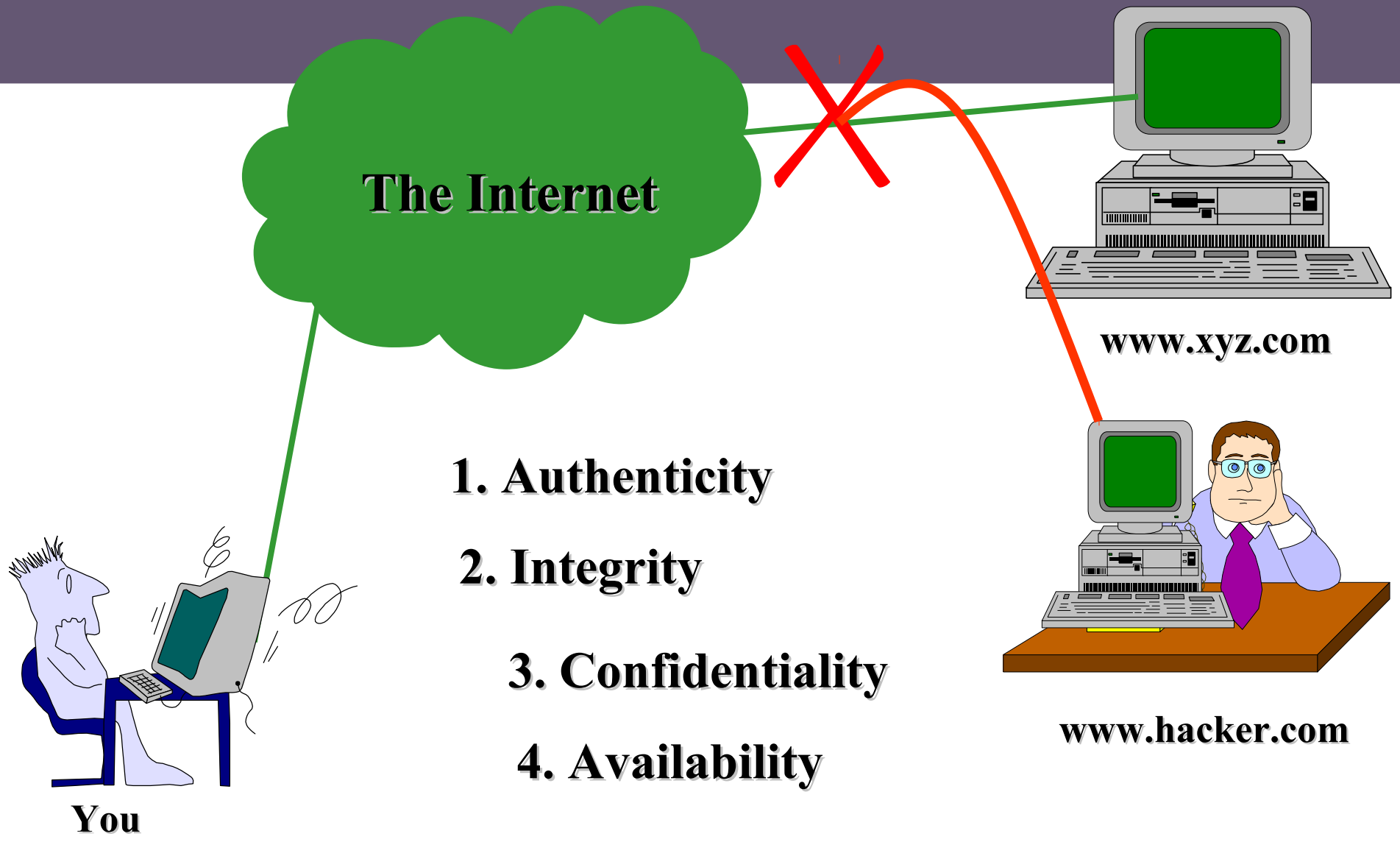


www.hacker.com

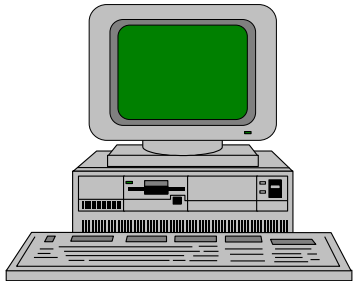


You

Security Requirements and User Needs

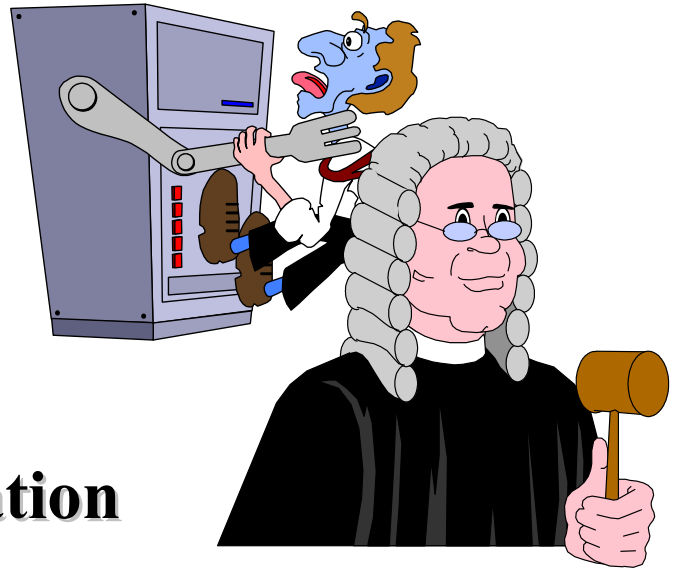


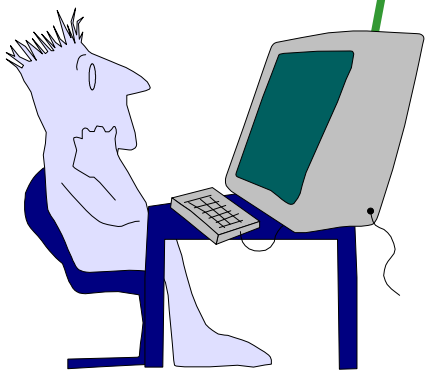
Security Requirements and User Needs



www.xyz.com

1. Authenticity
2. Integrity
3. Confidentiality
4. Availability
5. Non-repudiation





You

Solutions

Protection at Two Levels :

1. **Lower Level (Channel protection)**
   (Communication security services)

2. **Application/User Level**
   (Application level security services)

Solutions

Lower Level (Channel Protection)

-- protection at the packet (message) level
-- no protection at the document level
-- communications security only
-- efficient for network protection
-- not suitable for application security services
-- security services between the browser and server, not between user and application(s)

Solutions

Higher Level (Applications Protection)

-- protection at the documents level
-- no protection at the communication level
-- communications security implicit
-- not efficient for network protection
-- suitable for application security services
-- security services between the user
and server applications

Internet Cryptographic Protocols

- **Cybercash :** Electronic Funds Transactions, RFC1898
- **DNSSEC :** Domain Name System, RFC2065
- **IPSec :** Packet-Level Encryption, RFC2401
- **PCT :** TCP/IP-level Encryption
- **PGP :** E-Mail, RFC2015
- **S/MIME :** E-Mail, RFC2311,RFC2634
- **S-HTTP :** Web Browsing, RFC2660
- **SET :** Electronic Funds Transactions
- **SSL :** TCP/IP-level Encryption, Netscape
- **SSH :** Remote Login
- **TLS :** TCP/IP-level Encryption, RFC2246

Secure Socket Layer History

- SSL 1.0 Netscape 1994
- S-HTTP (web only)
- SSL 2.0 Netscape (buggy)
- PCT Microsoft (loser) 1996
- SSL 3.0 Netscape
- TLS 1.0 IETF 1999
- TLS 1.2 now dominant

TLS: Transport Layer Security

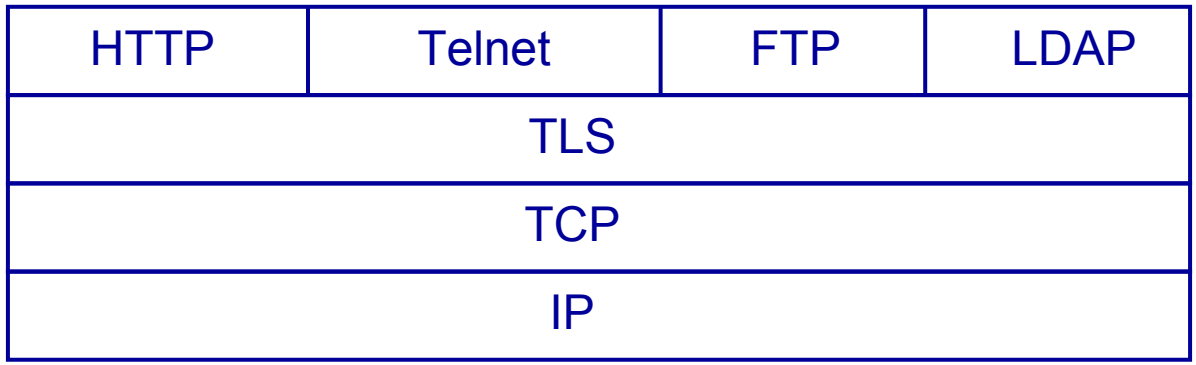
- *formerly known as*
  SSL: Secure Sockets Layer
- Addresses issues of privacy, integrity and authentication
  - What is it?
  - How does it address the issues?
  - How is it used

TLS

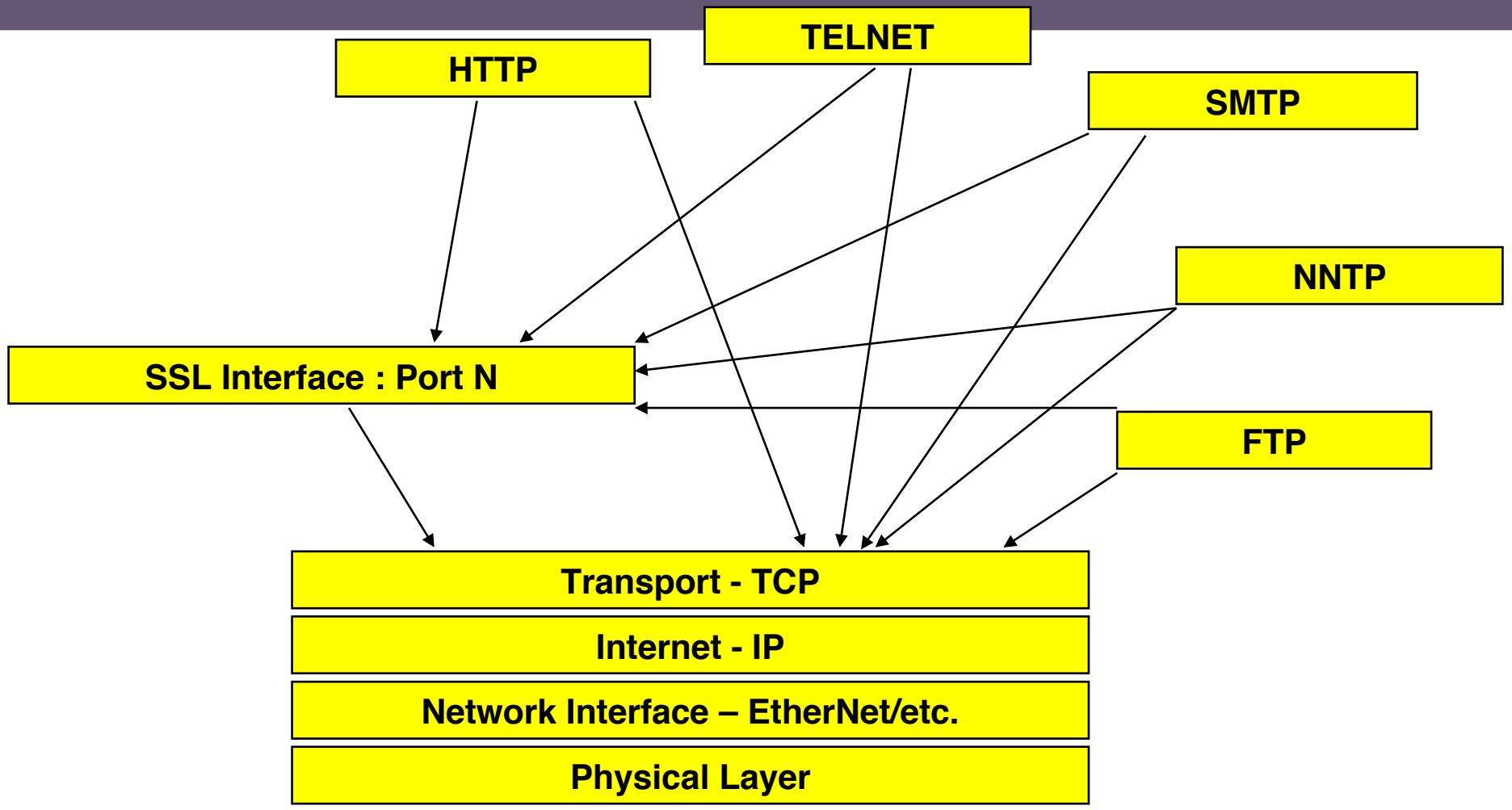
- “TLS, more commonly known as SSL”
- RFC2246 : TLS Protocol Version 1.0 1/99
- RFC2487 : SMTP over TLS
- RFC2712 : Adding Kerberos to TLS
- RFC2716 : PPP TLS
- RFC2817 : Upgrading to TLS within HTTP/1.1
- RFC2818 : HTTP over TLS
- RFC2830 : TLS for Lightweight Directory Access Protocol (LDAP)

What is TLS?

- Protocol layer
- Requires reliable transport layer (e.g. TCP)
- Supports any application protocols



Protocol Stack

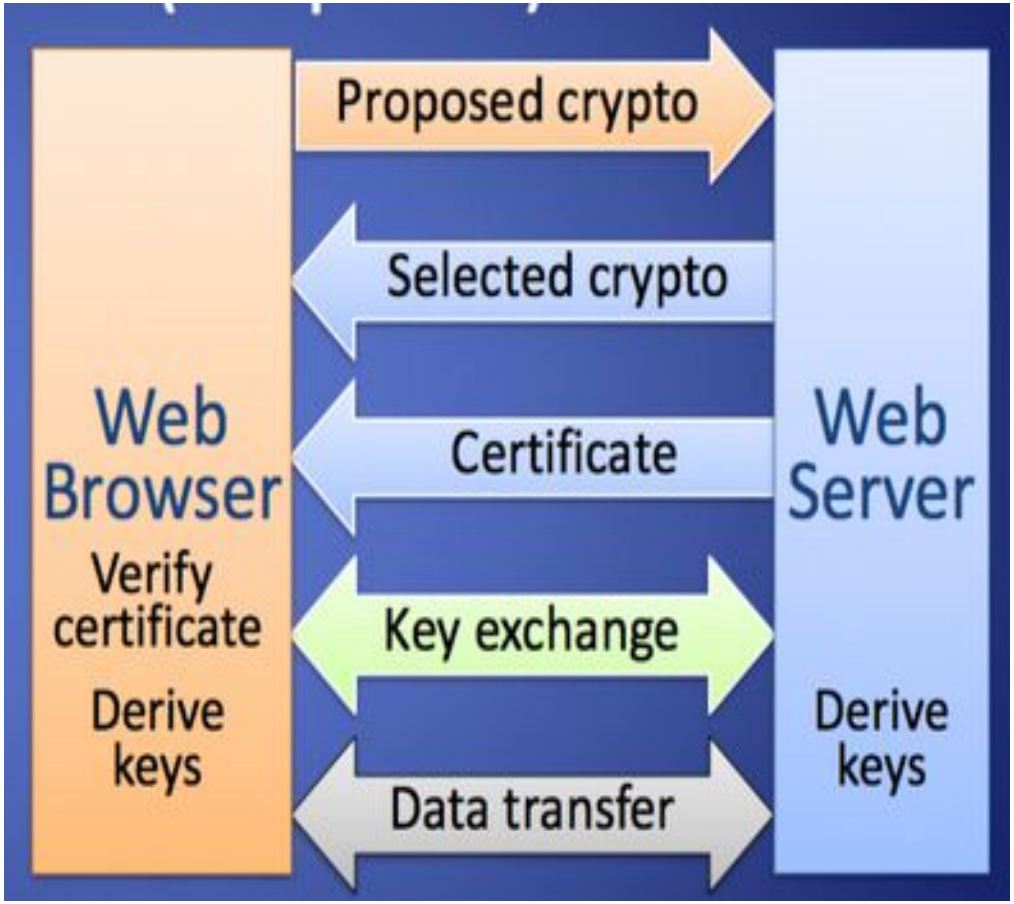


TLS: Overview

- Establish a session
  - Agree on algorithms
  - Share secrets
  - Perform authentication
- Transfer application data
  - Ensure privacy and integrity

TLS Overview

- Browser sends supported crypto algorithms
- Server picks strongest algorithms it supports
- Server sends certificate (chain)
- Client verifies certificate (chain)
- Client and server agree on secret value R by exchanging messages
- Secret value R is used to derive keys for symmetric encryption and hash-based authentication of subsequent data transfer

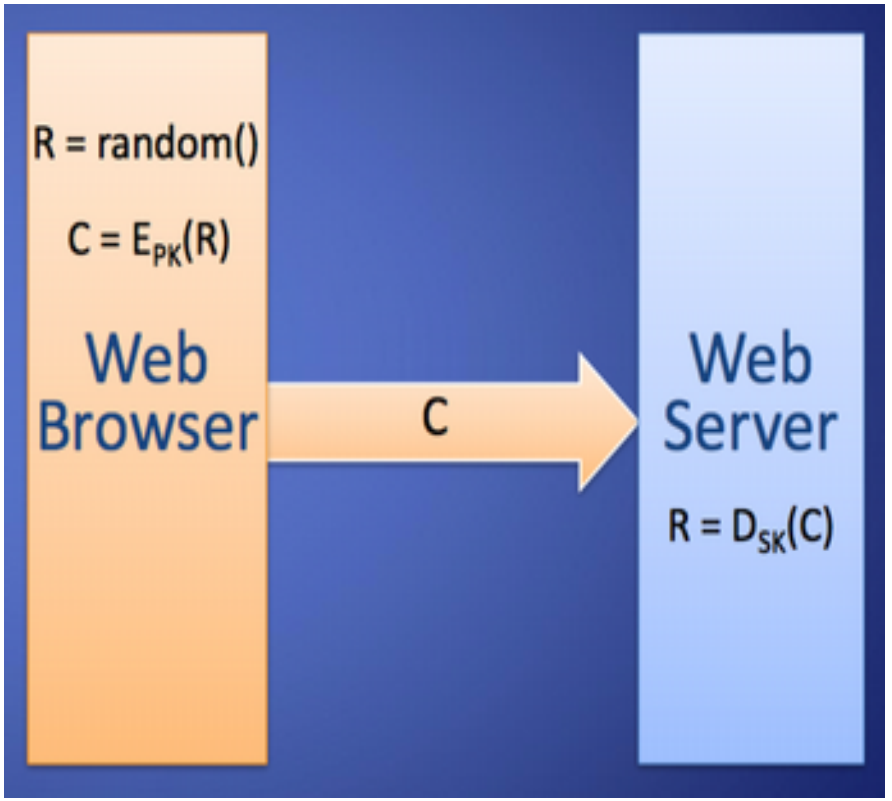


TLS:Key Exchange

- Need secure method to exchange secret key
- Use public key encryption for this
  - "key pair" is used - either one can encrypt and then the other can decrypt
  - slower than conventional cryptography
  - share one key, keep the other private
- Choices are RSA or Diffie-Hellman

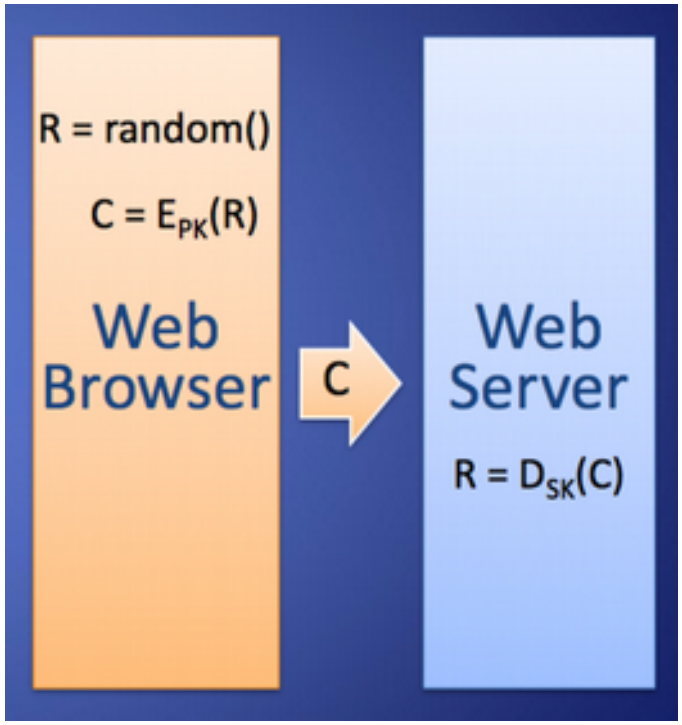
Basic Key Exchange

- Called RSA key exchange for historical reasons
- Client generates random secret value R
- Client encrypts R with public key, PK, of server C = EPK(R)
- Client sends C to server
- Server decrypts C with private key, SK, of server R = DSK(C)

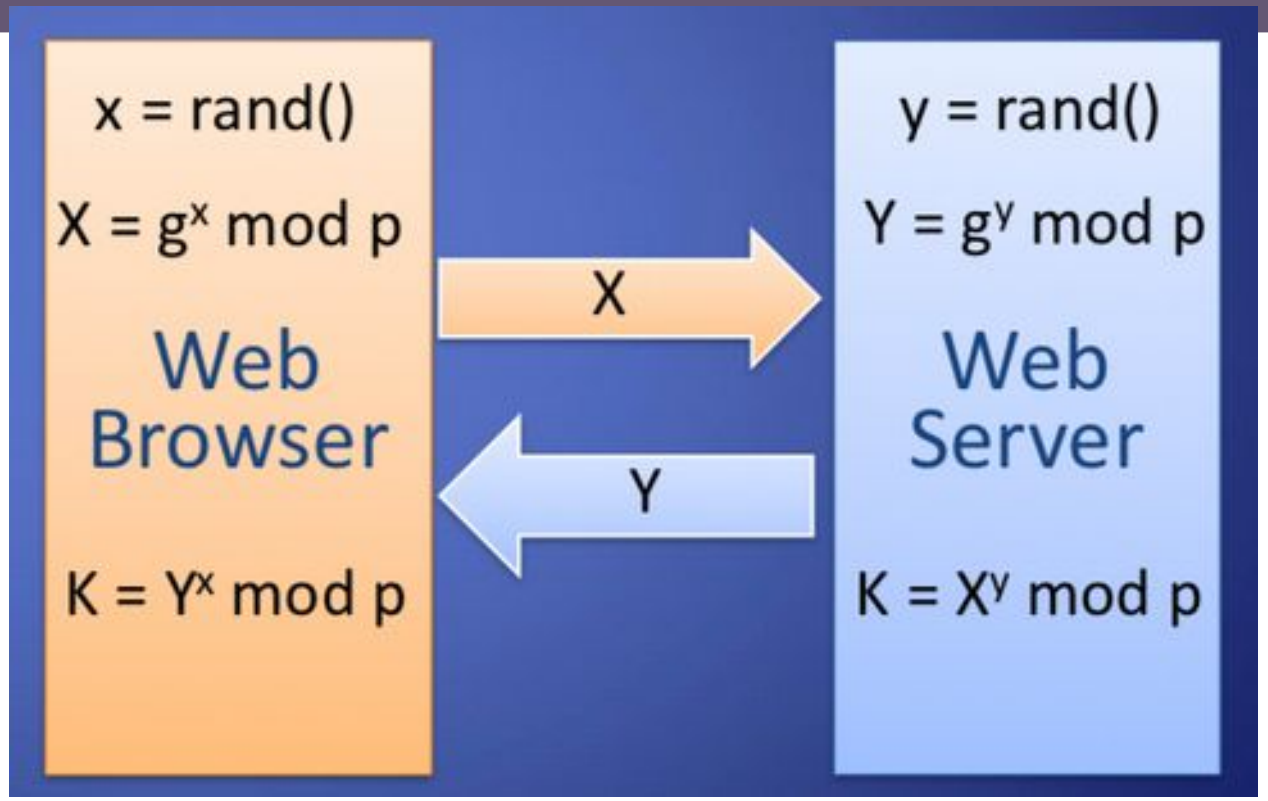


Forward Secrecy

- Compromise of public-key encryption private keys does not break confidentiality of past messages
- TLS with basic key exchange does not provide forward secrecy
- Attacker eavesdrop and stores communication
- If server's private key is compromised, attacker finds secret value R in key exchange and derives encryption keys

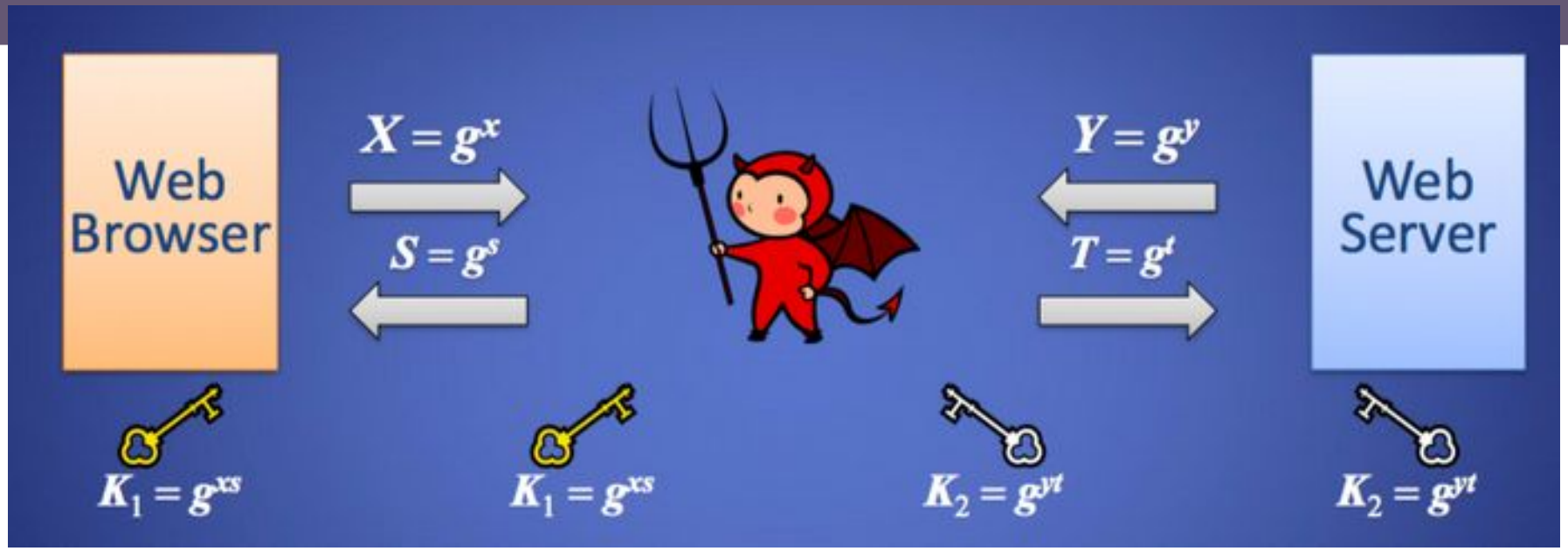


Diffie Hellman Key Exchange



Achieves forward secrecy

Attacker in the Middle



Solution:
Browser and server send signed X and Y respectively
Requires each to know the public key of the other

TLS: Privacy

- Encrypt message so it cannot be read
- Use conventional cryptography with shared key
  - DES, 3DES, AES
  - RC2, RC4
  - IDEA

TLS Encrypts

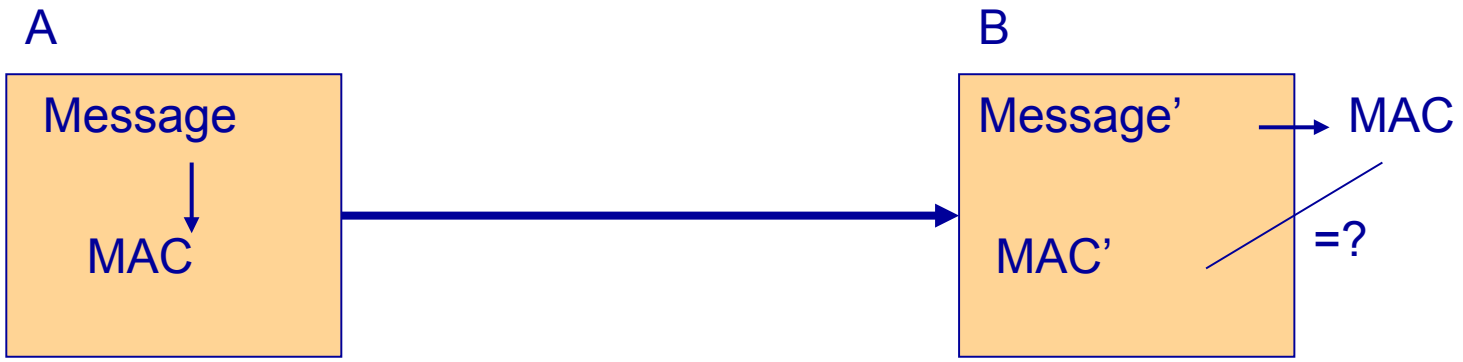
- ALL Browser-Server and Server-Browser except which-browser is talking to which-server
- URL of requested document
- Contents of requested document
- Contents of any submitted form fill-outs
- Cookies sent from browser to server
- Cookies sent from server to browser
- Contents of HTTP header
- Javascript communications
- Etc.

TLS: Integrity

- Compute fixed-length Message Authentication Code (MAC)
  - Includes hash of message
  - Includes a shared secret
  - Include sequence number
- Transmit MAC with message

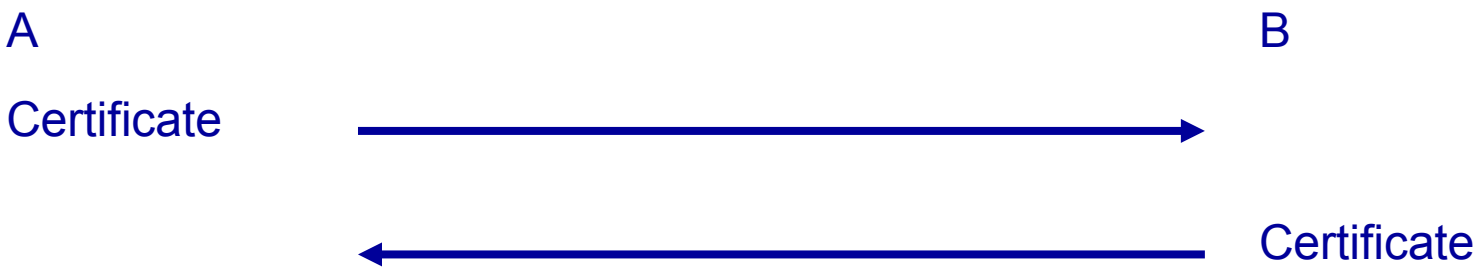
TLS: Integrity

- Receiver creates new MAC
  - should match transmitted MAC
- TLS allows MD5, SHA-1

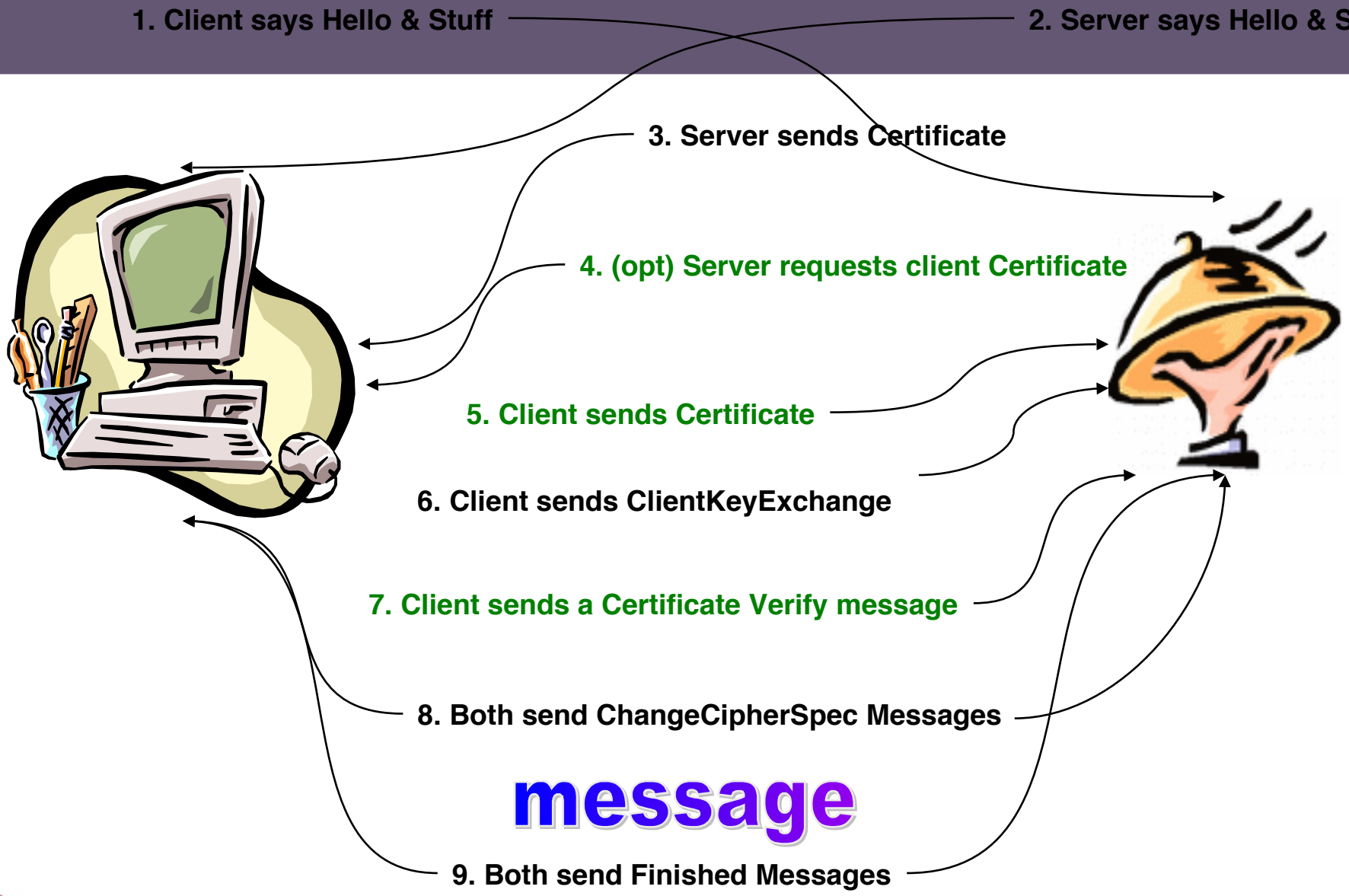


TLS: Authentication

- Verify identities of participants
- Client authentication is optional
- Certificate is used to associate identity with public key and other attributes

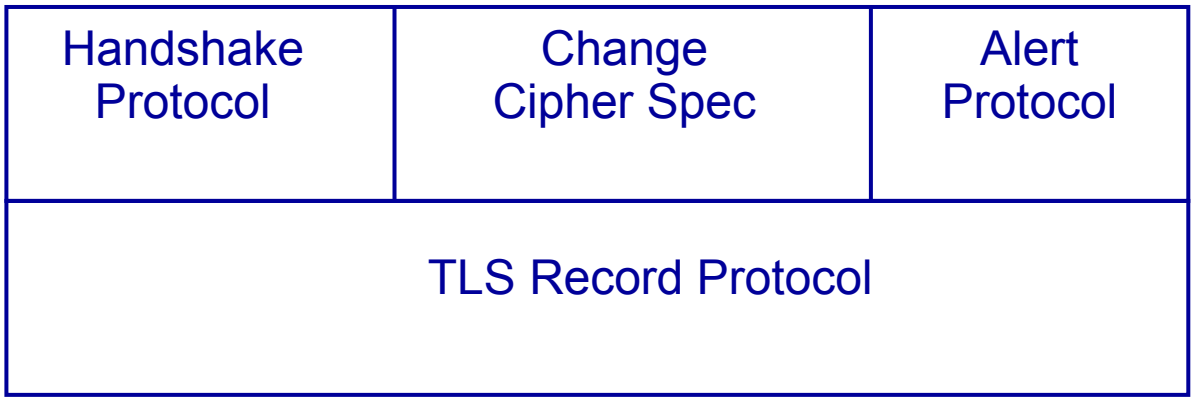


TLS Transaction

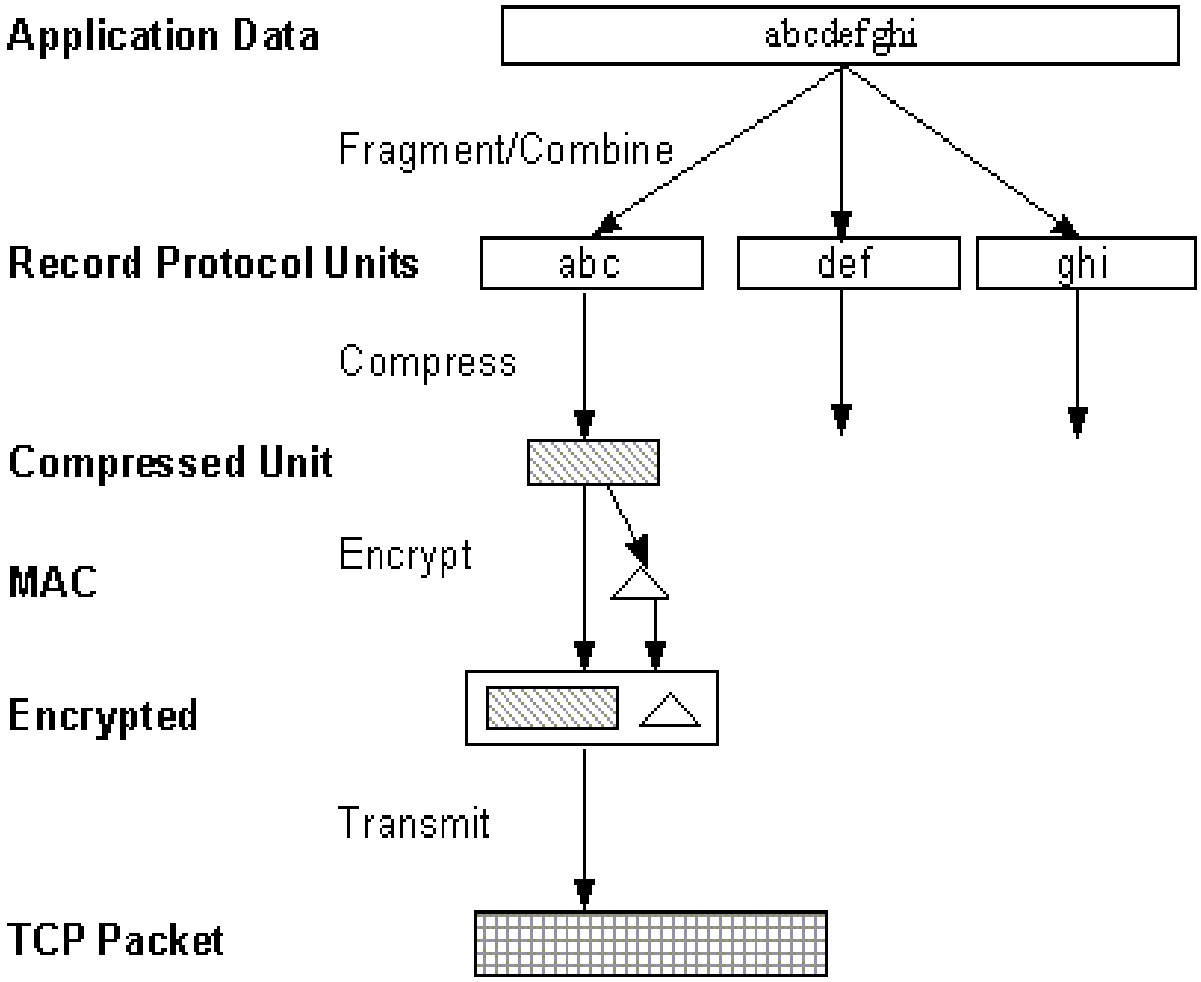


TLS: Architecture

- TLS defines Record Protocol to transfer application and TLS information
- A session is established using a Handshake Protocol



TLS: Record Protocol



TLS: HTTP Application

- HTTP most common TLS application
  - https://
- Requires TLS-capable web server
- Requires TLS-capable web browser
  - Netscape Navigator
  - Internet Explorer
  - Cryptozilla
    - Netscape Mozilla sources with SSLeay

Public Key Certificates

- X.509 Certificate associates public key with identity
- Certification Authority (CA) creates certificate
  - Adheres to policies and verifies identity
  - Signs certificate
- User of Certificate must ensure it is valid

Validating a Certificate

- Must recognize accepted CA in certificate chain
  - One CA may issue certificate for another CA
- Must verify that certificate has not been revoked
  - CA publishes Certificate Revocation List (CRL)

X.509: Certificate Content

- Version
- Serial Number
- Signature Algorithm Identifier
  - Object Identifier (OID)
  - e.g. id-dsa: {iso(1) member-body(2) us(840) x9-57 (10040) x9algorithm(4) 1}
- Issuer (CA) X.500 name
- Validity Period (Start,End)
- **Subject X.500 name**
- **Subject Public Key**
  - Algorithm
  - Value
- Issuer Unique Id (Version 2 ,3)
- Subject Unique Id (Version 2,3)
- Extensions (version 3)
  - optional
- **CA digital Signature**

Subject Names

- X.500 Distinguished Name (DN)
- Associated with node in hierarchical directory (X.500)
- Each node has Relative Distinguished Name (RDN)
  - Path for parent node
  - Unique set of attribute/value pairs for this node

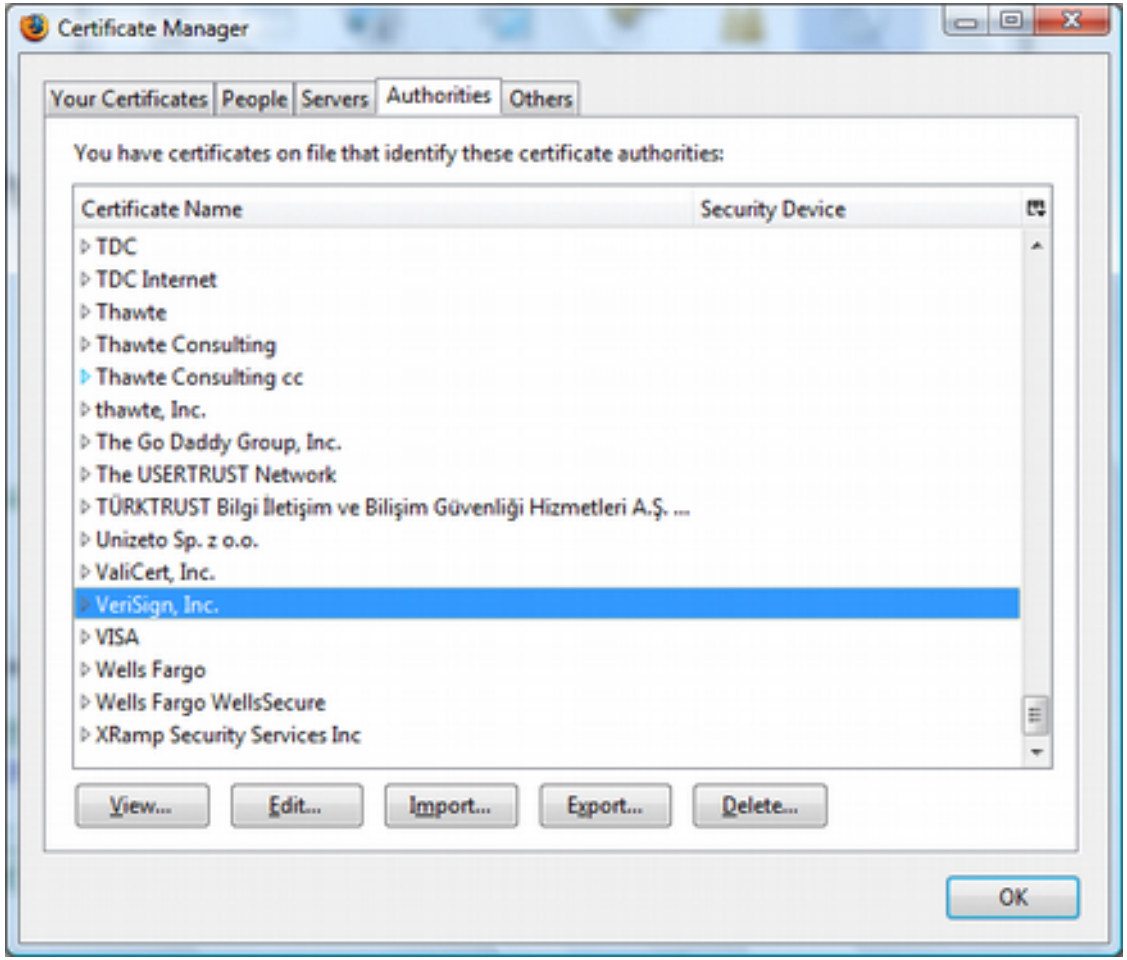
Example Subject Name

- Country at Highest Level (e.g. US)
- Organization typically at next level (e.g. CertCo)
- Individual below (e.g. Common Name “Kasun” with Id = 1)

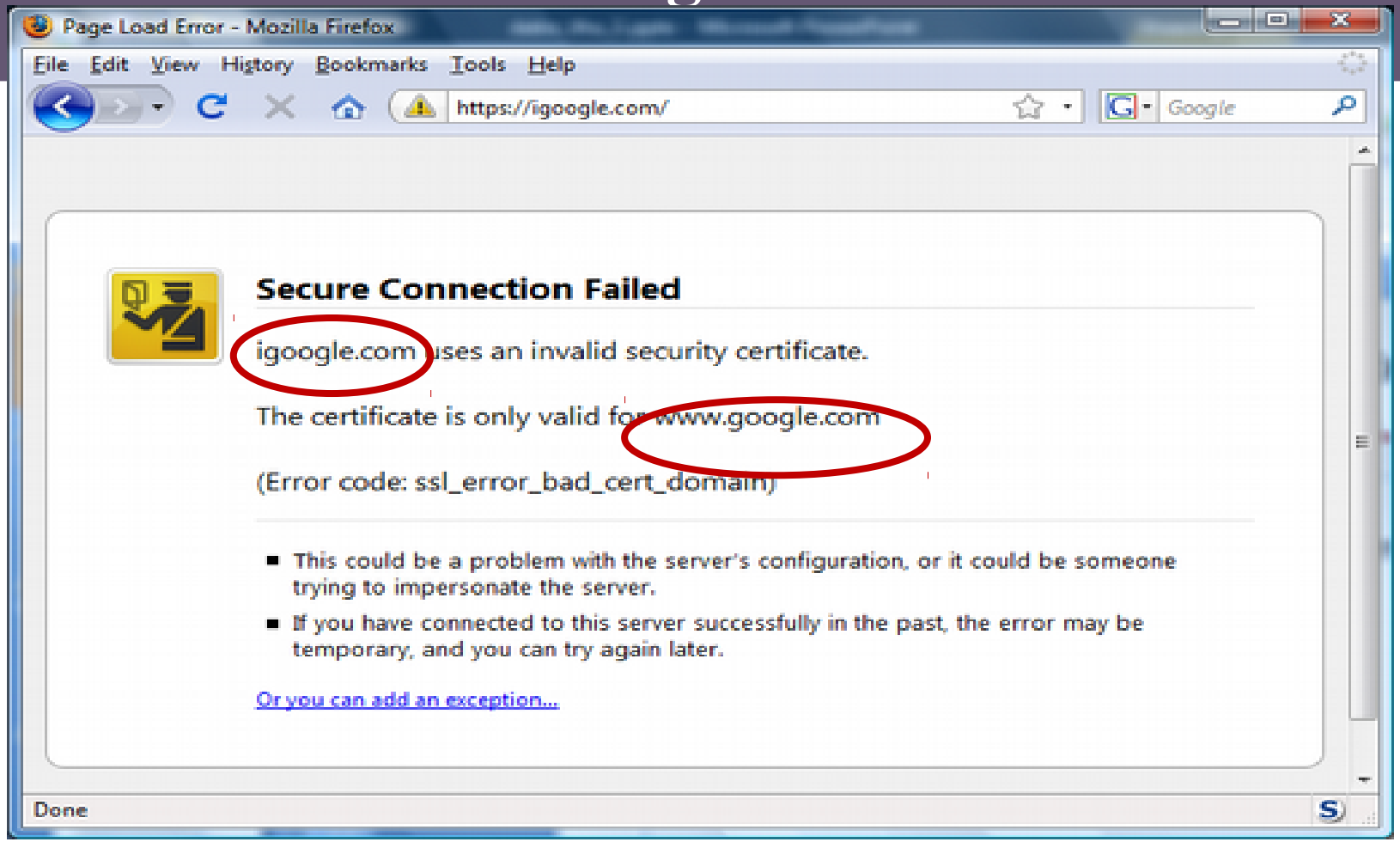
  DN = {
  - C=LK;
  - O=UCSC;
  - CN=Kasun, ID=1}

Certificate Authorities

Browsers accept certificates from a large number of CAs



Firefox: Invalid cert dialog



Firefox 3.0: <u>Four clicks</u> to get firefox to accept cert

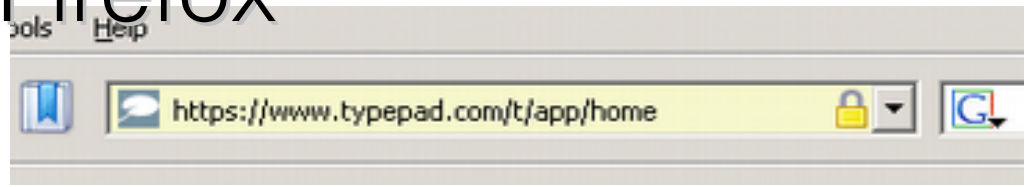
- page is displayed with full HTTPS indicators

SSL Indicators

- Microsoft IE

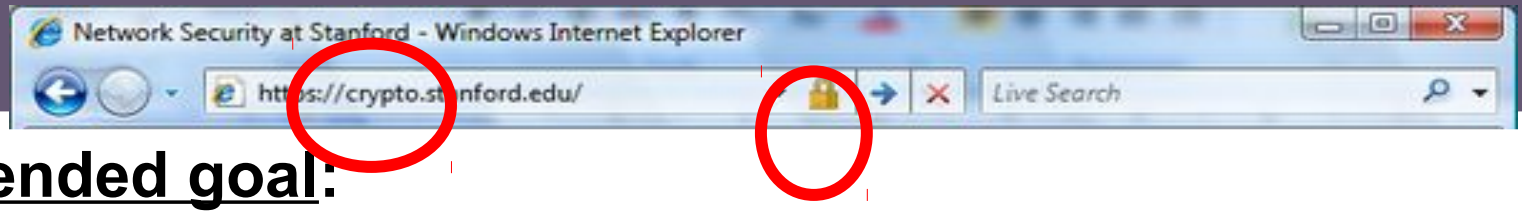
- Mozilla

- Firefox



- Safari

The lock icon: SSL indicator



Intended goal:

- Provide user with identity of page origin
- Indicate to user that page contents were not viewed or modified by a **network attacker**

In reality:

- Origin ID is not always helpful
- Many other problems (next few slides)

The lock icon: SSL indicator

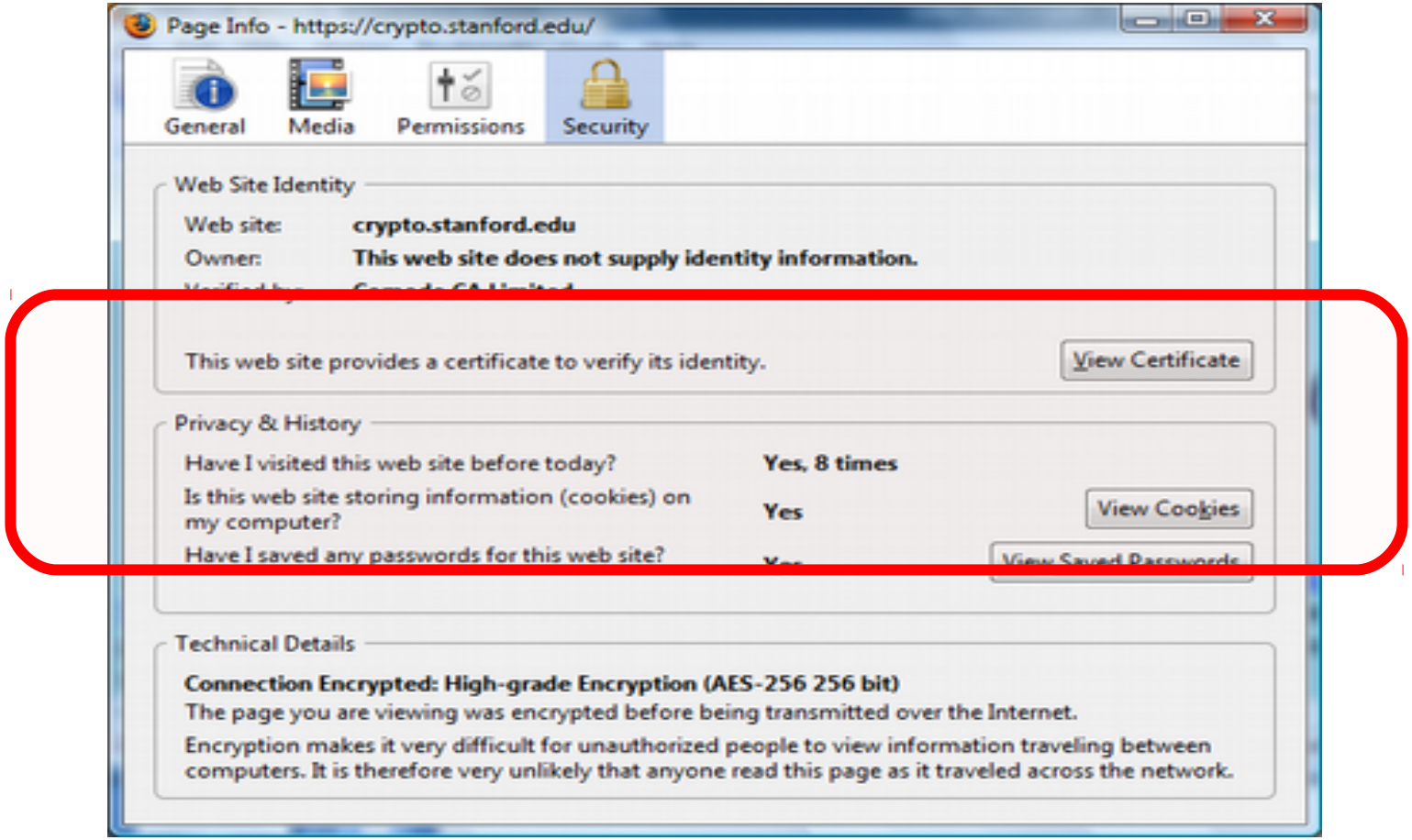
- **All elements on the page fetched using HTTPS**

  (with some exceptions)

- **For all elements:**
  - HTTPS cert issued by a CA trusted by browser
  - HTTPS cert is valid (e.g. not expired)
  - CommonName in cert matches domain in URL

The lock UI: help users authenticate site

Firefox 3: clicking on bottom lock icon gives

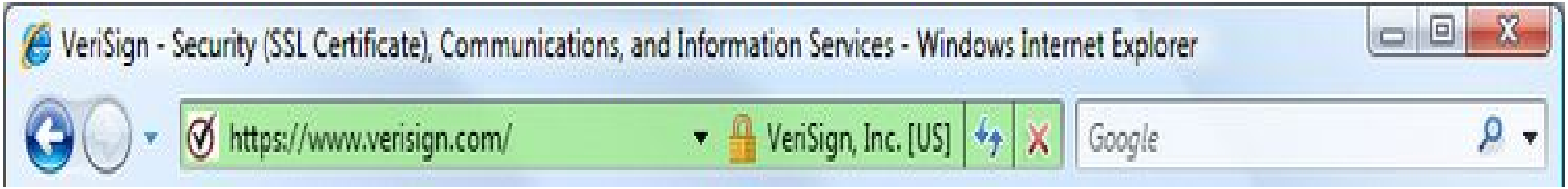


Version 3 Certificates

- Version 3 X.509 Certificates support alternative name formats as extensions
  - X.500 names
  - Internet domain names
  - e-mail addresses
  - URLs
- Certificate may include more than one name

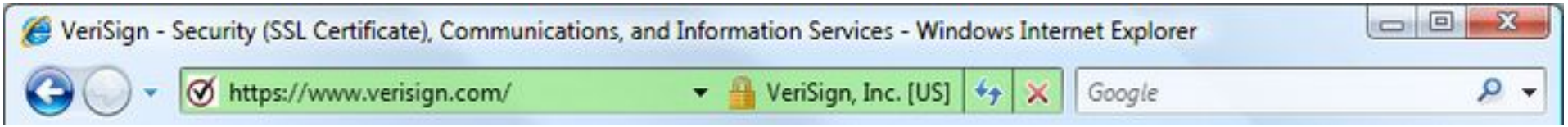
Extended validation (EV) certs

- **Harder to obtain than regular certs**
  - requires human lawyer at CA to approve cert request
- **Designed for banks and large e-commerce sites**



- **Helps block "semantic attacks":** www.bankofthe**vv**est.com

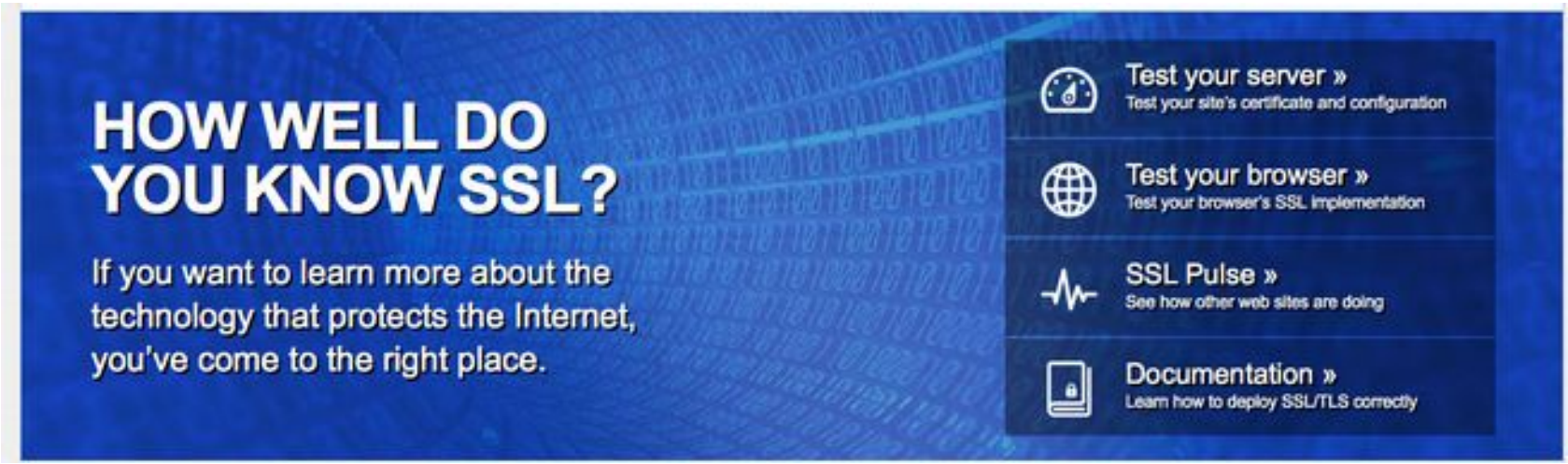
Extended validation (EV) certs



An Extended Validation Certificate (EV) is an X.509 public key certificate issued according to a specific set of identity verification criteria. These criteria require extensive verification of the requesting entity's identity by the certificate authority (CA) before a certificate is issued.

Certificates issued by a CA under the EV guidelines are not structurally different from other certificates (and hence provide no stronger cryptography than other, cheaper certificates),

SSLABS – www.ssllabs.com



Secure Sockets Layer – Apache

- Compile and install mod_ssl module.
- Create a public/private key pair.
- Get public key signed by certificate authority, yielding a certificate.
- Install certificate and configure Apache to find it.
- Restart

Create Self-signed Certificate

You can generate a self-signed host certificate using the following command:

```
openssl req -new -x509 -out host.pem
```

(Your private key will be saved to privkey.pem file and self-signed certificate will be saved to host.pem file.)

Creating a Certificate Request

To create a certificate request, use the following command:

openssl req -new -nodes -out req.pem -keyout key.pem

(Your private key will be saved to key.pem file and certificate request will be saved to req.pem file.)

req.pem:

```
-----BEGIN CERTIFICATE REQUEST-----
MIIBlDCB/gIBADBVMQswCQYDVQQGEwJMSzEQMA4GA1UEBxMHQ29sb21ibzEMMAoG
A1UEChMDQ01CMQ0wCwYDVQQLEwRVQ1NDMRcwFQYDVQQDEw51Y3NjLmNtYi5hYy5s
azCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA9XZEtFxoVbGhH9nrWKRi1avK
lMKKobVkgS99b9bcwnJ6zh7ZXwoiNBO1UNyDUuWrxxlZxcChnzds0UvEHVJatPYM
8+XwQpOmobIK/3E9f9SYh6OVbNxAIoLAXXoHBzV8YysyuxqEPFqmZW94TnfTUFWC
TTuwKPIourOZI1zhyW8CAwEAAaAAMA0GCSqGSIb3DQEBBAUAA4GBABBDlwxgDxqd
wpnfGUuRiIsp2C5KxHFAsVKvVwpRhlgdihcrYXpY2xNq1OTnqqS2dts2pO+xPuEP
nARAEnFABPxsqn95/mr+T91bah/2eBuhbJ9TjzxY9wWebTNMrk9CFygqlYldniizd
mhWMWQuqSnXSS5oC/+itEtAd64hWHv0Q
-----END CERTIFICATE REQUEST-----
```

Obtaining a Server Certificate

Convince a Certificate Authority to Sign your Certificate:

•Submit the req.pem file to Verisign or Thawte for signing (pay the fee) or
•Submit the req.pem file to **www.cacert.org** or **ca.cmb.ac.lk** (**Free**).

They will eventually mail you back a signed certificate.

Configure the <VirtualHost> block for the SSL-enabled site.

```
<VirtualHost 192.168.0.1:443>

DocumentRoot /var/www/html2
ServerName www.yourdomain.com
SSLEngine on
SSLCertificateFile /path/to/your_domain_name.crt
SSLCertificateKeyFile /path/to/your_private.key
SSLCertificateChainFile /path/to/CA.crt

</VirtualHost>
```

Adjust the file names to match your certificate and key files.

Authenticating with SSL

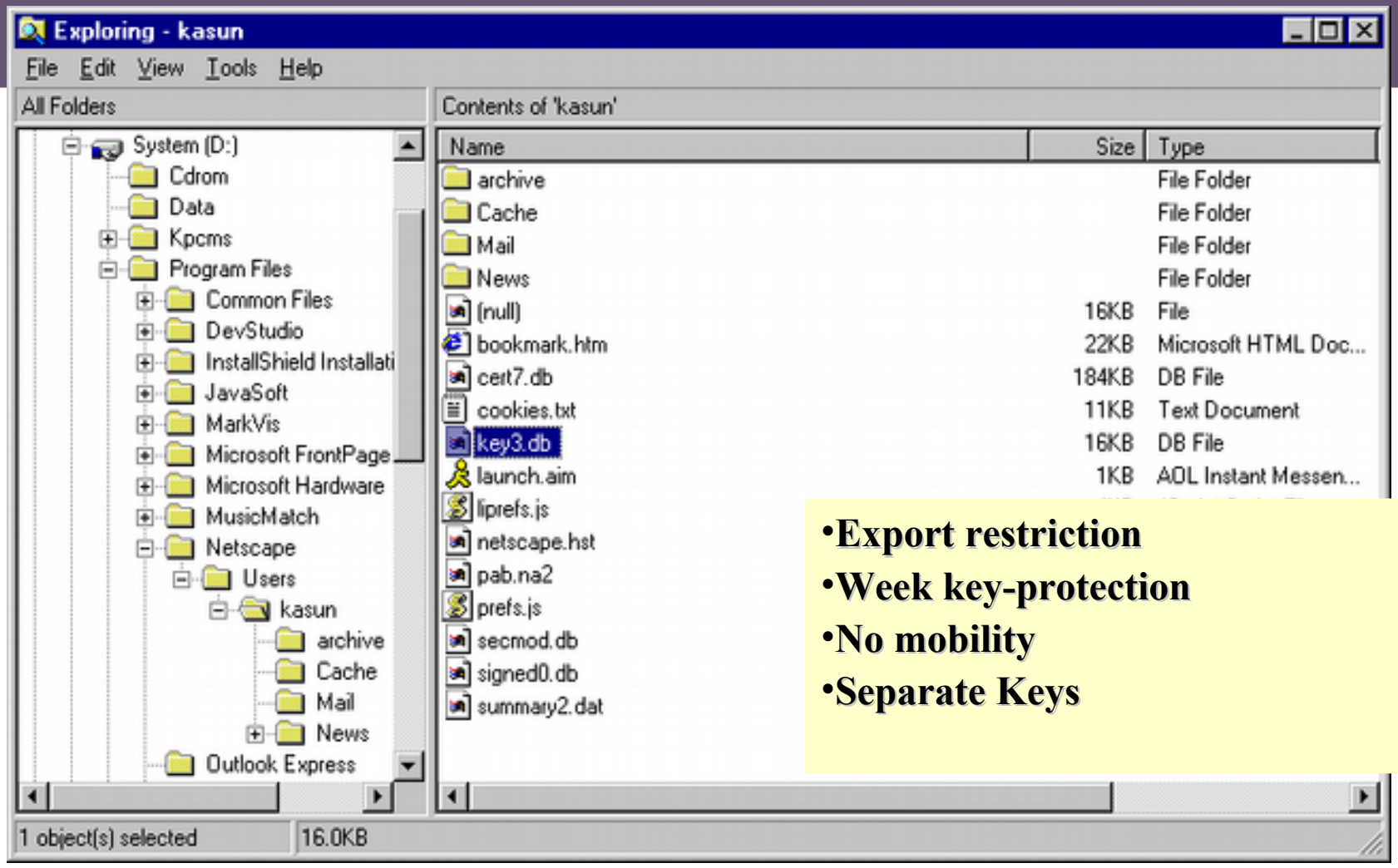
Give users of your intranet client certificates to authenticate with.
Advantages: No passwords to mess around with.
Disadvantages: Certificate management is **hard**.

Creating Client Certificates
OpenSSL will do that.

Problems with HTTPS and the Lock Icon

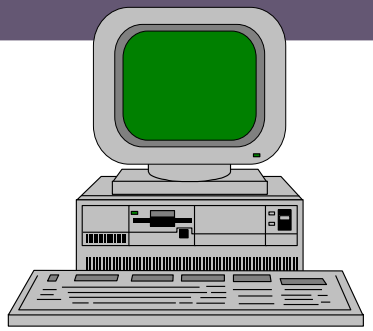
1. **Upgrade from HTTP to HTTPS**
2. **Semantic attacks on certs**
3. **Invalid certs**
4. **Mixed content**
   - HTTP and HTTPS on the same page
5. **Origin contamination**
   - Weak HTTPS page contaminates stronger HTTPS page

Private keys



- **Export restriction**
- **Week key-protection**
- **No mobility**
- **Separate Keys**

Man in the Middle



SSL Hello

Server Cert

SSL Hello

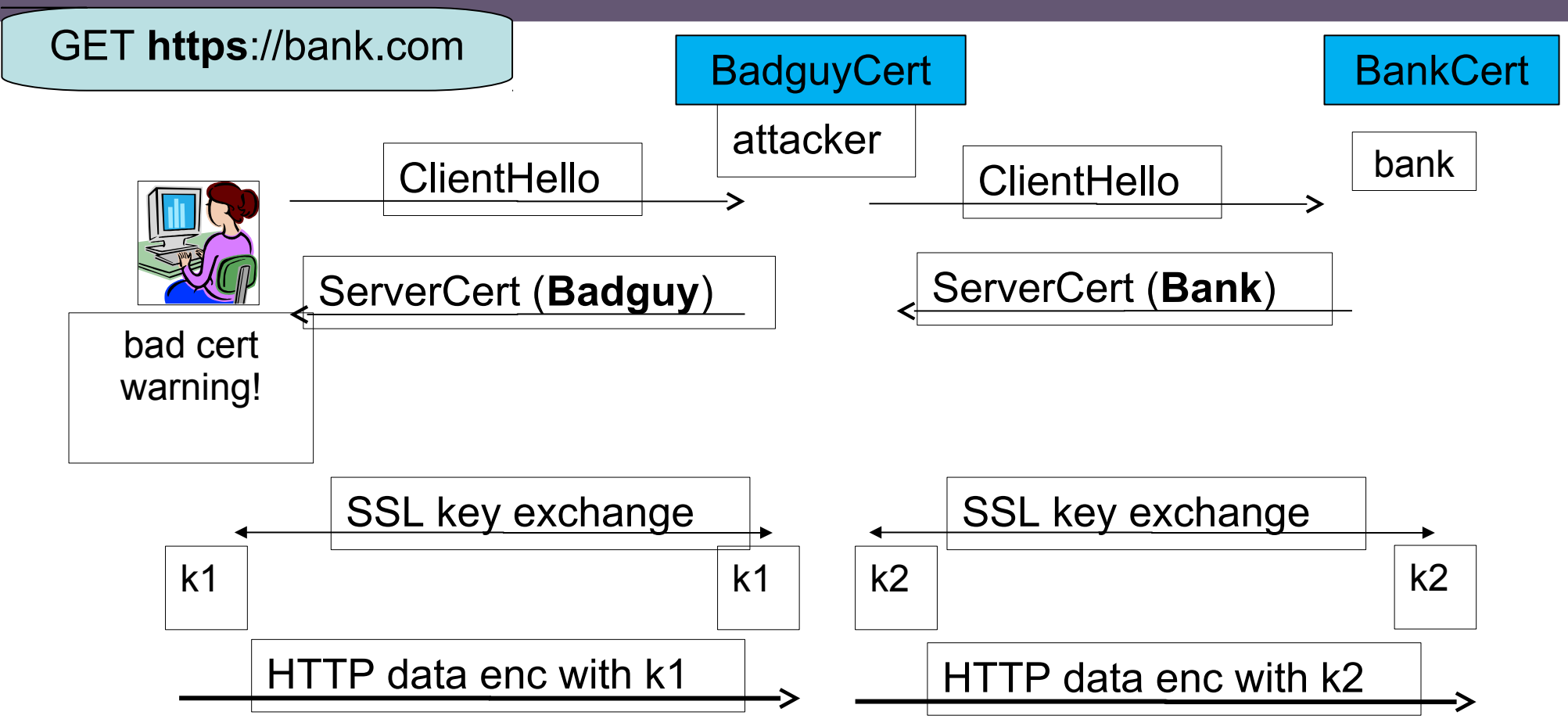
Attacker Cert

Attacker's proxy server establises SSL session with a user

Attacker's proxy server establises a session with the server

Attacker's proxy sever decrypts the data from the user and encrypts it back to the server

Man in the middle attack using invalid certs



Attacker proxies data between user and bank.
Sees all traffic and can modify data at will.

OWASP

Open Web Application Security Project

- http://www.owasp.org
  - Open group focused on understanding and improving the security of web applications and web services!
  - Hundreds of volunteer experts from around the world

OWASP
The Open Web Application Security Project
http://www.owasp.org

Discussion