

Lecture Notes – Computer Security

Author: Dr. Aaron Striegel
University of Notre Dame striegel@nd.edu

Topic: Encryption – Certificates

See earlier lecture for starting point

X.509 Example

keytool Solaris

```
[striegel@Blaster working]$ keytool -genkey -dname "cn=Aaron Striegel, ou=CSE
o=Notre Dame, c=US" -alias Striegel -keypass ShhKeepSecret! working/mykeystore
-storepass K#YSTOREpwd -validity 180
```

```
[striegel@Blaster striegel]$ keytool -list -keystore working/mykeystore
Enter keystore password: K#YSTOREpwd
```

```
Keystore type: jks
Keystore provider: SUN
```

Your keystore contains 1 entry

```
striegel, Sep 13, 2004, keyEntry,
Certificate fingerprint (MD5): FE:9D:D6:E2:87:5B:A0:7B:02:44:8A:27:A1:E3:ED:8D
[striegel@Blaster striegel]$
```

```
[striegel@Blaster striegel]$ keytool -export -alias Striegel -rfc -keystore
working/mykeyst>
```

```
Enter keystore password: K#YSTOREpwd
-----BEGIN CERTIFICATE-----
MIICuDCCAnUCBEFFsTMwCwYHkoZIZjgEAwUAMEEzCzAJBgNVBAYTA1VTMRkwFwYDVoQLEx
bz1Ob3RyZSBYXW1lMRcwFQYDVoQDEw5BYXJvbiBTdHJpZWdlbDAAeFw0wNDA5MTMxNDM5
NTAzMtIxNDM5NDdAMEEzCzAJBgNVBAYTA1VTMRkwFwYDVoQLExBDU0UgZz1Ob3RyZSBY
XW1lMRcwFQYDVoQDEw5BYXJvbiBTdHJpZWdlbDCCAbGwggEsBgqhkhjOOAQBMIIIBHwKB
gQD9f1OBHXUSKVLfSpwu7OTn9hG3UjzvrRADDHj+AtlEmaUVdQCJR+1k9jVj6v8X1ujD2y5t
VbNeB04AdNG/yZmC3a5lQpaSfn+gEexAiwk+7qdf+t8Yb+DtX58aophUPBPuD9tPFHsMCN
VQTWhaRMvZ1864rYdcq7/IiAxmd0UgBxwIVAjdgUI8VIwvMSPK5gqLrhAvwWBz1AoGBAPfhoIX
Wmz3ey7yrXDa4V715lK+7+jrqqv1XTAs9B4JnUVlXjrrUWU/mcQcQgYC0SRzXI+hMKBYTt88
JMozIpuE8FngLVHyNKOCjrh4rs6Z1k6jfwv6ITVi8ftiegEk08yk8b6oUZCJqIPf4Vrlnwa
Si2ZegHtVJWQBTDv+z0kqA4GFAAKBgQDbnAZRbgsRbR/AjxL1GCZ8G+NewdEngRgnerd1
V0luz4cNj+R1NK9C5BBjyp+43uojvtO37msi4hVcLJtaWFSRDvqJAoCEPhrqc9B1LRz
IW+SOjqfoOfPFJH+vKfpZhuRu2BwWFenqQXMampfi2unD+gP7o+3+fbEPWU02HYBLOj
ALBgcqhkhjOOAQDBQADMAAwLQIVAlnoYN5mYxwJzUiHlyd5Zic5Tw8ZAhR2NO3kK9iCq
qOyrPQvAH6wMrRoYQ==
-----END CERTIFICATE-----
```

SSL – Secure Socket Layer

Overview – Networking

Layer 7	App	Web browser, e-mail client
Layer 4	Transport	TCP or UDP
Layer 3	Network	IP
Layer 2/1	Data, Physical	802.11, 802.3 (Ethernet)

SSLv3 (currently used)

TLS (Transport Layer Security)

Connection
Session

Mechanisms used to transport data in a SSL session
Association between two peers

What do you need for a session?

Session ID for the session

Peer's X.509v3 certificate

Compression method

Cipher spec for message and message authentication code (MAC)

Better name is MIC – Message Integrity Check

Master secret of 48 bits shared with the peer

What do you need for the connection?

Random data for the server and client

Server and client write keys

Server and client MAC keys

Initialization vectors for ciphers (if applicable)

Server / client sequence #'s

Order of events

Use PKI to exchange keys (interchange)

Handshake to decide the exchange method

Encipher using a classical cipher (symmetric)

RSA 512 bit w/DES 40-bits and SHA for MAC

MAC (Checksum/Digest)

Hash (K_{MAC-WS} , OuputPad + Hash(K_{MAC-WS} , InputPad Seq# SSL Type Block))

Hash twice using the input and output pad

HMAC – Generic term for algorithm that uses a keyless hash and
a key to produce a keyed hash function

Why? Restriction on imports / exports

h = keyless hash function

b = block size (bytes)

l = hash size

k = key

$k \leq b$ Otherwise use $h(k)$ to produce k of length b

k' is k padded with zeros to make it by bytes long

ipad sequence of 00110110 repeated b times

opad sequence of 01011100 repeated b times

HMAC-h(k, m) = h(k' XOR opad + h(k' XOR ipad + m))
+ is concatenation

What could we use as a hash?

SHA, MD5, etc.

Master secret

Master = MD5 (pre + SHA("A" + pre + rand_1 + rand_2) +
MD5 (pre + SHA("BB" + pre + rand_1 + rand_2) +
MD5 (pre + SHA("CCC" + pre + rand_1 + rand_2))

pre -> shared data

rand_1 1 and rand_2 are shared random numbers

Think Needham-Schroeder

SSL Record Protocol (lower layer)

Take something unsecure and send it securely

HTTP -> HTTPS

Message gets compressed

MAC is computed

Compressed block + MAC are encrypted

SSL Handshake

How do we decide what keys to use / shared info?

Step 1 : Client initiates SSL handshake

Can also be done by the server

C -> S: { Version + Rand_1 + Session_ID + Cipher_List + Comp_List }

Version SSL version number (3)

Rand_1 Nonce of Timestamp and 28 random bytes

Session_ID 0 (new) or existing #

Cipher_List What ciphers does the client understand?

Comp_List What compression does the client understand?

Step 2: Server responds

S -> C : { version + rand_2 + session_ID + Cipher + Comp }

Version What will be used Max (Client, Server)

Cipher Cipher to use

Comp Compression mech to use

Session_ID ID of this session (must be non-zero)
Rand_2 Nonce similar to earlier nonce

Step 3: Server authenticates

S->C {Server_Cert}
The server's X.509 certificate

Step 4: Server sends the parameters for the communication

S-> C : { mod + exp + E (K_{priv-S}, Hash (Rand_1 + Rand_2 + Mod + Exp))}

mod, exp Used to verify
Rand_1, Rand_2 From initial exchange
Encrypt the hash with the private key of the server
Allows for decryption using the public key
Get the public key from the X.509v3 certificate

Step 5: Server may request certificate from client

S -> C: { Cert_Type + Good_CAs }
What type of certificate does the server want?
Who are trusted parties by the server?

Step 6: Server ends the second round

S -> {End_Round_2}

Step 7: Client responds with certificate

Recall chaining / verification of earlier
This step is optional

Step 8: Begin the key exchange

C -> S : {pre}

pre is the pre master secret

Both parties compute the master secret

Step 9: Client sends a verification message

C -> S: { h (master + opad + h(m + master + ipad))

HMAC to send the verification

master is the master secret computed from pre
ipad and opad are defined for the HMAC
m is the concatenation of messages 1 through 8

Step 10: Client instructs server to start using cipher

C -> S: $h(\text{master} + \text{opad} + h(\text{messages} + 0x434C4E54 + \text{master} + \text{ipad}))$

messages include message 9

Step 11: Server responds with finished message

S -> C: $h(\text{master} + \text{opad} + h(\text{messages} + 0x53525652 + \text{master} + \text{ipad}))$

Messages include message 9

Enciphered communications can now begin

Now the official PKI exchange begins

C -> S Sends the symmetric key using the server's public key as
verified by the certificate and handshake protocol