Web Security

Dr. Arjan Durresi
Louisiana State University
Baton Rouge, LA 70810
Durresi@csc.lsu.Edu

These slides are available at:

http://www.csc.lsu.edu/~durresi/csc4601-04/

How the Web Works: Other Elements

- ❑ Hyper-text markup language (html).
- ❑ Other application specific document.
  - ○ E.G., MIME, graphics, video/audio, postscript, Java applets, etc.
- ❑ Browsers.
  - ○ Display html documents and embedded graphics.
  - ○ Run Java program.
  - ○ Start helper applications.
  - ○ ...

Overview

- ❑ How web works
- ❑ Threats
- ❑ SSL, Architecture, protocols
- ❑ TLS
- ❑ SET

Web Security

- ❑ http://www.w3.org/Security/Faq
- ❑ Web now widely used by business, government, individuals
- ❑ But Internet & Web are vulnerable
- ❑ Have a variety of threats
  - ○ integrity
  - ○ Confidentiality - Revealing private information on server
  - ○ denial of service
  - ○ Authentication - Execute unauthorized programs
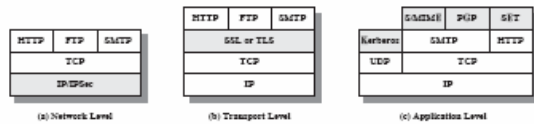- ❑ Need added security mechanisms

---

How the Web Works - HTTP

- ❑ Hypertext transfer protocol (http).
- ❑ Clients request "documents" (or scripts) through URL.
- ❑ Server response with "documents".
- ❑ Documents are not interpreted by http.
- ❑ Stateless protocol, request are independent.

Threats

	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 •Vulnerability 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

Location of Security

HTTP	FTP	SMTP
TCP		
IP/IPSec		

(a) Network Level

HTTP	FTP	SMTP
SSL or TLS		
TCP		
IP		

(b) Transport Level

S/MIME	PGP	SET
Kerberos	SMTP	HTTP
UDP	TCP	
IP		

(c) Application Level

SSL Architecture

❑ **SSL session**
  ○ an association between client & server
  ○ created by the Handshake Protocol
  ○ define a set of cryptographic parameters
  ○ may be shared by multiple SSL connections
  ○ Once established – operating state
  ○ During Handshake Protocol – pending read, write states
❑ **SSL connection**
 ○ a transient, peer-to-peer, communications link
  ○ associated with 1 SSL session

SSL (Secure Socket Layer)

❑ Transport layer security service
❑ originally developed by Netscape
❑ version 3 designed with public input
❑ subsequently became Internet standard known as TLS (Transport Layer Security)
❑ uses TCP to provide a reliable end-to-end service
❑ SSL has two layers of protocols
  ○ SSL record protocol provides basic security services
  ○ 3 higher-layer protocols:
    ❑ Handshake, change cipher spec, alert

Session and Connection

❑ Session parameters:
  ○ ID, peer certificate, compression method, cipher spec, master secret, is resumable.
❑ Connection parameters:
  ○ Server and client random, server write MAC secret, client write MAC secret, server write key, client write key, IV, sequence number.

SSL Architecture

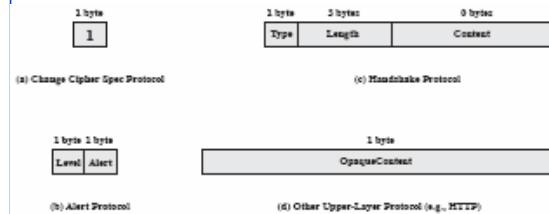| SSL Handshake Protocol	SSL Change Cipher Spec Protocol	SSL Alert Protocol	HTTP
SSL Record Protocol			
TCP			
IP			
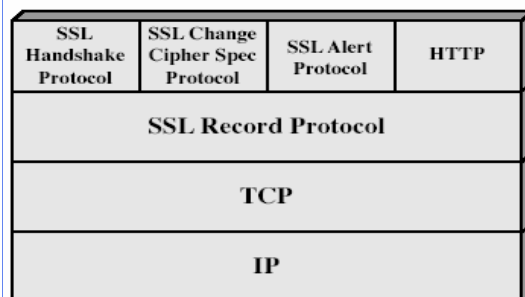
SSL Record Protocol

❑ **Confidentiality**
  ○ using symmetric encryption with a shared secret key defined by Handshake Protocol
  ○ IDEA, RC2-40, DES-40, DES, 3DES, Fortezza, RC4-40, RC4-128
  ○ message is compressed before encryption
❑ **Message integrity**
  ○ using a MAC with shared secret key
 ○ similar to HMAC but with different padding

SSL Record Protocol

- ❑ Layered protocol:
  - ○ Fragment application data into blocks
  - ○ Compress data
  - ○ Apply message authentication code (MAC) = h($m|s$) for message m and secret s
  - ○ Encrypt with client (cw) or server (sw) write key
  - ○ Transmit over TCP
- ❑ Specify content type for higher protocols
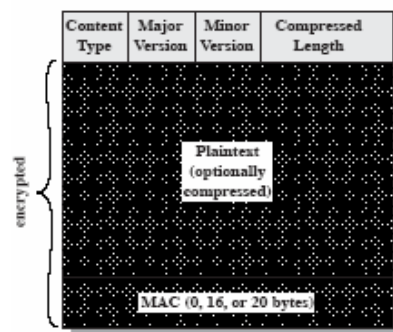

SSL Record Protocol Payload

SSL Record Protocol Operation
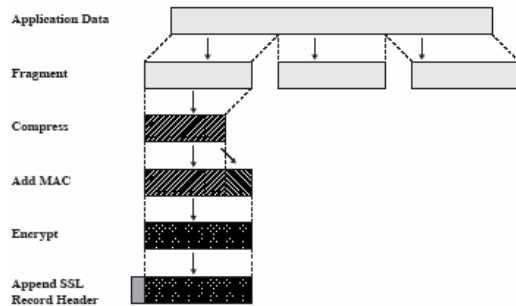
SSL Change Cipher Spec Protocol

- ❑ One of 3 SSL specific protocols which use the SSL Record protocol
- ❑ A single message
- ❑ Causes pending state to become current
- ❑ Hence updating the cipher suite in use

SSL Record Format

SSL Alert Protocol

- ❑ Conveys SSL-related alerts to peer entity
- ❑ Severity
  - ○ warning or fatal
- ❑ Specific alert
  - ○ unexpected message, bad record mac, decompression failure, handshake failure, illegal parameter
  - ○ close notify, no certificate, bad certificate, unsupported certificate, certificate revoked, certificate expired, certificate unknown
- ❑ Compressed & encrypted like all SSL data

SSL Handshake Protocol

- ❑ Allows server & client to:
  - ○ authenticate each other
  - ○ to negotiate encryption & MAC algorithms
  - ○ to negotiate cryptographic keys to be used
- ❑ It is used before any application data is transmitted
- ❑ Comprises a series of messages in phases
  - ○ Establish Security Capabilities
  - ○ Server Authentication and Key Exchange
  - ○ Client Authentication and Key Exchange
  - ○ Finish

SSL Handshake Protocol Message Types

Message Type	Parameters
hello_request	null
client_hello	version, random, session id, cipher suite, compression method
server_hello	version, random, session id, cipher suite, compression method
certificate	chain of X.509v3 certificates
server_key_exchange	parameters, signature
certificate_request	type, authorities
server_done	null
certificate_verify	signature
client_key_exchange	parameters, signature
finished	hash value

Establish Security Capabilities

- ❑ Used to initiate a logical connection and to establish the security capabilities associated with it
- ❑ Initiated by client – **client_hello_message,** parameters:
  - ○ Version – the highest SSL version understood by the client
  - ○ Random – 32-bit timestamp and 28 bytes random, serve as nonce
  - ○ Session ID – a variable length Session ID. A nonzero value indicates the client wants to update existing connection. Zero – new one
  - ○ CipherSuite – list of combinations of cryptographic algorithms supporte by client in decreasing order of preference
  - ○ Compression method – list supported methods
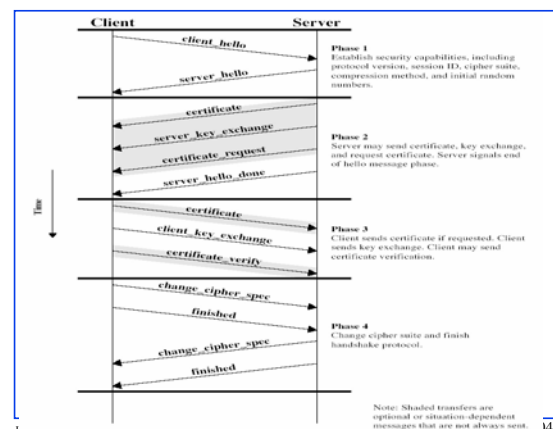
Establish Security Capabilities

- ❑ The server responds with **server_hello** message with the same parameters as the client_hello
  - ○ Version
  - ○ Random - independent of client
  - ○ Session ID
  - ○ CipherSuite – the single cipher selected
  - ○ Compression – the method selected

Server Authentication and Key Exchange

- ❑ Server begins this phase by sending its certificates
- ❑ Next sender server sends a **server_key_exchange** message (not sent if a certificate with fixed DH was sent or when RSA will be used)
- ❑ Next a nonanonymous server (server not using DH) can request a certificate from the client
- ❑ Final message **server_done** message

Client Authentication and Key Exchange

- Client verifies the certificate of server and the parameters are acceptable
- If server has requested a certificate, the client sends it with a **certificate** message.
- Next **client_key_exchange_message**
- Finally the client sends a **certificate_verify** message

To provide explicit verification of a client certificate (following any client certificate with signing capability)

Finish

- Client send **change_cipher_spec**
- Then **finished message** using the new key

Cryptographic Computations

- Master secret creation
  - A pre-master-secret is exchanged first.
    - RSA, or Diffie-Hellman.
  - Both sides compute master secret based on pre-master-secret.
- Generation of cryptographic parameters.
  - Client/server write MAC secrets, client/server write keys, client/server write IV are generated from master secret.

Cryptographic Computations: Details (1)

- Client generates a 48-byte pre-master-secret s_p
- Master secret:
  - $s_m = \text{MD5}(s_p|\text{SHA}(\text{'A'}|\,s_p|r_c|r_s))\,|$
    $\text{MD5}(s_p|\text{SHA}(\text{'BB'}|\,s_p|r_c|r_s))\,|$
    $\text{MD5}(s_p|\text{SHA}(\text{'CCC'}|\,s_p|r_c|r_s))$
  - Where $r_{c,s}$: client, server random

Cryptographic Computations: Details (2)

- Session key: same as above, but use the master secret in place of s_p to generate byte stream to cut out:
  - Client, server MAC secret
  - Client, server write key
  - Client, server IV

TLS (Transport Layer Security)

- IETF standard RFC 2246 similar to SSLv3
- with minor differences
  - in record format version number
  - uses HMAC for MAC
  - a pseudo-random function expands secrets
  - has additional alert codes
  - some changes in supported ciphers
  - changes in certificate negotiations
  - changes in use of padding

TLS Function P_hash
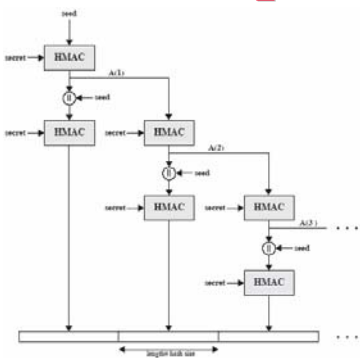
Key Features of SET

- ❑ Confidentiality of information using DES:
  - ○ Cardholder account and payment information is secured over the network
  - ○ Prevents merchant from learning the cardholder's credit card number, which is only provided to the issuing bank
- ❑ Integrity of data using RSA digital signatures, SHA-1 hash:
  - ○ Payment information: order information, personal data and payment instructions
- ❑ Cardholder account authentication using X.509v3 digital certificates with RSA signatures – merchant verify that a cardholder is a legitimate user of a valid credit card
- ❑ Merchant authentication – cardholder verify that a merchant has a relationship with a financial institution allowing it to accept payment cards

Secure Electronic Transactions (SET)

- ❑ Open encryption & security specification
- ❑ To protect Internet credit card transactions
- ❑ Developed in 1996 by Mastercard, Visa etc
- ❑ Not a payment system
- ❑ Rather a set of security protocols & formats
- ❑ SET provides:
  - ○ Secure communications amongst parties
  - ○ Trust from use of X.509v3 certificates
  - ○ Privacy by restricted info to those who need it

SET Components
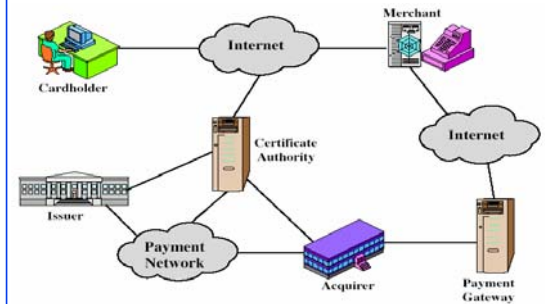
Business Requirements for SET

- ❑ Confidentiality of payment and ordering information - use encryption
- ❑ Integrity of all transmitted data – use digital signatures
- ❑ Authentication of cardholder – use digital signatures and certificates
- ❑ Authentication that a merchant can accept credit card transactions - use digital signatures and certificates
- ❑ Ensure the use of the best security practices and system design techniques
- ❑ Neither depends on transport security nor prevent their use (IPSec, SSL/TLS)
- ❑ Facilitate interoperability among software and network protocols – independent of hardware, software platforms

SET Transaction

1. Customer opens account
2. Customer receives a certificate – X.509v3 digital certificate signed by the bank. The certificate verifies the customer's RSA public key and its expiration date.
3. Merchants have their own certificates – one for the public key to sign messages and the other for key exchange
4. Customer places an order
5. Merchant is verified
6. Order and payment are sent – the payment information cannot be read by the merchant
7. Merchant requests payment authorization from payment gateway
8. Merchant confirms order
9. Merchant provides goods or service
10. Merchant requests payment

Dual Signature

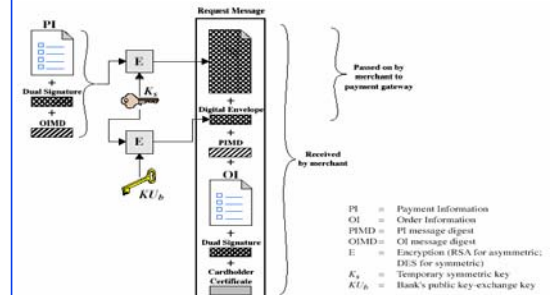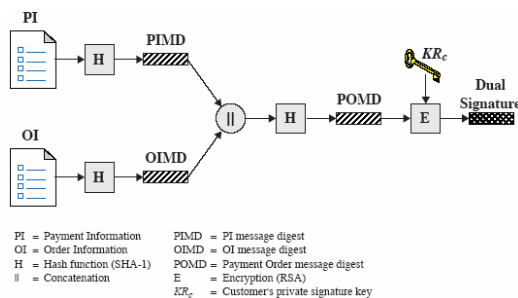- Customer creates dual messages
  - order information (OI) for merchant
  - payment information (PI) for bank
- Neither party needs details of other
- But **must** know they are linked
- Use a dual signature for this
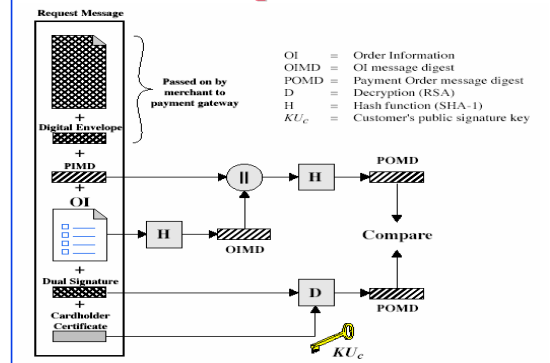 - signed concatenated hashes of OI & PI

Purchase Request – Customer

PI = Payment Information
OI = Order Information
PIMD = PI message digest
OIMD = OI message digest
E = Encryption (RSA for asymmetric; DES for symmetric)
K_s = Temporary symmetric key
KU_b = Bank's public key-exchange key

Construction Dual Signature

PI = Payment Information   PIMD = PI message digest
OI = Order Information   OIMD = OI message digest
H = Hash function (SHA-1)   POMD = Payment Order message digest
|| = Concatenation   E = Encryption (RSA)
   KR_c = Customer's private signature key

Purchase Request – Merchant

OI = Order Information
OIMD = OI message digest
POMD = Payment Order message digest
D = Decryption (RSA)
H = Hash function (SHA-1)
KU_c = Customer's public signature key

Purchase Request – Merchant

1. Verifies cardholder certificates using CA sigs
2. Verifies dual signature using customer's public signature key to ensure order has not been tampered with in transit & that it was signed using cardholder's private signature key
   H(PIMD||H(OI)) and D_{KUC}[DS] if they are equal
3. Processes order and forwards the payment information to the payment gateway for authorization (described later)
4. Sends a purchase response to cardholder

Payment Gateway Authorization

1. Verifies all certificates
2. Decrypts digital envelope of authorization block to obtain symmetric key & then decrypts authorization block
3. Verifies merchant's signature on authorization block
   $H(H(PI)||OIMD$ and $D_{KUC}[DS]$ – if they are equal
4. Decrypts digital envelope of payment block to obtain symmetric key & then decrypts payment block
5. Verifies dual signature on payment block
6. Verifies that transaction ID received from merchant matches that in PI received (indirectly) from customer
7. Requests & receives an authorization from issuer
8. Sends authorization response back to merchant

Payment Capture

- ❑ Merchant sends payment gateway a payment capture request
- ❑ Gateway checks request
- ❑ Then causes funds to be transferred to merchants account
- ❑ Notifies merchant using capture response

Summary

- ❑ Have considered:
  - ○ need for web security
  - ○ SSL/TLS transport layer security protocols
  - ○ SET secure credit card payment protocols