Secure Sockets Layer (SSL) / Transport Layer Security (TLS)

[redacted]

Network Security Products
S31213

Example

- http://www.greatstuff.com
- Wants credit card number
- Look at lock on browser
- Use https instead of http

History

A protocol designed by Netscape in late 1994 to provide communications security over the Internet

GOALS:

- message privacy
- message integrity
- mutual authentication

SSL Versions

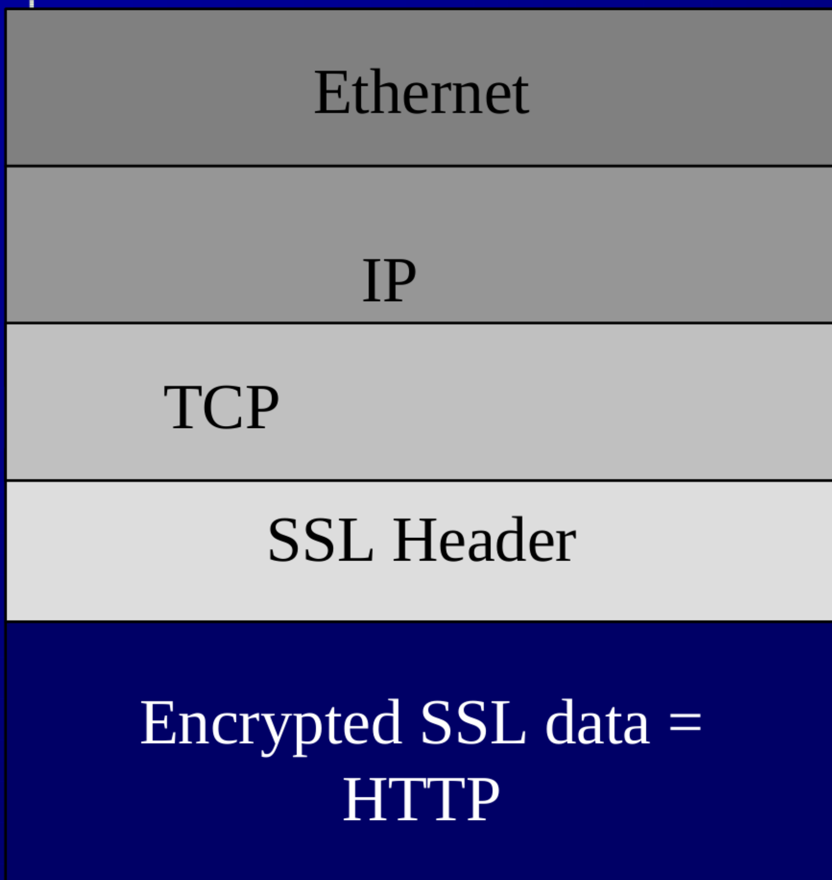
- 1.0: July 1994 – not released publicly
- 2.0: Dec 1994
- 3.0: Nov 1995
- 3.1: Jan 1999: RFC 2246 (TLS 1.0)
- 3.2: Apr 2006: RFC 4346 (TLS 1.1)
- 3.3: Aug 2008: RFC 5246 (TLS 1.2)

SSL Version Usage

SSL 3.0:	49%
SSL 3.1: (TLS 1.0)	48%
SSL 2.0:	3%
other:	< 0.0001 %

- Even though SSL 3.2 (TLS 1.1) and SSL 3.3 (TLS 1.2) have been out for a while, they are not seen!

Location of SSL Protocols



- Independent of packet boundaries
- Multiple SSL records can be sent per packet
- SSL records can span packets

TCP ports used by SSL

- IANA has over 60 ports specified for SSL/TLS use!
- Some ports seen more than others
  - https 443 ~63% of SSL/TLS traffic
  - pop3s 995 ~1 % of traffic

NON IANA TCP ports

- SSL/TLS can use ANY port!
  - tor 9001 10%
  - ? 4090 4% (mobile ip server)
  - tor 11375 2%
  - p2p 16613 1% (limewire)
  - p2p 44348 1% (limewire)
  - p2p ? 18% (limewire or other)

Note: all statistics ignore SSL sent under protocols other than IP

SSL Operation

- Application calls SSL connect routines to set up channel
- **Public Key** cryptography is used during handshake to authenticate parties and exchange session key
- **Symmetric Key** cryptography (using session key) is used to encrypt the data

Public Key Algorithms

- Key Exchange used to derive session keys for encryption:
  - RSA
  - Diffie-Hellman (DH / EDH / ADH)
  - Elliptic Curve Diffie-Hellman (ECDH / ECDHE)
  - Pre-Shared Key (PSK)
  - Secure Remote Password (SRP)
  - Fortezza
  - Kerberos
- Authentication mechanisms
  - RSA
  - DSA
  - None (Anonymous)

Symmetric Key Algorithms

- Work horse of algorithms
- Can offer near perfect secrecy
- Block - encrypt data block
  - RC2 – 128 bit key
  - DES - 56 bit key or Triple DES
  - IDEA - 128 bit key, PGP
  - AES – 128 or 256 bit key
  - SEED – 128 bit key
  - CAMELLIA – 128 or 256 bit key
- Stream - encrypt byte by byte
  - RC4 - 128 bit key

Message Digests

- Hash functions
  - All output is influenced by all input
  - If an input bit is changed, every output bit has 50% chance of changing
  - Improbable for different inputs to have the same hash
  - MD5(128 bit), SHA-1(160 bit)

Key Exchange Usage

- 65% RSA
- 20% ADH
- 1% DHE / RSA
- 0.7% RSA Export

The SSL Handshake

- Handshake determines:
  - SSL version (2 or 3.x)
  - Cipher suite
    - public key scheme (Diffie-Hellman, RSA)
    - symmetric key scheme (DES, RC4)
    - key length
    - hashing routine (SHA1, MD5)
  - Compression Scheme

Client Messages		Server Messages
Client Hello	→	
	←	Server Hello
	←	Certificate *
	←	Server Key Exchange *
	←	Certificate Request *
	←	Server Hello Done
Certificate *	→	
Client Key Exchange	→	
Certificate Verify *	→	
Change Cipher Spec	→	
Finished	→	
	←	Change Cipher Spec
	←	Finished
Application Data	↔	Application Data

Client Hello - Version 2

```
SSL 2.0 handshake message <1 byte>
Message length            <1 byte>
Client hello message      <1 byte = 01>
Preferred SSL Version     <2 bytes>

Cipher suite length       <2 bytes>
Session ID length         <2 bytes>
Client Hello Random length<2 bytes – usually 0x10>
```

Cipher suites <Set of 3 byte cipher suites>
Session ID If present, resumed session
Client Hello Random

Client Hello – Version 3/TLS

SSL 3/TLS handshake	<1 byte>
Version	<2 bytes>
Message length	<2 bytes>
Client hello message	<1 byte = 01>
Length	<3 bytes>
Version	<2 bytes
Client Hello Random	**<32 bytes>**
Session ID length	<1 byte – usually 0x20 or 0>
Session ID	**If present, resumed session**
Cipher suite length	<2 bytes>
Cipher suites	**<Set of 2 byte cipher suites>**
Compression length	<1 byte>
Compression methods	

Server Hello – Version 3/TLS

SSL 3/TLS message	<1 byte – 0x16>
Version	<2 bytes>
Length	<2 bytes>
Server hello message	<1 byte – 0x02>
Length	<3 bytes>
Version	<2 bytes>
Server hello random	**<32 bytes>**
Session ID length	<1 byte – usually 0x20>
Session ID	
Cipher selected	**<2 byte cipher suite>**
Compression	<1 byte>

SSL Certificates

- X.509 version number
- name of entity the certificate is validating
- public key of entity
- issuer name, the Certificate Authority
- unique serial number
- validity period
- digital signature

Sample Parsed Certificate

Certificate:

Data:

Version: 1 (0x0)

Serial Number:

f4:bf:15:eb:73:ef:e2:16

Signature Algorithm: sha1WithRSAEncryption

Issuer: C=CA, ST=server-ca-state, L=server-ca-city, O=server-ca-company, OU=server-ca-section, CN=server-ca-name/emailAddress=server-ca@server.ca.com

Validity

Not Before: Apr 24 21:07:13 2008 GMT

Not After : May 24 21:07:13 2008 GMT

Subject: C=SE, ST=server-state, L=server-city, O=server-company, OU=server-section, CN=server-name/emailAddress=server@server.com

Certificate (con't)

Subject Public Key Info:

Public Key Algorithm: rsaEncryption

RSA Public Key: (1024 bit)

Modulus (1024 bit):

00:ad:e3:64:3f:45:75:44:be:b8:5f:ab:74:35:e0:12:ef:2f:41:23:ca:10:96:2e:e3:1a:48:da:c4:ef:
8d:ca:67:d9:11:8a:9f:45:6c:f2:7c:e9:cb:fd:51:9b:5d:0b:02:1b:9d:fa:9c:28:ae:8c:ef:43:eb:cc:
7e:50:27:52:2d:af:28:7c:89:c5:37:43:01:f8:e5:98:03:9d:fe:dc:d2:ba:74:84:86:be:6f:f6:93:c6:
5a:15:36:85:11:9e:24:f1:c0:c7:e8:05:d1:91:86:7f:0d:58:be:f8:80:8b:1a:f0:0b:f5:0d:28:10:1e:
b1:fe:9f:61:9b:27:15:06:b7

Exponent: 65537 (0x10001)

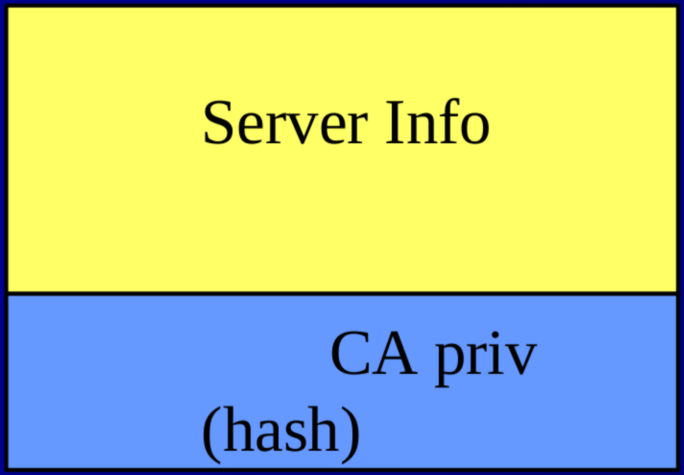
Signature Algorithm: sha1WithRSAEncryption

05:5e:a6:5a:eb:9c:ab:f6:2e:67:b2:7e:91:45:40:47:56:3d:76:5b:9a:d2:82:63:16:9a:d1:5a:4d:a0:
87:ed:2e:98:2a1a:4e:d9:04:bb:b0:b6:28:f6:a3:0b:f9:74:6f:c2:e1:dd:98:08:63:ff:2d:53:c5:b7:7c:
a8:c7:66:ea:6a:1a:cc:f9:4b:52:b1:bd:60:5e:d7:8c:aa:82:01:09:ef:15:d9:3a:98:45:0d:f1:9a:2c:be:
07:db:72:4c:b9:a2:90:c1:d1:06:fd:81:76:19:c5:4d:bf:30:df:81:c5:22:6b:5e:09:3f:9e:bc:b8:67:d5:
12:bb:24:da:7d

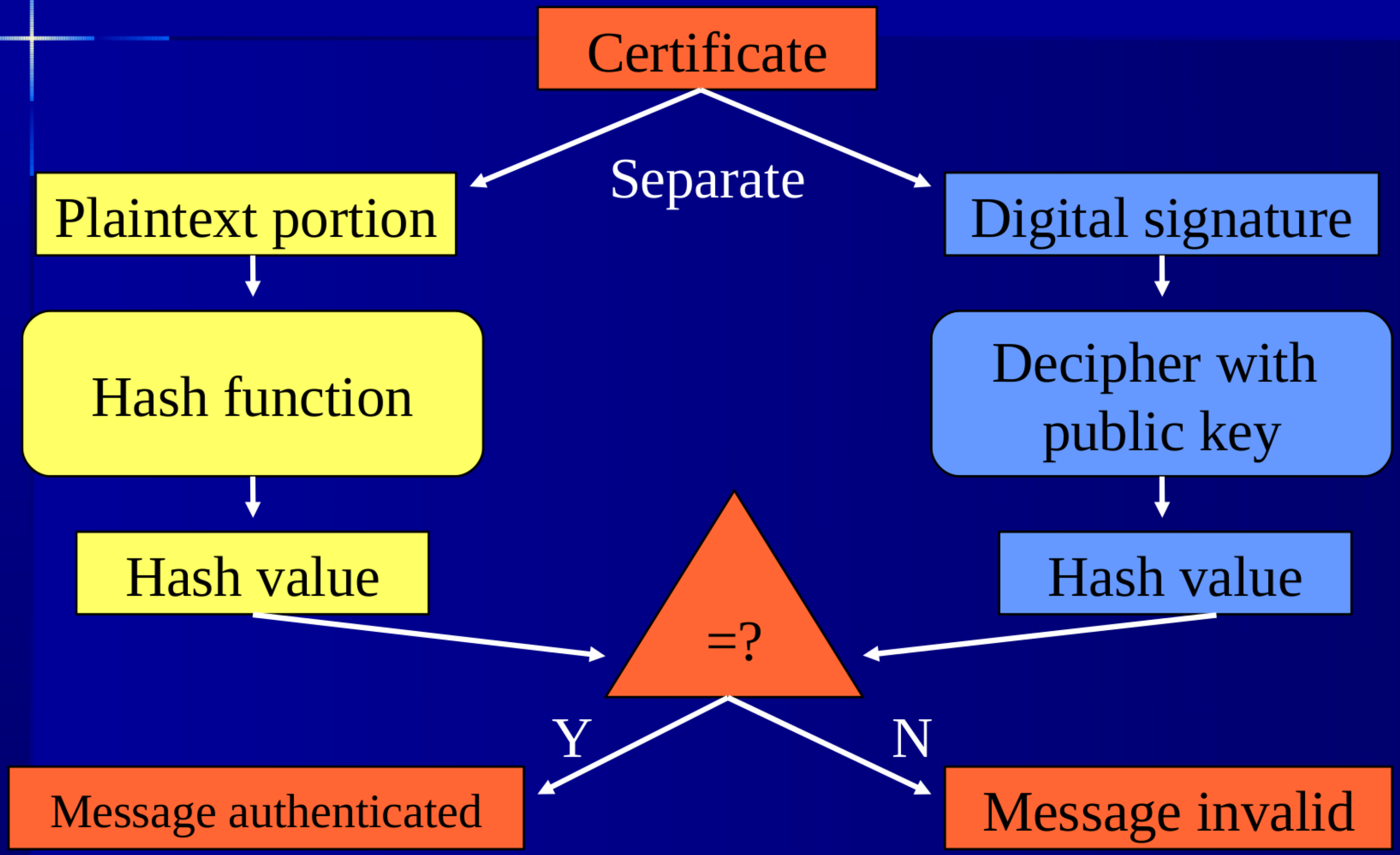
Certificate Authority

- Someone both parties trust
- Issuer of Certificates
- Many standard ones listed in browser options
  - VeriSign
  - GTE CyberTrust Root CA
  - Thawte Server

Certificate



X.509 Certificates



Key Generation

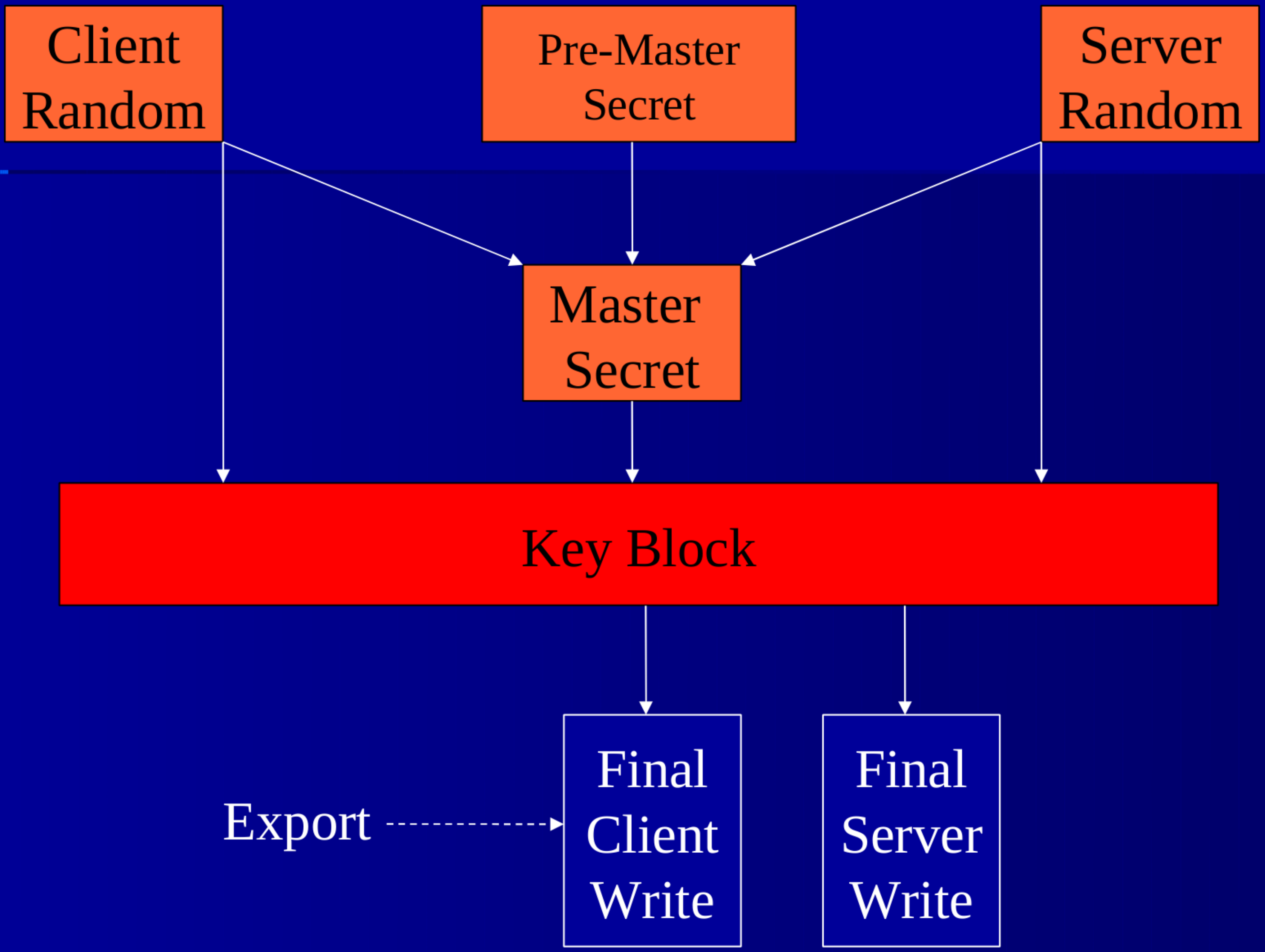
- Uses three random numbers to create session key
  - Client Random
  - Server Random
  - Pre-Master Secret
- Series of hash functions and bit selections

Pre-Master Secret

- 48 random bytes
- Either:
  - RSA: Sent in Client Key Exchange message encrypted with the public key of the server
  - Diffie-Hellman: Parameters are sent so that both sides can agree on a pre-master secret (either in the client key exchange message or the client certificate)
- THE security behind SSL/TLS

Master Secret

- Master secret is same across a session/resumed session.
- Used for generating encryption keys, MAC secrets and IVs.
- Formed differently for SSL and TLS, but both use a combination of:
  - SHA1
  - MD5
  - Client Random
  - Server Random
  - Pre-Master Secret
  - Fixed Constant (eg, “A” “client write key”)


Client Random
Pre-Master Secret
Server Random
Master Secret
Key Block
Export
Final Client Write
Final Server Write

Key Block

- **Generated per session. Generated differently for SSL/TLS, but both use:**
  - **SHA1**
  - **MD5**
  - **Client Random**
  - **Server Random**
  - **Master Secret**
  - **Fixed Constant (eg, “A” “client write key”)**

- The length of the key block generated depends upon the cipher suite used.

Session Keys

From the Key Block, pull out the keys as follows:

- **Client Write MAC Secret (Hash size bytes)**
- **Server Write MAC Secret (Hash size bytes)**
- **Client Write Key (Key Material Length)**
- **Server Write Key (Key Material Length)**
- **Client Write IV (IV Size)**
- **Server Write IV (IV Size)**

Example: 3DES_EDE_CBC_SHA

2 x 24 byte keys, 2 x 20 byte MAC secrets,2 x 8 byte IVs
= 104 bytes of key

Resumed Session

Client Messages		Server Messages
Client Hello	→	
	←	Server Hello
	←	Change Cipher Spec
	←	Finished
Change Cipher Spec	→	
Finished	→	
Application Data	↔	Application Data

Resumed Sessions

- Client sends session ID
- If stored in server cache, may use previous session information (Master key). Sends the same session ID back to client.
- Client does not send a Key Exchange, Server does not send a certificate
- Both use stored Master Key and skip first part of key generation

SSL Exploitation

- Not impossible!
- RSA key exchange “easy” to do because of fixed key.
- EDH key exchange not exploitable by the “easy” way. ☹

RSA Keys (Stating the Obvious)

If the Key Exchange type is RSA:

- If we can get a hold of the server's RSA private key, we can decrypt the Client Key Exchange message and read the pre-master secret key. No other heavy work need be done.
- Valid for life of certificate

Debain SSL

- Publically known weakness in the RNG for specific version of Debian openssl
- Creates finite set of RSA keys
- If Debian modulus is observed, lookup the private key in table (2 ^ 15) * 6 for each key size.
- Decrypt the traffic!

RSA Exploitation Steps

- Is it the key exchange RSA? (server hello)
  - If so, is the modulus match a known private key? (server certificate)
    - If so, is there 2-sided collect?
      - If so, do we have:
        - Client Hello
        - Server Hello
        - Client Key Exchange

DECRYPTION!

RSA Resumed Sessions

- Most traffic decrypted is resumed sessions (about 9 resumed sessions for every initial session).
- To decrypt a resumed session, you need:
  - Master key (initial session)
  - Client random (resumed session)
  - Server random (resumed session)

Problems in processing

- Literally millions of sessions per day
- Need to have good filtering and selection
- Need both sides of conversation
- USSID 18 issues

Network Traffic Problems

- Correctly reconstructing SSL session
  - Port reuse
  - Match client and server using time stamp
  - Match resumed sessions to initial sessions, using SSL session id or SSL session ticket

State Needed to Decrypt

- Cipher Suite
- Master Key
- Client Random
- Server Random
- Session ID / Session Ticket
- Index for finished message
- Index for application data

Contact Information

[redacted]

Network Security Products

S31213

[redacted]

Questions?