Security – Keys, Digital Signatures and Certificates II

CS3524 Distributed Systems and Security

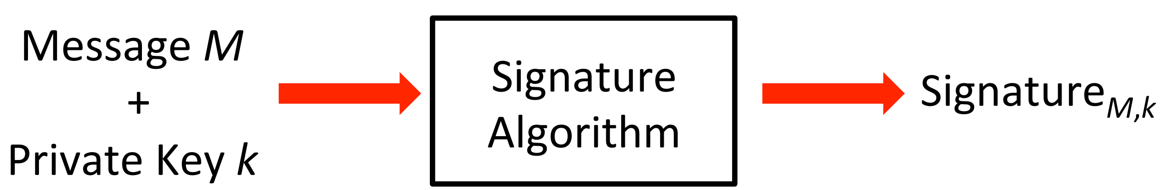
Lecture 20

Digital Signatures

- Public key cryptography can also be used for creating digital signatures
  - Identifies reliably the originator of a sent digital object (file / document, message etc.)
- Encryption of message with private key
  - Instead of a sender using the public key of the receiver, the sender’s private key is used to create a unique signature

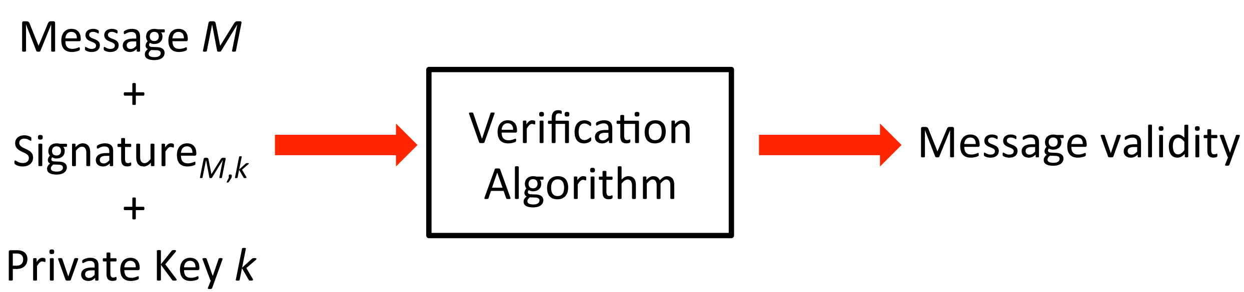
Digital Signatures
Creation from a Message

- A digital signature for a particular message is created by encoding a message with the private key
  - The message itself is not secret as anybody can decode it with the public key
- Only the person with the private key can produce the signature



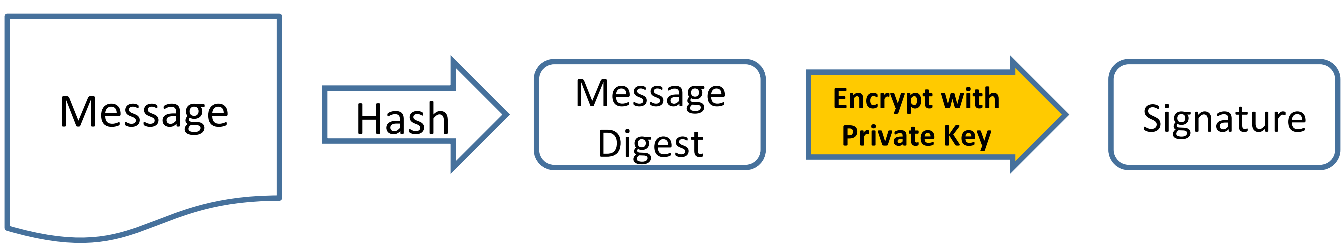
Digital Signature
Verification

- A transmitted digital signature can be verified with the public key, identifies uniquely the sender.
- As public key is public, anyone can verify that the signature is valid



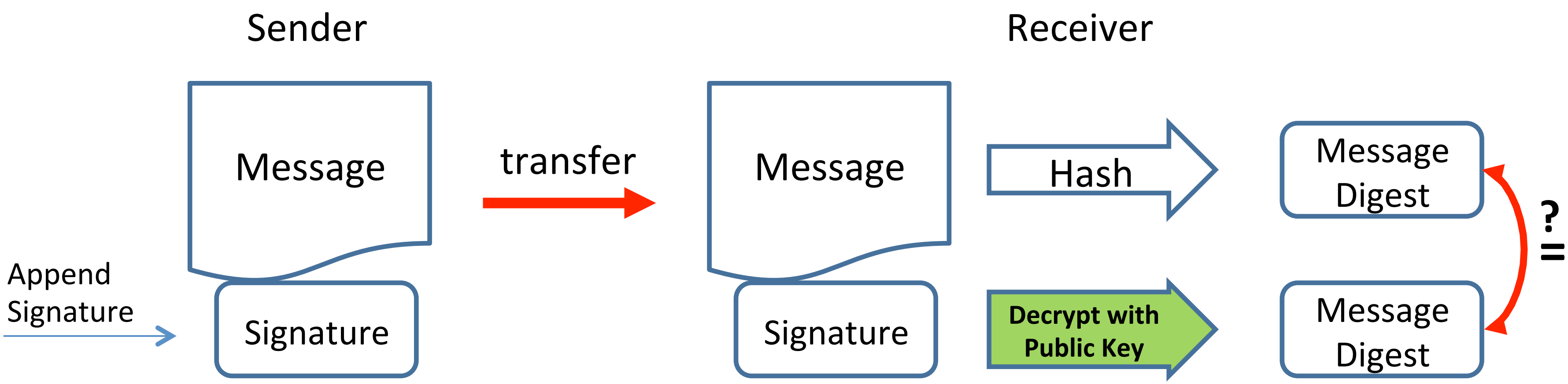
Digital Signature – Message Digest

- Encoding the whole message to produce a digital signature is not feasible
- Solution: create a “Message Digest”
  - Is a kind of “fingerprint” of a message, which is much shorter
  - the message digest is encrypted with the Private Key to create a unique signature for the message
- Calculation of the message digest
  - Apply a hash function to the message



Digital Signature – Message Digest

- Sender:
  - Creates signature from message digest
  - Appends signature to message and sends it
- Receiver
  - Recreates the message digest from the received message
  - Decrypts the signature with the public key of the sender
  - Compares the two results to check origin of message



Message Authentication

- Use own private key K_{PRIV} to authenticate own message
  - Generate digest D from message M:
    - D = hash(M)
  - Encrypt digest D with private key K_{PRIV} to create signature S:
    - S = encrypt(D, K_{PRIV})
- Send both the message and the signature to the recipient
  - Send: <M, encrypt(D, K_{PRIV})>

Message Integrity

- Use the public key K_{PUB} of the sender to check the integrity of the message
  - Receive: <M, encrypt(D, K_{PRIV})>
  - Generate digest D' from received message M:
    - D' = hash(M)
  - Decrypt signature S of sender with public key K_{PUB} to derive the sender's digest D:
    - D = decrypt(encrypt(D, K_{PRIV}), K_{PUB})
- Check Integrity:
  - If $D' = D$, then we have evidence to trust that the message has not been tampered with

Digital Signatures

- Encryption + Digital Signature provides the following benefits:
  - Integrity
  - Confidentiality
  - Authentication
    - The signature is unique not only to the person who owns the (private) key, but also to the message associated with it
  - Non-repudiation
    - A person signing a message cannot deny signing the message

Problem with Key Distribution Authenticity of Public Keys

- We know:
  - Messages can be encrypted with the public key of the recipient – can only decode the message with the private key
  - People therefore publish their public keys
- Problem
  - How do we know that a public key is indeed owned by a person we want to send a confidential message to?
  - Imposters can pretend to be a particular person / organisation
    - Question: am I encrypting my private message with the bank's public key, or with a public key published by an imposter?

Digital Certificate

- Helps to address the problem of public-key distribution and authenticity
  - Certification:
    - digital certificates are introduced to prove authenticity of public keys
  - Validation:
    - ability to check that the binding of a public key to a certificate is authentic
- Is a “statement of originality” by a third party, the “Certificate Authority” (CA)
  - Binds a public key to a particular owner
  - A digital certificate itself is authenticated with a digital signature, signed by the CA with its private key
    - One’s trust that a certified public key is original relies on one’s trust in the validity of the CA’s key

Public Key Infrastructure

- A Public Key Infrastructure provides the environment for the management of digital certificates
- Elements of a PKI
  - “Certificate Authority” (CA)
    - Is a trusted third party that issues and verifies digital certificates
    - Uses its own private key to sign digital certificates
  - “Registration Authority” (RA)
    - verifies the identity of users requesting information from the CA
  - A central repository for storing public keys

Digital Certificate

- Format
  - Issued by: Identity of the issuing Certificate Authority
  - Issued to: Identity of the receiver of the Certificate
  - Valid dates
    - Start date
    - Expiration date
  - Key Info
    - Contains the public key of the receiver of the certificate, e.g. the public key element {n,e} for RSA digital signatures

Problems with Certificates

- What happens if someone loses their key, or if a key is stolen?
- What happens if a CA's key is compromised?
- What if the keyholder's information changes?

On-Line Validation

- We could ask the issuing CA if the certificate we are looking at is still good
  - This is very similar to a credit card check
- PRO
  - Immediate notification of certificate revocation
- CON
  - Do we really want to ask about every certificate?
  - Can a CA handle such an onslaught of certificate queries?

Revocation Lists

- These are lists of “bad” certificates that are published regularly by the CAs and stored locally by the end user
- PRO
  – We don’t have to contact the CA to check status
  – Reduces the communication requirements of the CA
- CON
  – Certificates can go bad before we get the latest list
  – These lists can be huge

Can we Trust a Certificate Authority?

- Do we really know who these CAs are?
- There are (as of March 2011) almost 650 CAs!
- Firefox ships with around 150 trusted CAs, Internet Explorer uses over 300
- Do you know who Verisign is?
- Do you know who Comodo is?
- Why should we trust these companies?

Can we Trust a Certificate Authority?

- How do the CAs verify the identity of their applicants?
  - Is there a separation between the Certificate Authority CA, issuing the certificates, and the Registration Authority RA?
- If the registration is not done correctly
  - Maybe a site “amaz0n.com” may be able to get a certificate and spoof the “amazon.com” web site?
  - If they are able to get a valid certificate from these CAs, we might never notice

Attack on Comodo

- http://news.cnet.com/8301-31921_3-20046588-281.html
- Comodo is a New Jersey based company that issues SSL certificates
- Attacker obtained the username and password for a Comodo Trusted Partner in Southern Europe
- This login was used to acquire certificates fraudulently

SSL - Secure Socket Layer

Now known as

TLS – Transport Layer Security

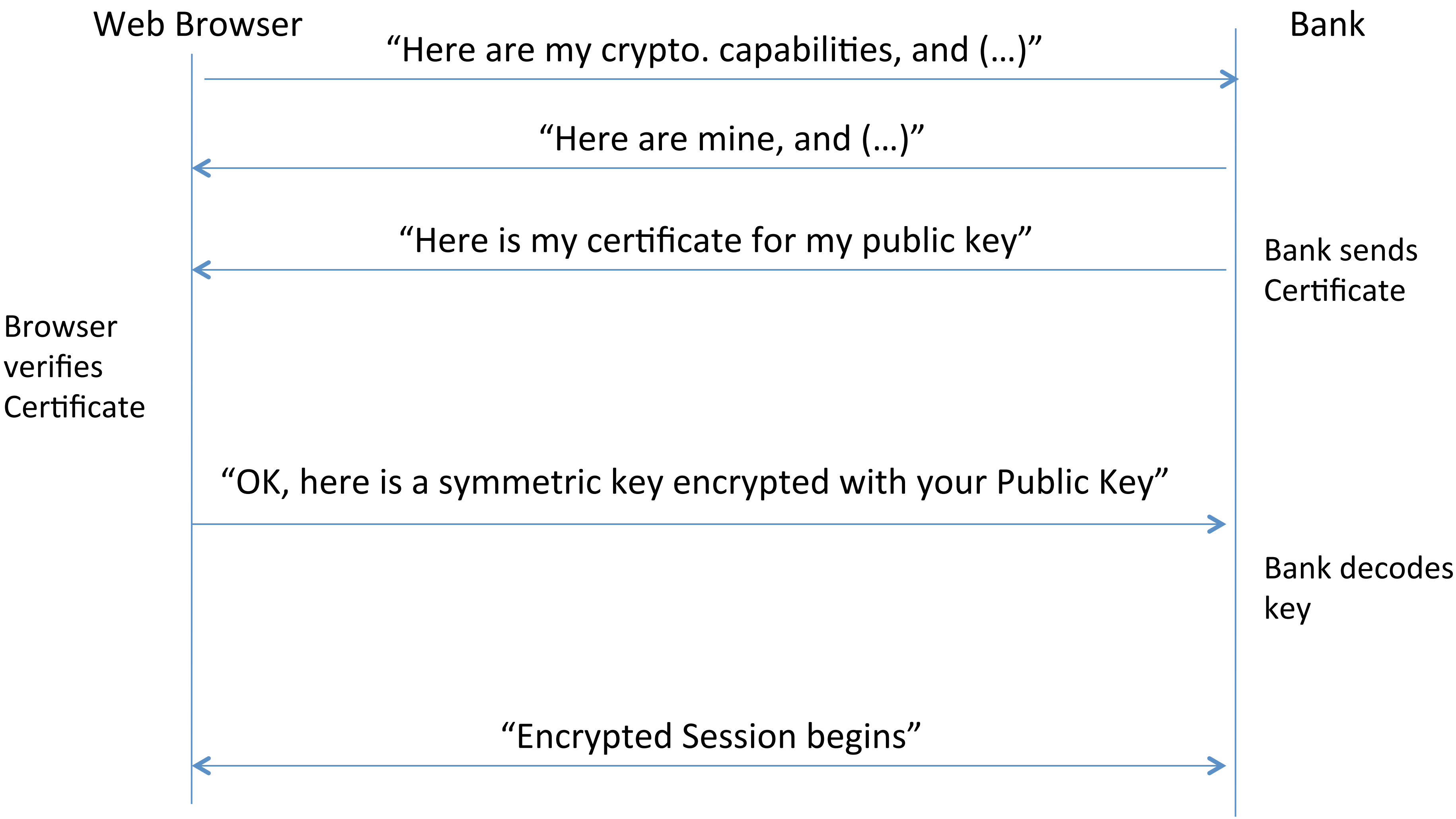
SSL, Secure Socket Layer

- Is a protocol designed to provide a data encryption and authentication between a web client and a web server.
- Provides secure communication for the Transport Layer (via TCP/IP)
  - Layer 3: IP (Internet Protocol), contains addressing and (limited) control information
  - Layer 4: TCP (Transport Control Protocol), provides end-to-end connection-oriented packet delivery
- Secure communication should not disturb TCP/IP networking
- Should be “transparent” to applications, allows them to communicate with servers in a secure fashion
- SSL has become Transport Layer Security Protocol TLS

SSL Secure Socket Layer

- SSL can be viewed as a security layer that sits between the application layer and the transport layer
  - The client hands over data to SSL (e.g. An HTTP message for a web server), SSL then encrypts this data and writes it to a TCP socket
  - The server receives this encrypted data via its TCP socket, SSL takes this data, decrypts it and directs this data to the server for processing
- SSL uses symmetric key cryptography for encryption and decryption of data that is transferred
- To Do:
  - Verify that server is trustworthy (certificates)
  - Exchange a symmetric key between server and client

SSL Handshake
(One-way, RSA abstracted)



The SSL Handshake Protocol

- In order to establish an SSL session, the web browser and web server go through the SSL Handshake Protocol
- Two tasks
  - Authenticate the Server and negotiate Cipher Suite
    - Browser and server negotiate the Cipher Suite to be determine the key distribution method and encryption algorithm for communication
    - Server sends certificate to web browser
  - Generate a shared symmetric key
    - Browser uses the server's public key contained in certificate to send a randomly generated symmetric key to the server
- SSL uses RSA Public Key cryptography for (a) testing the certificate and (b) for exchanging a shared symmetric key between server and browser

Cipher Suite Negotiation

- Client and Server choose a cipher suite (SSL 3.0 defines 31 cipher suites)
- A Cipher Suite is defined by the following components
  - Key exchange Method
    - SSL 2.0 supports only RSA key exchange
    - SSL 3.0 supports:
      - RSA key exchange when certificates are used
      - Diffie-Hellman key exchange when there has been no prior communication between client and server
  - Cipher for Data Transfer
    - SSL uses symmetric key cryptography, various choices of encryption algorithms:
      - DES, 3DES, IDEA, etc.
  - Method for creating the Message Authentication Code (MAC) / Message Digest Function choice
    - No Digest
    - MD5
    - Secure Hash Algorithm (SHA-1)

SSL Server Authentication

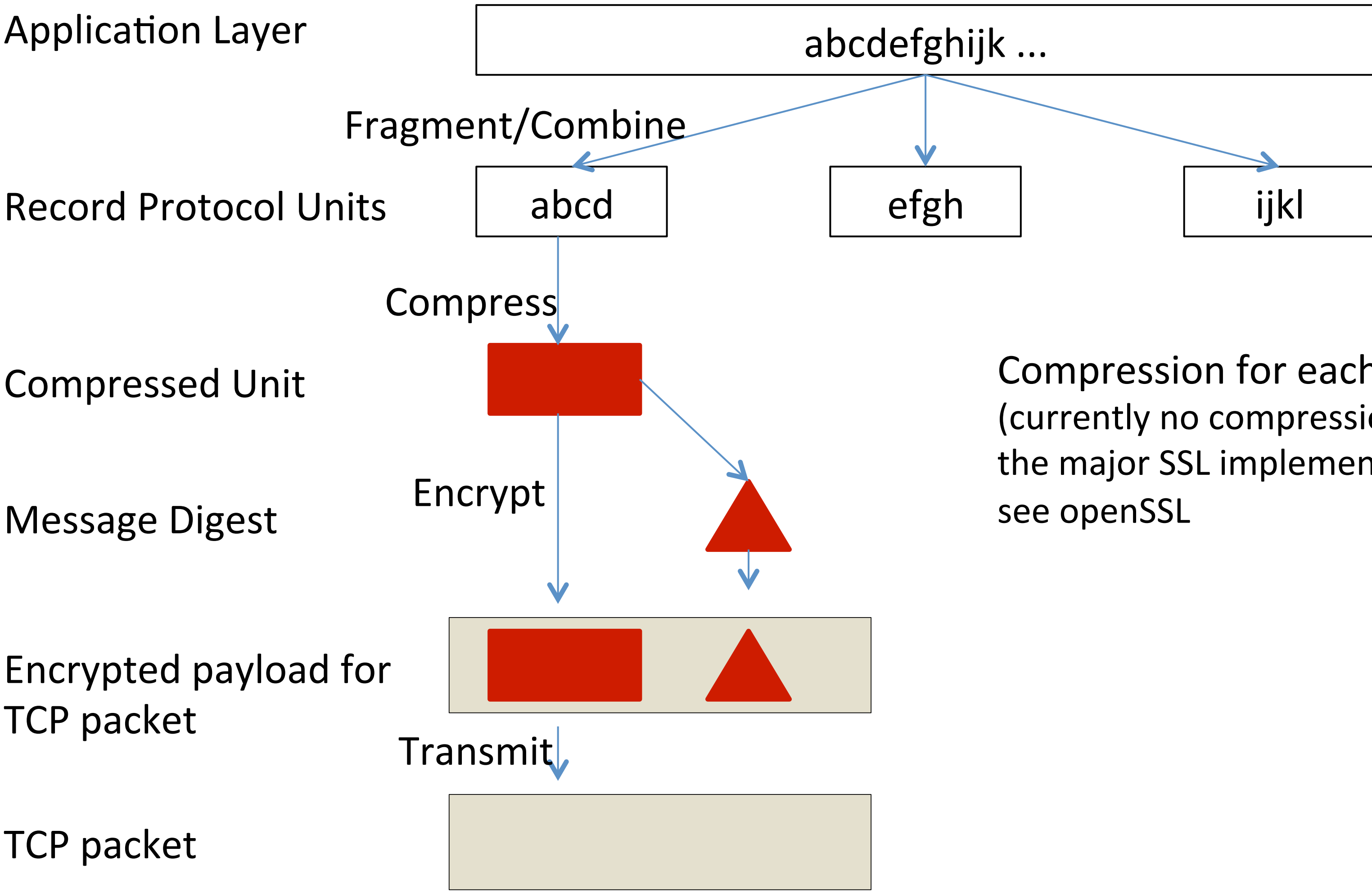
- SSL Server Authentication
  - SSL-enabled browsers maintain a list of the Public Keys of trusted CA's (Certificate Authorities)
  - When the web browser contacts the web server, the server submits a certificate, containing the server's public key (certificate is signed by one of the CA's known by the web browser)

SSL Handshake Phases

1. Browser sends SSL version and cryptographic preferences (e.g. RSA for key exchange, DES for communication) to server
2. Server sends SSL version, cryptographic preferences, and certificate; certificate (certified by some CA) contains server's encrypted RSA Public Key
3. Browser lookup of certificates CA; if CA is in the browsers list of trusted CA's, the CA's Public Key is used to (a) validate the certificate and (b) decrypt the server's RSA Public Key
4. Browser generates a symmetric secret (private) key (called the "session key"), encodes it with the server's RSA Public Key and sends it to the server
5. Server uses its RSA private key to decrypt the received session key for this SSL communication
6. Handshake finished, browser and server start communicating using the secret session key

1-way (server) authentication and RSA key exchange.

Data Transfer
SSL Record Protocol



Compression for each unit (currently no compression in most of the major SSL implementations), but see openSSL

SSL Communication by Web Browser

- What does the padlock symbol mean shown by a web browser?
  - Both browser and web server will use SSL for communication
  - All data transmitted in both directions will be encrypted
  - The browser recognizes the authority of the Certificate Authority that issued and signed the web server's certificate and holds the CA's public key
  - The web domain of the web server has been registered with the CA and is indeed a legitimately registered web domain

The SSL Process

- Phase 1: Handshake using SSL Handshake Protocol
  - To agree on secret keys and algorithms for phase 2
    - Negotiate the Cipher Suite to be used for data transfer
    - Establish / share a session key between client and server
  - To authenticate server
  - [To authenticate client (only for 2-way authentication mode)]
- SSL uses public key cryptography for the handshake
- SSL uses symmetric key cryptography for:
  - Encryption and decryption of data that is transferred

SSL Handshake Protocol (RSA)