

Web Engineering (SE-206)

Lecture # 27, Week#9
Spring, 2020

Testing WebApps

- Testing is the process of exercising a WebApp with the intent of finding (and ultimately correcting) errors.
- Tests must be design to uncover errors in WebApps that are implemented in:
 - different operating systems
 - browsers [or other interface devices such as set-top boxes, personal digital assistants (PDAs), and mobile phones]
 - hardware platforms
 - communications protocols
 - “backroom” applications

The “Dimensions” of Quality - I

- Reviews and testing examine one or more of the following quality dimensions:
 - *Content* is evaluated at both a syntactic and semantic level.
 - At the syntactic level, spelling, punctuation, and grammar are assessed for text-based documents. At a semantic level, correctness (of information presented), consistency (across the entire content object and related objects), and lack of ambiguity are all assessed.
 - *Function* is tested to uncover errors that indicate lack of conformance to stakeholder requirements. Each WebApp function is assessed for correctness, instability, and general conformance to appropriate implementation standards (e.g., Java or XML language standards).
 - *Structure* is assessed to ensure that it properly delivers WebApp content and function, is extensible, and can be supported as new content or functionality is added.

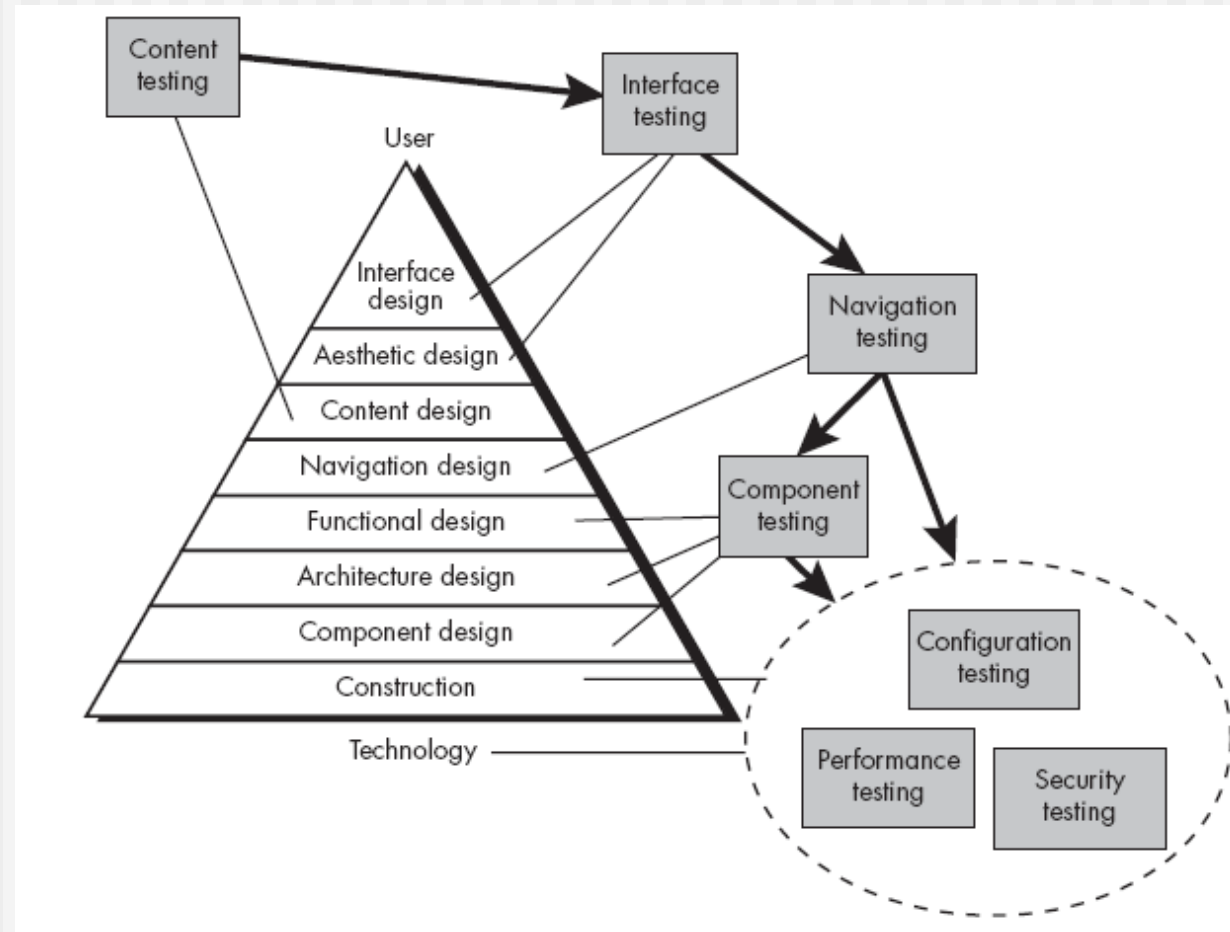
The “Dimensions” of Quality - II

- *Usability* is tested to ensure that each category of user is supported by the interface and can learn and apply all required navigation syntax and semantics.
- *Navigability* is tested to ensure that all navigation syntax and semantics are exercised to uncover any navigation errors (e.g., dead links, improper links, erroneous links).
- *Performance* is tested under a variety of operating conditions, configurations, and loading to ensure that the system is responsive to user interaction and handles extreme loading without unacceptable operational degradation.
- *Compatibility* is tested by executing the WebApp in a variety of different host configurations on both the client and server sides. The intent is to find errors that are specific to a unique host configuration.
- *Interoperability* is tested to ensure that the WebApp properly interfaces with other applications and/or databases.
- *Security* is tested by assessing potential vulnerabilities and attempting to exploit each. Any successful penetration attempt is deemed a security failure.

Testing Strategy

1. The content model for the WebApp is reviewed to uncover errors.
2. The interface model is reviewed to ensure that all use cases have been accommodated.
3. The design model for the WebApp is reviewed to uncover navigation errors.
4. The user interface is tested to uncover errors in presentation and/or navigation mechanics.
5. Selected functional components are unit tested.
6. Navigation throughout the architecture is tested.
7. The WebApp is implemented in a variety of different environmental configurations and is tested for compatibility with each configuration.
8. Security tests are conducted in an attempt to exploit vulnerabilities in the WebApp or within its environment.
9. Performance tests are conducted.
10. The WebApp is tested by a controlled and monitored population of end users. The results of their interaction with the system are evaluated for content and navigation errors, usability concerns, compatibility concerns, and WebApp reliability and performance.

The Testing Process



Content Testing

- Content testing combines both reviews and the generation of executable test cases.
 - **Reviews** are applied to uncover semantic errors in content.
 - **Executable testing** is used to uncover content errors that can be traced to dynamically derived content that is driven by data acquired from one or more databases.
- Content testing has three important objectives:
 - **to uncover syntactic errors** (e.g., typos, grammar mistakes) in text-based documents, graphical representations, and other media,
 - **to uncover semantic errors** (i.e., errors in the accuracy or completeness of information) in any content object presented as navigation occurs, and
 - **to find errors in the organization or structure of content** that is presented to the end user.

Content Testing - Checklist

- Is the information up to date and factually accurate?
- Is the information concise and to the point?
- Is the layout of the content object easy for the user to understand?
- Can information embedded within a content object be found easily?
- Have proper references been provided for all information derived from other sources?
- Is the information presented consistent internally and consistent with information presented in other content objects?
- Can the content be interpreted as being offensive or misleading, or does it open the door to litigation?
- Does the content infringe on existing copyrights or trademarks?
- Does the content contain internal links that supplement existing content? Are the links correct?
- Does the aesthetic style of the content conflict with the aesthetic style of the interface?

Content Testing – Dynamic Content

- When content is created dynamically using information maintained within a database, the following issues are considered:
 - The original client-side request for information is rarely presented in the form [e.g., structured query language (SQL)] that can be input to a database management system (DBMS).
 - The database may be remote to the server that houses the WebApp.
 - *What happens if the WebApp is accessible but the database is not?*
 - Raw data acquired from the database must be transmitted to the WebApp server and properly formatted for subsequent transmittal to the client.
 - The dynamic content object(s) must be transmitted to the client in a form that can be displayed to the end user.

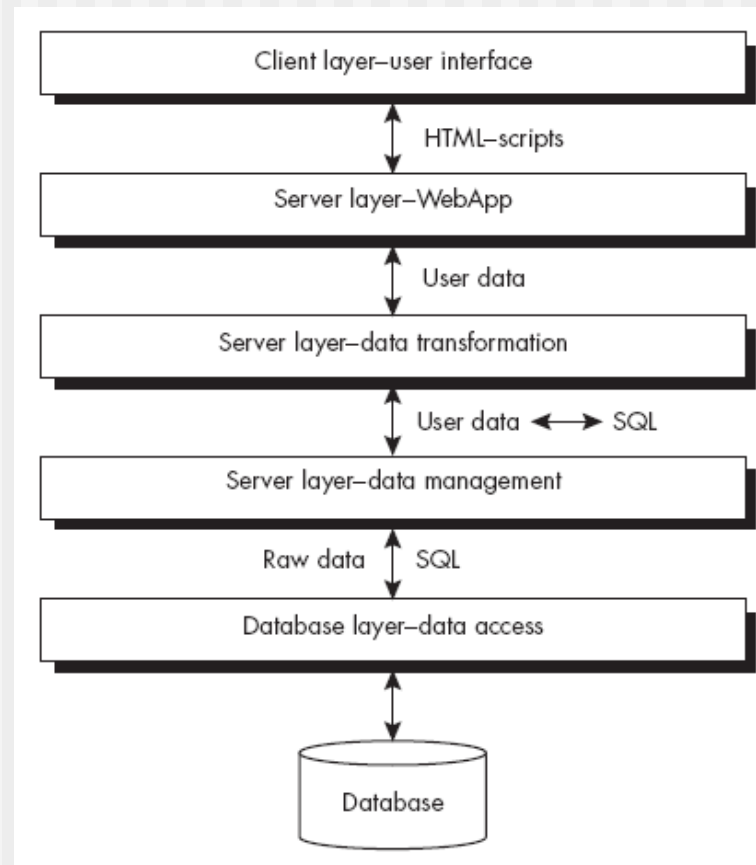
Web Engineering (SE-206)

Lecture # 28, Week#10
Spring, 2020

Content Testing – Dynamic Content

- When content is created dynamically using information maintained within a database, the following issues are considered:
 - The original client-side request for information is rarely presented in the form [e.g., structured query language (SQL)] that can be input to a database management system (DBMS).
 - The database may be remote to the server that houses the WebApp.
 - *What happens if the WebApp is accessible but the database is not?*
 - Raw data acquired from the database must be transmitted to the WebApp server and properly formatted for subsequent transmittal to the client.
 - The dynamic content object(s) must be transmitted to the client in a form that can be displayed to the end user.

Content Testing - Database



User Interface Testing

- Verification and validation of a WebApp user interface occurs at three distinct points in the WebE process.
 - **During communication** (Chapter 4) and modeling (Chapter 7), the interface model is reviewed to ensure that it conforms to customer requirements and to other elements of the analysis model.
 - **During design** (Chapter 9), the interface design model is reviewed to ensure that generic quality criteria established for all user interfaces have been achieved and that application-specific interface design issues have been properly addressed.
 - **During testing** (Chapter 15), the focus shifts to the execution of application-specific aspects of user interaction as they are manifested by interface syntax and semantics. In addition, testing provides a final assessment of usability.

UI Testing Strategy

- Interface features are tested to ensure that design rules, aesthetics, and related visual content are available to the user without error.
- Individual interface mechanisms are tested in a manner that is analogous to unit testing.
- Each interface mechanism is tested within the context of a use case or navigation pathway for a specific user category.
- The complete interface is tested against selected use cases and navigation pathways to uncover errors in the semantics of the interface.
- The interface is tested within a variety of environments (e.g., operating systems, browsers) to ensure that it will be compatible.

User Interface – Testing specific elements (1)

- When a user interacts with a WebApp, the interaction occurs through one or more interface mechanisms. Each mechanism must be tested:
 - **Links.** Navigation mechanisms that link the user to some other content object or function.
 - **Forms.** A structured document containing blank fields that are filled in by the user.
 - **Client-side scripting.** A list of programmed commands in a scripting language (e.g., JavaScript) that handle information input via forms or other user interactions.
 - **Dynamic HTML.** Provides access to content objects that are manipulated on the client side using scripting or cascading style sheets (CSSs).
 - **Client-side pop-up windows.** Small windows that pop up without user interaction.
 - **Server-side scripts.** Black-box tests are conducted with an emphasis on data integrity and script processing once validated data has been received. In addition, performance testing can be conducted.

User Interface – Testing specific elements (2)

- When a user interacts with a WebApp, the interaction occurs through one or more interface mechanisms. Each mechanism must be tested:
 - **Streaming and push content.** *Streaming content* is encountered when material (usually audio or video) is downloaded in a manner that allows it to be displayed while it is still being downloaded (rather than having to wait for the entire content to be downloaded). *Push content* is encountered when content objects are downloaded automatically from the server side rather than waiting for a request from the client side. Both streaming and push content present testing challenges.
 - **Cookies.** A block of data sent by the server and stored by a browser as a consequence of a specific user interaction. The content of the data is WebApp-specific (e.g., user identification data or a list of items that have been selected for purchase by the user).
 - **Application-specific interface mechanisms.** Include one or more “macro” interface mechanisms such as a shopping cart, credit card processing, or a shipping cost calculator.

Usability Testing

- Similar to interface semantics testing in the sense that it evaluates:
 - the degree to which users can interact effectively with the WebApp
 - the degree to which the WebApp guides users' actions, provides meaningful feedback and enforces a consistent interaction approach.
- Determines the degree to which the WebApp interface makes the user's life easy

Usability Testing

- Define a set of usability testing categories and identify goals for each.
- Design tests that will enable each goal to be evaluated.
- Select participants who will conduct the tests.
- Log the details of the participants' interaction with the WebApp while testing is conducted.
- Develop a mechanism for assessing the usability of the WebApp.
- Usability testing can occur at a variety of different levels of abstraction:
 - (1) the usability of a specific interface mechanism (e.g., a form) can be assessed
 - (2) the usability of a complete Web page (encompassing interface mechanisms, data objects, and related functions) can be evaluated, or
 - (3) the usability of the complete WebApp can be considered.

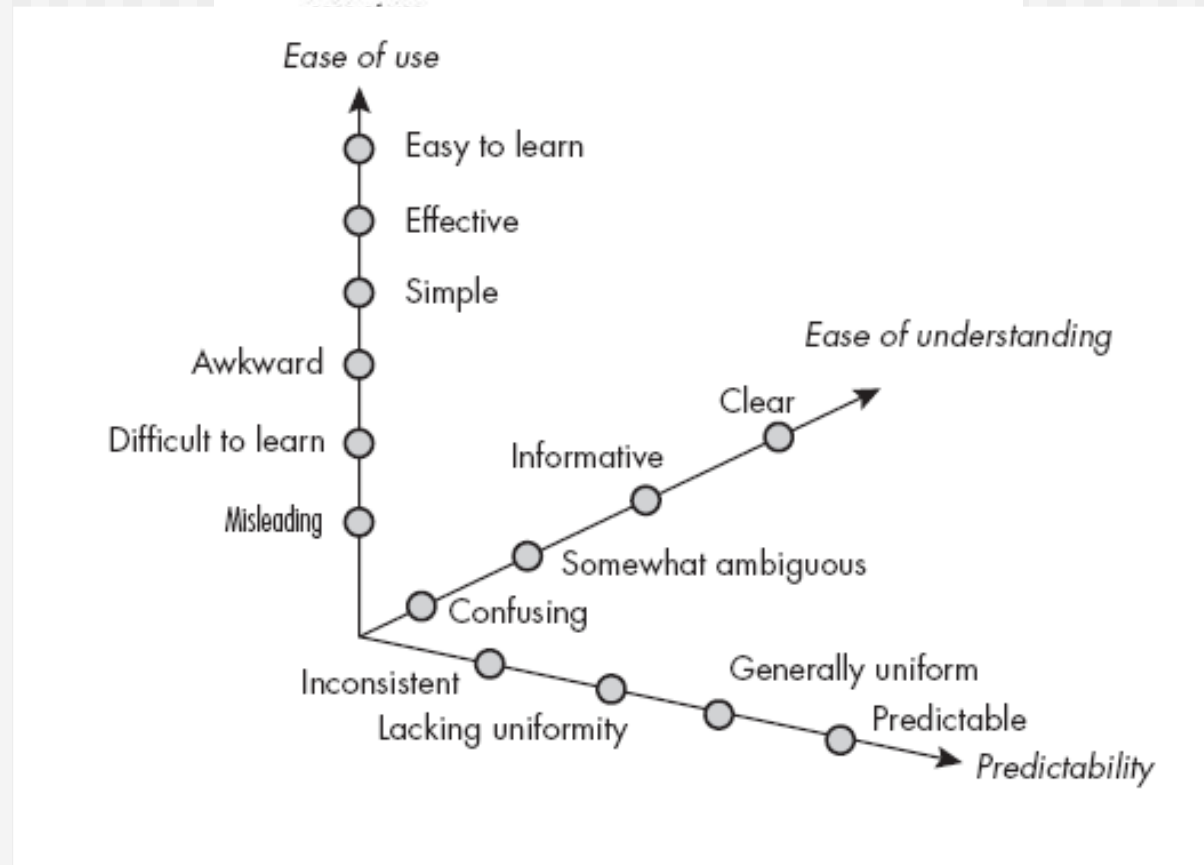
Usability Test Categories

- **Interactivity.** Are interaction mechanisms (e.g., pull-down menus, buttons, pointers) easy to understand and use?
- **Layout.** Are navigation mechanisms, content, and functions placed in a manner that allows the user to find them quickly?
- **Readability.** Is text well written and understandable? Are graphic representations intuitive and easy to understand?
- **Aesthetics.** Do the layout, color, typeface, and related characteristics lead to ease of use? Do users “feel comfortable” with the look and feel of the WebApp?
- **Display characteristics.** Does the WebApp make optimal use of screen size and resolution?
- **Time sensitivity.** Can important features, functions, and content be used or acquired in a timely manner?
- **Personalization.** Does the WebApp appropriately tailor itself to the specific needs of different user categories or individual users?

Usability Evaluation: Checklist

- Is the system usable without continual help or instruction?
- Do the rules of interaction help a knowledgeable user to work efficiently?
- Do interaction mechanisms become more flexible as users become more knowledgeable?
- Has the system been tuned to the physical and social environment in which it will be used?
- Are users aware of the state of the system? Do users know where they are at all times?
- Is the interface structured in a logical and consistent manner?
- Are interaction mechanisms, icons, and procedures consistent across the interface?
- Does the interaction anticipate errors and help the user correct them?
- Is the interface tolerant of errors that are made?
- Is the interaction simple?

Qualitative Assessment of Usability



Compatability Testing

- WebApps operate in complex (and often unpredictable) environments
 - Different browsers, screen resolutions, operating systems, plug-ins, access bandwidths, etc.
- Serious errors can be caused by obscure combinations
- Most common problem is deterioration in usability:
 - Download speeds may become unacceptable
 - Missing plug-ins may make content unavailable
 - Browser differences can change page layout or legibility
 - Forms may be improperly organized.
- Compatibility testing strives to uncover these problems before the WebApp goes online.
 - First step is to define a set of “commonly encountered” client-side configurations and their variants.
 - Next, derive a series of compatibility validation tests (from existing interface tests, navigation tests, performance tests, and security tests).

Component-Level Testing

- *Component-level testing*, also called *function testing*, focuses on a set of tests that attempt to uncover errors in WebApp functions
- Applies the following test-case design methods:
 - Equivalence partitioning
 - Boundary value analysis
 - Path testing

Selecting Components to Test

- Which functionality in the Web site is most critical to its purpose?
- Which areas of the site require the heaviest database interaction?
- Which aspects of the site's CGI, applets, ActiveX components, and so on are most complex?
- What types of problems would cause the most complaints or the worst publicity?
- What areas of the site will be the most popular?
- What aspects of the site have the highest security risks?

Navigation Testing

- Each of the following navigation mechanisms should be tested [Spl01]:
 - **Navigation links.** These mechanisms include internal links within the WebApp, external links to other WebApps, and anchors within a specific Web page.
 - **Redirects.** These links come into play when a user requests a nonexistent URL or selects a link whose destination has been removed or whose name has changed.
 - **Bookmarks.** Although bookmarks are a browser function, the WebApp should be tested to ensure that a meaningful page title can be extracted as the bookmark is created and that dynamic pages are bookmarked appropriately.
 - **Frames and framesets.** Each frame contains the content of a specific Web page; a frameset contains multiple frames and enables the display of multiple Web pages at the same time.
 - **Site maps.** A site map provides a complete table of contents for all Web pages.
 - **Internal search engines.** An internal (local) search engine allows the user to perform a key word search within the WebApp to find needed content.

Navigation Semantics

- As navigation design is conducted, you create “a set of information and related navigation structures that collaborate in the fulfillment of a subset of related user requirements” [Cac02].
- These are sometimes referred to as *navigation semantic units* (NSUs) and are defined by a set of navigation paths (called “ways of navigating”) that connect navigation nodes (e.g., Web pages, content objects, or functionality).
- Taken as a whole, each NSU allows a user to achieve specific requirements defined by one or more use cases for a user category.
- Navigation testing exercises each NSU to ensure that these requirements can be achieved.

Navigation Semantic Testing - I

- Is the NSU achieved in its entirety without error?
- Is every *navigation node* (a destination defined for an NSU) reachable within the context of the navigation paths defined for the NSU?
- If the NSU can be achieved using more than one navigation path, has every relevant path been tested?
- If guidance is provided by the user interface to assist in navigation, are directions correct and understandable as navigation proceeds?
- Is there a mechanism (other than the browser back arrow) for returning to the preceding navigation node and to the beginning of the navigation path?
- Do mechanisms for navigation within a large navigation node (e.g., anchor point links for a long Web page) work properly?
- If a function is to be executed at a node and the user chooses not to provide input, can the remainder of the NSU be completed?

Navigation Semantic Testing - II

- If a function is executed at a node and an error in function processing occurs, can the NSU be completed?
- Is there a way to discontinue the navigation before all nodes have been reached, but then return to where the navigation was discontinued and proceed from there?
- Is every node reachable from the site map? Are node names meaningful to end users?
- If a node within an NSU is reached from some external source, is it possible to process to the next node on the navigation path? Is it possible to return to the previous node on the navigation path?
- Do users understand their location within the content architecture as the NSU is executed?

Configuration Testing

- Configuration variability and instability are important factors that make Web engineering a challenge.
 - Hardware, operating system(s), browsers, storage capacity, network communication speeds, and a variety of other client-side factors are difficult to predict for each user.
- The job of configuration testing is to test a set of probable client-side and server-side configurations to ensure that the user experience will be the same on all of them and to isolate errors that may be specific to a particular configuration.

Testing Strategy

- **Server-side.** configuration test cases are designed to verify that the projected server configuration [i.e., WebApp server, database server, operating system(s), firewall software, concurrent applications] can support the WebApp without error.
- **Client-side.** On the client side, configuration tests focus more heavily on WebApp compatibility with configurations that contain one or more permutations of the following components:
 - **Hardware.** CPU, memory, storage, and printing devices
 - **Operating systems.** Linux, Macintosh OS, Microsoft Windows, a mobile-based OS
 - **Browser software.** FireFox, Internet Explorer, Safari, Mozilla/Netscape, Opera, and others
 - **User interface components.** Active X, Java applets, and others
 - **Plug-ins.** QuickTime, RealPlayer, and many others
 - **Connectivity.** Cable, DSL, regular modem, industry-grade connectivity (e.g., T1 lines)

Web Engineering (SE-206)

Lecture # 29, Week#10
Spring, 2020

Security and Performance Testing

- Security and performance testing address the three distinct elements of the WebApp infrastructure
 - the server-side environment that provides the gateway to Internet users
 - the network communication pathway between the server and the client machine
 - the client-side environment that provides the end user with a direct interface to the WebApp.
- **Security testing** focuses on unauthorized access to WebApp content and functionality along with other systems that cooperate with the WebApp on the server side.
- **Performance testing** focuses on the operating characteristics of the WebApp and on whether those operating characteristics meet the needs of end users.

Security Testing

- One or more of the following security elements is implemented [Ngu01]:
 - **Firewalls.** A filtering mechanism that is a combination of hardware and software that examines each incoming packet of information to ensure that it is coming from a legitimate source, blocking any data that are suspect.
 - **Authentication.** A verification mechanism that validates the identity of all clients and servers, allowing communication to occur only when both sides are verified.
 - **Encryption.** An encoding mechanism that protects sensitive data by modifying it in a way that makes it impossible to read by those with malicious intent. Encryption is strengthened by using *digital certificates* that allow the client to verify the destination to which the data are transmitted.
 - **Authorization.** A filtering mechanism that allows access to the client or server environment only by those individuals with appropriate authorization codes (e.g., user ID and password).
- Security tests should be designed to probe each of these security technologies in an effort to uncover security holes that can be exploited by those with malicious intent.

Performance Testing

- Objectives:
 - Does the server response time degrade to a point where it is noticeable and unacceptable?
 - At what point (in terms of users, transactions, or data loading) does performance become unacceptable?
 - What system components are responsible for performance degradation?
 - What is the average response time for users under a variety of loading conditions?
 - Does performance degradation have an impact on system security?
 - Is WebApp reliability or accuracy affected as the load on the system grows?
 - What happens when loads that are greater than maximum server capacity are applied?
 - What is the impact of poor performance on company revenues?
- **Load testing** determines how the WebApp and its server-side environment will respond to various loading conditions.
- **Stress testing** is a continuation of load testing, but in this instance the variables, N , T , and D are forced to meet and then exceed operational limits.

Performance Testing

Load testing can also be used to assess recommended connection speeds for users of the WebApp. Overall throughput P is computed in the following manner:

$$P = N \times T \times D$$

As an example, consider a popular sports news site. At any given time, 4000 concurrent users submit a request (a transaction T) once every 30 seconds on average. Each transaction requires the WebApp to download a news article that averages 12 kbytes in length. Therefore,

$$N = 4000 \text{ users}$$

$$T = 0.033 \text{ transactions per second per user}$$

$$D = 12 \text{ kbyte per transaction}$$

and throughput can be calculated as

$$P = 4000 \times 0.033 \times 12 \approx 1600 \text{ kbyte/s}$$

The network connection for the server would therefore have to support this average data rate and should be tested to ensure that it does.

Web Engineering (SE-206)

Lecture # 30, Week#10
Spring, 2020

Transport Layer Security (TLS)

One of the most widely used security services

Can be embedded in specific packages

Most browsers come equipped with TLS, and most Web servers have implemented the protocol

Defined in RFC 5246

Could be provided as part of the underlying protocol suite and therefore be transparent to applications

Is an Internet standard that evolved from a commercial protocol known as Secure Sockets Layer (SSL)

Is a general purpose service implemented as a set of protocols that rely on TCP



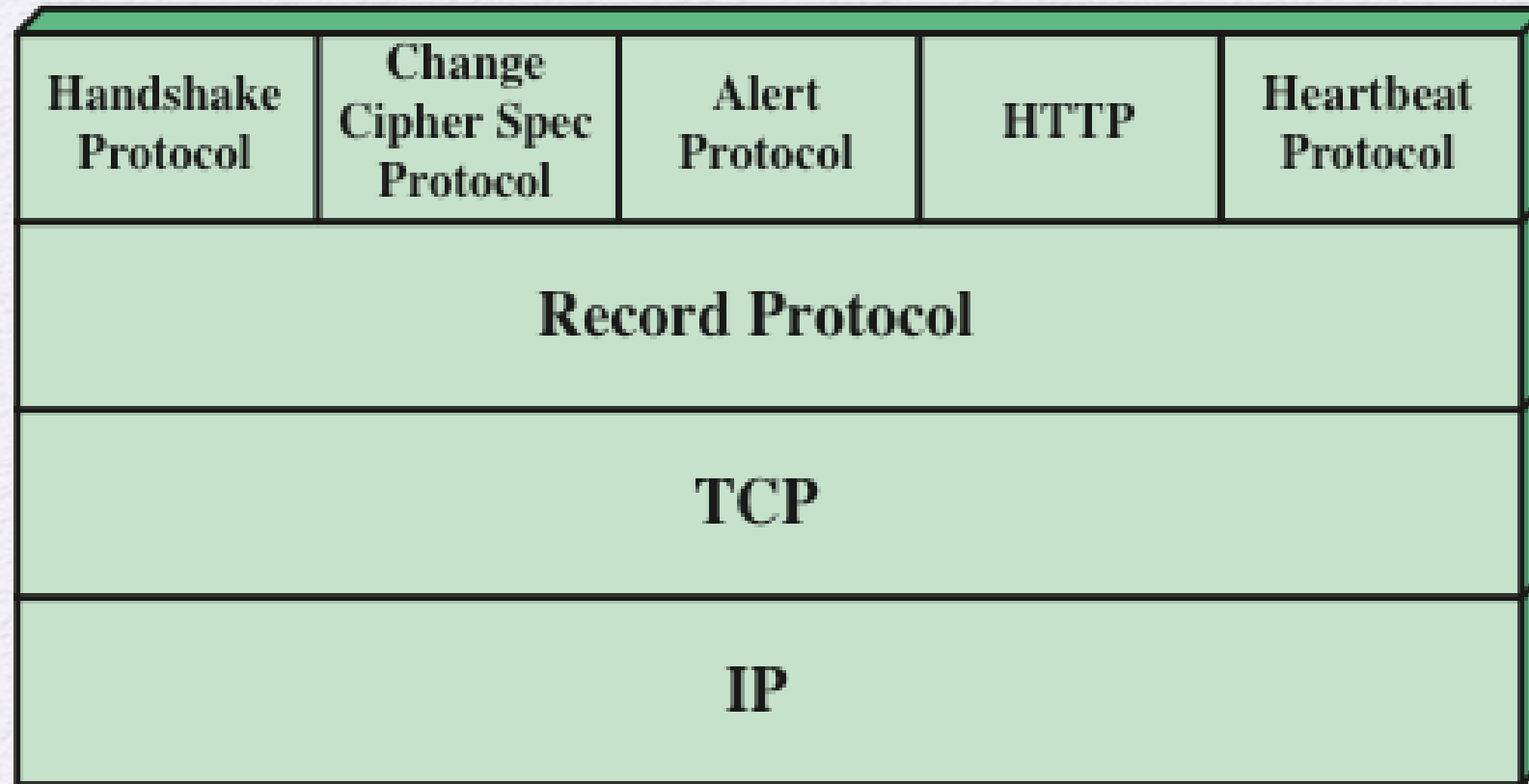
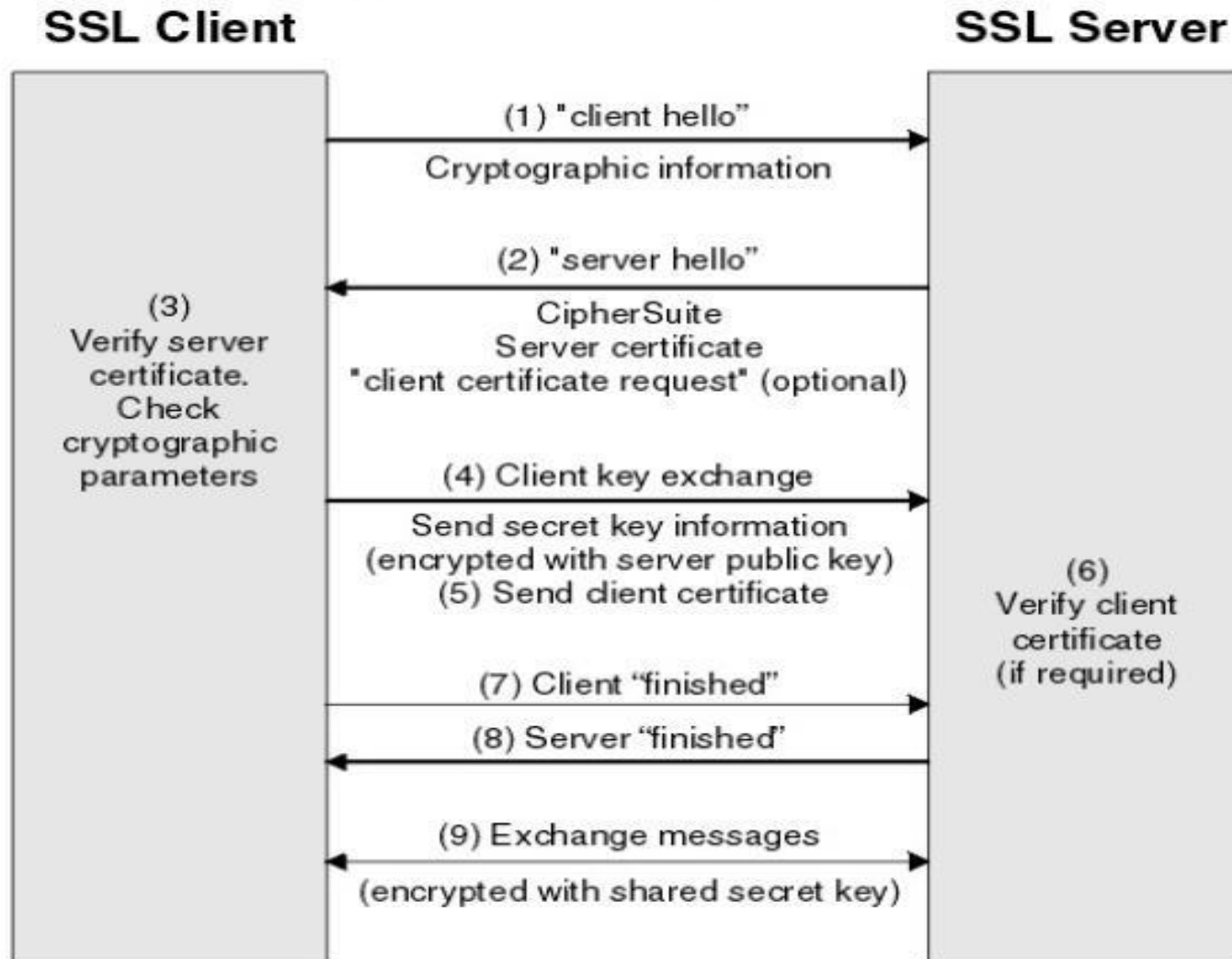


Figure 17.2 SSL/TLS Protocol Stack

SSL Handshake Protocol

- Allows server & client to:
 - authenticate each other
 - to negotiate encryption & MAC algorithms and keys
- Comprises a series of messages exchanged in phases:
 1. Establish Security Capabilities (to agree on encryption, MAC, and key-exchange algorithms)
 2. Server Authentication and Key Exchange
 3. Client Authentication and Key Exchange
 4. Finish

Figure 1. Overview of the SSL or TLS handshake



1. Client Hello

Information that the server needs to communicate with the client using SSL. This includes the SSL version number, cipher settings, session-specific data.

2. Server Hello

Information that the server needs to communicate with the client using SSL. This includes the SSL version number, cipher settings, session-specific data.

3. Authentication and Pre-Master Secret

Client authenticates the server certificate. (e.g. Common Name / Date / Issuer) Client (depending on the cipher) creates the pre-master secret for the session, Encrypts with the server's public key and sends the encrypted pre-master secret to the server.

4. Decryption and Master Secret

Server uses its private key to decrypt the pre-master secret. Both Server and Client perform steps to generate the master secret with the agreed cipher.

5. Encryption with Session Key

Both client and server exchange messages to inform that future messages will be encrypted.

HTTPS

(HTTP over SSL)

- Refers to the combination of HTTP and SSL to implement secure communication between a Web browser and a Web server
- The HTTPS capability is built into all modern Web browsers
- A user of a Web browser will see URL addresses that begin with https:// rather than http://
- If HTTPS is specified, port 443 is used, which invokes SSL
- Documented in RFC 2818, *HTTP Over TLS*
 - There is no fundamental change in using HTTP over either SSL or TLS and both implementations are referred to as HTTPS
- When HTTPS is used, the following elements of the communication are encrypted:
 - URL of the requested document
 - Contents of the document
 - Contents of browser forms
 - Cookies sent from browser to server and from server to browser
 - Contents of HTTP header

Connection Initiation

For HTTPS, the agent acting as the HTTP client also acts as the TLS client

The client initiates a connection to the server on the appropriate port and then sends the TLS ClientHello to begin the TLS handshake

When the TLS handshake has finished, the client may then initiate the first HTTP request

All HTTP data is to be sent as TLS application data

There are three levels of awareness of a connection in HTTPS:

At the HTTP level, an HTTP client requests a connection to an HTTP server by sending a connection request to the next lowest layer

- Typically the next lowest layer is TCP, but it may also be TLS/SSL

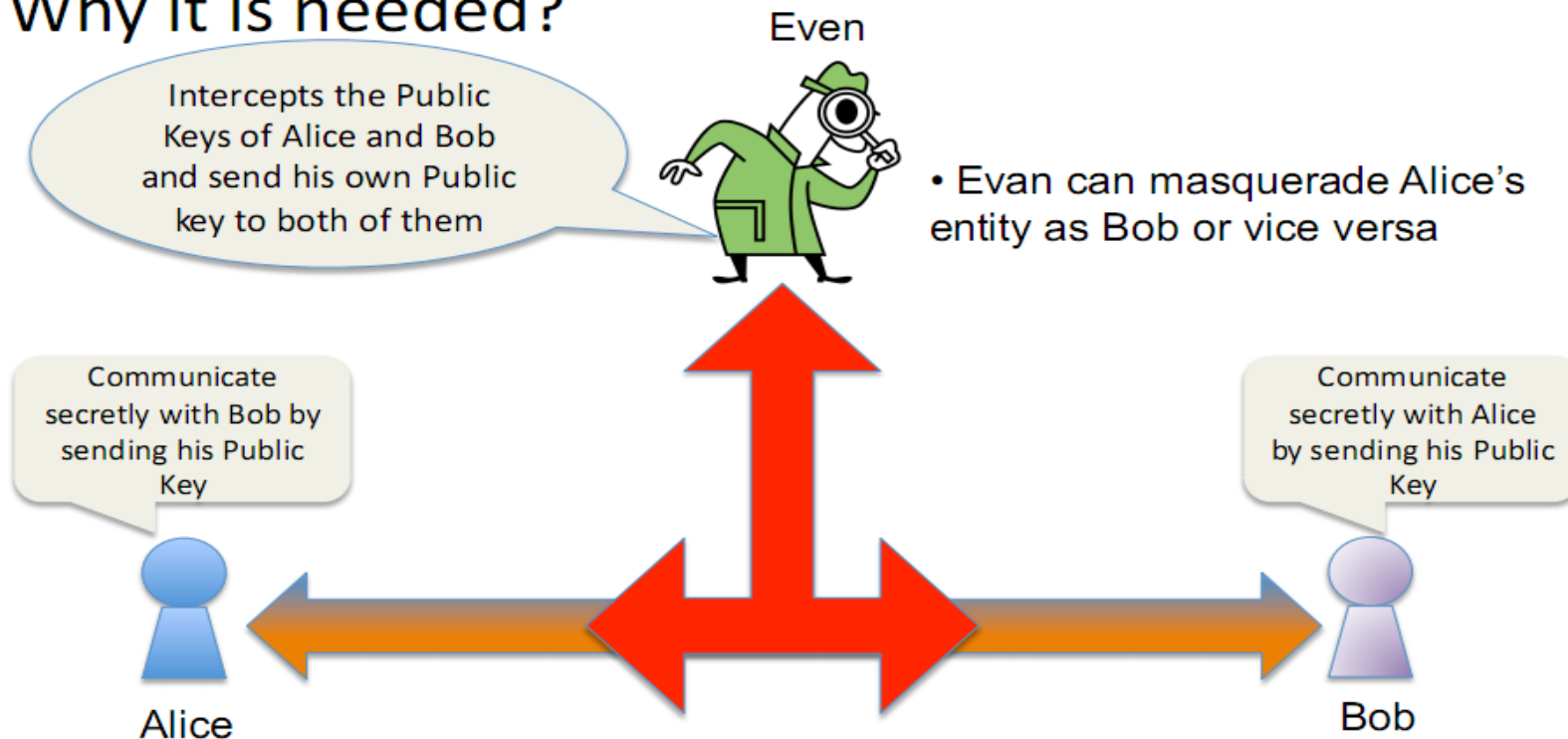
At the level of TLS, a session is established between a TLS client and a TLS server

- This session can support one or more connections at any time

A TLS request to establish a connection begins with the establishment of a TCP connection between the TCP entity on the client side and the TCP entity on the server side

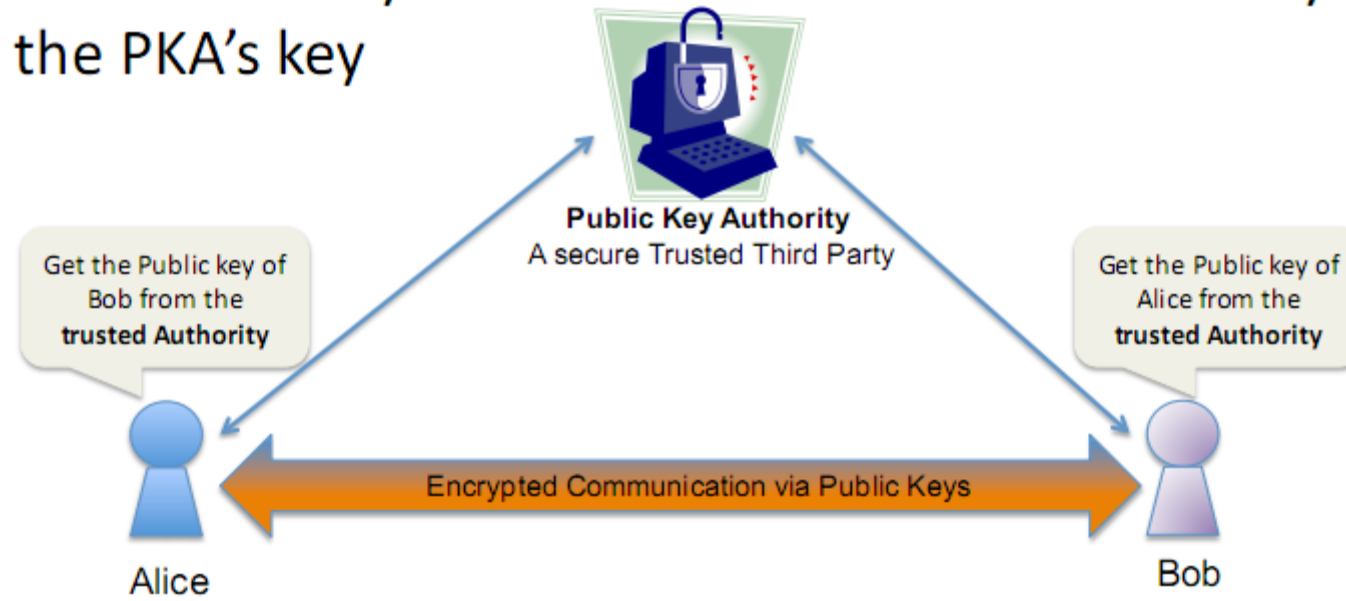
Public Key Authority (PKA)

- A well-trusted Authority is required to distribute keys from the directory
- Why it is needed?

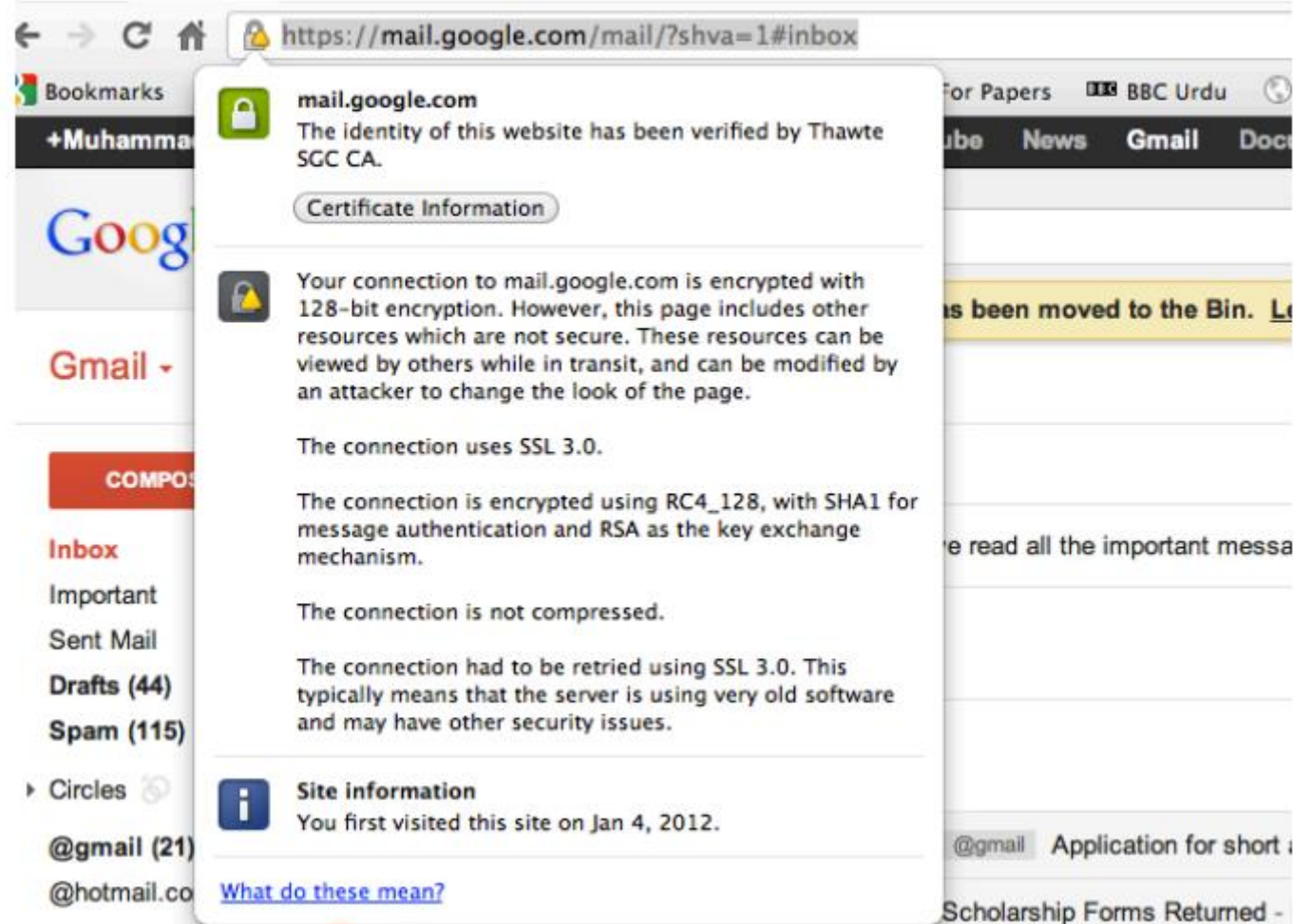


Public Key Authority (PKA)

- **Primary Role:** to digitally sign and publish the public key bound to a given user
- PKA does this with its own private key, so that trust in the user key relies on one's trust in the validity of the PKA's key



Public Key Certificates



The screenshot shows a web browser window with the address bar displaying `https://mail.google.com/mail/?shva=1#inbox`. A security warning pop-up is visible over the page content. The pop-up has a green lock icon and the text "mail.google.com" followed by "The identity of this website has been verified by Thawte SGC CA." Below this is a "Certificate Information" button. The main body of the warning contains a yellow triangle icon and the following text: "Your connection to mail.google.com is encrypted with 128-bit encryption. However, this page includes other resources which are not secure. These resources can be viewed by others while in transit, and can be modified by an attacker to change the look of the page." Below this, it states "The connection uses SSL 3.0.", "The connection is encrypted using RC4_128, with SHA1 for message authentication and RSA as the key exchange mechanism.", "The connection is not compressed.", and "The connection had to be retried using SSL 3.0. This typically means that the server is using very old software and may have other security issues." At the bottom of the pop-up is a "Site information" section with an information icon, stating "You first visited this site on Jan 4, 2012." and a link "What do these mean?". The background of the browser shows the Gmail interface with the "Inbox" selected in the left sidebar.

mail.google.com
The identity of this website has been verified by Thawte SGC CA.

Certificate Information

Your connection to mail.google.com is encrypted with 128-bit encryption. However, this page includes other resources which are not secure. These resources can be viewed by others while in transit, and can be modified by an attacker to change the look of the page.

The connection uses SSL 3.0.

The connection is encrypted using RC4_128, with SHA1 for message authentication and RSA as the key exchange mechanism.

The connection is not compressed.

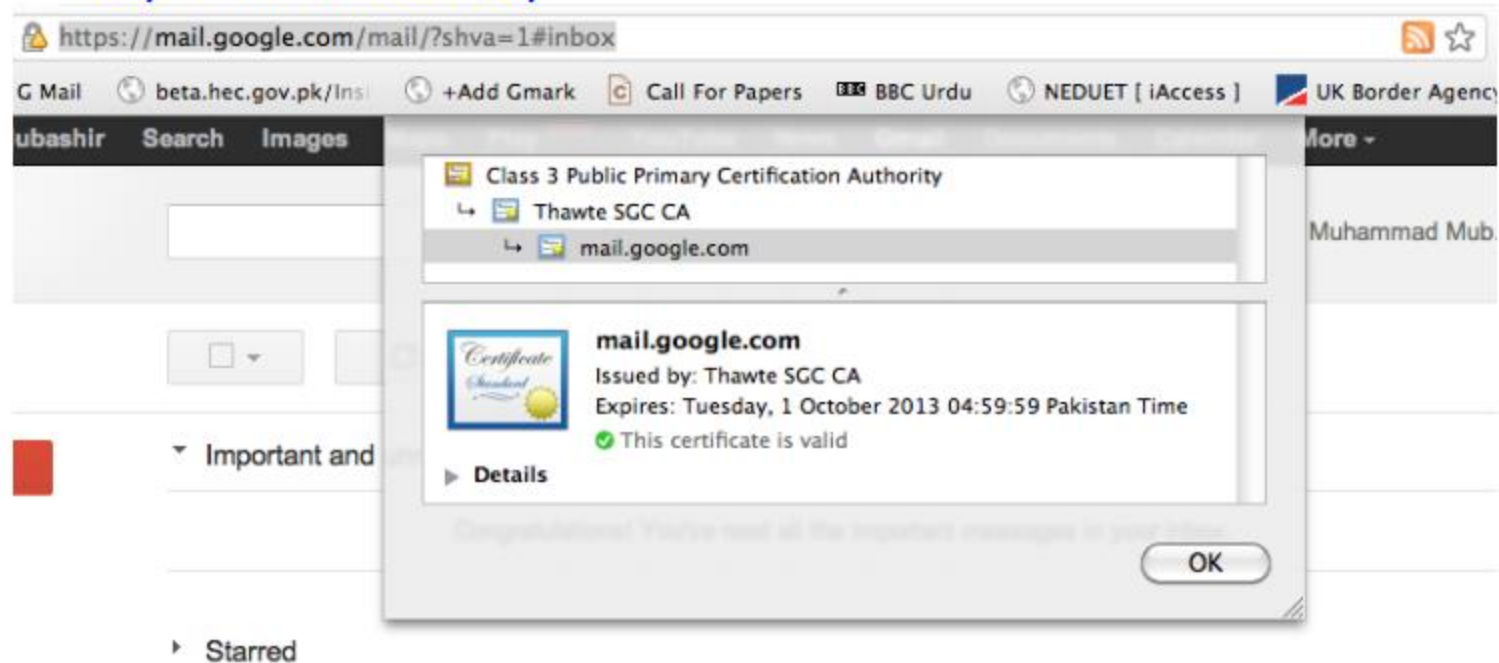
The connection had to be retried using SSL 3.0. This typically means that the server is using very old software and may have other security issues.

Site information
You first visited this site on Jan 4, 2012.

[What do these mean?](#)

Public Key Certificates

- In cryptography, a public key certificate (also known as a digital certificate or identity certificate) is an electronic document which uses a digital signature to bind a public key with an identity



X.509 Public Key Certificate

Subject Name
Country US
State/Province California
Locality Mountain View
Organization Google Inc
Common Name mail.google.com

The entity which is identified

X.509 is an ITU-T Standard for Public Key Certificates

Issuer Name
Country ZA
Organization Thawte Consulting (Pty) Ltd.
Common Name Thawte SGC CA

Serial Number 2B 9F 7E E5 CA 25 A6 25 14 20 47 82 75 3A 9B 89
Version 3

Used to uniquely identify the certificate

Signature Algorithm SHA-1 with RSA Encryption (1 2 840 113549 1 1 5)
Parameters none

Not Valid Before Wednesday, 26 October 2011 05:00:00 Pakistan Time
Not Valid After Tuesday, 1 October 2013 04:59:59 Pakistan Time

Public Key Info

Algorithm RSA Encryption (1 2 840 113549 1 1 1)
Parameters none
Public Key 128 bytes : AF 39 15 98 68 E4 92 FE ...
Exponent 65537

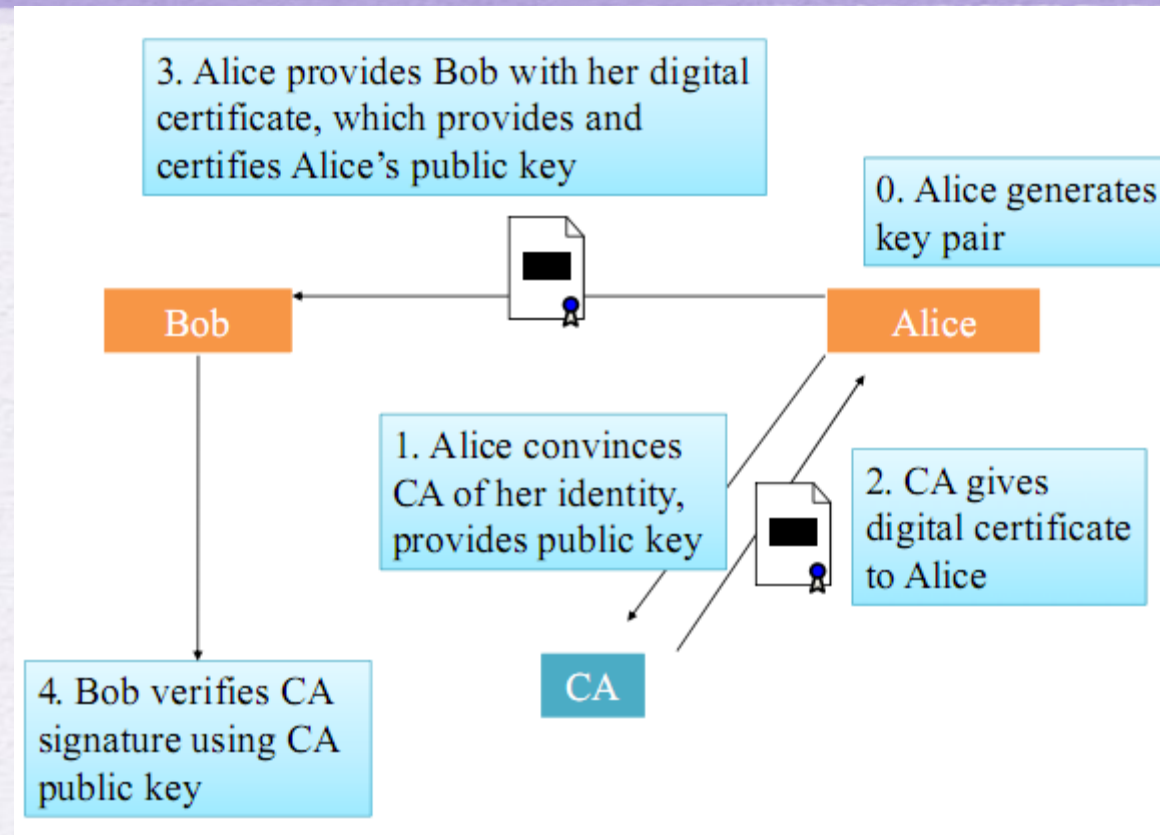
Key Size 1024 bits
Key Usage Encrypt, Verify, Derive

Signature 128 bytes : 35 80 11 CD 52 3E 84 29 ...

A Complete Picture

Public Key Infrastructure (PKI)

- A system for the **creation**, **storage**, and **distribution** of **digital certificates** which are used to verify that a particular **Public Key** belongs to a certain **Entity**.
- The PKI creates **digital certificates** which map public keys to entities, securely stores these certificates in a central repository, and revokes them if needed.
- A PKI consists of
 - **Certificate Authority (CA)**: Both issues and verifies the digital certificates.
 - **Registration Authority** which verifies the identity of users requesting information from the CA
 - **Repository**: A central directory -- i.e. a secure location in which to store and index keys.
 - **Archive**: Store sufficient information to be used for solving future disputes of old documents



Public Key Infrastructure (PKI)

