Learning Network Security with SSL The OpenSSL Way

Shalendra Chhabra

schhabra@cs.ucr.edu.

Computer Science and Enginering

University of California, Riverside

http://www.cs.ucr.edu/šchhabra

Slides Available from http://www.cs.ucr.edu/šchhabra/scale05.pdf

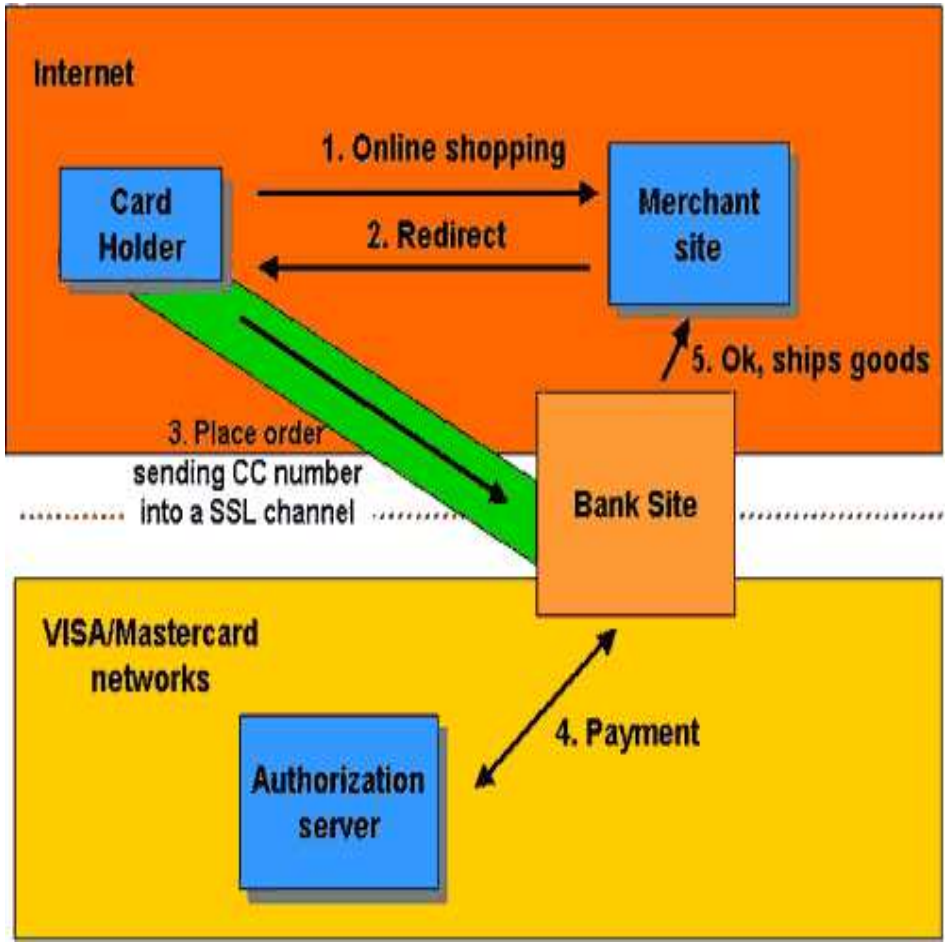
Cryptography and Its Goals

- Confidentiality (secrecy)
- Integrity (anti-tampering)
- Authentication
- Non-repudiation
- Snooping (passive eavesdropping)
- Tampering
- Spoofing
- Hijacking
- Capture-replay

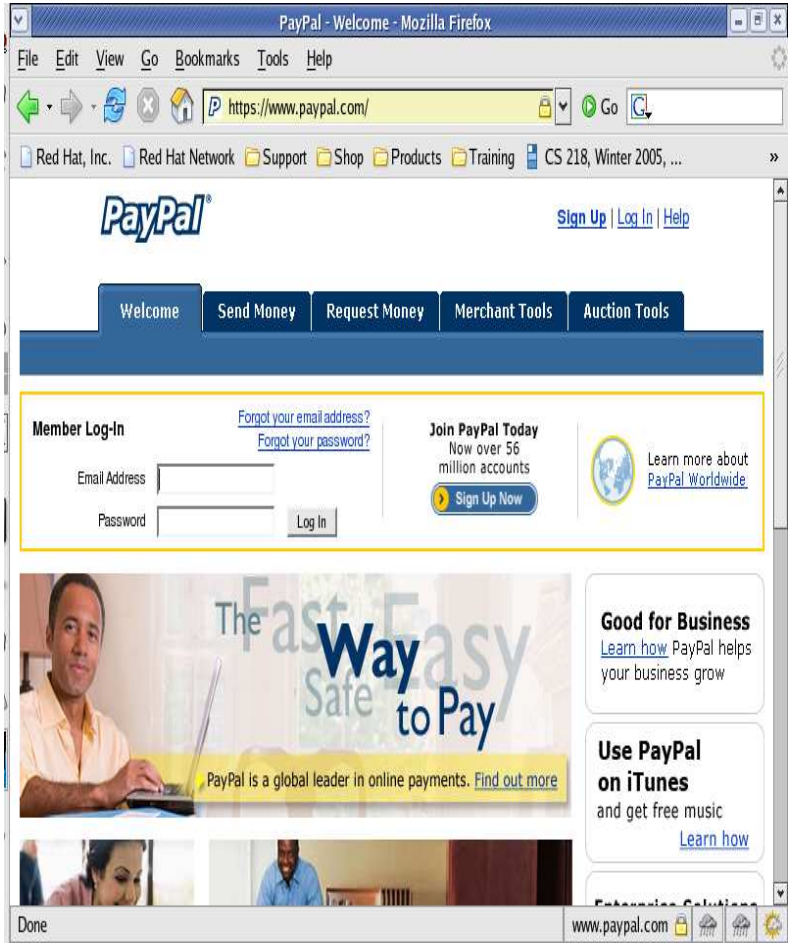
Crytographic Algorithms

- Symmetric Key Encryption: `DES, 3DES, AES, IDEA, BLOWFISH: Faster`
- Public Key Encryption: `Diffie Hellman (1976, New Directions in Cryptography), RSA, DSA: Slower`
- Cryptographic Hash Functions : `MD2, MD5 (16 byte), SHA (20 bytes): One Way, Fixed Output, Collision Free`
- HMAC: `Message Authentication Codes based on Hash Functions are called HMAC`
- Digital Signatures: `Hash signed with the Private Key`

A Glimpse of How Transactions in ECommerce Work (Generally)



http and https - Watch this "Lock"!



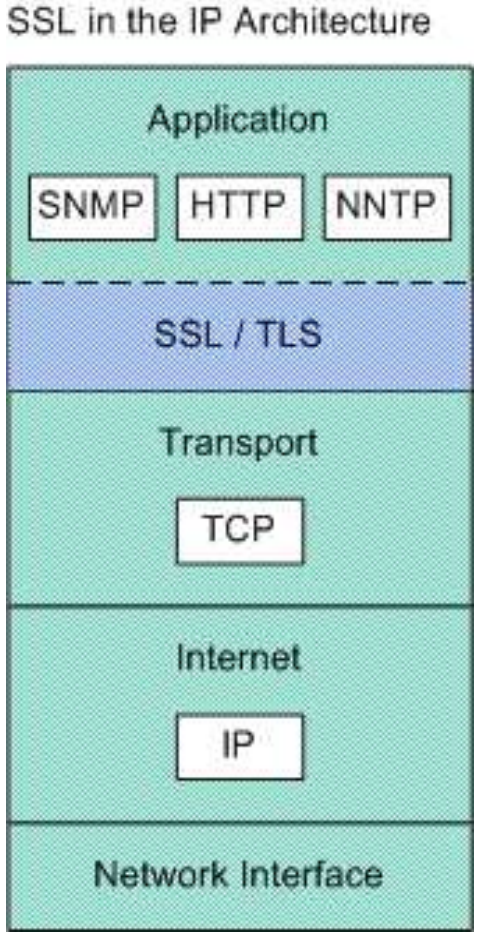
Secure Socket Layer and Transport Layer Security

- History and Versions: SSL/TLS
  Developed by Netscape, 1996 and then served as a basis for TLS, an IETF standard protocol

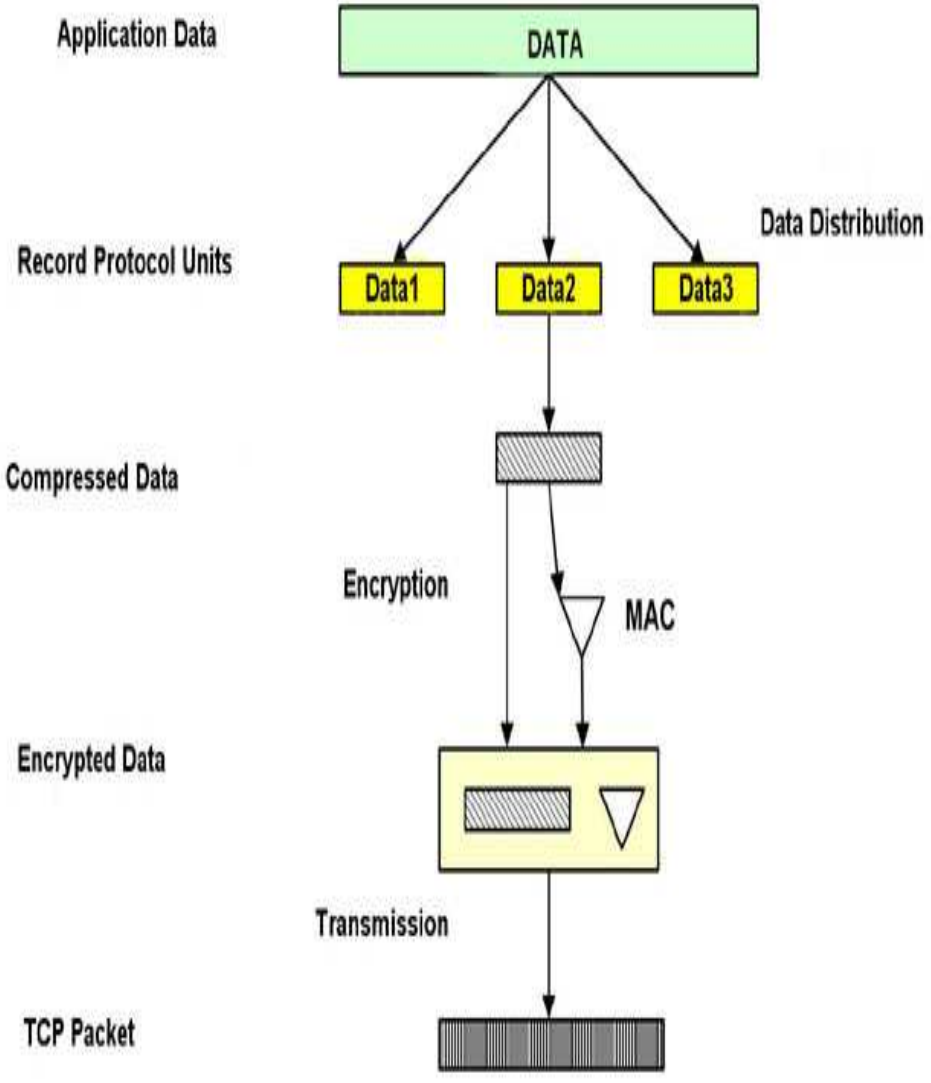
- TLS 1.1 - RFC 2246 12/2004, Expires 06/2005

- SSL v 3.0 - Internet Draft Expires 9/96
- “https”: HTTP Over TLS: RFC 2818
- Early Weak Keys
  Earlier a restriction of 40-bit keyspace small enough to be broken by Brute Force Search. Modern implementations use 128-bit (or longer) keys for symmetric key ciphers.

Protocol Stack with TLS

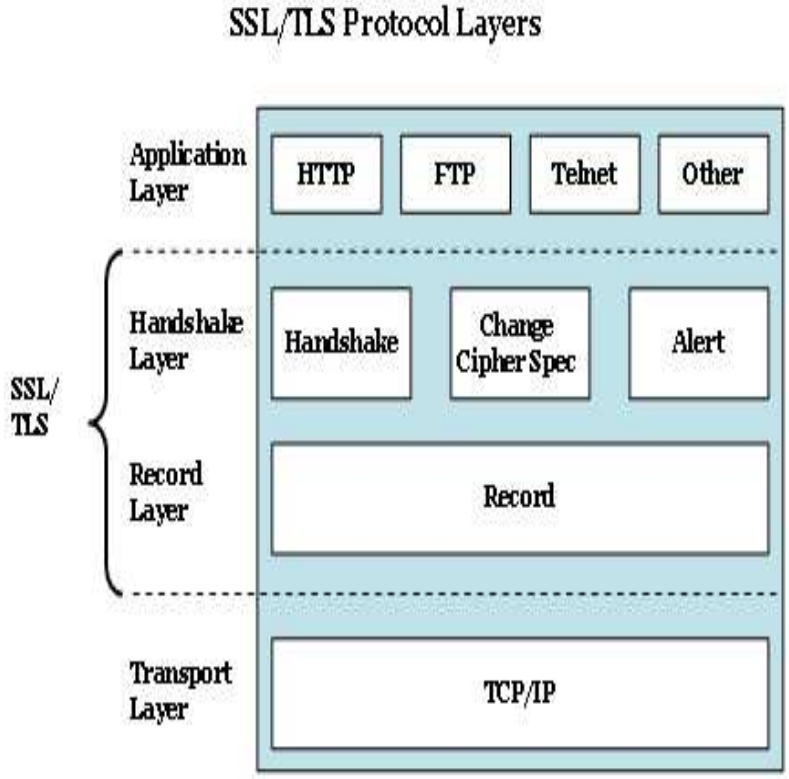


Flow of Application Data Through The Stack

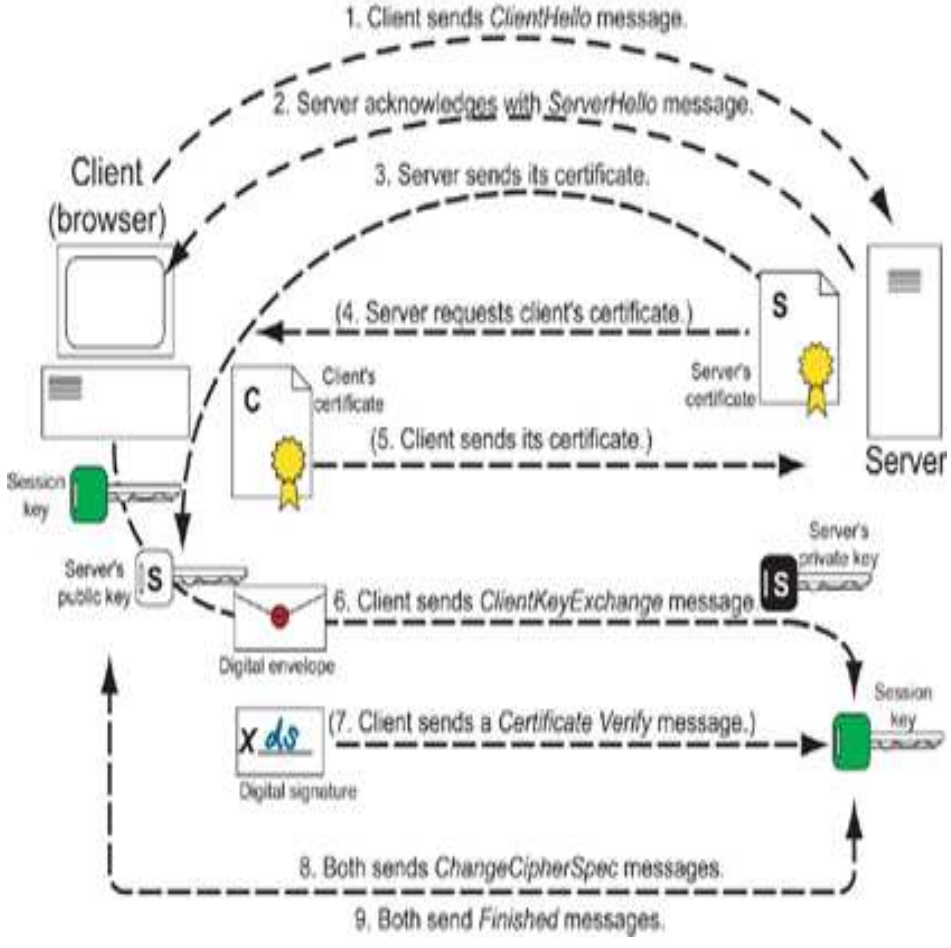


Architecture of TLS v 1.1

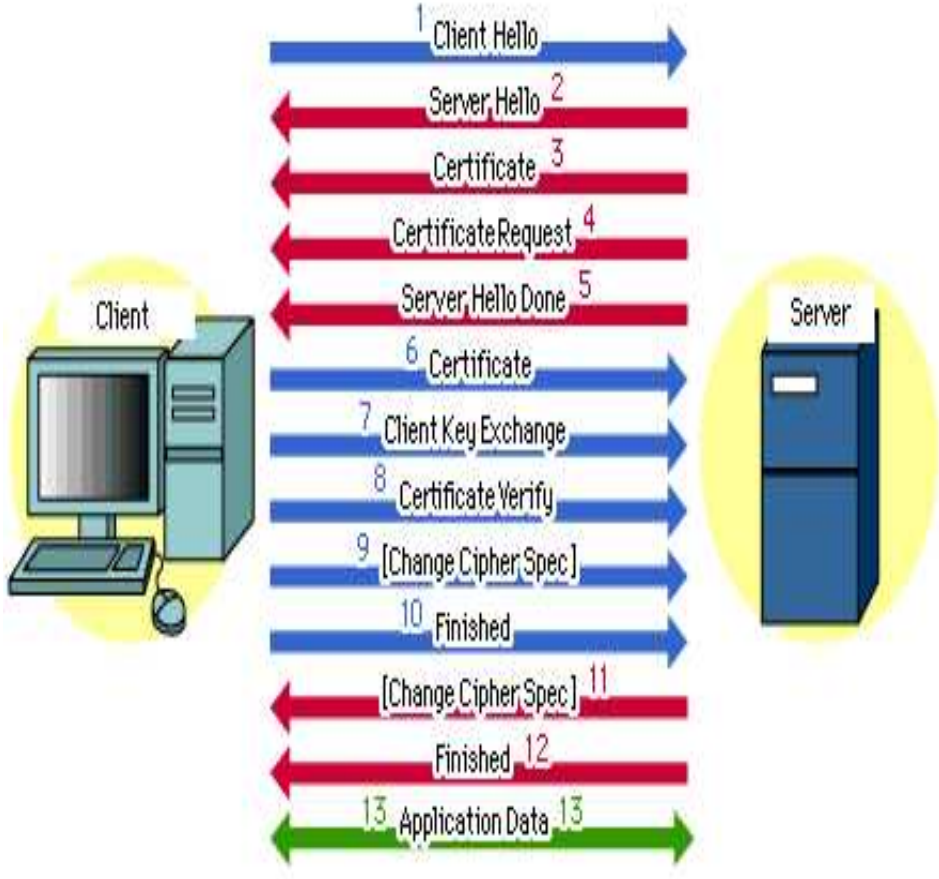
- TLS Handshake Protocol
- TLS Record Protocol
- TLS Change Cipher Spec Protocol
- TLS Alert Protocol



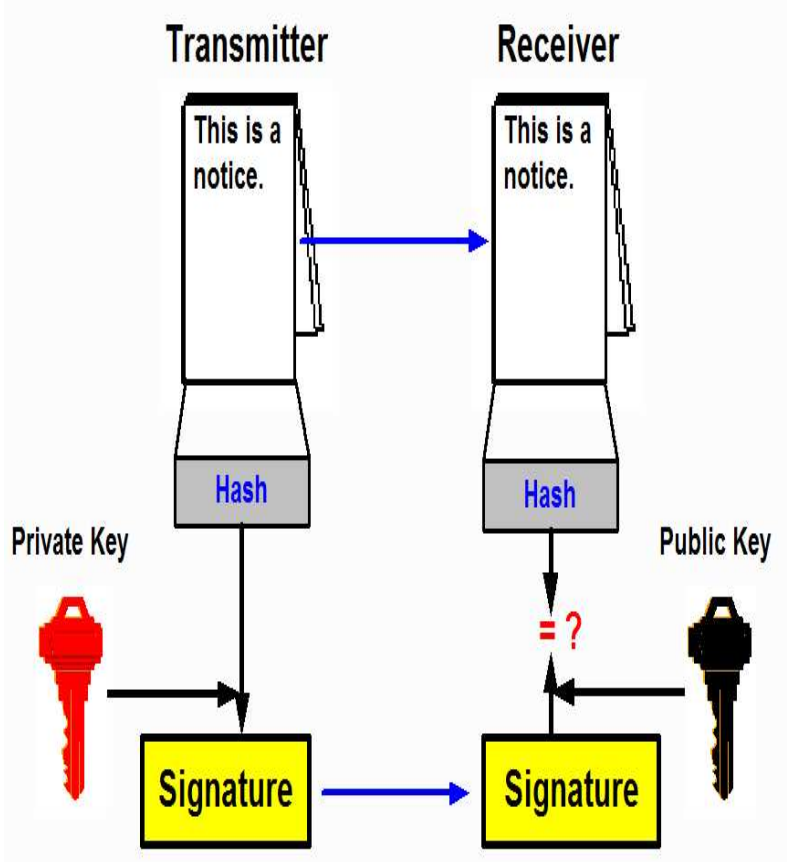
Message Flow for a Full Handshake



Message Flow for a Full Handshake



Digital Signature

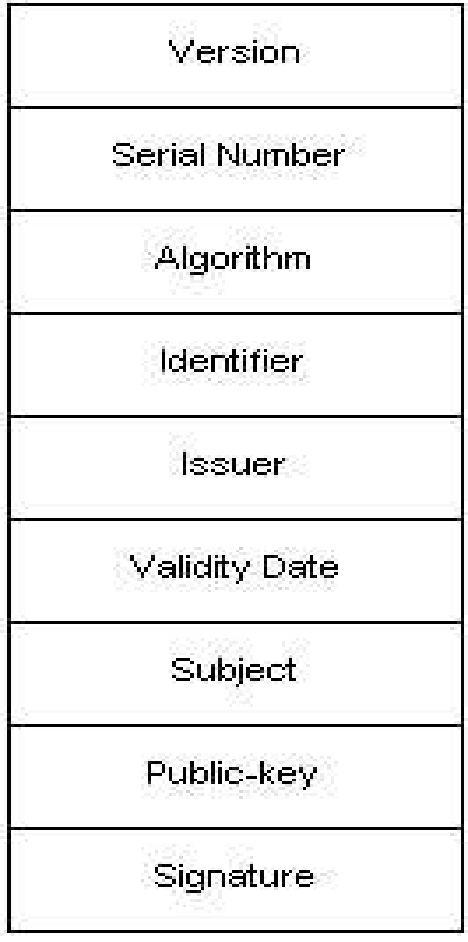


Format of ClientHello and ServerHello

struct { ProtocolVersion client_version;
Random random;
SessionID session_id;
CipherSuite cipher_suites<2..$2^{16} - 1$>;
CompressionMethod compression_methods<1..$2^{8} - 1$>;
} ClientHello;

struct { ProtocolVersion server_version;
Random random;
SessionID session_id;
CipherSuite cipher_suite;
CompressionMethod compression_method;
} ServerHello;

Format of an X509 certificate



OpenSSL

- Go to www.openssl.org
- Click on Source on the left
- Download the latest version of openssl:Oct 25 13:44:48 2004 openssl-0.9.7e.tar.gz
- As su install $tar -zxvf /usr/local/openssl-0.9.7e.tar.gz
- $cd /usr/local/openssl-0.9.7e
- $./configure
- $make
- $make install
- $openssl version

Command Line Interface

- Configuration Files
  `ca,req, x509`
- Format of the OpenSSL Configuration File:
  Organized in Sections and Each section contains a set of keys and each key has an associated value
- `cat /usr/share/ssl/openssl.cnf`
- `openssl dgst -sha1 file.txt`
- `openssl sha1 -out digst.txt file.txt`
- `openssl enc -des3 -salt -in file.txt -out ciphertext.bin`
- `openssl bf-cfb -salt -in file.txt -out ciphertext.bin -pass env:  HOME`
- `openssl base64 -in ciphertext.bin -out base64.txt`
- `openssl dhparam -out dhparam.pem -2 1024`
- `openssl dhparam -in dhparam.pem -noout -C`
- `openssl dsaparam -out dsaparam.pem 1024`
- `openssl genrsa -out rsaprivatekey.pem -passout pass:test -des3 1024`

Creating a Self-Signed Root Certificate

- `openssl req -x509 -newkey rsa -out cacert.pem -outform PEM`
- `cat cacert.pem`
- `openssl x509 -in cacert.pem -text -noout`

Writing some Client Server Programs using OpenSSL in C

- $cd /usr/local/openssl-0.9.7e/demos/ssl
- $ls
- $serv.cpp cli.cpp
- Compile Server i.e. serv.cpp
- $g++ -c serv.cpp
  or
- $g++ -I/usr/local/include -c serv.cpp
- $g++ serv.o -lssl -o servertest
  or
- $g++ serv.o /usr/local/lib/libssl.a /usr/local/lib/libcrypto.a -o servertest
- Compile Client i.e. cli.cpp in the same manner
- Wait we need Server Certificate and Private Key
- $openssl req -x509 -newkey rsa -out cert.pem -outform PEM
- $ cat cert.pem privkey.pem » foo-cert.pem
- $./servertest
- $./clientest
- $ Run SSLDump

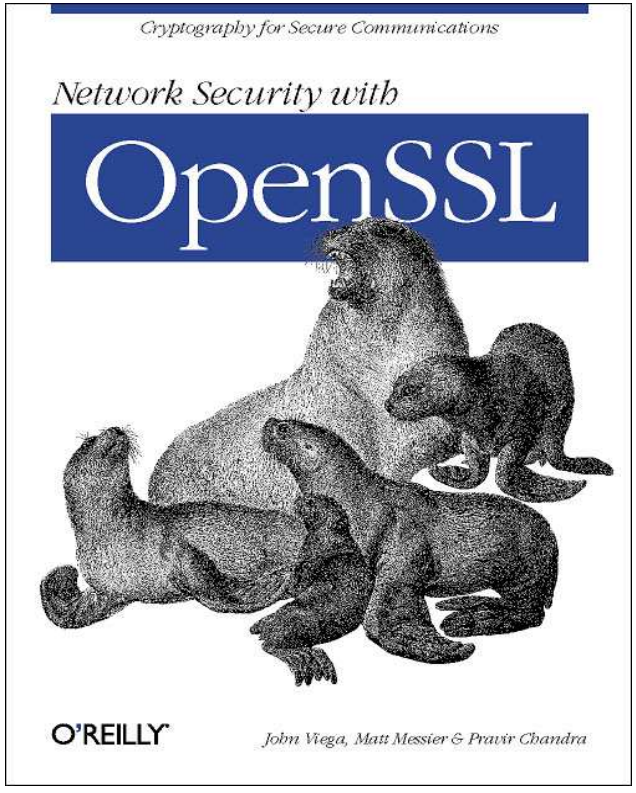
SSLDump and SSLSniffer

```
$openssl s_client -connect www.paypal.com:443
$ssldump -i eth0 port 443
New TCP connection 1: 192.168.1.103(32952) <-> 206.65.183.42(443)
1 1 0.0834 (0.0834) C>S Handshake
ClientHello
Version 3.1
resume [32]=
23 22 00 00 b9 8d c0 23 0e fe 0d cb b4 c8 89 e9
8e 8c 14 da e4 d5 2d 0a 56 ed c5 61 11 48 4a 84
cipher suites
Unknown value 0x39
Unknown value 0x38
Unknown value 0x35
Unknown value 0x33
Unknown value 0x32
TLS_RSA_WITH_RC4_128_MD5
TLS_RSA_WITH_RC4_128_SHA
.
2 3 0.1954 (0.0000) S>C ChangeCipherSpec
2 4 0.1954 (0.0000) S>C Handshake
2 5 0.1984 (0.0029) C>S ChangeCipherSpec
2 6 0.1984 (0.0000) C>S Handshake
2 7 0.1984 (0.0000) C>S application_data
```

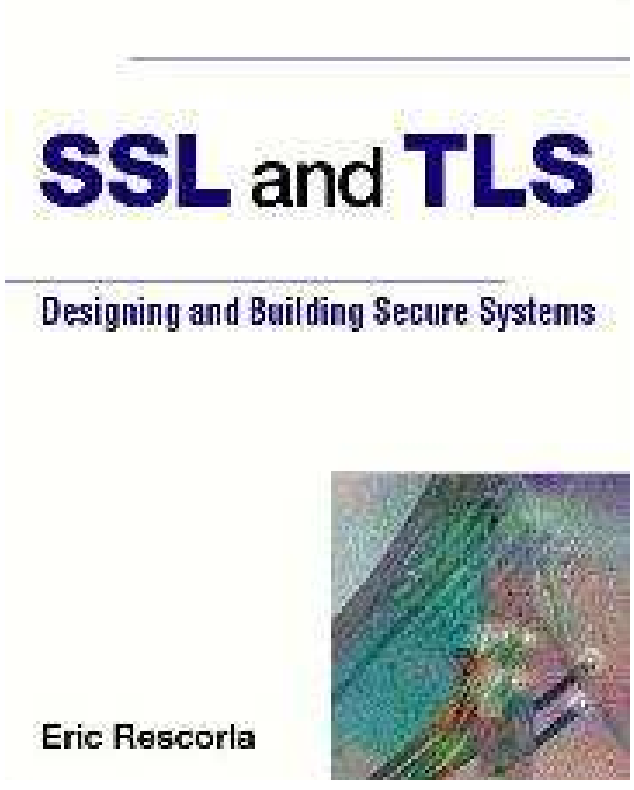
Security Analysis of SSL Protocol

- Version Rollback Attack
- Attacks on Handshack Protocol
- If using Public Key Crytography (Diffie Hellman) Man in the Middle Attack
- Analysis of the SSL 3.0 Protocol, D. Wagner and B. Schneier
  The Second USENIX Workshop on Electronic Commerce Proceedings, USENIX Press, November 1996, pp. 29-40.

"Network Security with OpenSSL"



"SSL and TLS: Designing and Building Secure Systems"



References

1. SSL 3.0 Specification: http://wp.netscape.com/eng/ssl3/
2. OpenSSL: http://www.openssl.org
3. SSLDump: http://www.rtfm.com/ssldump/
4. Network Security With OpenSSL by John Viega, Matt Messier and Pravir Chandra
5. Slides available from http://www.cs.ucr.edu/s̃chhabra/scale05.pdf