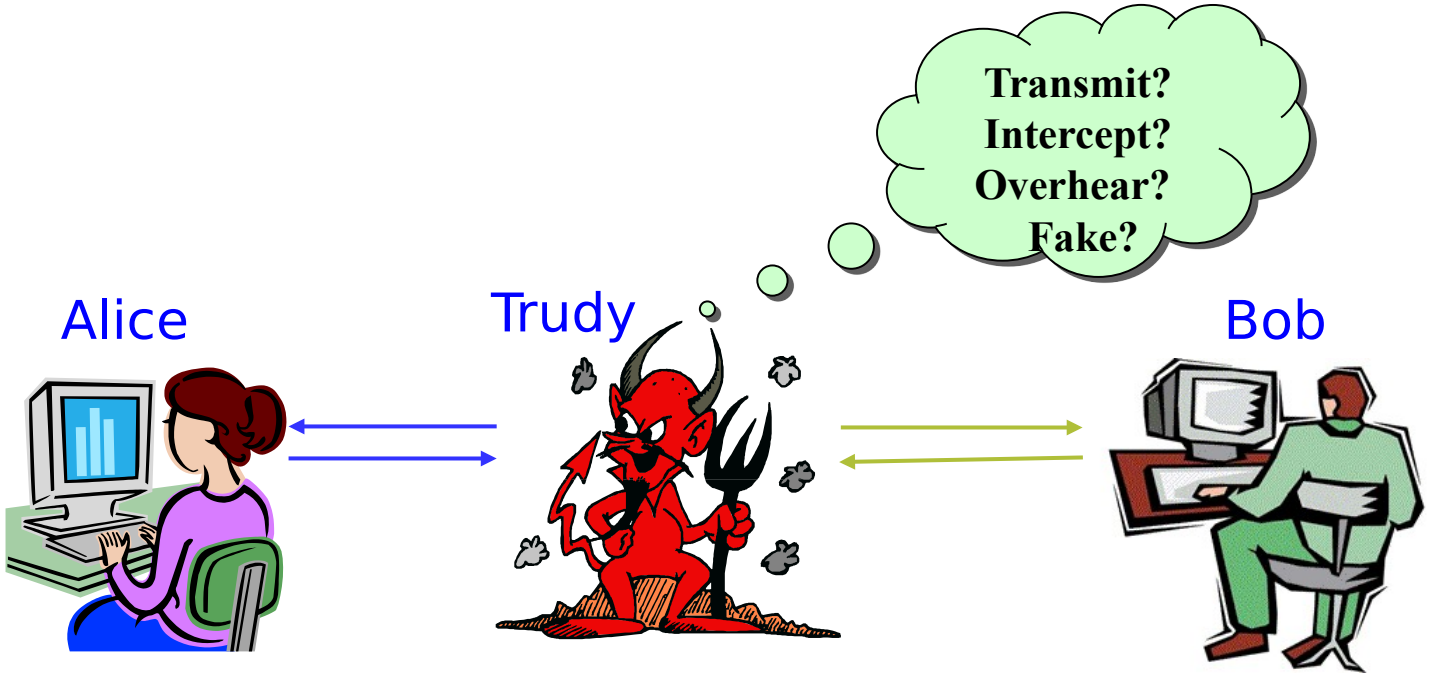
Authentication Protocols (CS-452)

Authentication Protocols

- Used to convince parties of each other's identity and to exchange session keys



Authentication Protocols

- Published protocols are often found to have flaws and need to be modified
- Key issues are
  - Confidentiality
    - To prevent masquerade and to prevent compromise of session keys, essential identification and session key information must be communicated in encrypted form.
  - Timeliness – to prevent replay attacks

Replay Attacks

- A valid signed message is copied and later resent
  - Simple replay: the opponent simply copies a message and replays it later
  - Repetition that can be logged: an opponent can replay a timestamped message within the valid time window
  - Repetition that cannot be detected: may arise because the original message could have been suppressed and thus did not arrive at its destination; only the replay message arrives.
  - Backward replay without modification: a replay back to the sender
    - When using symmetric encryption, the sender cannot easily recognize the difference between messages sent and messages received.

Replay Attacks

- Countermeasures
  - Attach a sequence number to each message used in an authentication exchange
    - Generally impractical – requires each party to keep track of the last sequence number for each claimant it has dealt with
  - Timestamps: party A accepts a message as fresh only if the message contains a timestamp that, in A's judgment, is close enough to A's knowledge of current time.
    - Needs synchronized clocks

Replay Attacks (Cont.)

- Countermeasures
  - Nonce: a random number that illustrates the freshness of a session.
    - Party A sends B a nonce and requires that the subsequent response received from B contains the correct nonce value.

Using Symmetric Encryption

- As discussed previously can use a two-level hierarchy of keys
- Usually with a trusted Key Distribution Center (KDC)
  - Each party shares own master key with KDC
  - KDC generates session keys used for connections between parties
  - Master keys used to distribute these to them

Needham-Schroeder Symmetric Key Protocol (Revisited)

- Original third-party key distribution protocol
- For session between A B mediated by KDC
- Protocol:

1. A->KDC: $ID_A \| ID_B \| N_1$

2. KDC -> A: $E_{Ka}[Ks \| ID_B \| N_1 \| E_{Kb}[Ks \| ID_A]]$

3. A -> B: $E_{Kb}[Ks \| ID_A]$

4. B -> A: $E_{Ks}[N_2]$

5. A -> B: $E_{Ks}[f(N_2)]$

Needham-Schroeder Symmetric Key Protocol (Revisited)

3. A -> B: $E_{Kb}[Ks||ID_A]$
4. B -> A: $E_{Ks}[N_2]$
5. A -> B: $E_{Ks}[f(N_2)]$

- Suppose that an attacker X has been able to compromise an old session key.

Attack: Needham-Schroeder Protocol (Revisited)

3. A -> B: $E_{Kb}[K_s||ID_A]$

4. B -> A: $E_{Ks}[N_2]$

5. A -> B: $E_{Ks}[\mathrm{f}(N_2)]$

- Suppose that an attacker X has been able to compromise an old session key.
- X can impersonate A and trick B into using the old key by simply replaying step 3.
- Unless B remembers indefinitely all previous session keys used with A, B will be unable to determine that this is a replay.
- X then intercepts the step 4 and sends bogus messages to B that appear to B to come from A using an authenticated session key.

Solution: Needham-Schroeder Protocol (Revisited)

- Use a timestamp T that assures A and B that the session key has only just been generated.
- Revised protocol:

1. A->KDC: $ID_A \| ID_B \| N_1$

2. KDC -> A: $E_{Ka}[Ks \| ID_B \| N_1 \| E_{Kb}[Ks \| ID_A \| T]]$

3. A -> B: $E_{Kb}[Ks \| ID_A \| T]$

4. B -> A: $E_{Ks}[N_2]$

5. A -> B: $E_{Ks}[f(N_2)]$

Timestamp

- Principals can verify the timeliness by checking:

$$|\text{Clock} - \text{T}| < \Delta t_1 + \Delta t_2$$

  - Δt_1: The estimated normal discrepancy between the KDC's clock and the local clock (principals' clock)
  - Δt_2: The expected network delay time

- Need to synchronize clock

Suppress-Replay Attacks

- Suppress-replay attacks: when the sender's clock is ahead of the receiver's clock, the opponent can intercept a message from the sender and replay it later when the timestamp in the message becomes current at the receiver's site.

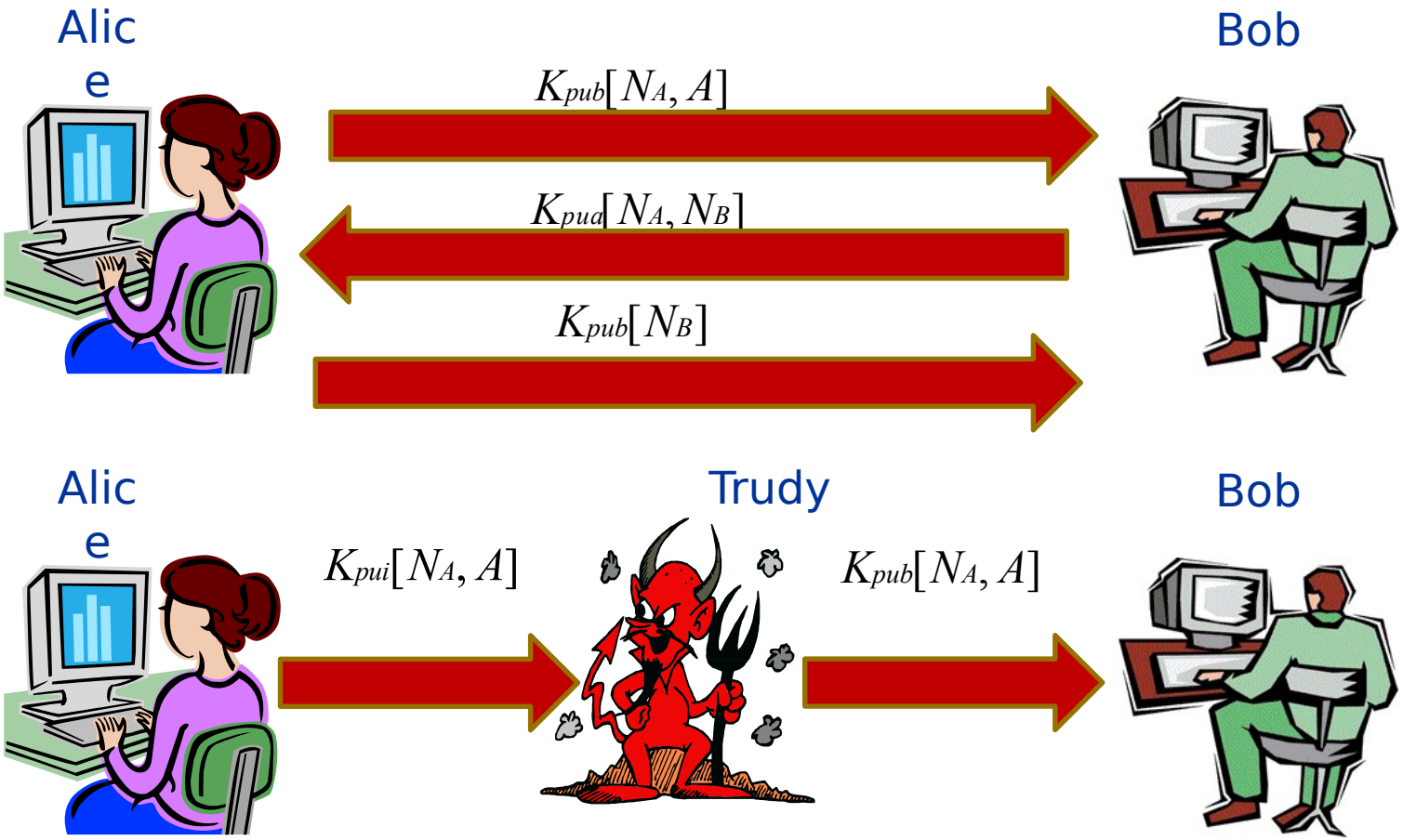
Suppress-Replay Attacks

- Suppress-replay attacks: when the sender's clock is ahead of the receiver's clock, the opponent can intercept a message from the sender and replay it later when the timestamp in the message becomes current at the receiver's site.
- Countermeasure:
  - Enforce the requirement that parties regularly check their clocks against the KDC's clock.
  - Rely on handshaking protocols using nonces.

Using Public-Key Encryption

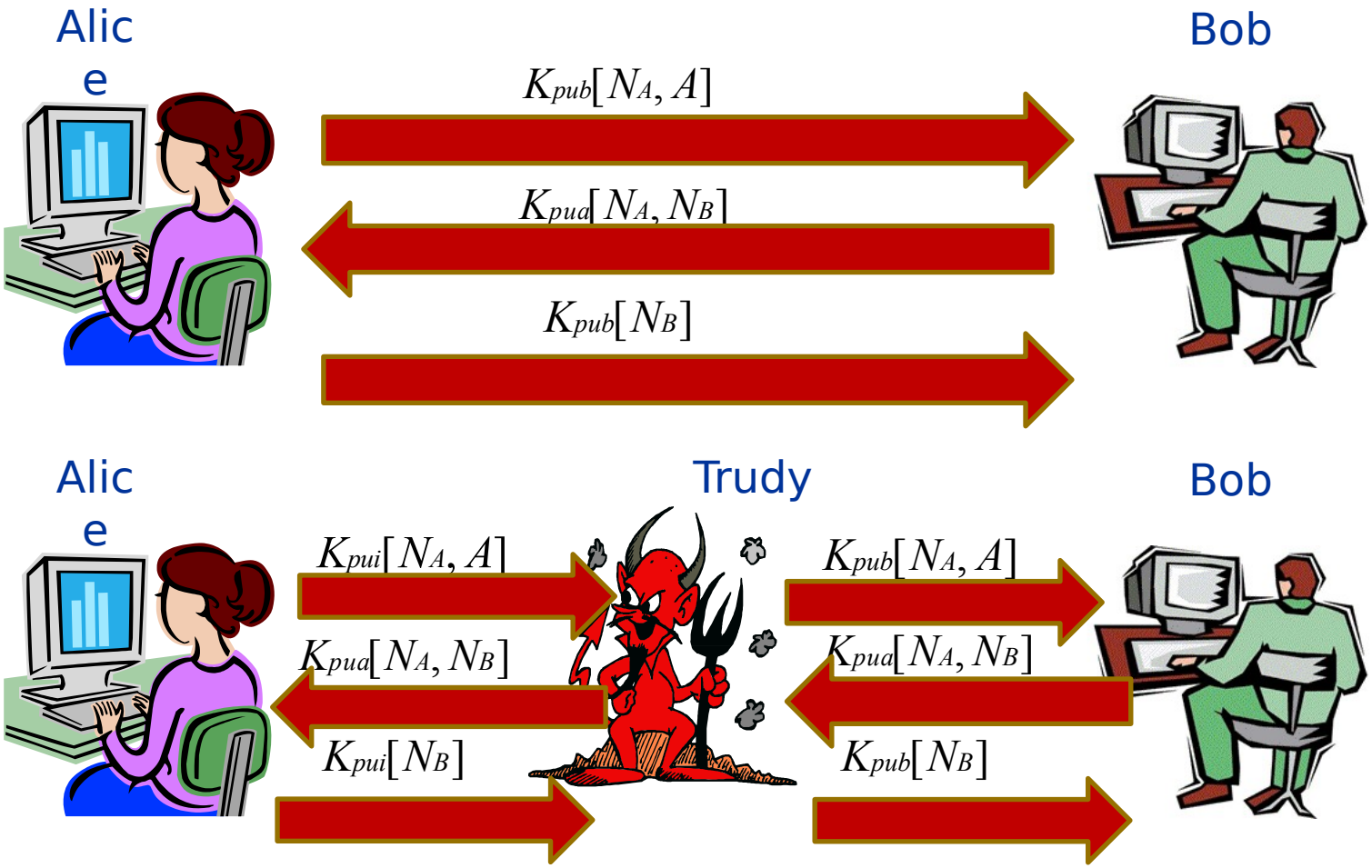
- Have a range of approaches based on the use of public-key encryption
- Need to ensure we have correct public keys for other parties
- Various protocols exist using timestamps or nonces

Needham-Schroeder Public Key Protocol



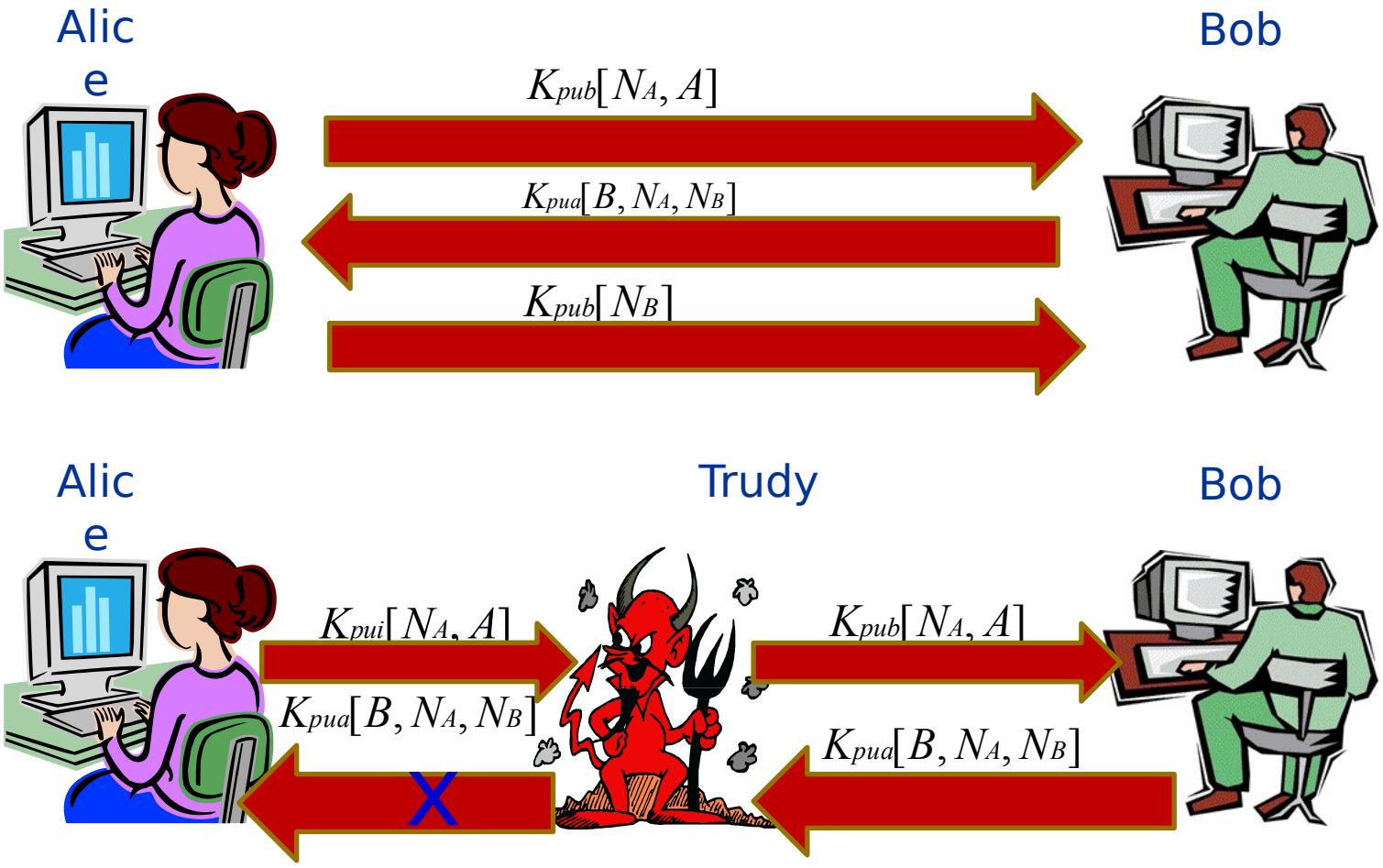
[Lowe, 1995]

Needham-Schroeder Public Key Protocol



[Lowe, 1995]

Needham-Schroeder-Lowe Public Key Protocol



One-Way Authentication

- Required when sender and receiver are not in communications at same time (eg. email)
- Have header in clear so can be delivered by email system
- May want contents of body protected & sender authenticated

Using Symmetric Encryption

- Can refine use of KDC but cannot have exchange of nonces:

1. A->KDC: $ID_A \| ID_B \| N_1$

2. KDC -> A: $\mathrm{E_{Ka}}[\mathrm{Ks} \| ID_B \| N_1 \| \mathrm{E}_{Kb}[Ks \| ID_A]\ ]$

3. A -> B: $E_{Kb}[Ks \| ID_A] \| \mathrm{E_{Ks}}[\mathrm{M}]$

Using Symmetric Encryption

- Can refine use of KDC but cannot have exchange of nonces:

  1. A->KDC: $ID_A \| ID_B \| N_1$

  2. KDC -> A: $\mathrm{E}_{\mathrm{Ka}}[\mathrm{Ks} \| ID_B \| N_1 \| \mathrm{E}_{Kb}[Ks \| ID_A]]$

  3. A -> B: $E_{Kb}[Ks \| ID_A] \| \mathrm{E}_{\mathrm{Ks}}[\mathrm{M}]$

- Guarantees that only the intended recipient of a message will be able to read it.
- Does not protect against replays
  - Could rely on timestamp in message, though email delays make this problematic

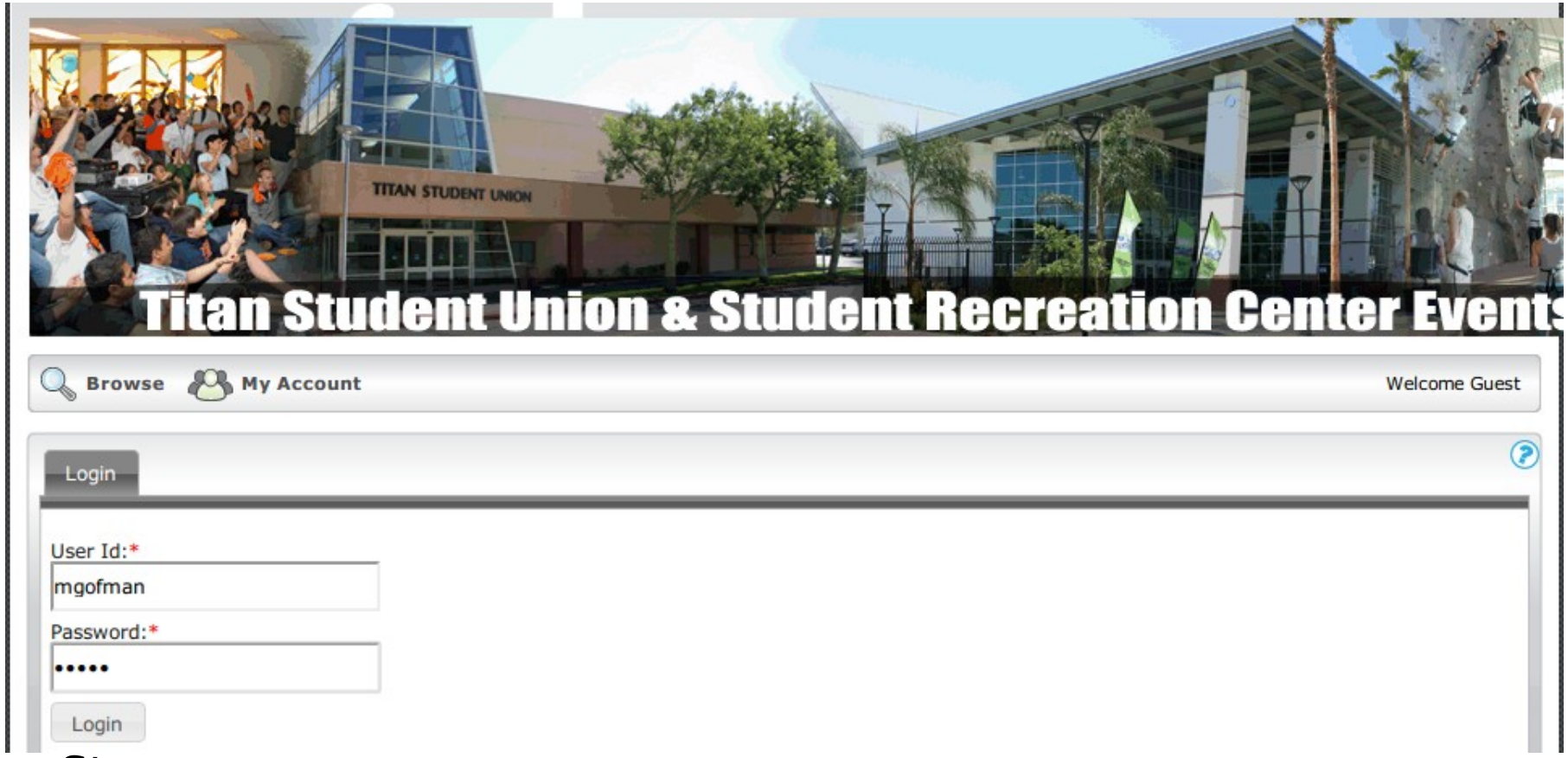
Secure Sockets Layer/Transport Layer Security (SSL/TLS) and Web Security

Web Security

- World Wide Web is fundamentally a client/server application running over internet and TCP/IP intranet.
- Web now widely used by business, government, individuals
- But Web is vulnerable

Example of a Web Vulnerability (1)

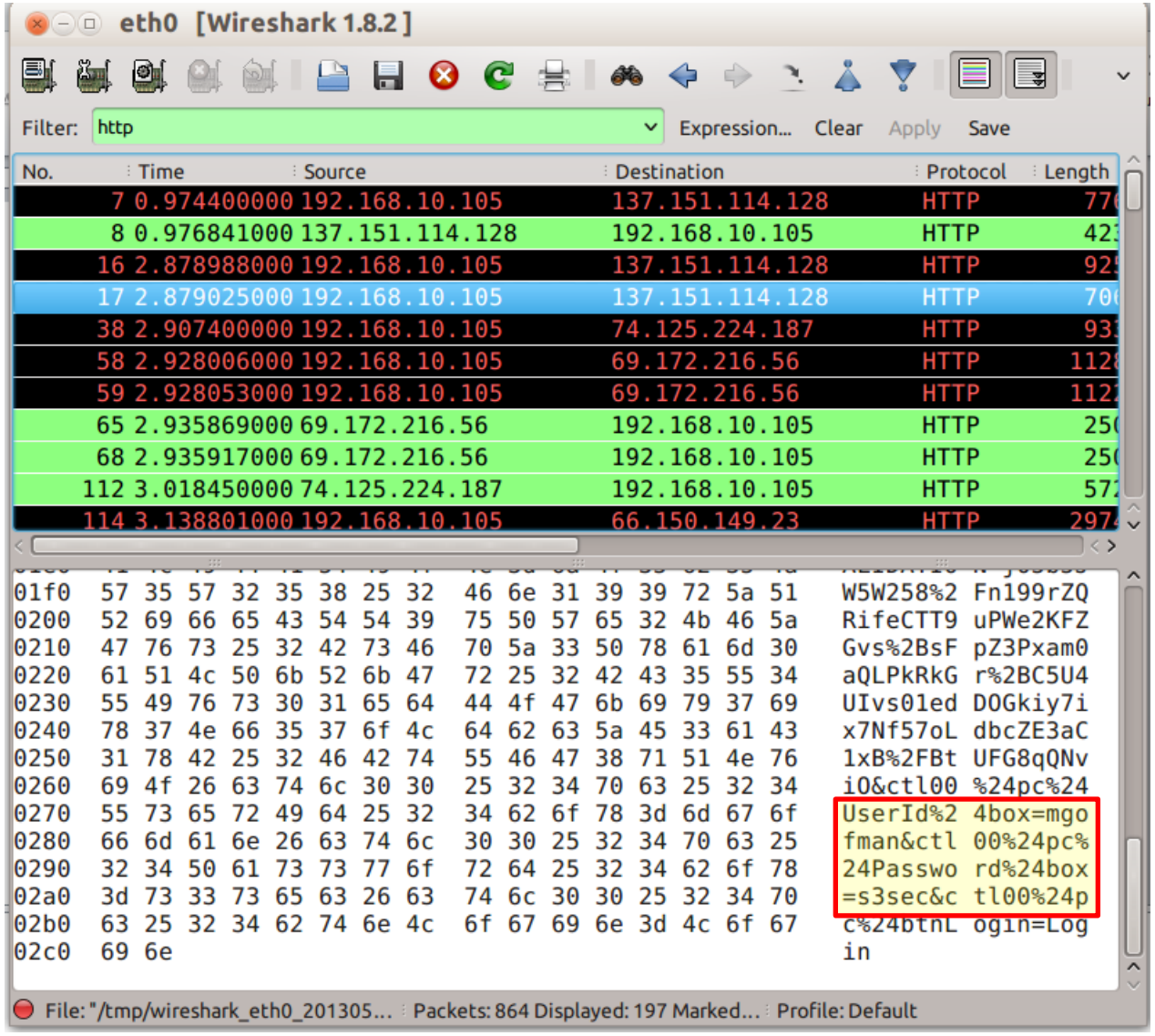
- Website: http://asi.fullerton.edu/VirtualEMS/Login.aspx



- Steps:
  - Enter user name: mgofman
  - Enter password: s3sec
  - Use WireShark packet sniffer to observe the website traffic…

Example of a Web Vulnerability (2)

- Traffic captured using WireShark:



Summary of Web-based Attacks

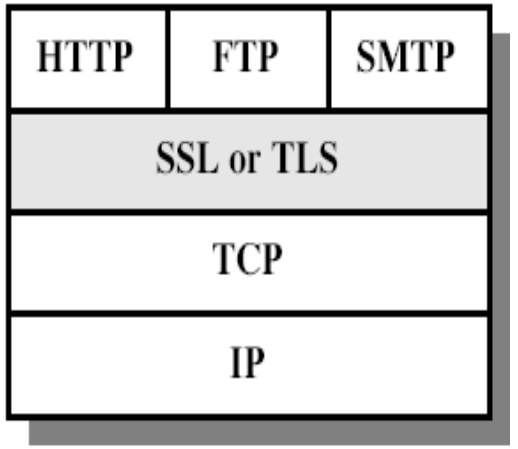
	Threats	Consequences	Countermeasures
Integrity	• Modification of user data • Trojan horse browser • Modification of memory • Modification of message traffic in transit	• Loss of information • Compromise of machine • Vulnerabilty to all other threats	Cryptographic checksums
Confidentiality	• Eavesdropping on the net • Theft of info from server • Theft of data from client • Info about network configuration • Info about which client talks to server	• Loss of information • Loss of privacy	Encryption, Web proxies
Denial of Service	• Killing of user threads • Flooding machine with bogus requests • Filling up disk or memory • Isolating machine by DNS attacks	• Disruptive • Annoying • Prevent user from getting work done	Difficult to prevent
Authentication	• Impersonation of legitimate users • Data forgery	• Misrepresentation of user • Belief that false information is valid	Cryptographic techniques

Hypertext Transfer Protocol Secure (HTTPS)

- A combination of the http protocol and a network security protocol
- Also known as Hypertext Transfer Protocol over Secure Socket Layer
- The administrator must create a public X.509 key certificate for the Web server.
- This certificate must be signed by a certificate authority.
  - SSL certificate providers: Verisign, Thawte, InstantSSL, Entrust, Baltimore, Geotrust etc.
- Web browsers are distributed with the public key of major certificate authorities so that they can verify certificates signed by them.

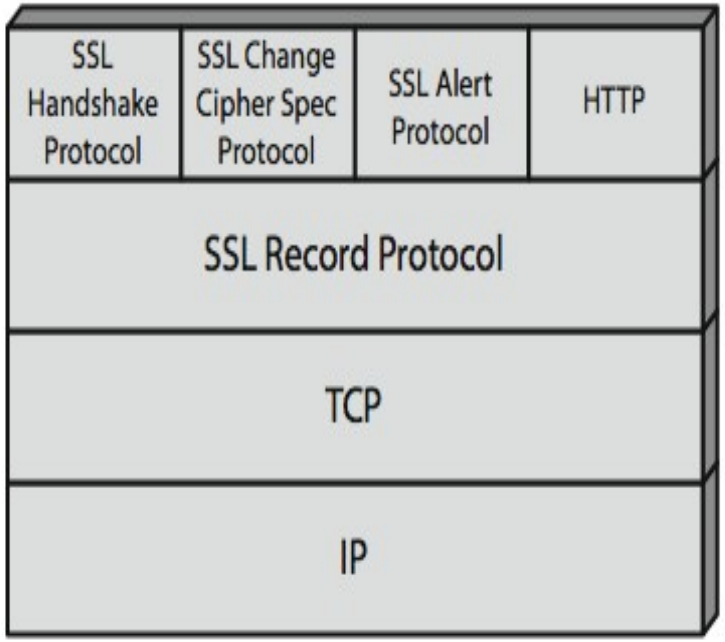
SSL/TLS (Secure Socket Layer/Transport Layer Security)

- A cryptographic protocol that provides security for communications over networks.
- One of the most widely used Web security mechanisms.
- Transport layer security service - designed to make use of TCP to provide a reliable end-to-end security service.
- Originally developed by Netscape
- Subsequently became Internet standard known as TLS (Transport Layer Security)



SSL/TLS Architecture

- Has two layers of protocols
  - Level 1:
    - SSL Record Protocol: provides basic security services to various higher-layer protocols.
  - Level 2:
    - Hypertext Transfer Protocol (HTTP): which provides the transfer service for Web client/server interaction, can operate on top of SSL.
    - Three higher-layer protocols: used in the management of SSL exchanges.



SSL/TLS Handshake Protocol

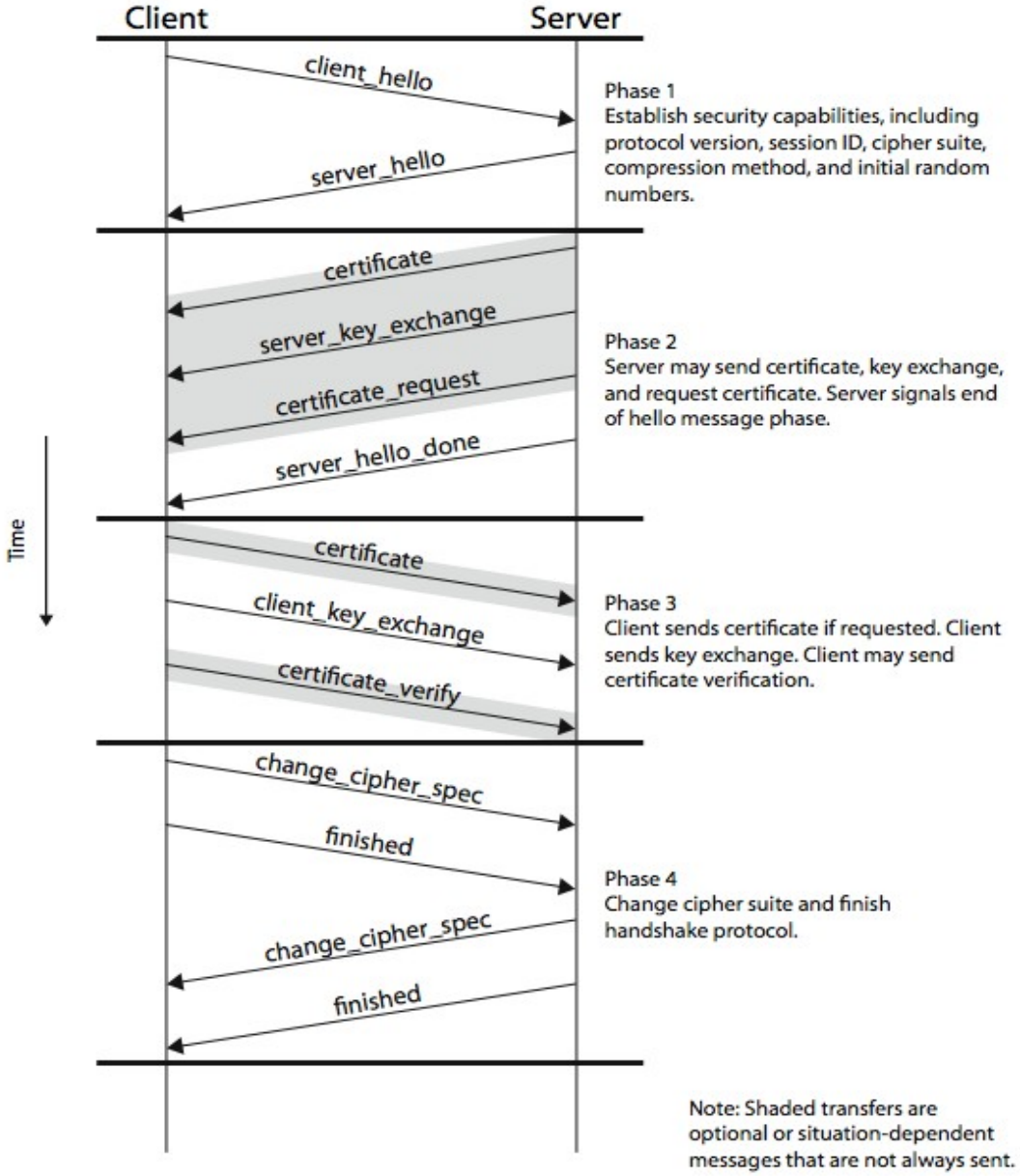
- Allows server and client to:
  - Authenticate each other
  - To negotiate encryption & MAC algorithms
  - To negotiate cryptographic keys to be used to protect data sent in an SSL record.

SSL/TLS Handshake Protocol

Comprises a series of messages in phases

1. Establish Security Capabilities:

   (a) The client initiates a logical connection and establish the security capabilities: protocol version, session ID, cipher suite (cryptographic algorithms supported by the client), compression method.

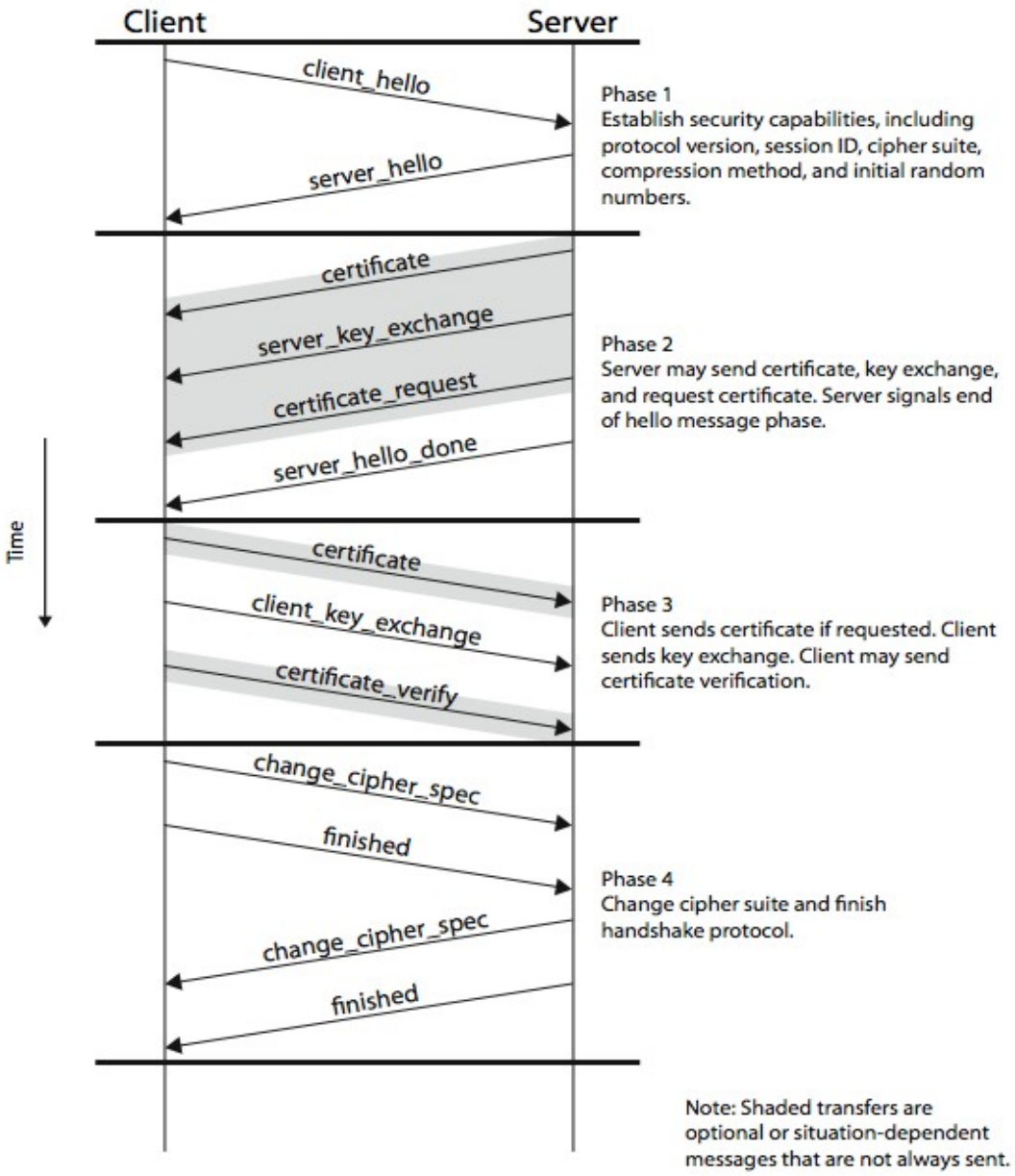


SSL/TLS Handshake Protocol

Comprises a series of messages in phases

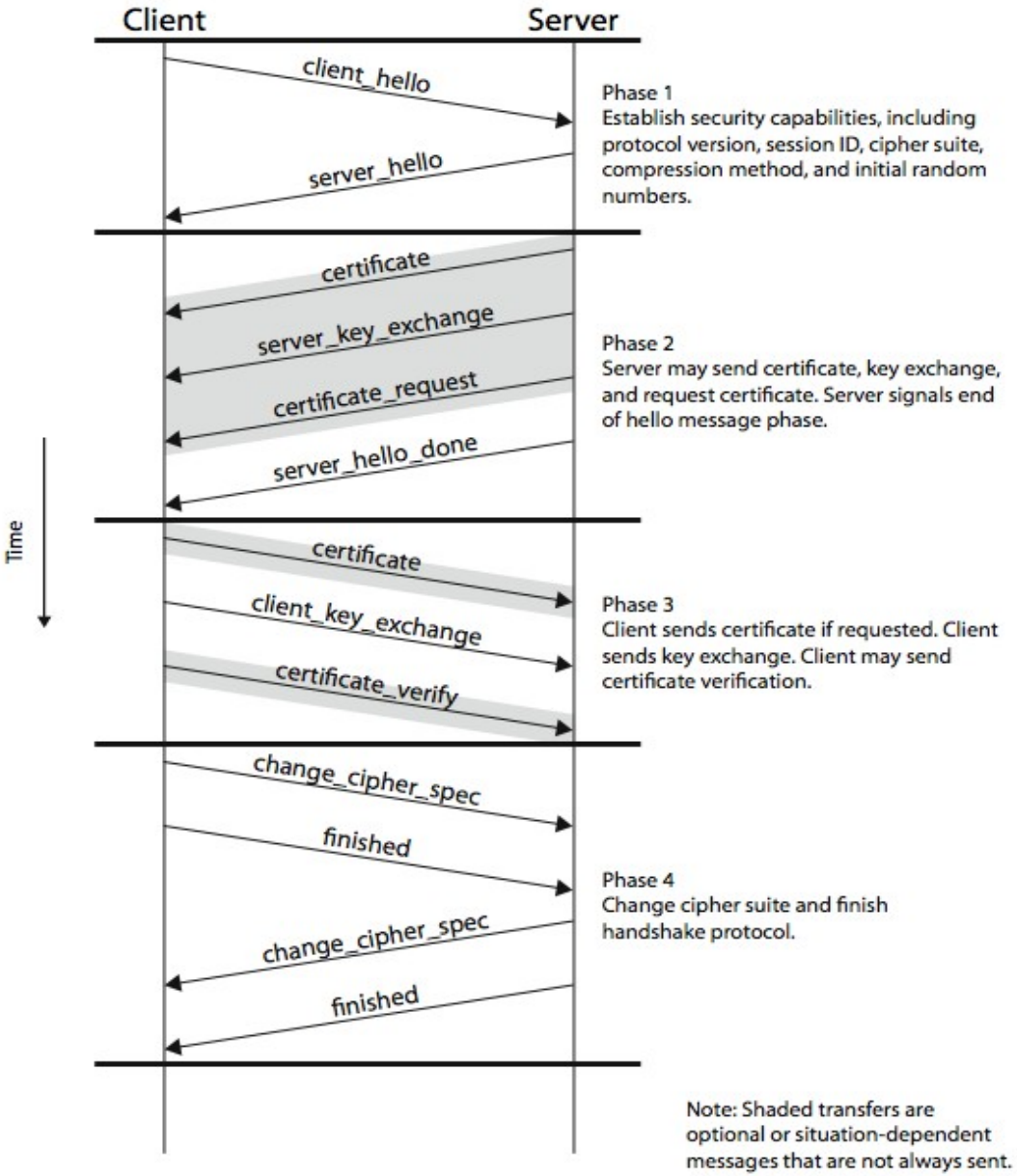
1. Establish Security Capabilities:

   (b) the server picks the strongest cipher and hash function that it also supports and notifies the client of the decision



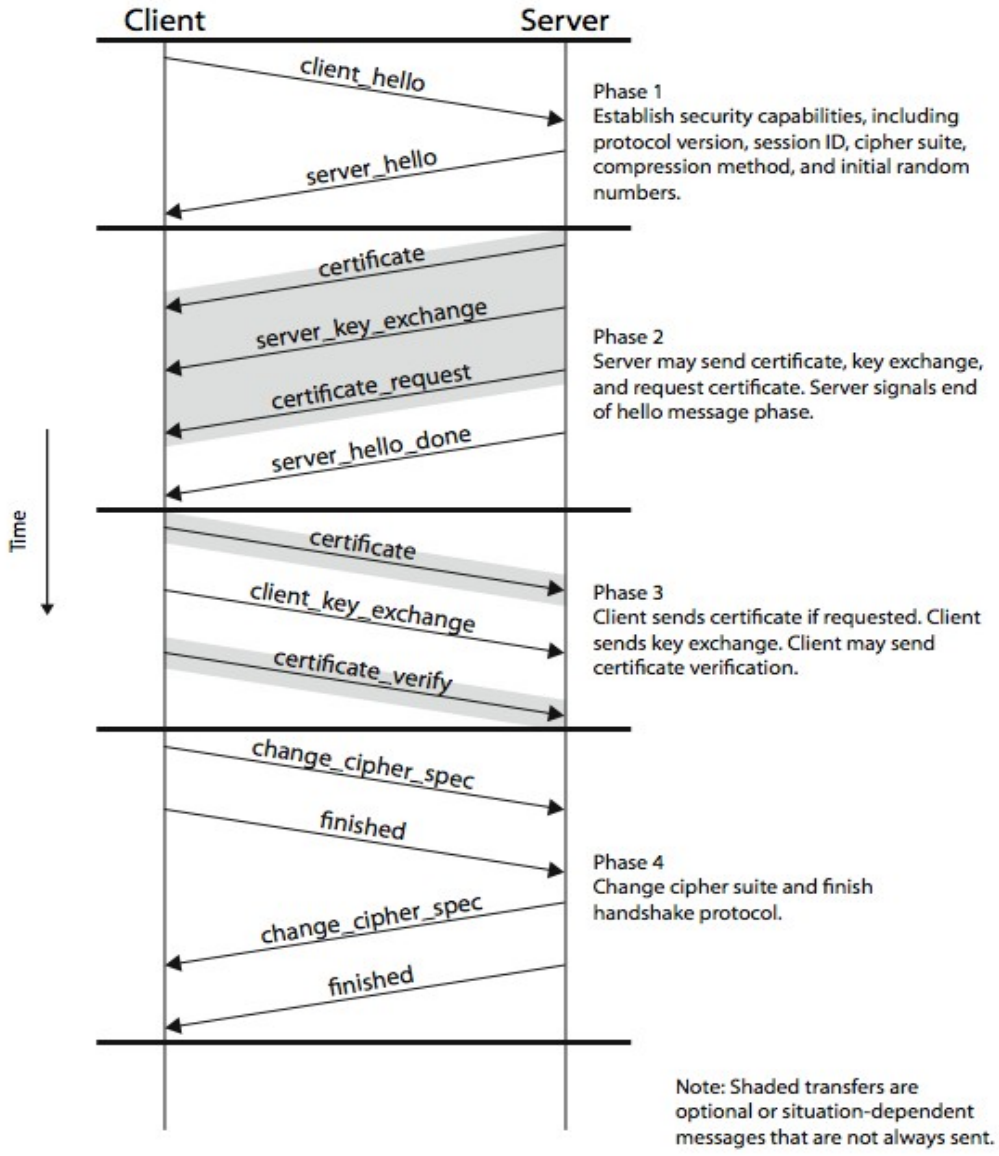
SSL/TLS Handshake Protocol

2. Server Authentication and Key Exchange: 1) sends certificate if it needs to be authenticated; 2) sends a server_key_exchange message, and request certificate; 3) signals the end of hello message phase.



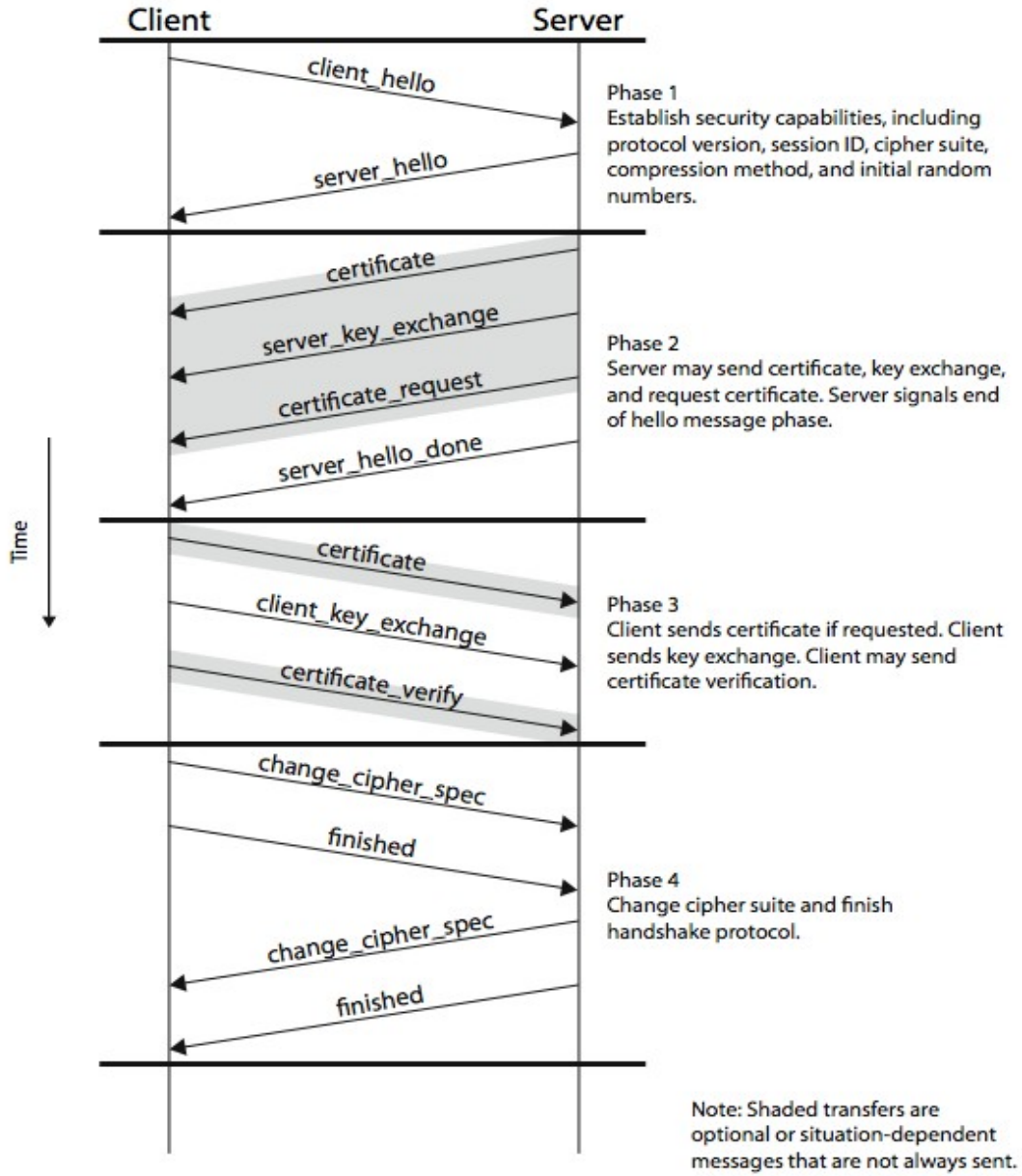
SSL/TLS Handshake Protocol

3. Client Authentication and Key Exchange: sends certificate if requested and encrypts a random key with the server's public key, and sends the result to the server.



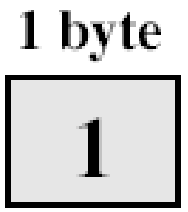
SSL/TLS Handshake Protocol

4. Change cipher suite and finish handshake protocol.



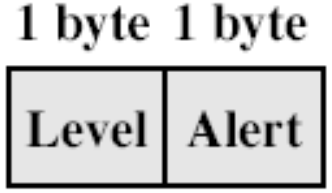
SSL/TLS Change Cipher Spec Protocol

- Notify the receiving party that subsequent records will be protected under the just-negotiated cipherspec and keys.
- Consists of a single message, which consists of a single byte with the value 1.
- Causes pending state to become current - updating the cipher suite in use



SSL/TLS Alert Protocol

- Conveys SSL-related alerts to peer entity
- Consists of two bytes – the first takes the value: warning (1) or fatal (2); the second contains a code that indicates the specific alert.



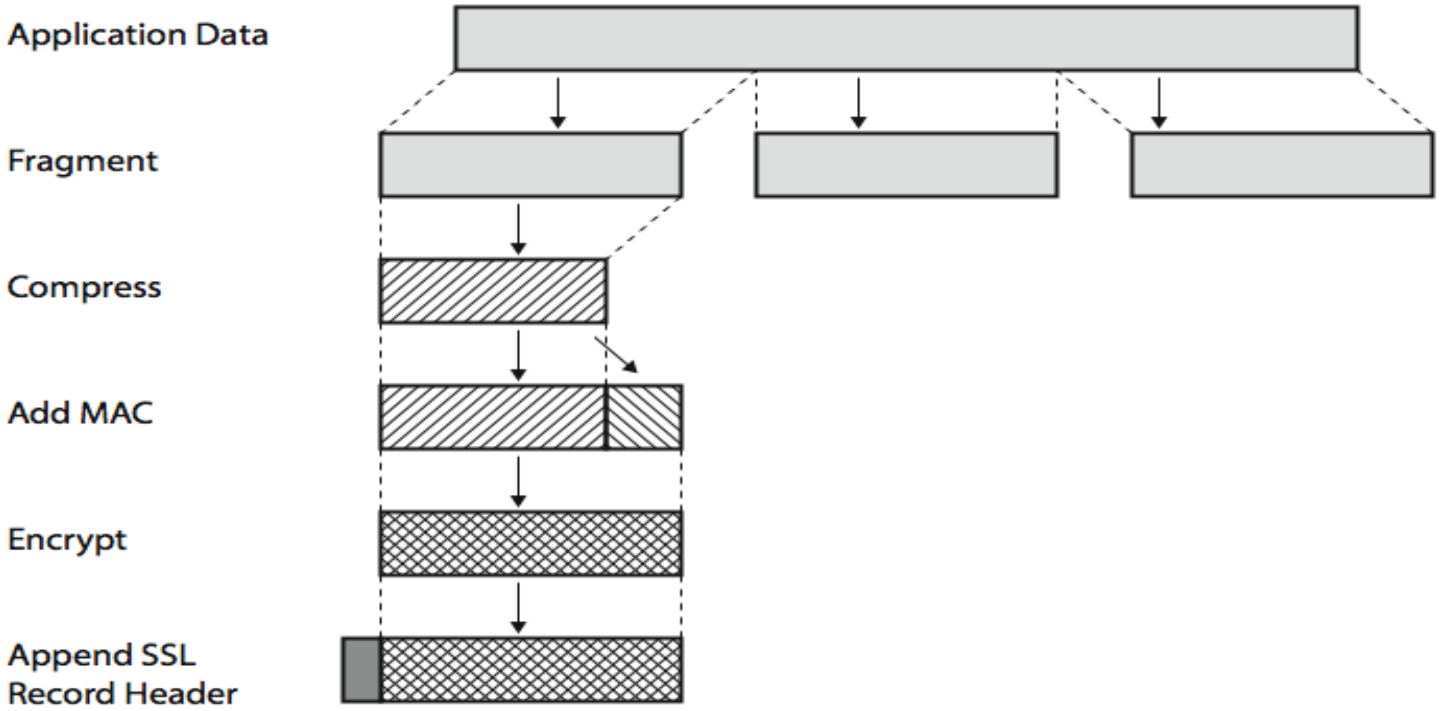
  - Fatal: unexpected message, decompression failure, handshake failure, illegal parameter
    - SSL immediately terminates the connection
  - Warning: close notify (the sender will not send any more message of this connection), bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown.

SSL/TLS Record Protocol Services

- Provides two services for SSL connections
  - Confidentiality:
    - encrypt SSL payloads
  - Message integrity:
    - use a shared secret key to form MAC.

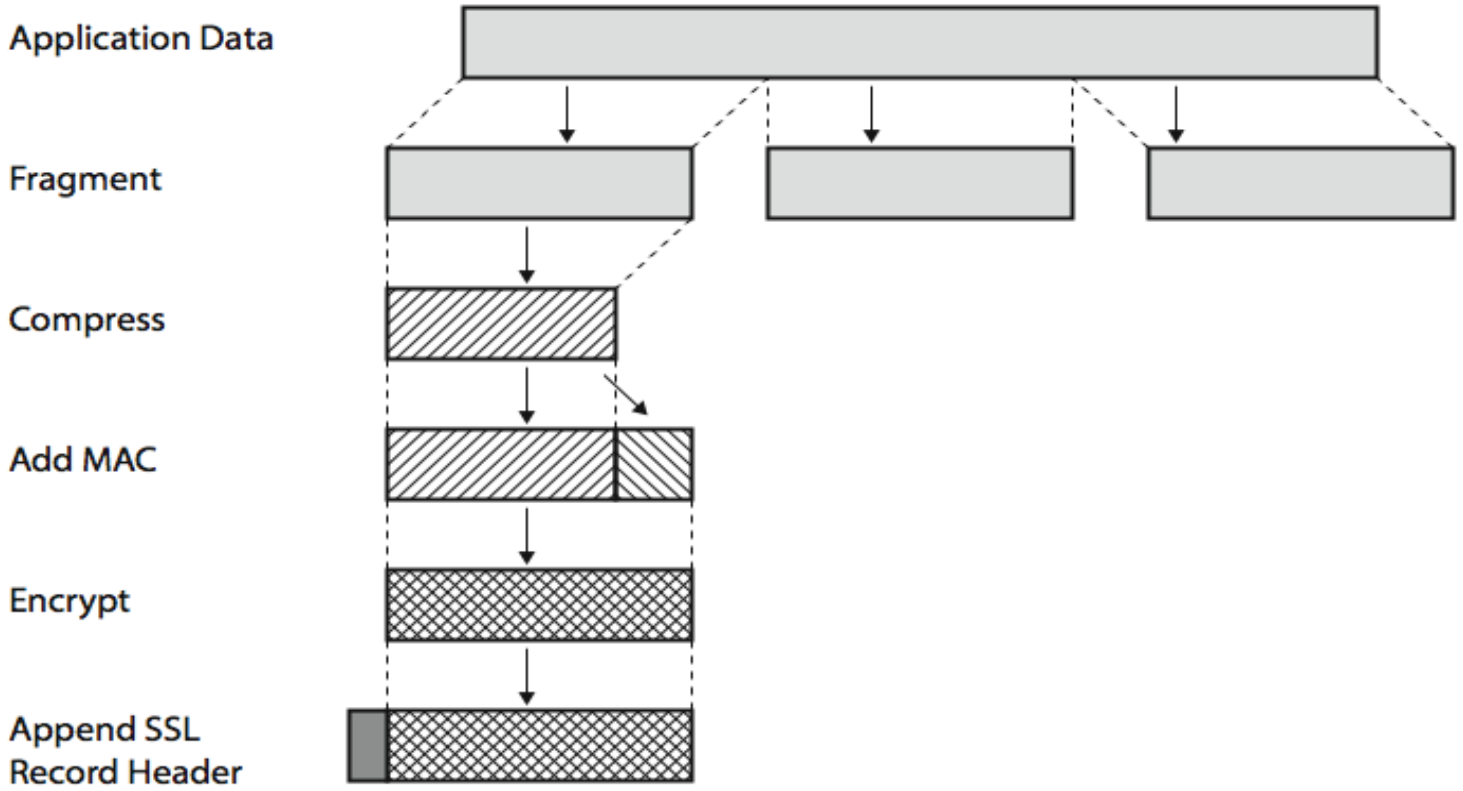
SSL/TLS Record Protocol Operation

- Fragmentation: message is fragmented into blocks of 2^{14} bytes or less
- Compression (optional): lossless and may not increase the content length by more than 1024 byte (for very short block, it is possible that the output is longer)



SSL/TLS Record Protocol Operation

- MAC: Compute the message authentication code over the compressed data.
- Encryption: the compressed message plus the MAC are encrypted using symmetric encryption.



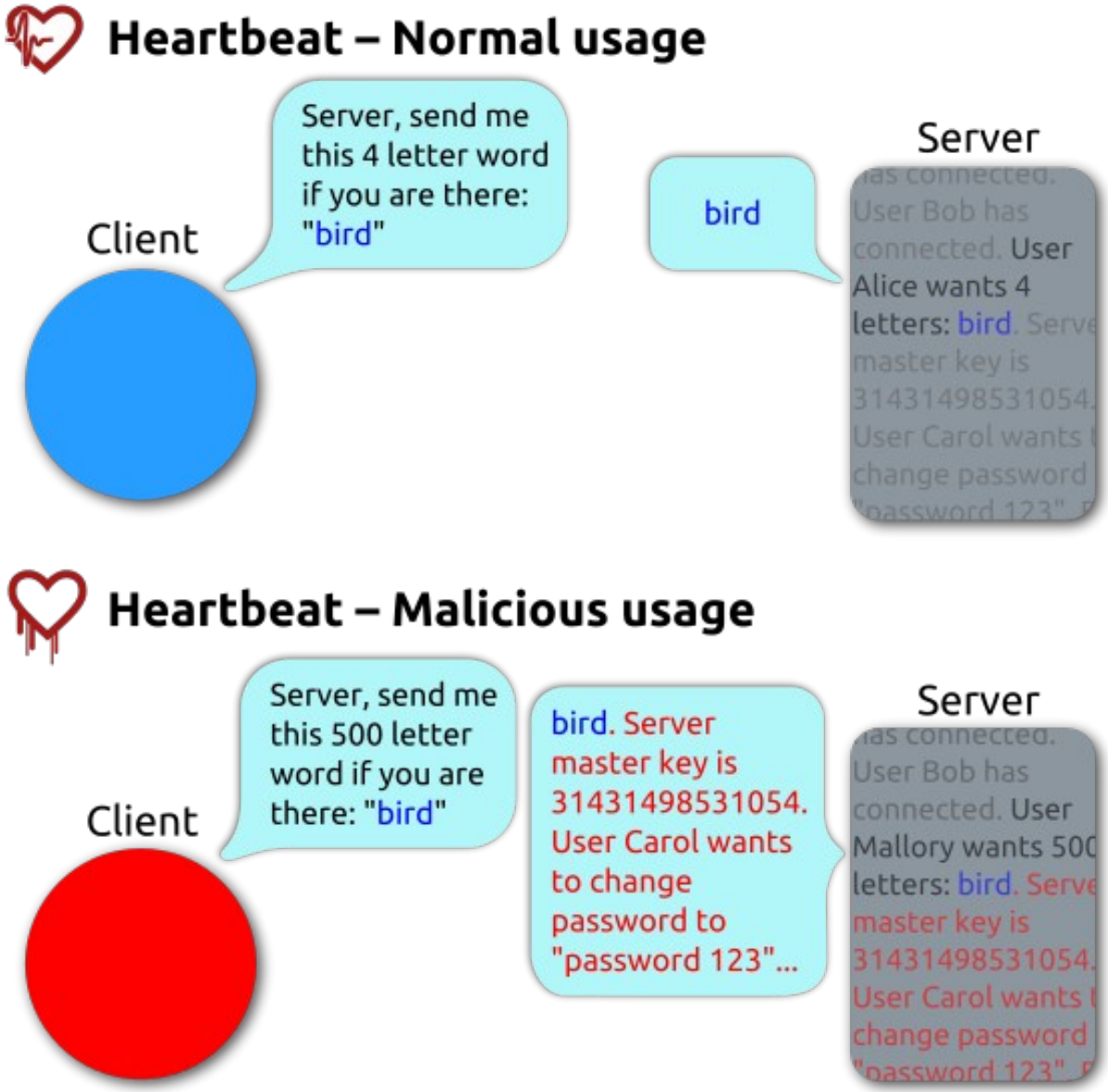
HeartBleed Vulnerability

- Introduced into OpenSSL code in 2011 by Robin Seggelmann (a Ph.D. student at the University of Duisburg-Essen).
  - Seggleman implemented a “heartbeat” function into OpenSSL which allows one side to check if the other side is still up and running.
- Stephen Henson, in charge of OpenSSL core development, did not spot Seggelmann’s bug.
  - Result: the vulnerable code was introduced into the production version… persisted till 2014.

HeartBleed Vulnerability

- SSL heartbeats are used for one side (server or client) to check if the other side is alive and well.
  - Send an N byte message to the other side. The other side will echo the same N bytes back to the sender.
- The implementation bug:
  - The sender sends an X < N byte message, but tells the receiver that the message is actually N bytes.
  - The other side will echo the X bytes and N-X bytes in the memory adjacent to the first X bytes.
    - That memory can contain keys, certificates, passwords, and other information.
    - Can steal up to 64 KB of data per heartbeat message.

HeartBleed Vulnerability Example:



Password Management

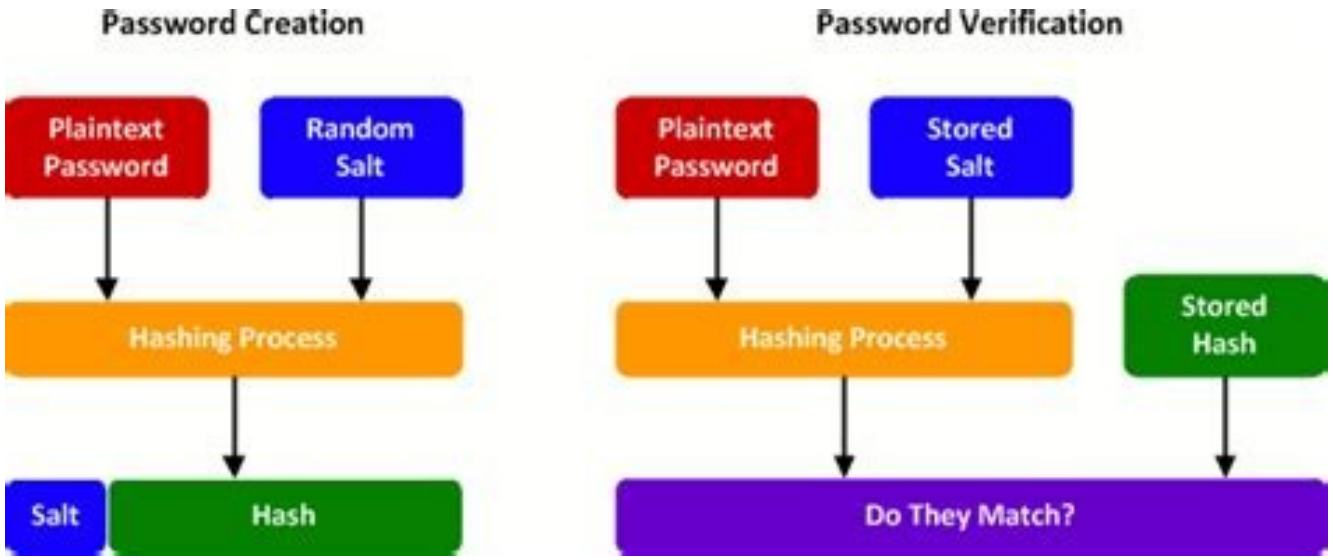
Password Management

- Front-line defense against intruders
- Users supply both:
  - login – determines whether the user is authorized to gain access to a system, and the privileges of that user.
  - password – to identify them
- Should protect password file on system
  - One-way function: the system stores only the value of a function based on the user's password.
  - Access control: access to the password file is limited to one or a very few accounts.

Unix Password Management

- The user selects a password.
- Multiple encryption/hashing schemes supported for storing the password
- Good explanation: http://en.wikipedia.org/wiki/Shadow_password

Linux Password and Verification: Basic Idea



- Basic idea:
  - Storing the password: the user provides the password
    - The password is hashed and is stored in the database indexed by the user name
    - Before hashing the password, the Salt value (a random number) is added to the password to help frustrate rainbow table attacks.
  - Verification: the user enters user name and password. The entry for the user is looked up in the database. If hash(entered password || salt) == hash in the database, the password is correct

Managing Passwords - Education

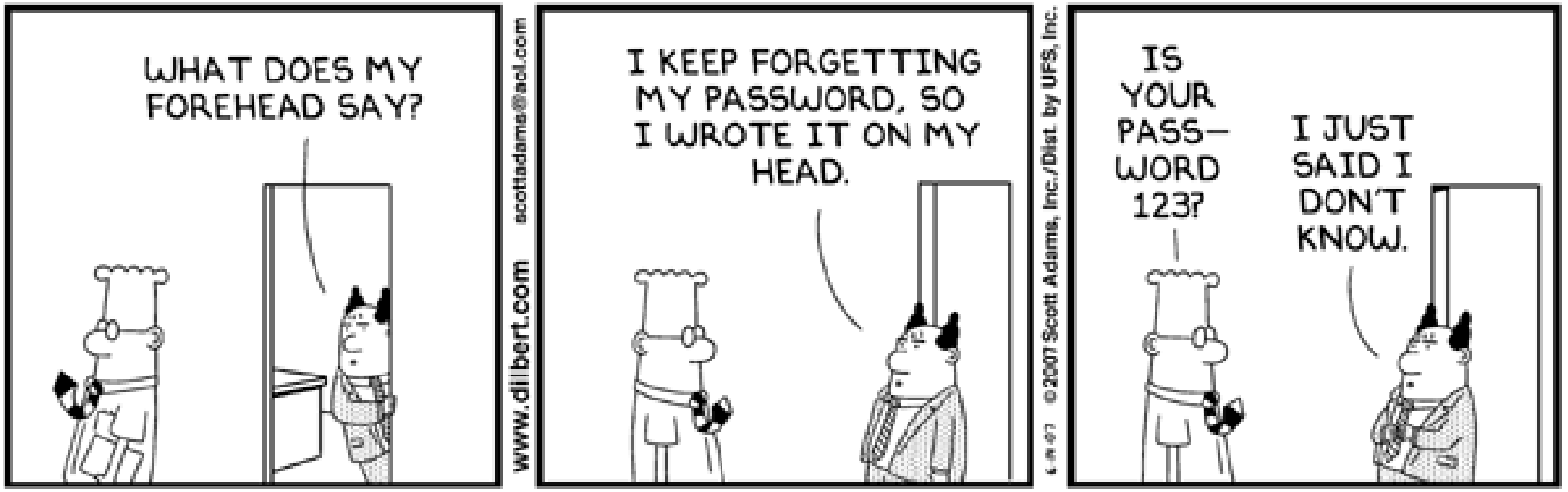
- Studies have shown that many people use short passwords or guessable passwords
- Need some approach to counter this
- Educate on importance of good passwords
  - Give guidelines for good passwords
    - Minimum length (>6)
    - Require a mix of upper & lower case letters, numbers, punctuation
    - Not dictionary words
  - But likely to be ignored by many users

Managing Passwords - Computer Generated

- Let computer create passwords

Managing Passwords - Computer Generated

- Let computer create passwords
  - If the passwords are quite random in nature, users will not be able to remember them.
  - Even if the password is pronounceable, the user may have difficulty remembering it.
  - Have history of poor user acceptance.



© Scott Adams, Inc./Dist. by UFS, Inc.

Managing Passwords - Reactive Checking

- Periodically run password guessing tools
- Cracked passwords are disabled
- But is resource intensive
- Bad passwords are vulnerable till found

Managing Passwords - Proactive Checking

- Most promising approach to improving password security
- Allow users to select own password
- But have system verify it is acceptable
  - Simple rule enforcement
  - Compare against dictionary of bad passwords

Analyzing Websites for User-Visible Security Design Flaws

http://cups.cs.cmu.edu/soups/2008/proceedings/p117Falk.pdf

Banking trends

- An increasing number of people rely on secure websites to carry out their daily business
- Pew Internet: 42% of all internet users bank online
- Forbes.com conducted a survey on +900 people and divided users in:
  - Used online banking applications and paid bills online through their bank's website
  - Used online banking applications but not online bill payments
  - Used no online banking activities whatsoever
- Those who used online banking were satisfied with the services.
- Those who chose not to use online banking cited security concerns as a reason why they did not use the services

How banks deal with online security

- Due to the sensitive nature of these sites, security is a top priority.
- Hire security experts to conduct vulnerability assessments.
- Deploy encryption protocols such as SSL.
- Monitoring accounts for suspicious activities.
- Overall the online security has improved compared to few years ago.

Study Conducted in the Paper

- Conducted during Nov - Dec 2006.
- Analyses 214 U.S. financial institutions for user-visible security design flaws.
- Design flaws are a result of decisions made during the website design phase and they promote insecure user behaviour.
- These design features made it very difficult for someone to use the site securely.

Design Flaws

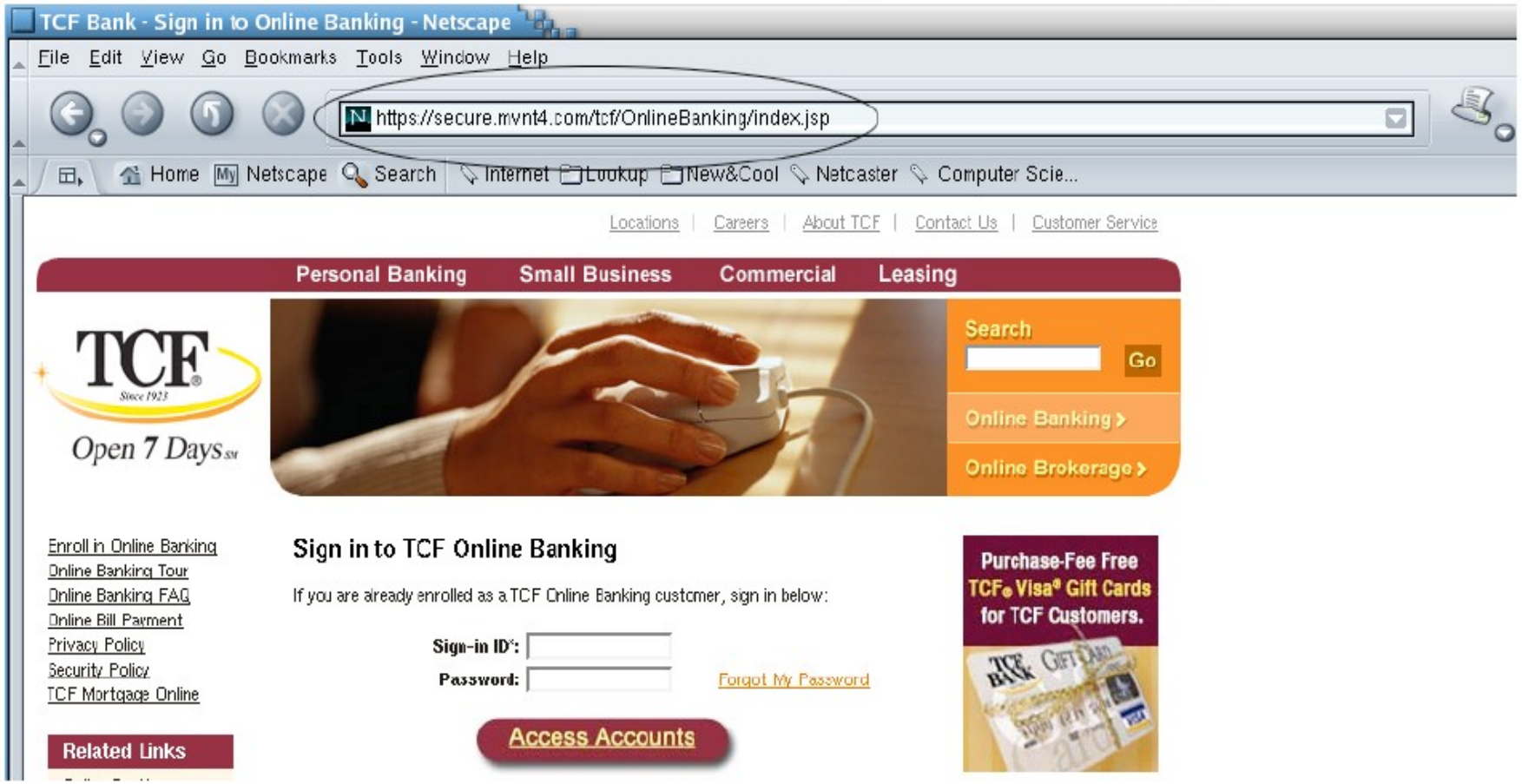
- Break in the chain of trust: websites forward users to new pages that have different domains without notifying the users

Break in the chain of trust

- Customer is redirected to a site that has a different domain name than the financial institution's site that was originally visited.
- The switch is usually done without warning customers about such redirection.
- It is up to the user to determine if the new site is really affiliated with the financial institution.

Example: Break in the chain of trust

Break in the chain of trust (Cont.)



- The signed certificate of http://secure.mvnt4.com, but TCF Bank was not the owner of this certificate. The owner is Metavante Corporation.

Example: Break in the chain of trust

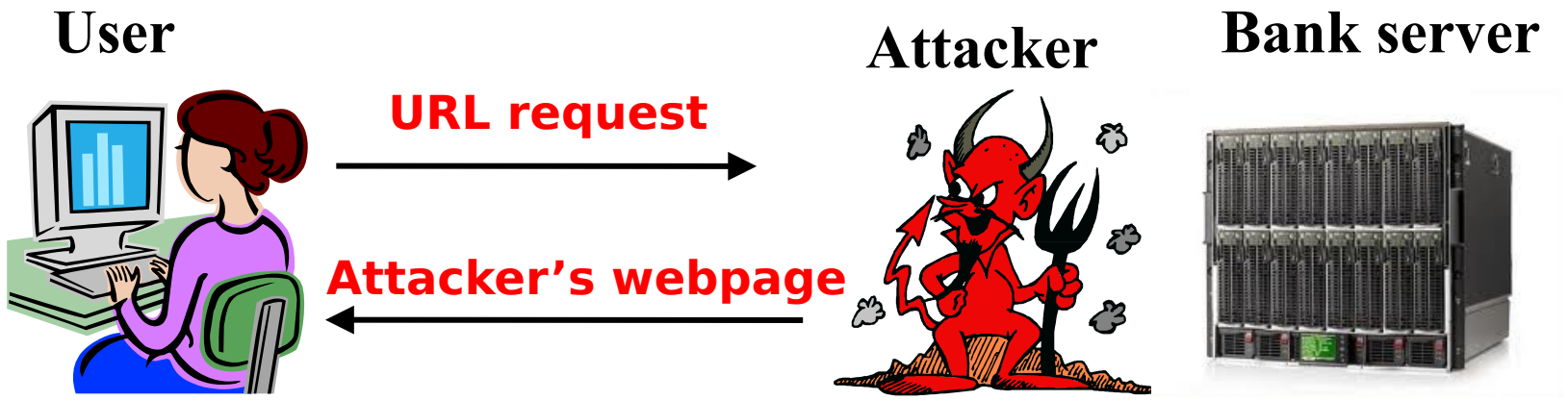
- University of Michigan credit union's website, users authenticate properly and are taken to a secure page.
- However, if an account holder decides to sign-up for Bill Pay, a new window pops up that belongs to a third-party.
- This window asks the user to enter information, such as mother's maiden name, SSN, account #, and birth date.
- No message is given, indicating that this pop-up from third-party website will occur.
- The credit union could have handled this design better, by either providing better disclosures or by not requiring the user to enter that information.

Design Flaws

- Break in the chain of trust: websites forward users to new pages that have different domains without notifying the users
- Presenting secure login options on insecure pages: Some sites present login forms that forward to a secure page but do not come from a secure page.

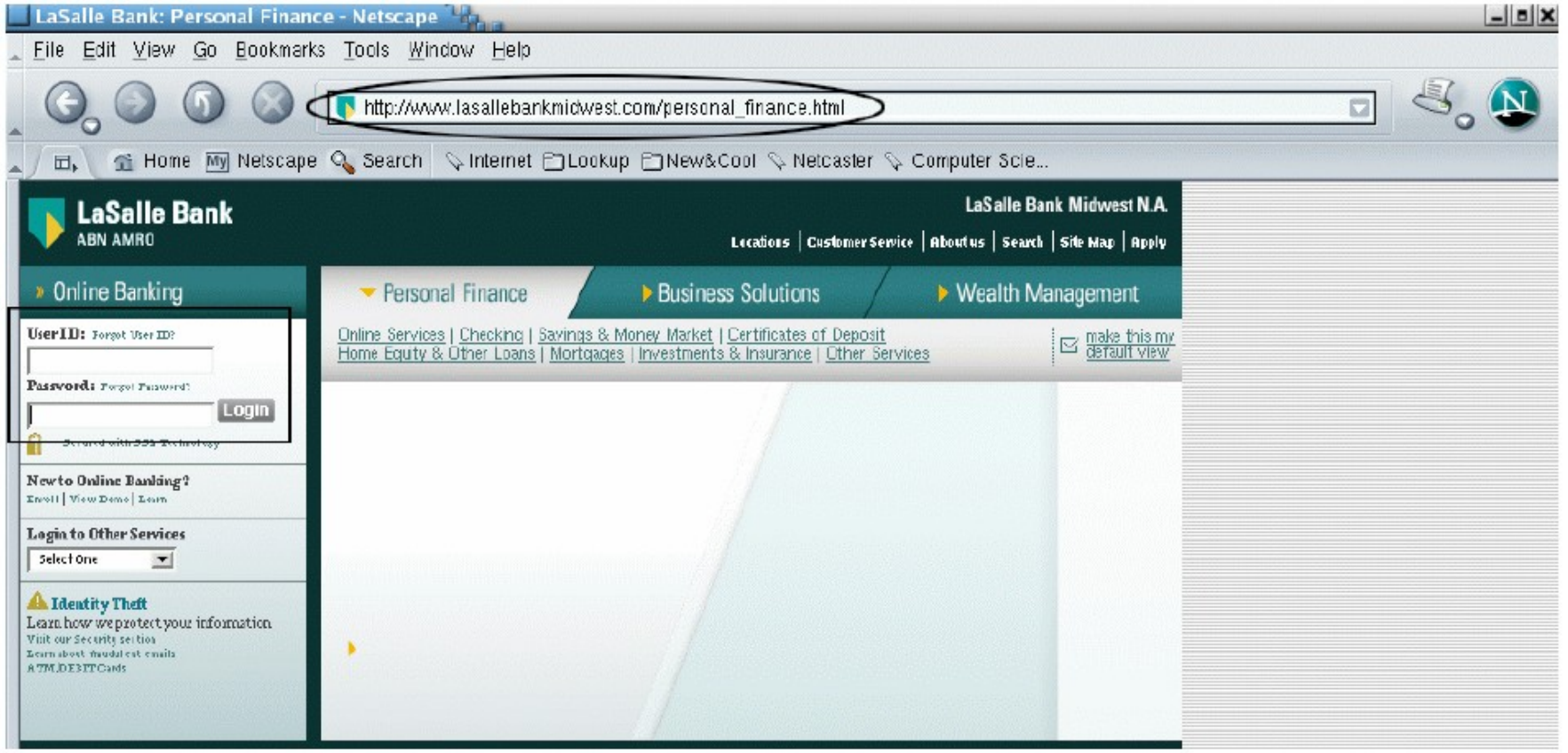
Presenting secure login options on insecure pages

- Login pages and options displayed on insecure pages leave users vulnerable to man-in-the-middle attacks.
  - They have no way of knowing if their usernames and passwords are being sent to a hacker site.



Presenting secure login options on insecure pages

- E.g. LaSalle Bank's -- http://www.lasallebank.com



Presenting secure login options on insecure pages

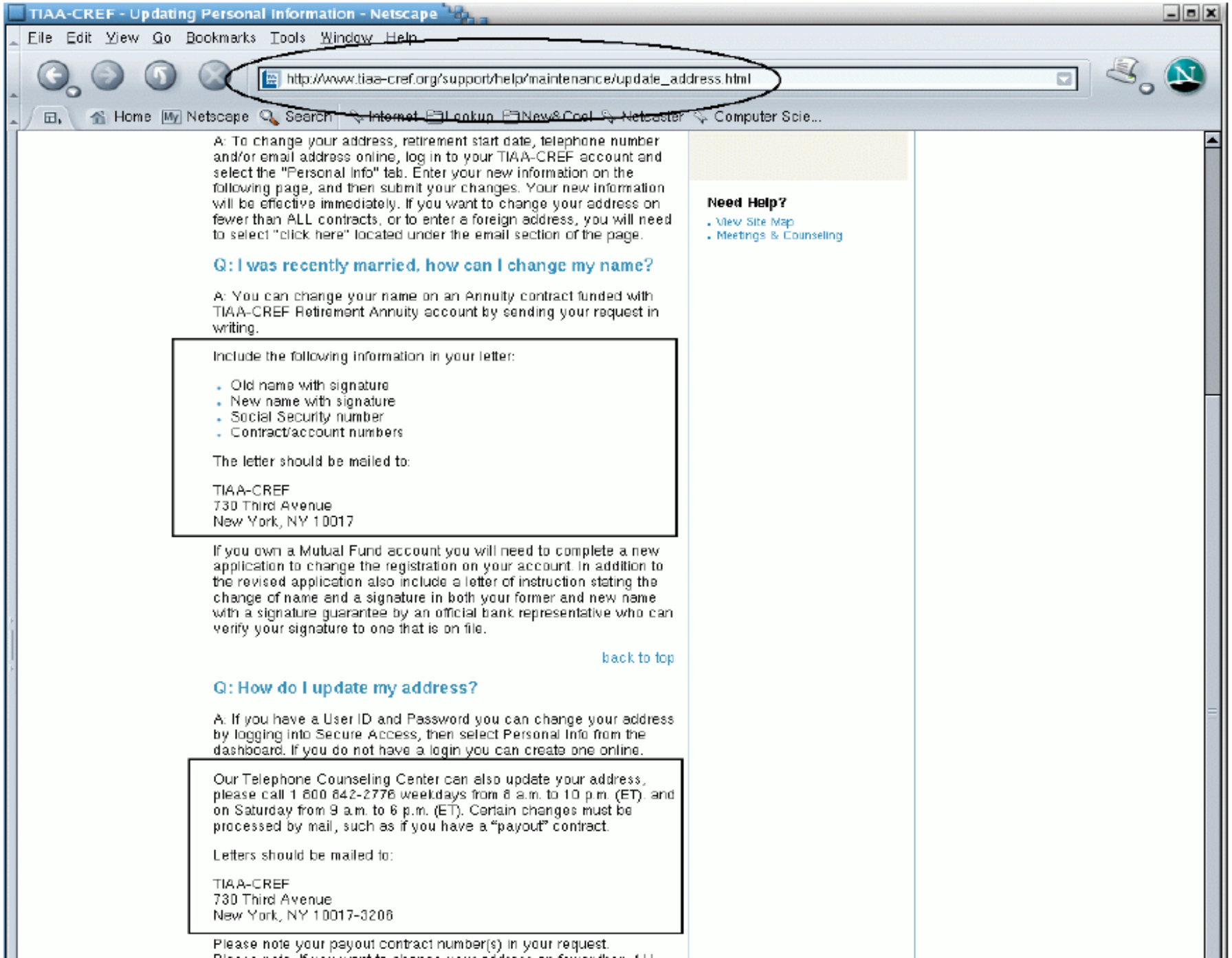
- Vanguard, a brokerage company, used to provide the login window on their home page (http page)
- Response: if a customer was concerned, the customer could hit the Submit button without entering a valid user id and password, and that would take the customer to an SSL protected login page.
- However, Vanguard modified their login process, moving the login window to an SSL-protected page.

Design Flaws

- Break in the chain of trust: websites forward users to new pages that have different domains without notifying the users
- Presenting secure login options on insecure pages: Some sites present login forms that forward to a secure page but do not come from a secure page.
- Contact information/security advice on insecure pages: some sites host their contact information etc. on insecure pages.

Contact information/security advice on insecure pages

- Contact information is considered security-relevant context because users rely on that information being correct for security-sensitive operations.
- Allows modification of the page by replacing the customer service phone numbers with bogus numbers
- Then crooks answer the phone and ask for SSN, birth date, or other confidential information


TIAA-CREF - Updating Personal Information - Netscape
File Edit View Go Bookmarks Tools Window Help
http://www.tiaa-cref.org/support/help/maintenance/update_address.html
Home Netscape Search Internet Lookup New&Cool Netcaster Computer Scie...
A: To change your address, retirement start date, telephone number and/or email address online, log in to your TIAA-CREF account and select the "Personal Info" tab. Enter your new information on the following page, and then submit your changes. Your new information will be effective immediately. If you want to change your address on fewer than ALL contracts, or to enter a foreign address, you will need to select "click here" located under the email section of the page.
Q: I was recently married, how can I change my name?
A: You can change your name on an Annuity contract funded with TIAA-CREF Retirement Annuity account by sending your request in writing.
Include the following information in your letter:
Old name with signature
New name with signature
Social Security number
Contract/account numbers
The letter should be mailed to:
TIAA-CREF
730 Third Avenue
New York, NY 10017
If you own a Mutual Fund account you will need to complete a new application to change the registration on your account. In addition to the revised application also include a letter of instruction stating the change of name and a signature in both your former and new name with a signature guarantee by an official bank representative who can verify your signature to one that is on file.
back to top
Q: How do I update my address?
A: If you have a User ID and Password you can change your address by logging into Secure Access, then select Personal Info from the dashboard. If you do not have a login you can create one online.
Our Telephone Counseling Center can also update your address, please call 1 800 842-2776 weekdays from 8 a.m. to 10 p.m. (ET) and on Saturday from 9 a.m. to 6 p.m. (ET). Certain changes must be processed by mail, such as if you have a "payout" contract.
Letters should be mailed to:
TIAA-CREF
730 Third Avenue
New York, NY 10017-3206
Please note your payout contract number(s) in your request.
Need Help?
View Site Map
Meetings & Counseling

Design Flaws

- Break in the chain of trust: websites forward users to new pages that have different domains without notifying the users
- Presenting secure login options on insecure pages: Some sites present login forms that forward to a secure page but do not come from a secure page.
- Contact information/security advice on insecure pages: some sites host their contact information etc. on insecure pages.
- Inadequate policies for user ids and passwords: It is important to maintain consistent and strong policies on passwords and user ids.

Inadequate policies for user ids and passwords

- Design flaws
  - The use of social security numbers and email addresses for user IDs
    - E.g. LaSalle Bank website, www.lasallebank.com
    - TIAA CREF, www.tiaa-cref.com
  - No policy on allowed passwords creates weak passwords making them vulnerable to dictionary attacks.
- 31% of the banks affected allow e-mail addresses as user names.
- They concluded that a strong username could be more important then a strong password.

Design Flaws

- Break in the chain of trust: websites forward users to new pages that have different domains without notifying the users
- Presenting secure login options on insecure pages: Some sites present login forms that forward to a secure page but do not come from a secure page.
- Contact information/security advice on insecure pages: some sites host their contact information etc. on insecure pages.
- Inadequate policies for user ids and passwords: It is important to maintain consistent and strong policies on passwords and user ids.
- Emailing security sensitive information insecurely

E-mailing security sensitive information insecurely

- Design flaw: the financial sites offer to send security-sensitive statements or passwords via emails.
- If passwords are e-mailed through an insecure mail server, an attacker could intercept unencrypted traffic on the network.

Detecting Design Flaws

- Use a tool for automatically detecting flaws
- They used wget to recursively download the financial institution websites and use scripts to recursively traverse and analyse the web pages

Detecting Breaking in the Chain of Trust

- For each web site, record the domain and search each page for URLs that did not match the domain.
- Looked for two cases:
  - Insecure pages making a transition to a secure page.
  - A secure page making a transition to a secure page.

Presenting Secure Login Options on Insecure Pages

- Search each web page for the string login.
- If so, search the same page for the strings username or user id or password.
- If such strings were found on the same page, we then verified whether the page was displayed using the http protocol.
- http ® contained the design flaw.

Contact Information/Security Advice on Insecure Pages

- Search each web page for the string contact, information, or FAQ.
- If those strings where found, check whether the page was protected with SSL.
- If not, then we considered it to contain the design flaw.

Inadequate Policies for User IDs and Passwords

- The use of email addresses for user IDs
  - Search for the string e-mail
  - If such a page also contained the strings login and user id, it was assumed to violate the property.
  - They manually confirmed the results, filtering out any false matches.

Inadequate Policies for User IDs and Passwords (Cont.)

- Inadequate password strength policies
  - Search for the string password (excluding the Login pages).
  - If the string is found, searched for the presence of one of the following strings: recommendation, strong, or setting.
  - If so, they made a conservative assumption that the website had a policy on setting strong passwords.

E-Mailing Security-Sensitive Information Insecurely

- Search for the presence of either of the two strings statements or password as well as the presence of the two strings sending and e-mail.
- In order to reduce the number of false positives, they assigned values based on proximity.
  - The closer the two sets of words, the higher the value or probability.

Results

- With automated tools (such as this one) false positives are possible
- They tried to manually eliminate them wherever was possible
- Especially the “break-in-chain-of-trust” test has a significant number false positives (30% reported but in fact there were only 17%)
- Most sites made an effort to provide good policies for user ids and passwords

What did they find?

- 30% of the sites broke the chain of trust
- 47% presented a login page on an insecure page
- 55% presented contact and other sensitive information on insecure pages
- 31% allowed e-mail addresses as user names
- 76% of sites have at least one design flaw
- 68% had 2 or more design flaws
- 10% of the sites had all five design flaws
- 24% of sites were completely free of design flaws

Acknowledgements

- Many slides borrowed from Dr. Ping Yang from State University of New York at Binghamton.