SharkFest '16 Europe

SSL troubleshooting

with Wireshark (and Tshark)

Tuesday, October 18th 2016

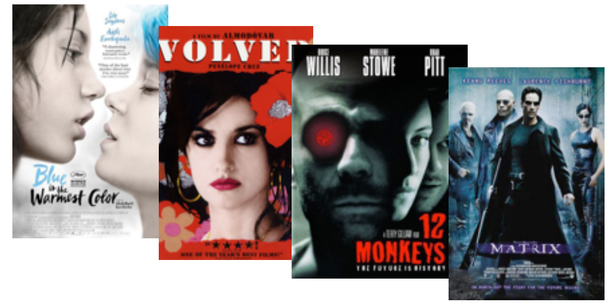
twitter: #sf16eu @SYNbit

Sake Blok
Packet Analyst | SYN-bit

About me...

EURONET INTERNET

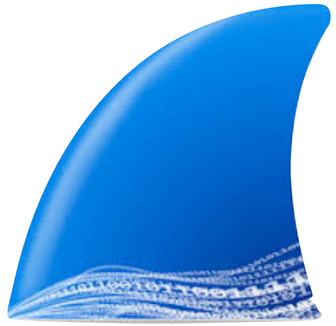
techno inside



ABN·AMRO

SYN-bit

deep traffic analysis



ion ip

f5

Haarlem

Do you trust these companies?

NETLOCK
Az Első Hitelesítés-szolgáltató

Dhimyotis

中華電信公開金鑰基礎建設

DigiNotar®
A VASCO COMPANY

About you…

Who?

…thinks SSL is just about encryption?
…has run into trouble setting up SSL connections?
…and had to troubleshoot SSL?
…had to analyze application traffic inside SSL?
…and to decrypt SSL traffic?
…but ran into problems decrypting?

Why…

…do we need SSL in the first place?

Confidentiality

Encryption / Decryption

Message Integrity

Message Digests / Signing

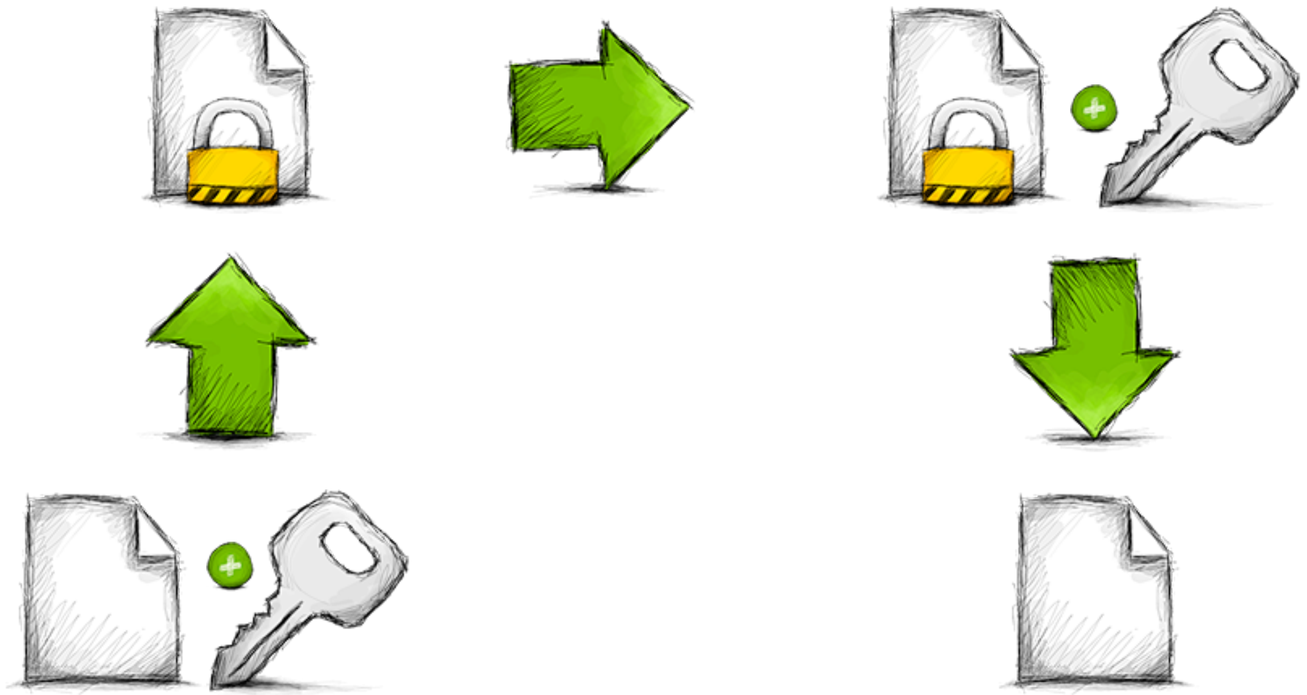
Authentication

Certificates / Certificate Authorities

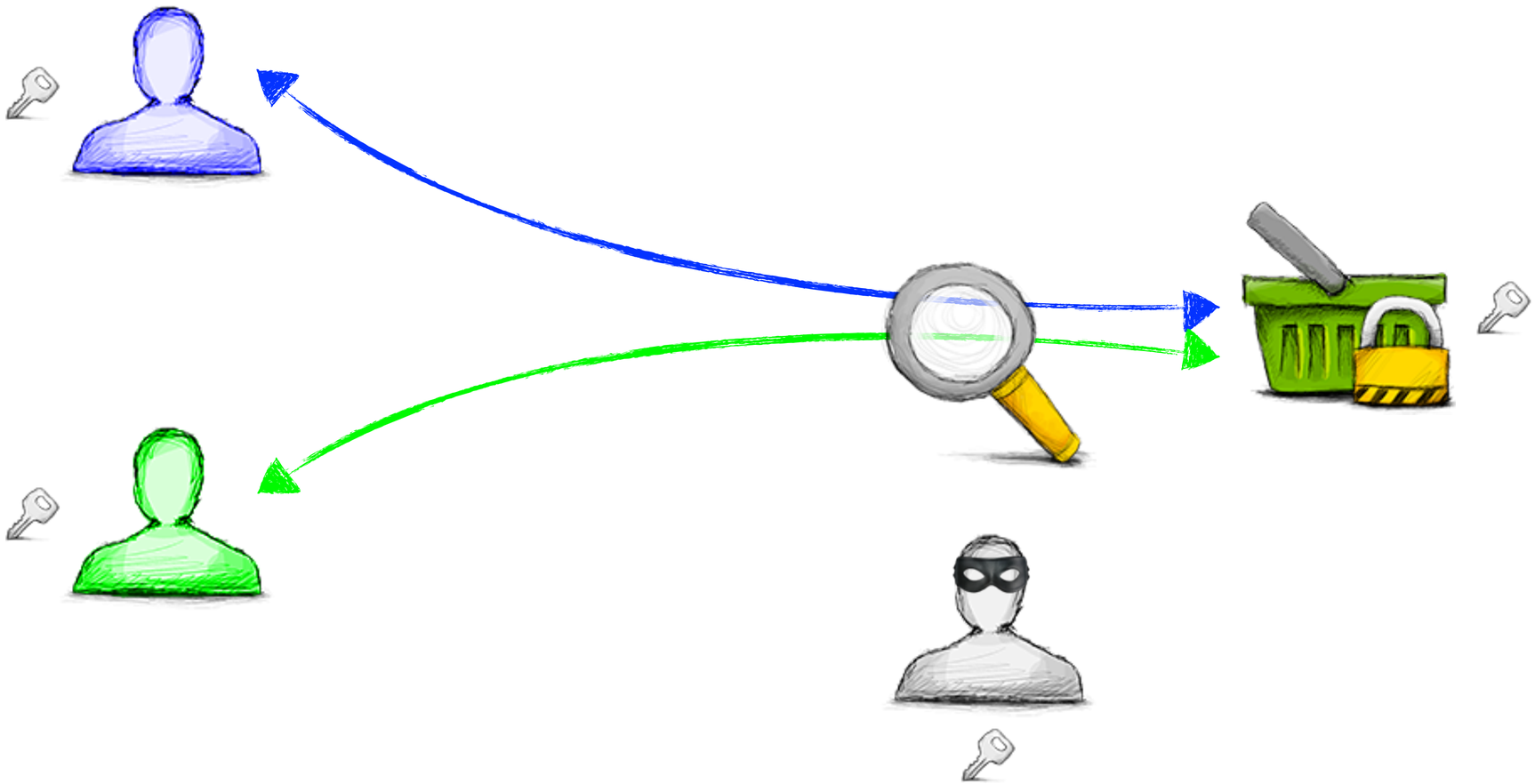
Cryptography 101

Symmetrical Encryption

- Same key for encryption and decryption
- Computatively "cheap"
- Short keys (typically 40-256 bits)
- DES, 3DES, AESxxx, RC4

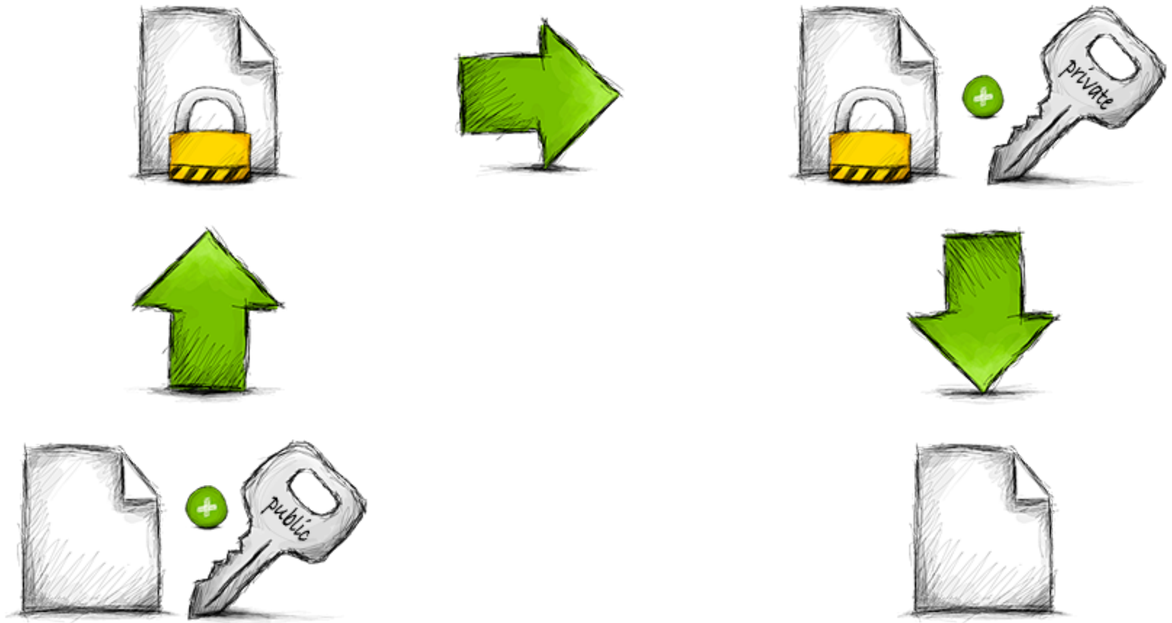


Symmetrical Session Key

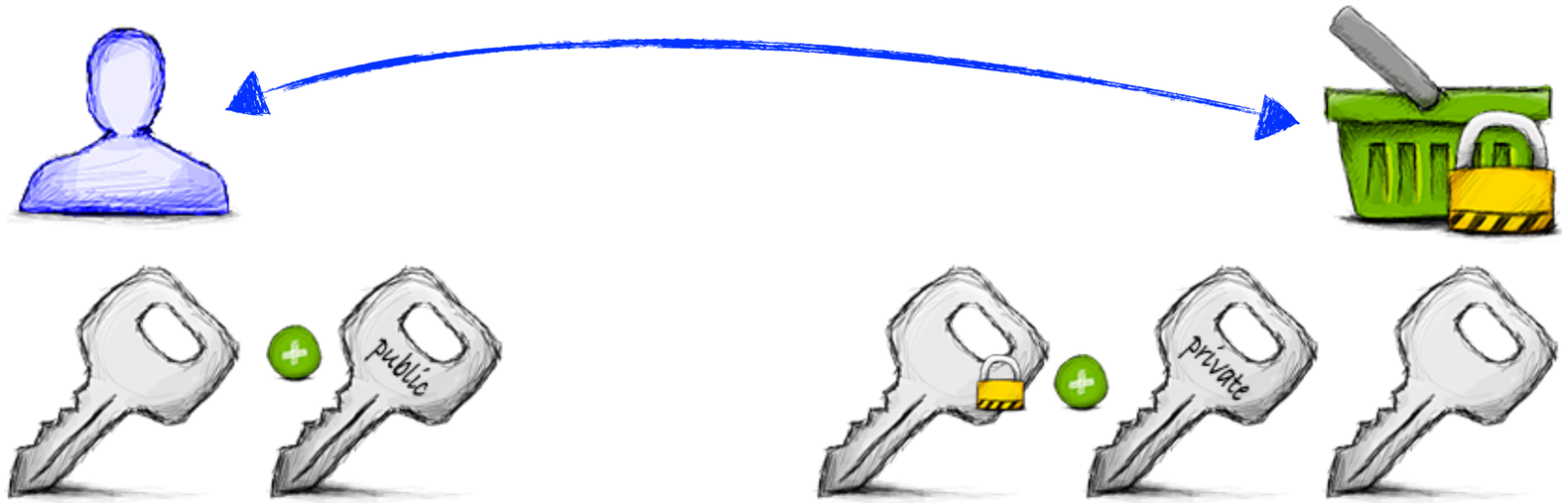


Asymmetrical Encryption

- One key for encryption, second key for decryption (both keys form a pair)
- Computatively "expensive"
- Long keys (RSA: 512-4096 bits)
- RSA, DSA, ECC



Secured Key Exchange



Man-in-the-Middle

Certificates

"In cryptography, a public key certificate (or identity certificate) is an **electronic document** which utilizes a **digital signature** to bind together a **public key** with an **identity**."
(From http://en.wikipedia.org/wiki/Digital_certificate)

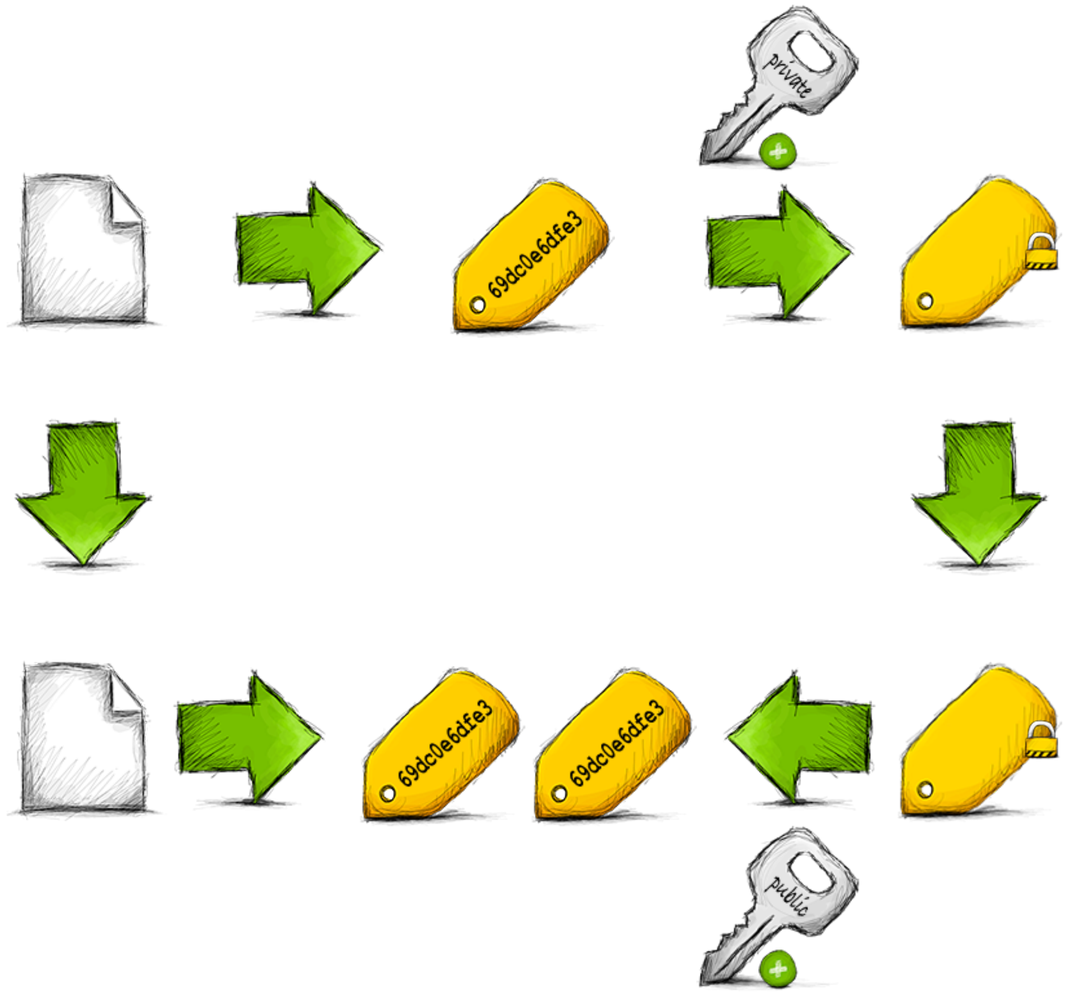
But what is signing??? And who is doing it?

Message Digest

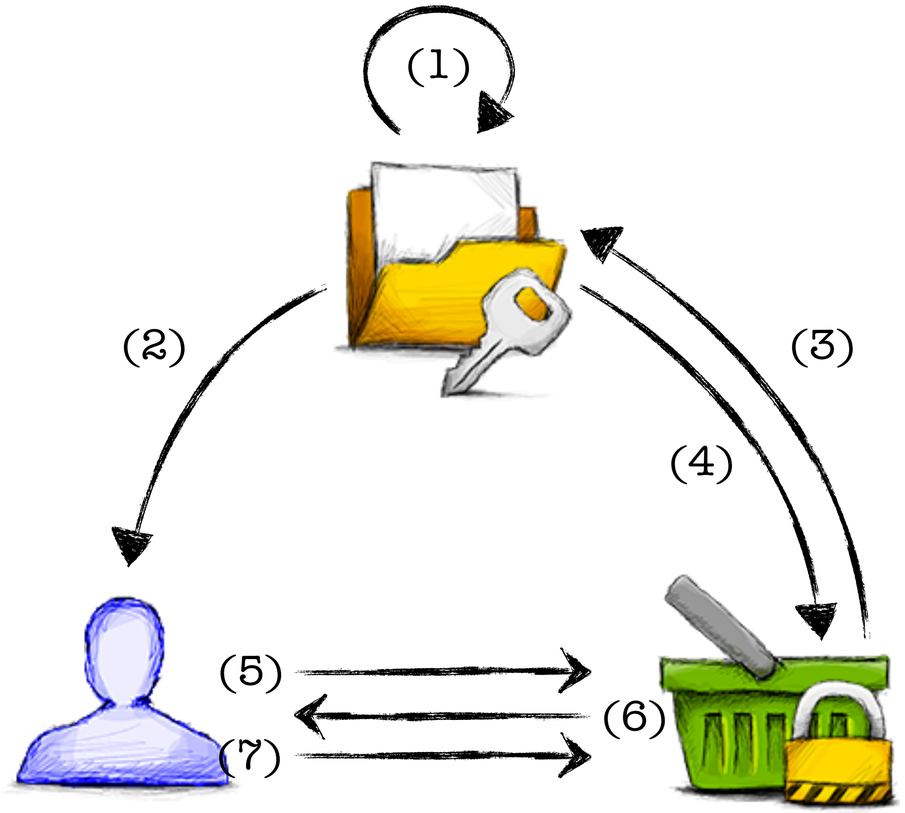
- Irreversible
  - original text not reproducable from the digest
- Collision-resistance
  - "Not possible" to create a message M'
    so that it has the same digest as message M
- MD5, SHA-1, SHA-2, SHA-3

Signing

- Create digest of message
- Encrypt digest with private key
- Authenticity and sender of message can be checked with public key



Certificate Authority



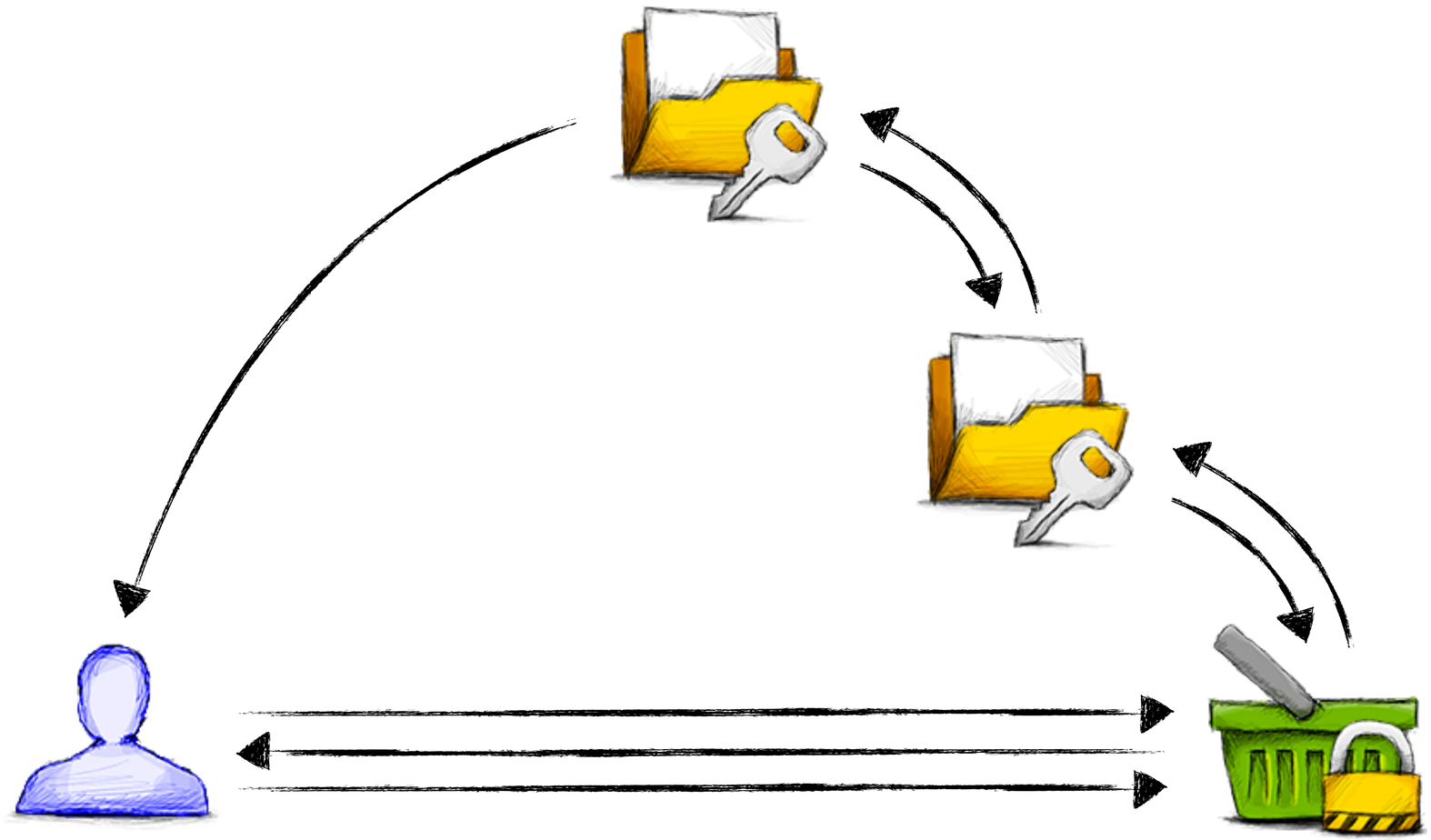
- Certificate Authority (CA) creates Identity & Keypair
- CA signs Identity & Public Key with Private Key to create its root-certificate (1)
- CA sends root-certificate to Clients (2)

- Server creates Identity & Keypair
- Server sends Identity & Public Key to Authority (3)
- **CA Checks Identity**
- CA Signs Identity & Public Key with Private Key
- CA sends back Certificate (4)

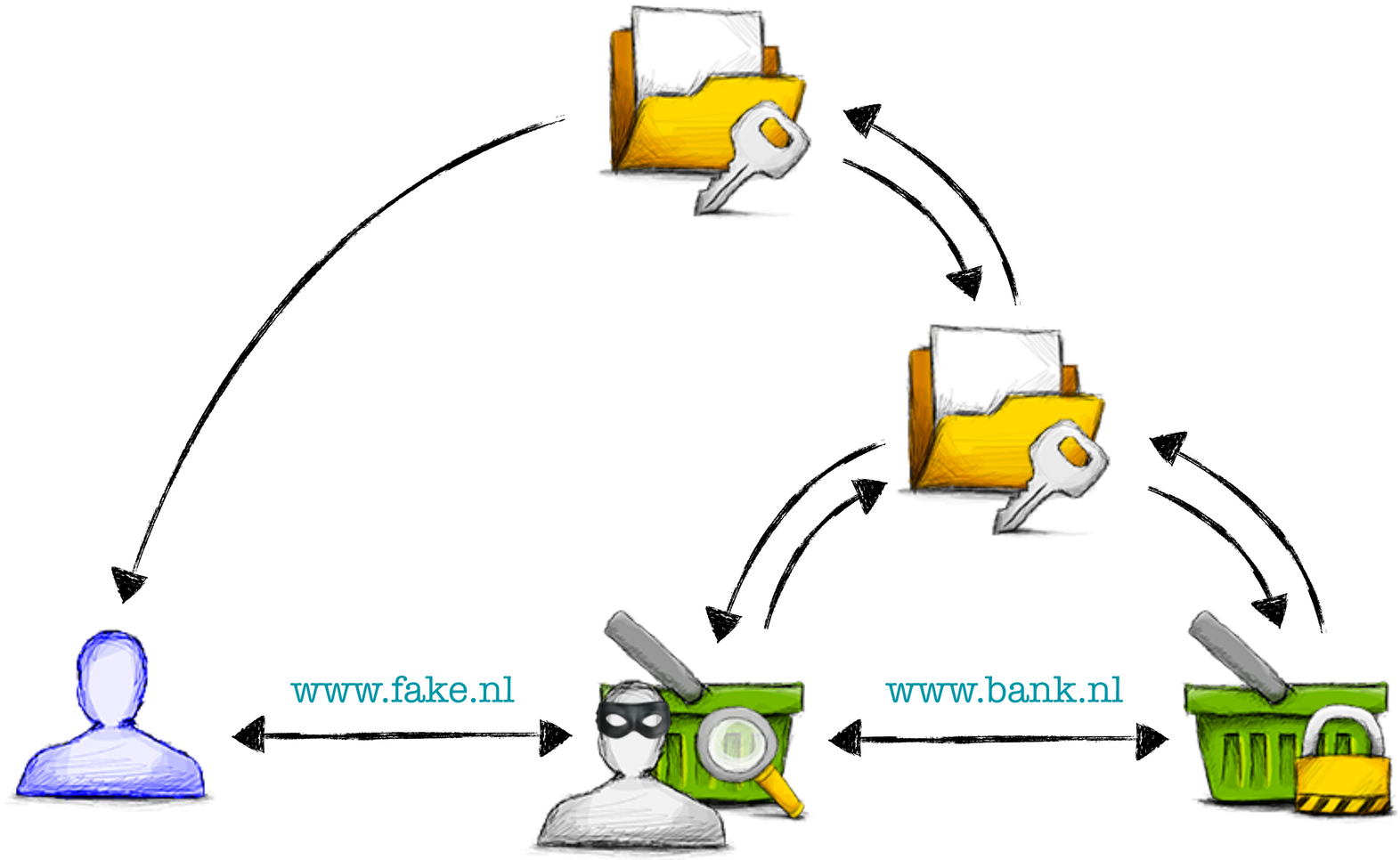
- Client connect to the Server (5)
- Server sends Certificate to Client (6)
- Client verifies Certificate with Public Key from the CA
- Client creates session keys
- Client encrypts session keys with the server Public Key
- Client sends encrypted keys to the server (7)
- Server decrypts session keys with its Private Key

- Client and server use session keys to exchange data

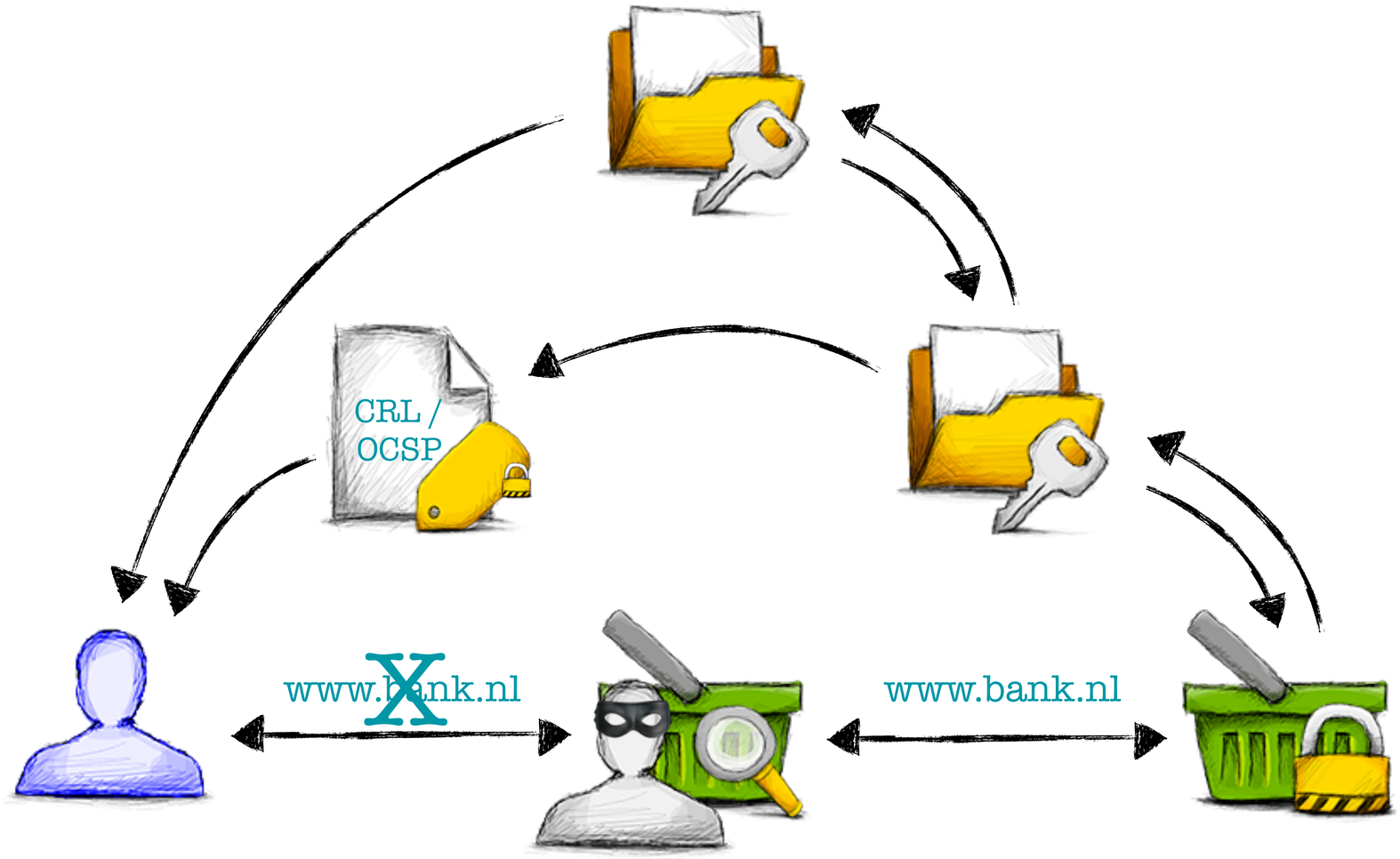
Intermediate CA's



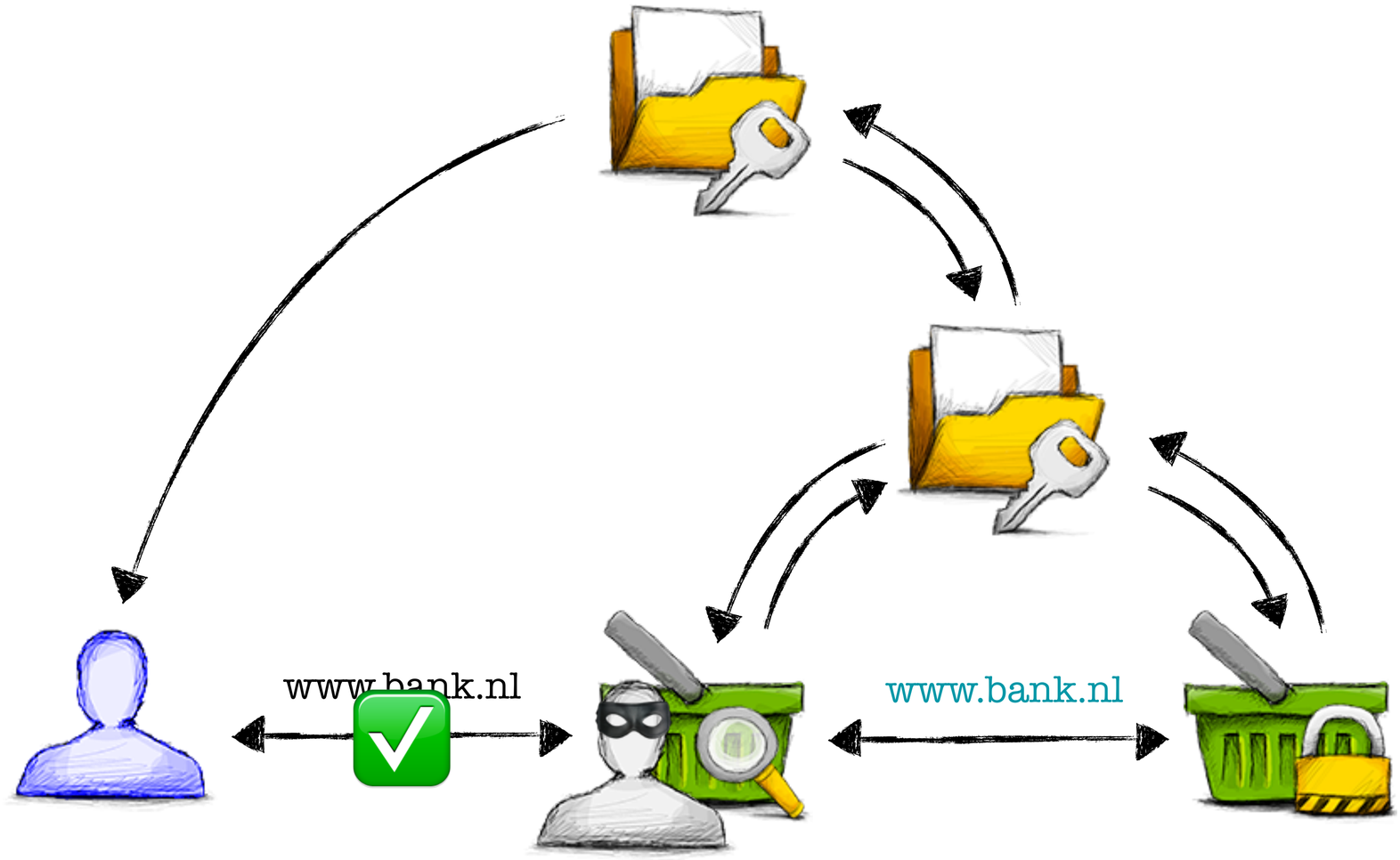
Man-in-the-Middle II



Compromised Server Key



Compromised CA



Trust!

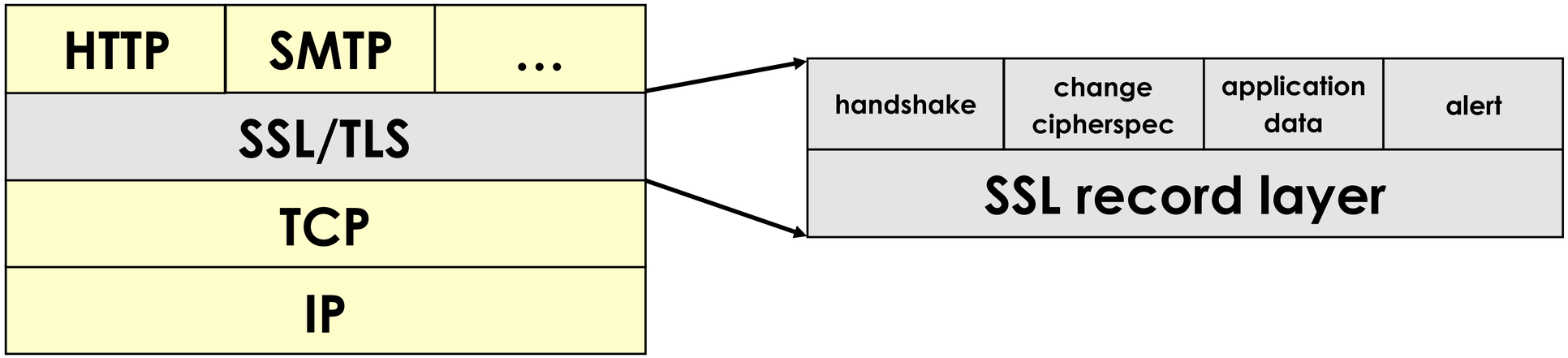
SSL/TLS protocol

SSL/TLS History

- SSLv1 by Netscape (unreleased, 1994)
- SSLv2 by Netscape (v2-draft,1994)
  - deprecated in 2011 (RFC 6176)
- SSLv3 by Netscape (v3-draft, 1995)
  - deprecated in June 2015 (RFC 7568)
- TLSv1.0, IETF (RFC 2246, 1999)
- TLSv1.1, IETF (RFC 4346, 2006)
- TLSv1.2, IETF (RFC 5246, 2008)
- TLSv1.3, IETF (draft 16, 2016)

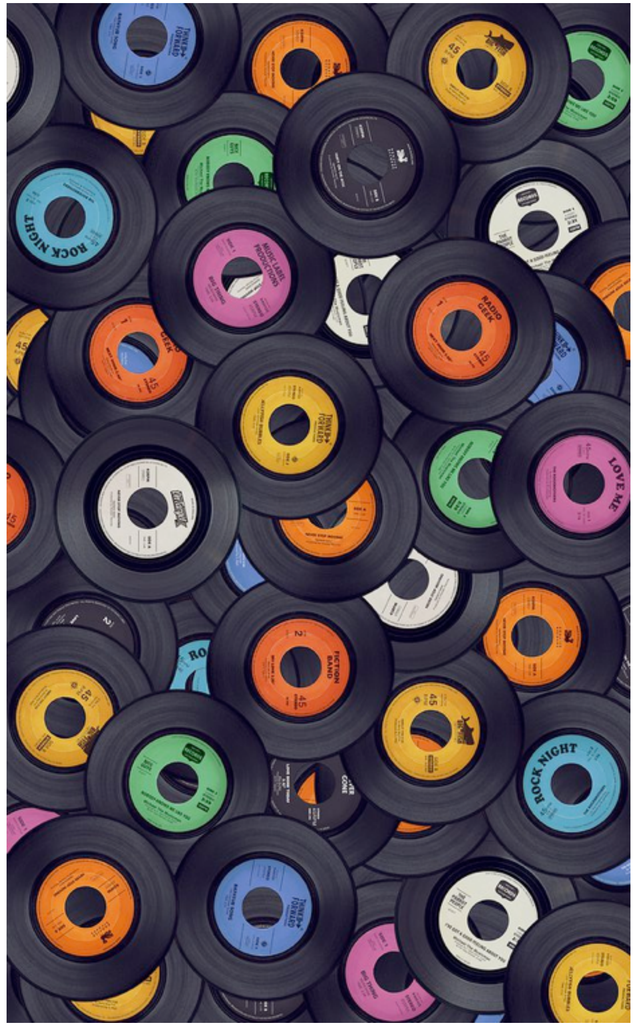
Place in TCP/IP stack

- Between transport and application layer
- Protocol independent



SSL Record Layer

- Provides fragmentation (max size 2^14)
- Multiple SSL messages (of one content type) per SSL Record allowed
- SSL Record can be split over multiple TCP-segments (2^14 > MSS!)
- One TCP-segment can contain multiple SSL Records (or fragments)



SSL Content Types

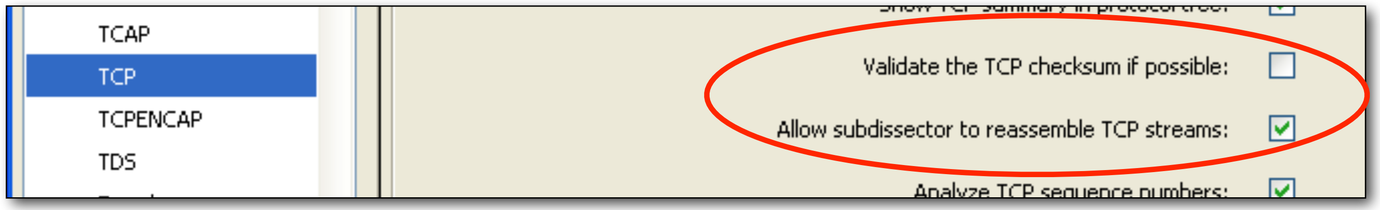
- Handshake Protocol (0x16)
  - responsible for authentication and session key setup

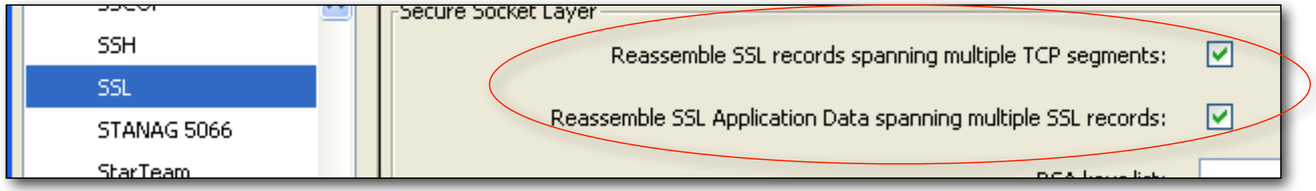
- ChangeCipherSpec Protocol (0x14)
  - Notify start of encryption

- Alert Protocol (0x15)
  - Reporting of warnings and fatal errors
- Application Protocol (0x17)
  - Actual encryption and transport of data

Troubleshooting SSL/TLS

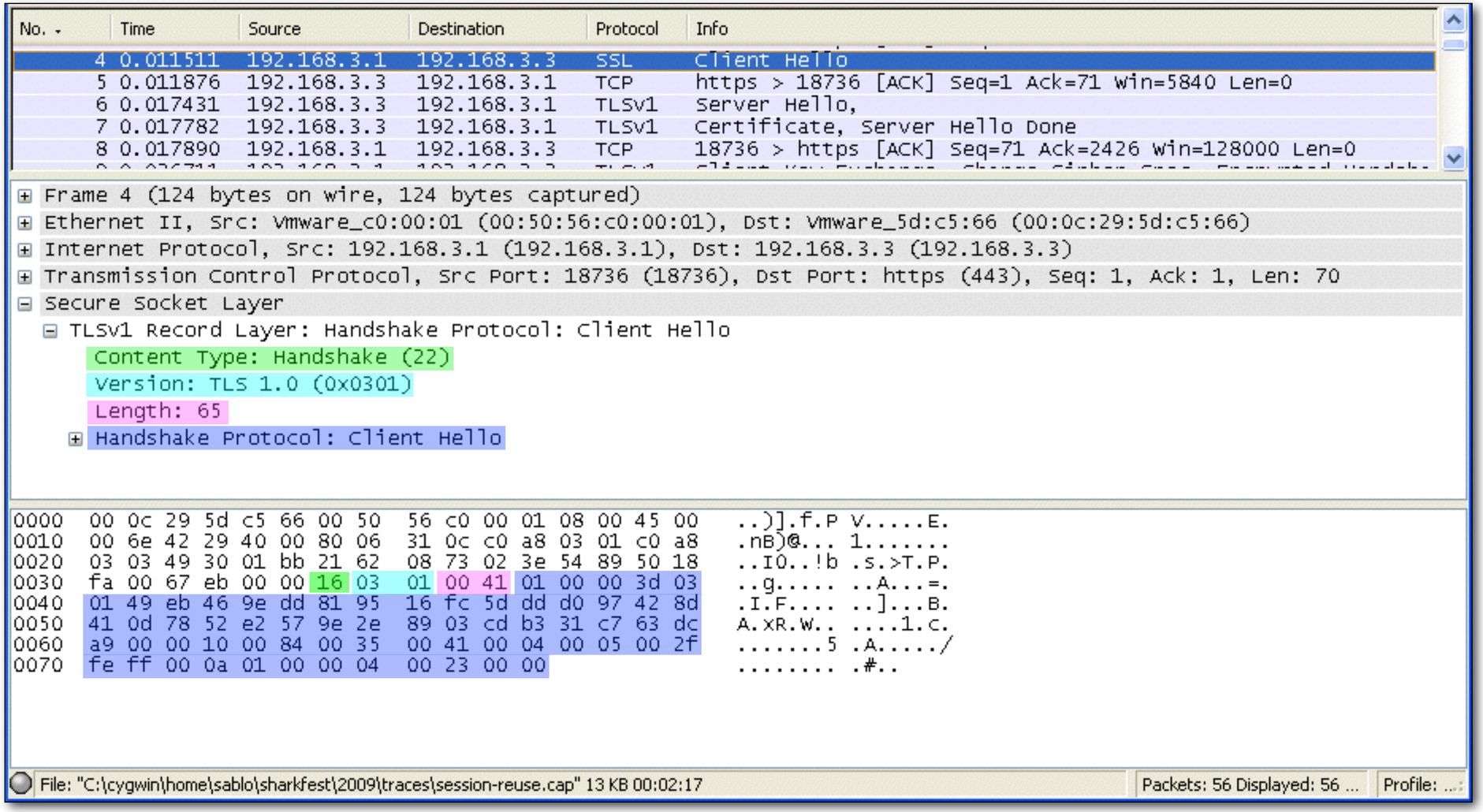
Choosing the right settings



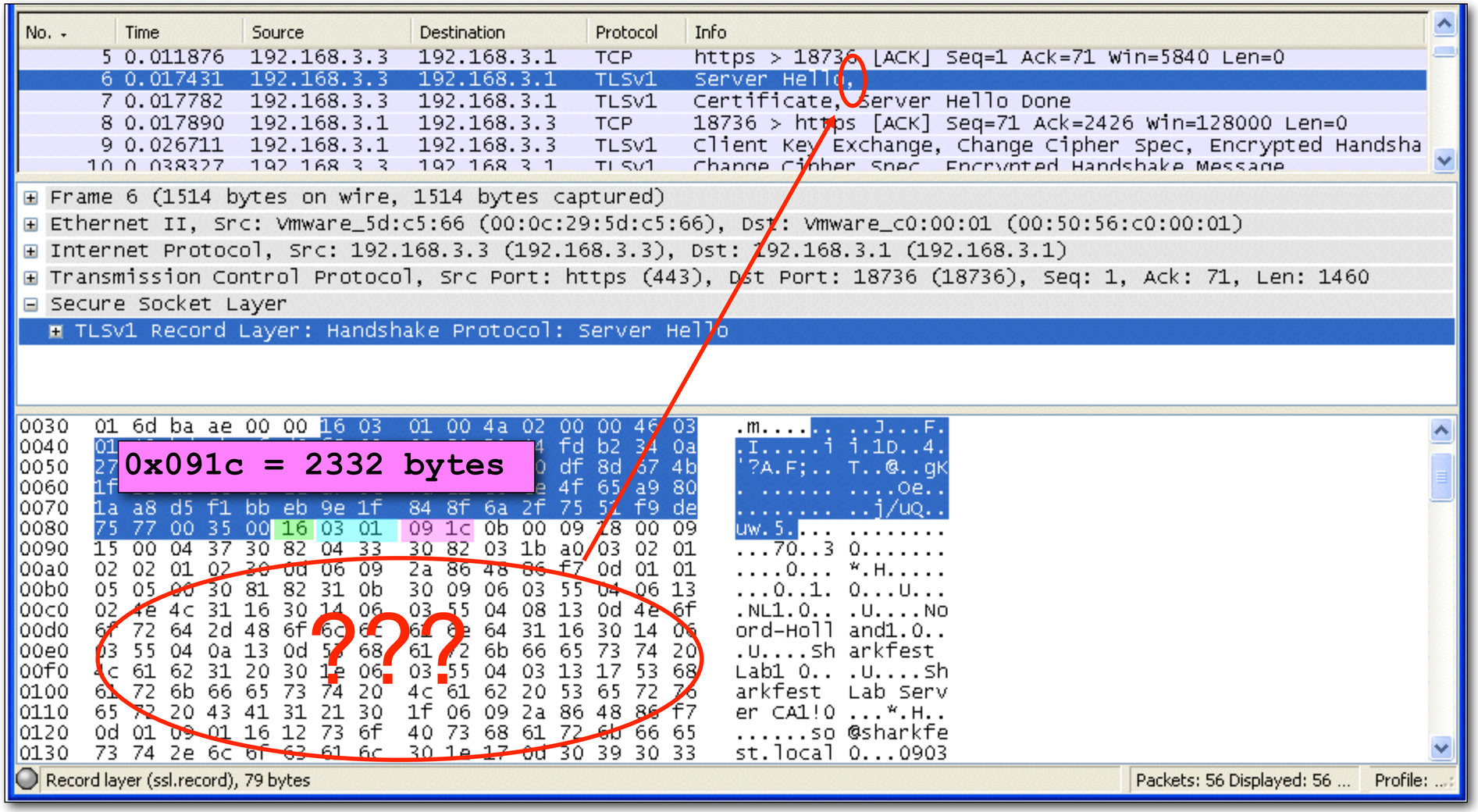


ip.defragment: TRUE
tcp.check_checksum: FALSE
tcp.desegment_tcp_streams: TRUE
ssl.desegment_ssl_records: TRUE
ssl.desegment_ssl_application_data: TRUE

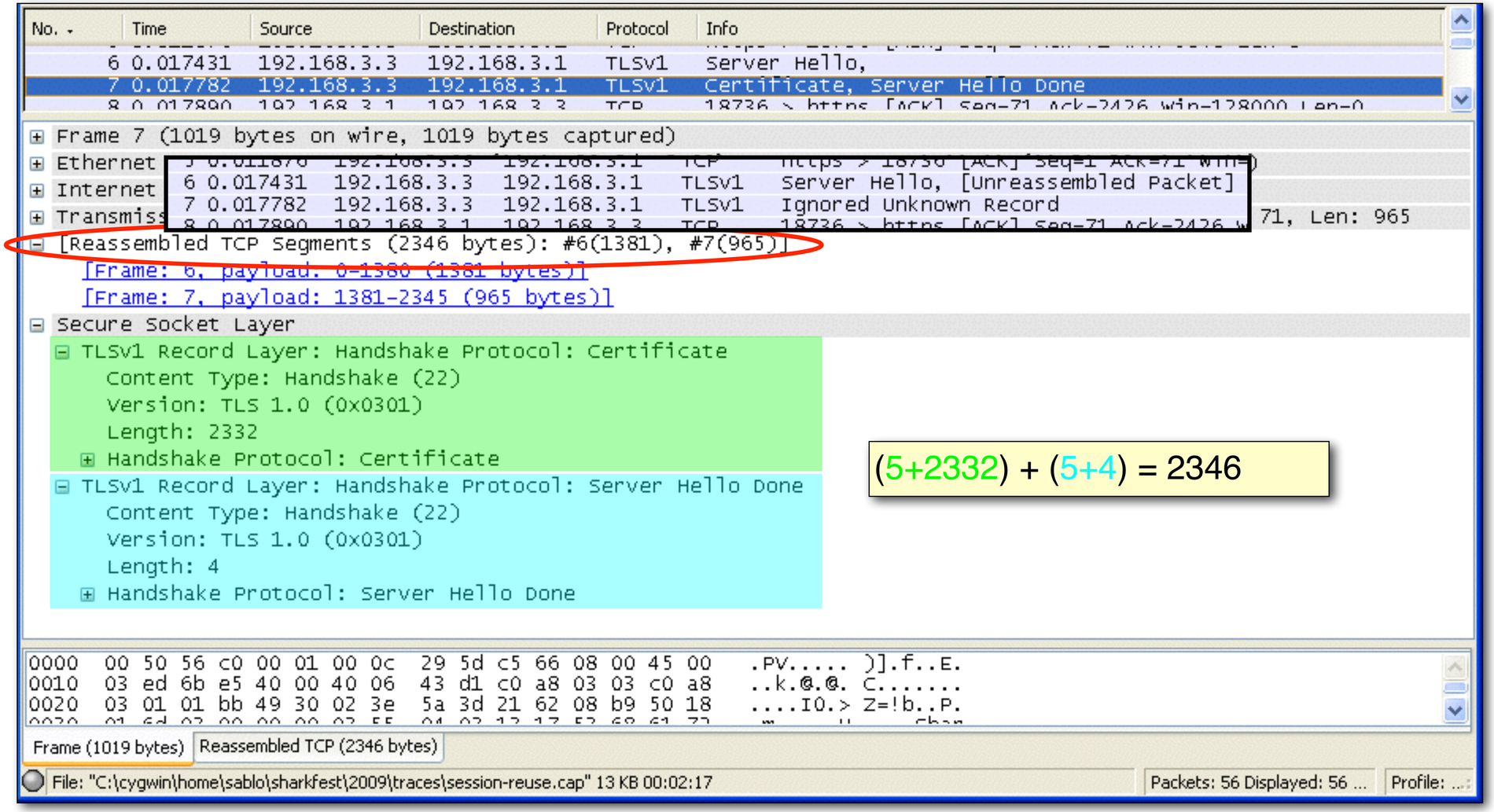
Analyzing the SSL record layer (1)



Analyzing the SSL record layer (2)

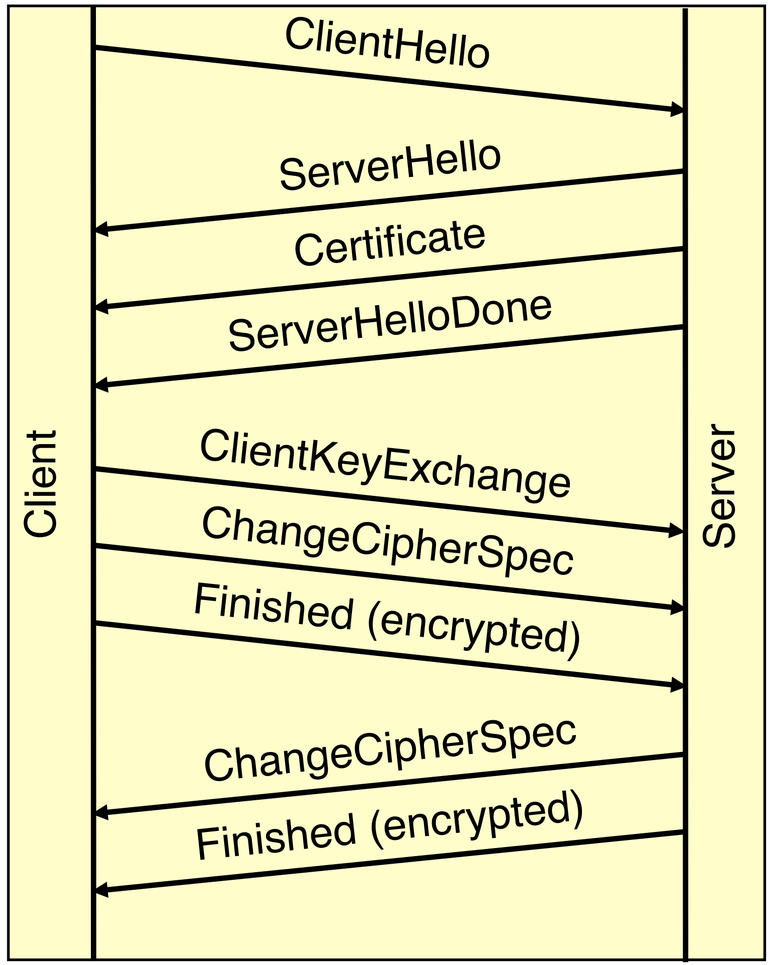


Analyzing the SSL record layer (3)



(5+2332) + (5+4) = 2346

Handshake with RSA key exchange



No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.3.1	192.168.3.3	TCP	18736 > https [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=1
2	0.000309	192.168.3.3	192.168.3.1	TCP	https > 18736 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 WS=
3	0.000357	192.168.3.1	192.168.3.3	TCP	18736 > https [ACK] Seq=1 Ack=1 Win=128000 Len=0
4	0.011511	192.168.3.1	192.168.3.3	TLSv1	Client Hello
5	0.011876	192.168.3.3	192.168.3.1	TCP	https > 18736 [ACK] Seq=1 Ack=71 Win=5840 Len=0
6	0.017431	192.168.3.3	192.168.3.1	TLSv1	Server Hello,
7	0.017782	192.168.3.3	192.168.3.1	TLSv1	Certificate, Server Hello Done
8	0.017890	192.168.3.1	192.168.3.3	TCP	18736 > https [ACK] Seq=71 Ack=2426 Win=128000 Len=0
9	0.026711	192.168.3.1	192.168.3.3	TLSv1	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Mes
10	0.038327	192.168.3.3	192.168.3.1	TLSv1	Change Cipher Spec, Encrypted Handshake Message

ClientHello (client)


```
Secure Socket Layer
  TLSv1 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 65
    Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 61
        Version: TLS 1.0 (0x0301)
      Random
          gmt_unix_time: Apr 19, 2009 17:43:26.000000000
          random_bytes: DD819516FC5DDDD097428D410D7852E2579E2E8903CDB331...
        Session ID Length: 0
        Cipher Suites Length: 16
      Cipher Suites (8 suites)
          Cipher Suite: TLS_RSA_WITH_CAMELLIA_256_CBC_SHA (0x0084)
          Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
          Cipher Suite: TLS_RSA_WITH_CAMELLIA_128_CBC_SHA (0x0041)
          Cipher Suite: TLS_RSA_WITH_RC4_128_MD5 (0x0004)
          Cipher Suite: TLS_RSA_WITH_RC4_128_SHA (0x0005)
          Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0x002f)
          Cipher Suite: SSL_RSA_FIPS_WITH_3DES_EDE_CBC_SHA (0xfeff)
          Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0x000a)
        Compression Methods Length: 1
      Compression Methods (1 method)
          Compression Method: null (0)
        Extensions Length: 4
      Extension: SessionTicket TLS
          Type: SessionTicket TLS (0x0023)
          Length: 0
          Data (0 bytes)
```

ServerHello (server)


```
Secure Socket Layer
  TLSv1 Record Layer: Handshake Protocol: Server Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 74
    Handshake Protocol: Server Hello
        Handshake Type: Server Hello (2)
        Length: 70
        Version: TLS 1.0 (0x0301)
      Random
          gmt_unix_time: Mar 16, 2009 02:30:23.000000000
          random_bytes: D6F56969813144FDB2340A273F419E463BF915549B0740DF...
        Session ID Length: 32
        Session ID: DB00C2AAD79CFDA109CE4F65A9801AA8D5F1BBEB9E1F848F...
        Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
        Compression Method: null (0)
```

Certificate I (server)


```
Secure Socket Layer
  TLSv1 Record Layer: Handshake Protocol: Certificate
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 2332
    Handshake Protocol: Certificate
        Handshake Type: Certificate (11)
        Length: 2328
        Certificates Length: 2325
      Certificates (2325 bytes)
          Certificate Length: 1079
        Certificate ()
          Certificate Length: 1240
        Certificate ()
  TLSv1 Record Layer: Handshake Protocol: Server Hello Done
```

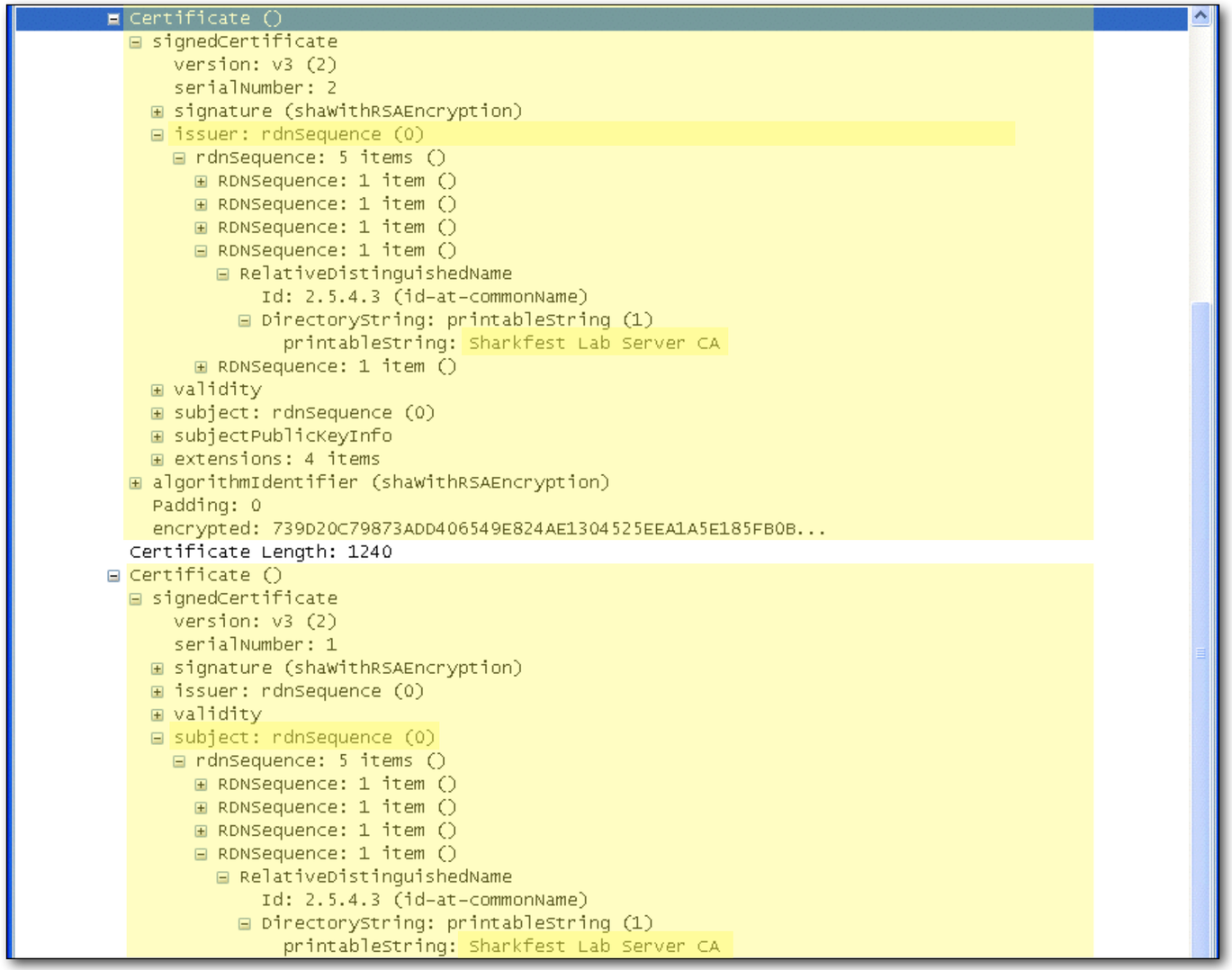
Certificate II (server)


```
Handshake Protocol: Certificate
    Handshake Type: Certificate (11)
    Length: 2328
    Certificates Length: 2325
  Certificates (2325 bytes)
      Certificate Length: 1079
    Certificate ()
      signedCertificate
          version: v3 (2)
          serialNumber: 2
        signature (shaWithRSAEncryption)
        issuer: rdnSequence (0)
        validity
        subject: rdnSequence (0)
        subjectPublicKeyInfo
        extensions: 4 items
      algorithmIdentifier (shaWithRSAEncryption)
          Algorithm Id: 1.2.840.113549.1.1.5 (shaWithRSAEncryption)
      Padding: 0
      encrypted: 739D20C79873ADD406549E824AE1304525EEA1A5E185FB0B...
      Certificate Length: 1240
    Certificate ()
```

Certificate III (server)


```
validity
subject: rdnSequence (0)
  rdnSequence: 5 items ()
    RDNSequence: 1 item ()
      RelativeDistinguishedName
          Id: 2.5.4.6 (id-at-countryName)
          CountryName: NL
    RDNSequence: 1 item ()
      RelativeDistinguishedName
          Id: 2.5.4.8 (id-at-stateOrProvinceName)
        DirectoryString: printableString (1)
            printableString: Noord-Holland
    RDNSequence: 1 item ()
      RelativeDistinguishedName
          Id: 2.5.4.10 (id-at-organizationName)
        DirectoryString: printableString (1)
            printableString: Sharkfest Lab
    RDNSequence: 1 item ()
      RelativeDistinguishedName
          Id: 2.5.4.3 (id-at-commonName)
        DirectoryString: printableString (1)
            printableString: public.sharkfest.local
    RDNSequence: 1 item ()
      RelativeDistinguishedName
          Id: 1.2.840.113549.1.9.1 (pkcs-9-at-emailAddress)
          SyntaxIA5String: co@sharkfest.local
subjectPublicKeyInfo
```

Certificate IV (server)



ServerHelloDone (server)


```
Secure Socket Layer
  TLSv1 Record Layer: Handshake Protocol: Certificate
  TLSv1 Record Layer: Handshake Protocol: Server Hello Done
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 4
    Handshake Protocol: Server Hello Done
        Handshake Type: Server Hello Done (14)
        Length: 0
```

ClientKeyExchange (client)


```
Secure Socket Layer
  TLSv1 Record Layer: Handshake Protocol: Client Key Exchange
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 134
    Handshake Protocol: Client Key Exchange
        Handshake Type: Client Key Exchange (16)
        Length: 130
  TLSv1 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  TLSv1 Record Layer: Handshake Protocol: Encrypted Handshake Message
```

ChangeCipherSpec (client)


```
Secure Socket Layer
  TLSv1 Record Layer: Handshake Protocol: Client Key Exchange
  TLSv1 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
      Content Type: Change Cipher Spec (20)
      Version: TLS 1.0 (0x0301)
      Length: 1
      Change Cipher Spec Message
  TLSv1 Record Layer: Handshake Protocol: Encrypted Handshake Message
```

Finished (client)

Without decryption:

```
Secure Socket Layer
  TLSv1 Record Layer: Handshake Protocol: Client Key Exchange
  TLSv1 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  TLSv1 Record Layer: Handshake Protocol: Encrypted Handshake Message
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 48
      Handshake Protocol: Encrypted Handshake Message
```

With decryption:

```
Secure Socket Layer
  TLSv1 Record Layer: Handshake Protocol: Client Key Exchange
  TLSv1 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  TLSv1 Record Layer: Handshake Protocol: Finished
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 48
    Handshake Protocol: Finished
        Handshake Type: Finished (20)
        Length: 12
        Verify Data
```

ChangeCipherSpec (server)


```
Secure Socket Layer
  TLSv1 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
      Content Type: Change Cipher Spec (20)
      Version: TLS 1.0 (0x0301)
      Length: 1
      Change Cipher Spec Message
  TLSv1 Record Layer: Handshake Protocol: Encrypted Handshake Message
```

Finished (server)

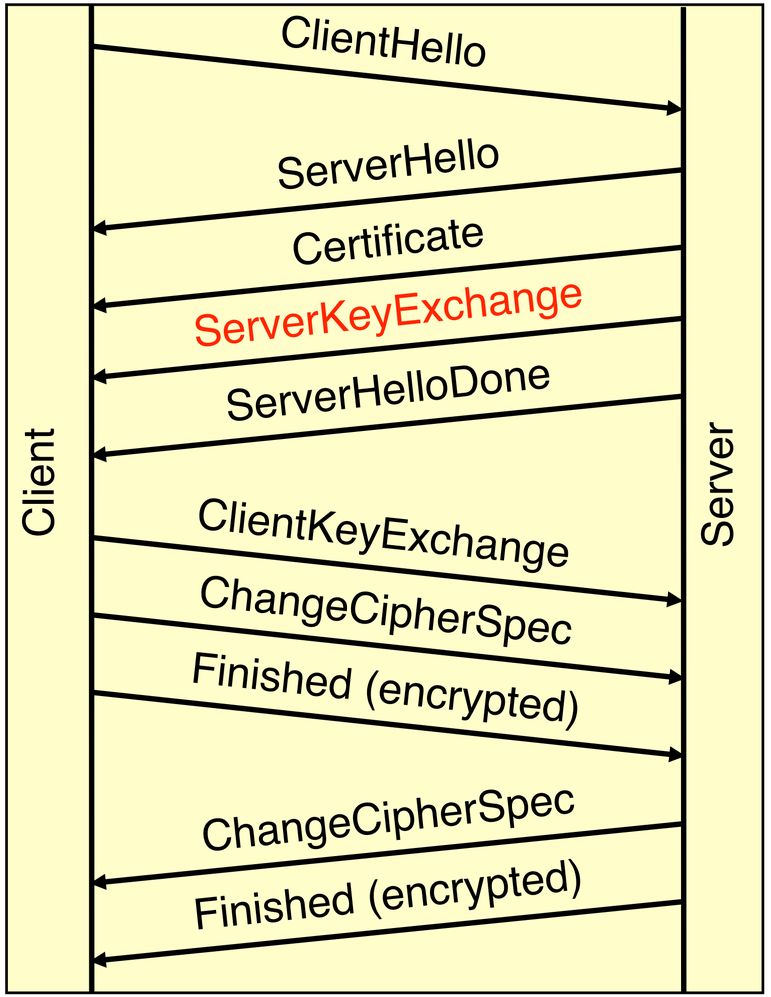
Without decryption:

```
Secure Socket Layer
  TLSv1 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  TLSv1 Record Layer: Handshake Protocol: Encrypted Handshake Message
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 48
      Handshake Protocol: Encrypted Handshake Message
```

With decryption:

```
Secure Socket Layer
  TLSv1 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  TLSv1 Record Layer: Handshake Protocol: Finished
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 48
    Handshake Protocol: Finished
        Handshake Type: Finished (20)
        Length: 12
        Verify Data
```

Handshake with DHE key generation

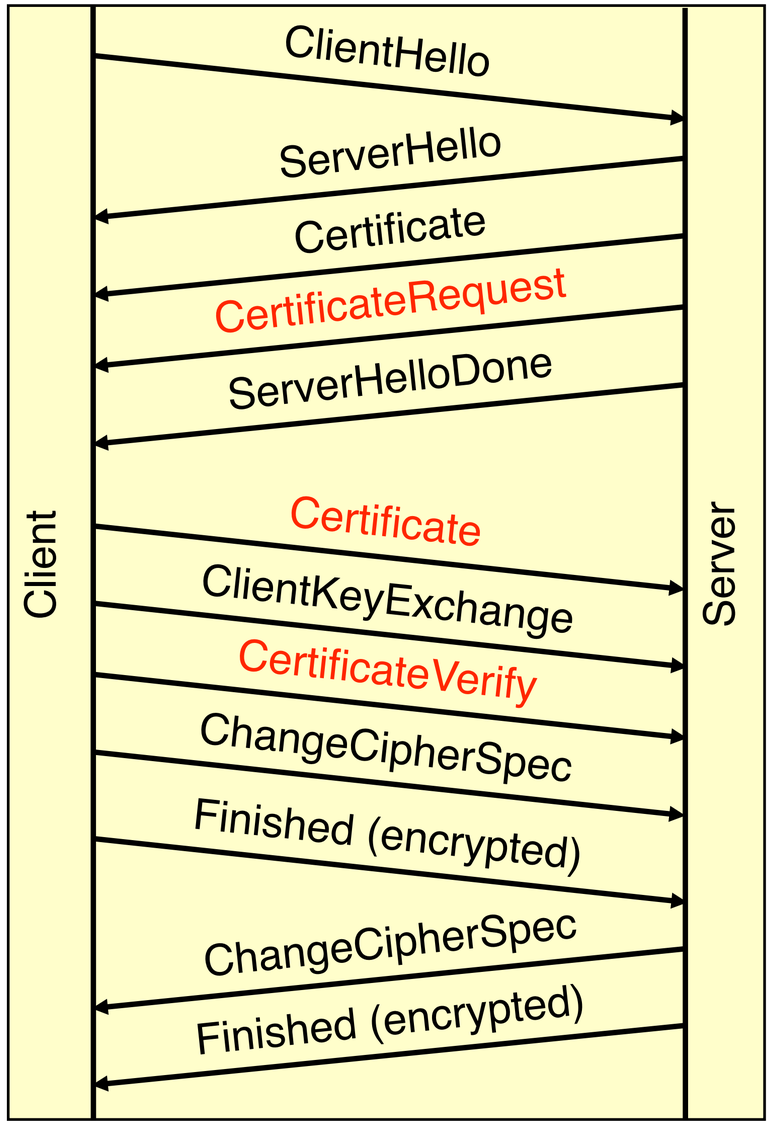


No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.3.1	192.168.3.3	TCP	42370 > https [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=1
2	0.000577	192.168.3.3	192.168.3.1	TCP	https > 42370 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=146
3	0.000618	192.168.3.1	192.168.3.3	TCP	42370 > https [ACK] Seq=1 Ack=1 Win=128000 Len=0
4	0.026109	192.168.3.1	192.168.3.3	SSL	Client Hello
5	0.026465	192.168.3.3	192.168.3.1	TCP	https > 42370 [ACK] Seq=1 Ack=107 Win=5840 Len=0
6	0.070925	192.168.3.3	192.168.3.1	TLSv1	Server Hello,
7	0.071108	192.168.3.3	192.168.3.1	TLSv1	Certificate, Server Key Exchange, Server Hello Done
8	0.071172	192.168.3.1	192.168.3.3	TCP	42370 > https [ACK] Seq=107 Ack=2828 Win=128000 Len=0
9	0.090279	192.168.3.1	192.168.3.3	TLSv1	Client Key Exchange, Change Cipher Spec, Encrypted Handshak
10	0.090657	192.168.3.3	192.168.3.1	TCP	https > 42370 [ACK] Seq=2828 Ack=305 Win=6912 Len=0
11	0.110494	192.168.3.3	192.168.3.1	TLSv1	Change Cipher Spec, Encrypted Handshake Message

ServerKeyExchange


```
Secure Socket Layer
  TLSv1 Record Layer: Handshake Protocol: Certificate
  TLSv1 Record Layer: Handshake Protocol: Server Key Exchange
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 397
    Handshake Protocol: Server Key Exchange
        Handshake Type: Server Key Exchange (12)
        Length: 393
  TLSv1 Record Layer: Handshake Protocol: Server Hello Done
```

Client Authentication



No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.3.1	192.168.3.4	TCP	14980 > https [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=1
2	0.000372	192.168.3.4	192.168.3.1	TCP	https > 14980 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=14(
3	0.000400	192.168.3.1	192.168.3.4	TCP	14980 > https [ACK] Seq=1 Ack=1 Win=128000 Len=0
4	0.015645	192.168.3.1	192.168.3.4	SSLv2	Client Hello
5	0.015824	192.168.3.4	192.168.3.1	TCP	https > 14980 [ACK] Seq=1 Ack=52 Win=5840 Len=0
6	0.017894	192.168.3.4	192.168.3.1	SSLv3	Server Hello,
7	0.017988	192.168.3.4	192.168.3.1	SSLv3	Certificate, Certificate Request, Server Hello Done
8	0.018015	192.168.3.1	192.168.3.4	TCP	14980 > https [ACK] Seq=52 Ack=2590 Win=128000 Len=0
9	4.089191	192.168.3.1	192.168.3.4	TCP	[TCP segment of a reassembled PDU]
10	4.089622	192.168.3.4	192.168.3.1	TCP	https > 14980 [ACK] Seq=2590 Ack=1512 Win=8768 Len=0
11	4.089949	192.168.3.1	192.168.3.4	SSLv3	Certificate, Client Key Exchange, Certificate Verify, Chan(
12	4.107141	192.168.3.4	192.168.3.1	SSLv3	Change Cipher Spec, Encrypted Handshake Message

CertificateRequest (server)


```
Secure Socket Layer
  SSLv3 Record Layer: Handshake Protocol: Certificate
  SSLv3 Record Layer: Handshake Protocol: Multiple Handshake Messages
      Content Type: Handshake (22)
      Version: SSL 3.0 (0x0300)
      Length: 167
    Handshake Protocol: Certificate Request
        Handshake Type: Certificate Request (13)
        Length: 159
        Certificate types count: 2
      Certificate types (2 types)
          Certificate type: RSA Sign (1)
          Certificate type: DSS Sign (2)
        Distinguished Names Length: 154
      Distinguished Names (154 bytes)
          Distinguished Name Length: 152
        Distinguished Name: ()
          RDNSequence: 1 item ()
            RelativeDistinguishedName
                Id: 2.5.4.3 (id-at-commonName)
              DirectoryString: printableString (1)
                  printableString: Sharkfest Lab Root CA
          RDNSequence: 1 item ()
          RDNSequence: 1 item ()
          RDNSequence: 1 item ()
          RDNSequence: 1 item ()
          RDNSequence: 1 item ()
    Handshake Protocol: Server Hello Done
```

Certificate (client)


```
Secure Socket Layer
  SSLv3 Record Layer: Handshake Protocol: Multiple Handshake Messages
      Content Type: Handshake (22)
      Version: SSL 3.0 (0x0300)
      Length: 2579
    Handshake Protocol: Certificate
        Handshake Type: Certificate (11)
        Length: 2309
        Certificates Length: 2306
      Certificates (2306 bytes)
          Certificate Length: 1060
        Certificate ()
          Certificate Length: 1240
        Certificate ()
    Handshake Protocol: Client Key Exchange
    Handshake Protocol: Certificate Verify
  SSLv3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  SSLv3 Record Layer: Handshake Protocol: Encrypted Handshake Message
```

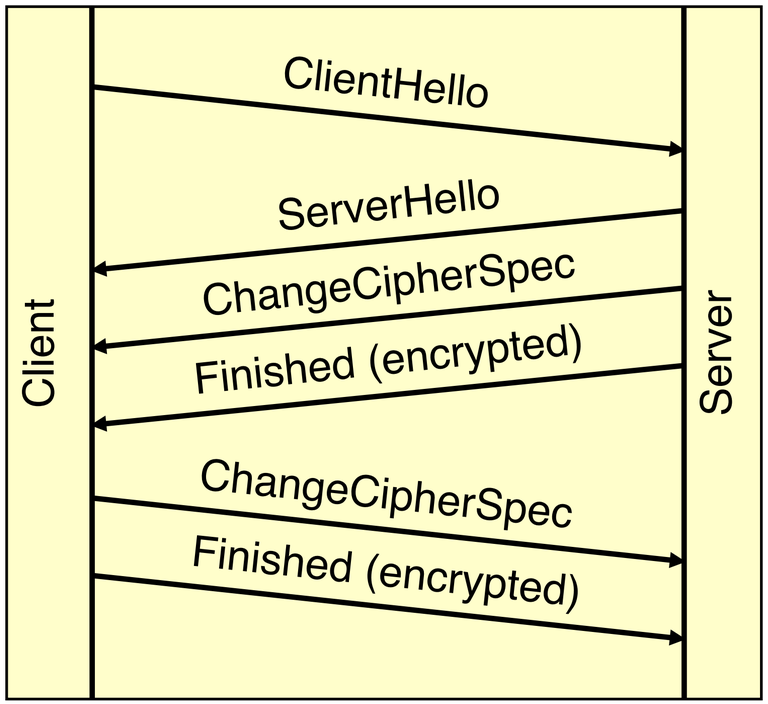
CertificateVerify (client)


```
Secure Socket Layer
  SSLv3 Record Layer: Handshake Protocol: Multiple Handshake Messages
      Content Type: Handshake (22)
      Version: SSL 3.0 (0x0300)
      Length: 2579
    Handshake Protocol: Certificate
    Handshake Protocol: Client Key Exchange
    Handshake Protocol: Certificate Verify
        Handshake Type: Certificate Verify (15)
        Length: 130
  SSLv3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  SSLv3 Record Layer: Handshake Protocol: Encrypted Handshake Message
```

Reusing SSL sessions

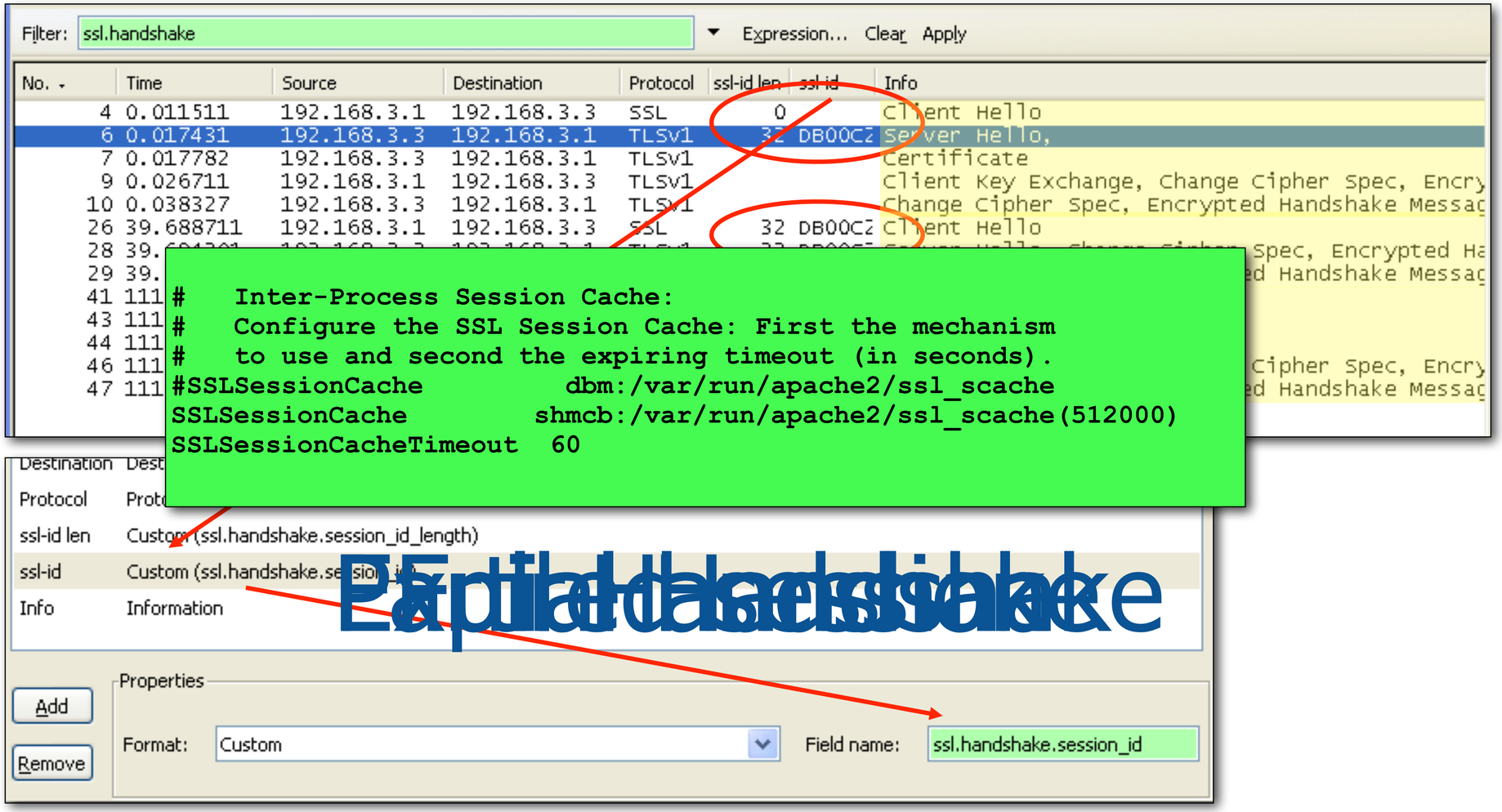
- Key negotiation "expensive"
- Cache session keys and re-use for new TCP sessions
- SSL session ID is used as Index
- Timeout on SSL session ID is an "absolute timeout" not an "idle timeout"
  - Old IE: 2 minutes, now 10 hours

Handshake of a Reused Session



No.	Time	Source	Destination	Protocol	Info
23	39.68772(	192.168.3.1	192.168.3.3	TCP	18774 > https [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=1
24	39.68810)	192.168.3.3	192.168.3.1	TCP	https > 18774 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 WS=
25	39.68814(	192.168.3.1	192.168.3.3	TCP	18774 > https [ACK] Seq=1 Ack=1 Win=128000 Len=0
26	39.68871)	192.168.3.1	192.168.3.3	TLSV1	Client Hello
27	39.68898:	192.168.3.3	192.168.3.1	TCP	https > 18774 [ACK] Seq=1 Ack=103 Win=5840 Len=0
28	39.69430)	192.168.3.3	192.168.3.1	TLSV1	Server Hello, Change Cipher Spec, Encrypted Handshake Message
29	39.71735(	192.168.3.1	192.168.3.3	TLSV1	Change Cipher Spec, Encrypted Handshake Message, Application Dat

SSL session reuse (new, reused and expired)



No SSL session reuse

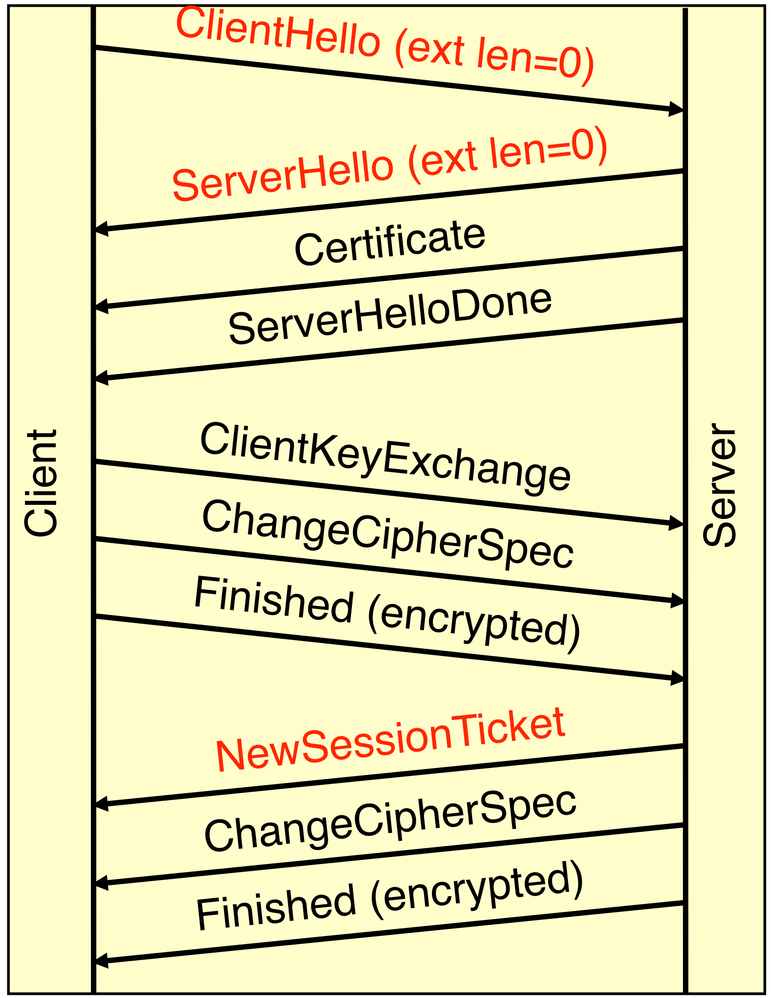
No.	Time	Source	Destination	Protocol	ssl-id len	ssl-id	Info
4	0.011833	192.168.3.1	192.168.3.3	TLSv1	32	5186BC	Client Hello
6	0.018800	192.168.3.3	192.168.3.1	TLSv1	0		Server Hello,
7	0.019128	192.168.3.3	192.168.3.1	TLSv1			Certificate
9	0.026392	192.168.3.1	192.168.3.3	TLSv1			Client Key Exchange, Change Cipher Spec, Encryp
10	0.037500	192.168.3.3	192.168.3.1	TLSv1			Change Cipher Spec, Encrypted Handshake Message

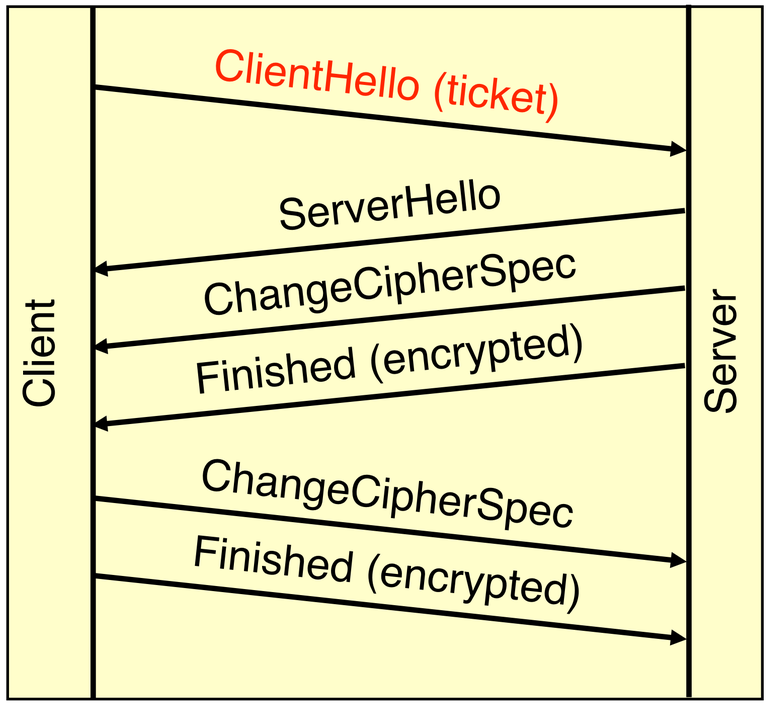
```
⊞ Frame 6 (1514 bytes on wire, 1514 bytes captured)
⊞ Ethernet II, Src: Vmware_5d:c5:66 (00:0c:29:5d:c5:66), Dst: Vmware_c0:00:01 (00:50:56:c0:00:01)
⊞ Internet Protocol, Src: 192.168.3.3 (192.168.3.3), Dst: 192.168.3.1 (192.168.3.1)
⊞ Transmission Control Protocol, Src Port: https (443), Dst Port: 17788 (17788), Seq: 1, Ack: 103, Len: 1460
⊟ Secure Socket Layer
  ⊟ TLSv1 Record Layer: Handshake Protocol: Server Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 42
    ⊟ Handshake Protocol: Server Hello
        Handshake Type: Server Hello (2)
        Length: 38
        Version: TLS 1.0 (0x0301)
      ⊞ Random
        Session ID Length: 0
        Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0x0035)
        Compression Method: null (0)
```

TLS session Tickets

- TLS session tickets (RFC 5077)
- Do not keep state on server, only on client
- TLS extension in ClientHello and ServerHello
- New SSL HandshakeType: **NewSessionTicket**

TLS session Tickets (2)





TLS session Tickets

4 0.015145	192.168.1.22	74.125.132.19	TLSv1	164	Client Hello
6 0.032365	74.125.132.19	192.168.1.22	TLSv1	1484	Server Hello
7 0.032767	74.125.132.19	192.168.1.22	TLSv1	350	Certificate, Server Hello Done
9 0.033752	192.168.1.22	74.125.132.19	TLSv1	252	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
10 0.051951	74.125.132.19	192.168.1.22	TLSv1	292	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
22 2.363423	192.168.1.22	74.125.132.19	TLSv1	360	Client Hello
26 2.383264	74.125.132.19	192.168.1.22	TLSv1	199	Server Hello, Change Cipher Spec, Encrypted Handshake Message

```
▽ Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 89
    Version: TLS 1.0 (0x0301)
```

```
▽ Handshake Protocol: Server Hello
    Handshake Type: Server Hello (2)
    Length: 53
    Version: TLS 1.0 (0x0301)
  ▷ Random
    Session ID Length: 0
    Cipher Suite: TLS_RSA_WITH_RC4_128_SHA
    Compression Method: null (0)
    Extensions Length: 13
  ▷ Extension: server_name
  ▷ Extension: renegotiation_info
  ▽ Extension: SessionTicket TLS
```

```
▽ Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 285
    Version: TLS 1.0 (0x0301)
  ▷ Random
    Session ID Length: 32
    Session ID: 73d2a649be4542fefe7b5cb6f4b15b5a48ae87f7597390b0...
    Cipher Suites Length: 20
  ▷ Cipher Suites (10 suites)
    Compression Methods Length: 1
  ▷ Compression Methods (1 method)
    Extensions Length: 192
  ▷ Extension: server_name
  ▽ Extension: SessionTicket TLS
      Type: SessionTicket TLS (0x0023)
      Length: 164
      Data (164 bytes)
```

```
▽ TLSv1 Record Layer: Handshake Protocol
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 174
  ▽ Handshake Protocol: New Session Ticket
      Handshake Type: New Session Ticket (4)
      Length: 170
```

Analyzing SSL Alerts

Without decryption:

```
14 12.494568  192.168.3.1   192.168.3.3   TLSv1    Application Data
15 12.495834  192.168.3.3   192.168.3.1   TLSv1    Application Data, Application Data
17 27.530927  192.168.3.3   192.168.3.1   TLSv1    Encrypted Alert
20 32.811207  192.168.3.1   192.168.3.3   TLSv1    Encrypted Alert
```

```
Secure Socket Layer
  TLSv1 Record Layer: Encrypted Alert
      Content Type: Alert (21)
      Version: TLS 1.0 (0x0301)
      Length: 32
      Alert Message: Encrypted Alert
```

With decryption:

```
14 12.494568  192.168.3.1   192.168.3.3   HTTP     GET / HTTP/1.1
15 12.495834  192.168.3.3   192.168.3.1   HTTP     HTTP/1.1 200 OK  (text/html)
17 27.530927  192.168.3.3   192.168.3.1   TLSv1    Alert (Level: Warning, Description: Close Notify)
20 32.811207  192.168.3.1   192.168.3.3   TLSv1    Alert (Level: Warning, Description: Close Notify)
```

```
Secure Socket Layer
  TLSv1 Record Layer: Alert (Level: Warning, Description: Close Notify)
      Content Type: Alert (21)
      Version: TLS 1.0 (0x0301)
      Length: 32
    Alert Message
        Level: Warning (1)
        Description: Close Notify (0)
```

Analyzing SSL Application Data

11	0.040173	192.168.3.1	192.168.3.3	TLSv1	491	Application Data
12	0.042446	192.168.3.3	192.168.3.1	TLSv1	496	Application Data, Application Data
14	12.494568	192.168.3.1	192.168.3.3	TLSv1	491	Application Data
15	12.495834	192.168.3.3	192.168.3.1	TLSv1	496	Application Data, Application Data
29	39.717354	192.168.3.1	192.168.3.3	TLSv1	550	Change Cipher Spec, Encrypted Handshake Message, Application Data
30	39.720262	192.168.3.3	192.168.3.1	TLSv1	496	Application Data, Application Data
48	111.230987	192.168.3.1	192.168.3.3	TLSv1	491	Application Data
49	111.233419	192.168.3.3	192.168.3.1	TLSv1	496	Application Data, Application Data

```
▽ Secure Sockets Layer
  ▽ TLSv1 Record Layer: Application Data Protocol: http
      Content Type: Application Data (23)
      Version: TLS 1.0 (0x0301)
      Length: 384
      Encrypted Application Data: 94c662e11c5c01813955dfc675754583ab4a70d65fddf8e9...
  ▽ TLSv1 Record Layer: Application Data Protocol: http
      Content Type: Application Data (23)
      Version: TLS 1.0 (0x0301)
      Length: 48
      Encrypted Application Data: 635e2a228ddc1aa5d7a2a89c809e6e693ec01f4cf5746fee...
```

Elliptic Curve Cryptology

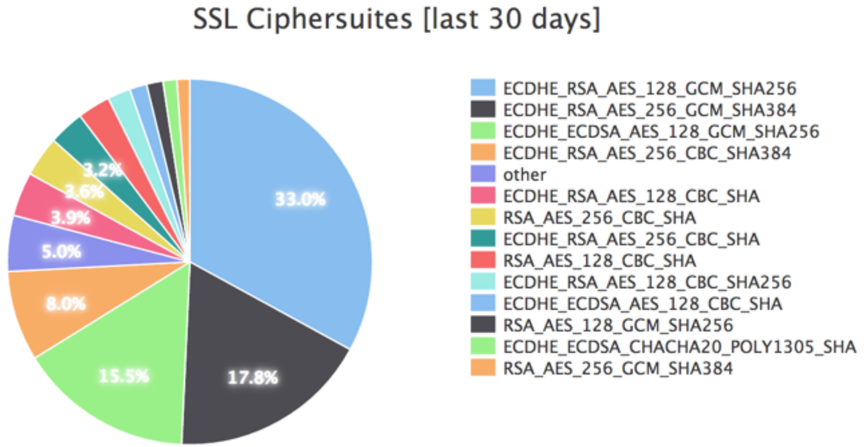
RSA versus ECC

- Smaller key sizes
  - less CPU cycles needed
  - less memory needed
  - smaller PDU's (less traffic)

Symmetric	ECC	DH/DSA/RSA
80	163	1024
112	233	2048
128	283	3072
192	409	7680
256	571	15360

- Good for mobile devices

- Most TLS connections now use a cipher with ECDHE key generation



- Of those connections, most still use a certificate with a RSA public key

ECDHE_ECDSA

- Server's certificate MUST contain an ECDSA-capable public key and be signed with ECDSA.
- Server sends its ephemeral ECDH public key and a specification of the corresponding curve in the ServerKeyExchange message. These parameters MUST be signed with ECDSA using the private key corresponding to the public key in the server's Certificate.
- The client generates an ECDH key pair on the same curve as the server's ephemeral ECDH key and sends its public key in the ClientKeyExchange message.
- Both client and server perform an ECDH operation and use the resultant shared secret as the premaster secret.

ECDHE_RSA

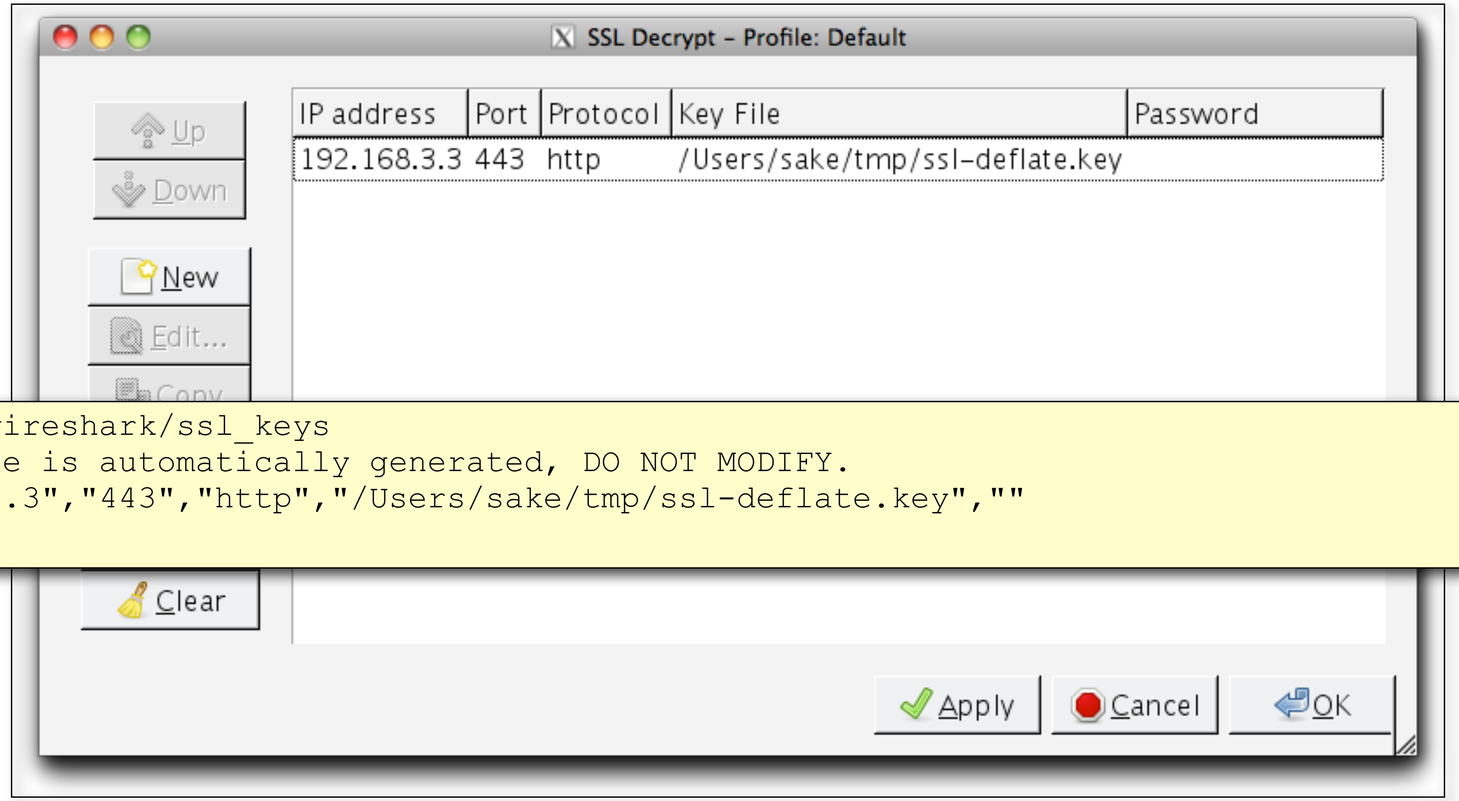
- This key exchange algorithm is the same as ECDHE_ECDSA except that :
- the server's certificate MUST contain an RSA public key authorized for signing
- and that the signature in the ServerKeyExchange message must be computed with the corresponding RSA private key
- The server certificate MUST be signed with RSA.

Decrypting SSL/TLS

Decrypting SSL traffic

- With RSA key exchange
  - Provide **server** private key to Wireshark
  - Only works when whole session (including **full handshake**) is in the tracefile
  - Also works with Client Authentication
- With ephemeral DH key generation (ServerKeyExchange present)
  - Use (pre-)masterkey logfile
  - Set SSLKEYLOGFILE environment variable (Firefox/Chrome)
- Exporting SSL session keys
  - File -> Export SSL session keys…

Providing the server private key




```
$ cat ~/.wireshark/ssl_keys
# This file is automatically generated, DO NOT MODIFY.
"192.168.3.3","443","http","/Users/sake/tmp/ssl-deflate.key",""
$
```

SSL debug logfile

SSL debug log:

```
ssl_init keys string:
192.168.3.3,443,http,c:\temp\public.sharkfest.local.key
ssl_init found host entry 192.168.3.3,443,http,c:\temp\public.sharkfest.local.key
ssl_init addr '192.168.3.3' port '443' filename 'c:\temp\public.sharkfest.local.key' password(only for p12 file) '(null)'
ssl_load_key: can't import pem data
```

PEM keyfile *without* passphrase:

```
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQDrHdbb+yGE6m6EZ03bXURpZCjch2H6g97ZAkJVGrjLZFfettBA
EYa8vYYxWsf8KBpEZeksSCsDA9MnU2H6QDjzqdOnaSWfeXMAr4OsCOpauStpreq7
q1hk8iOqy+f4KijRrhWplh1QW1A8gtSIg137pyUhW+WsfwxKwmzjGIC1SwIDAQAB
AoGBAMneA9U6KIxjb+JUg/99c7h9W6wEvTYHNTXjf6psWA+hpuQ82E65/ZJdszL6
...
b6QKMh16r5wd6smQ+CmhOEnqqyT5AIwwl2RIr9GbfIpTbtbRQw/EcQOCx9wFiEfo
tGSsEFi72rHK+DpJqRI9AkEA72gdyXRgPfGOS3rfQ3DBcImBQvDSCBa4cuU1XJ1/
MO93a8v9Vj87/yDm4xsBDsoz2PyBepawHVlIvZ6jDD0aXw==
-----END RSA PRIVATE KEY-----
```

PEM keyfile *with* passphrase:

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,F6C218D4FA3C8B66

FR2cnmkkFHH45Dcsty1qDiIUy/uXn+9m/xeQMVRxtiSAmBmnUDUFIFCDDiDc9yif
ERok2jPr2BzAazl5RBxS2TY/+7x0/dHD11sF3LnJUoNruo77TERxqgzOI0W1VDRA
...
ygw5JslxgiN18F36E/cEP5rKvVYvfEPMa6IsiRhfZk1jLAuZihVWc7JodDf+6RKV
yBXrK/bDtdEih+bOnYu+ZDvjAzVz9GhggCW4QHNboDpTxrrYPkj5Nw==
-----END RSA PRIVATE KEY-----
```

Converting keys

Removing passphrase:

```
root@mgmt# openssl rsa -in encrypted.key -out cleartext.key
Enter pass phrase for encrypted.key: <passphrase>
writing RSA key
root@mgmt#
```

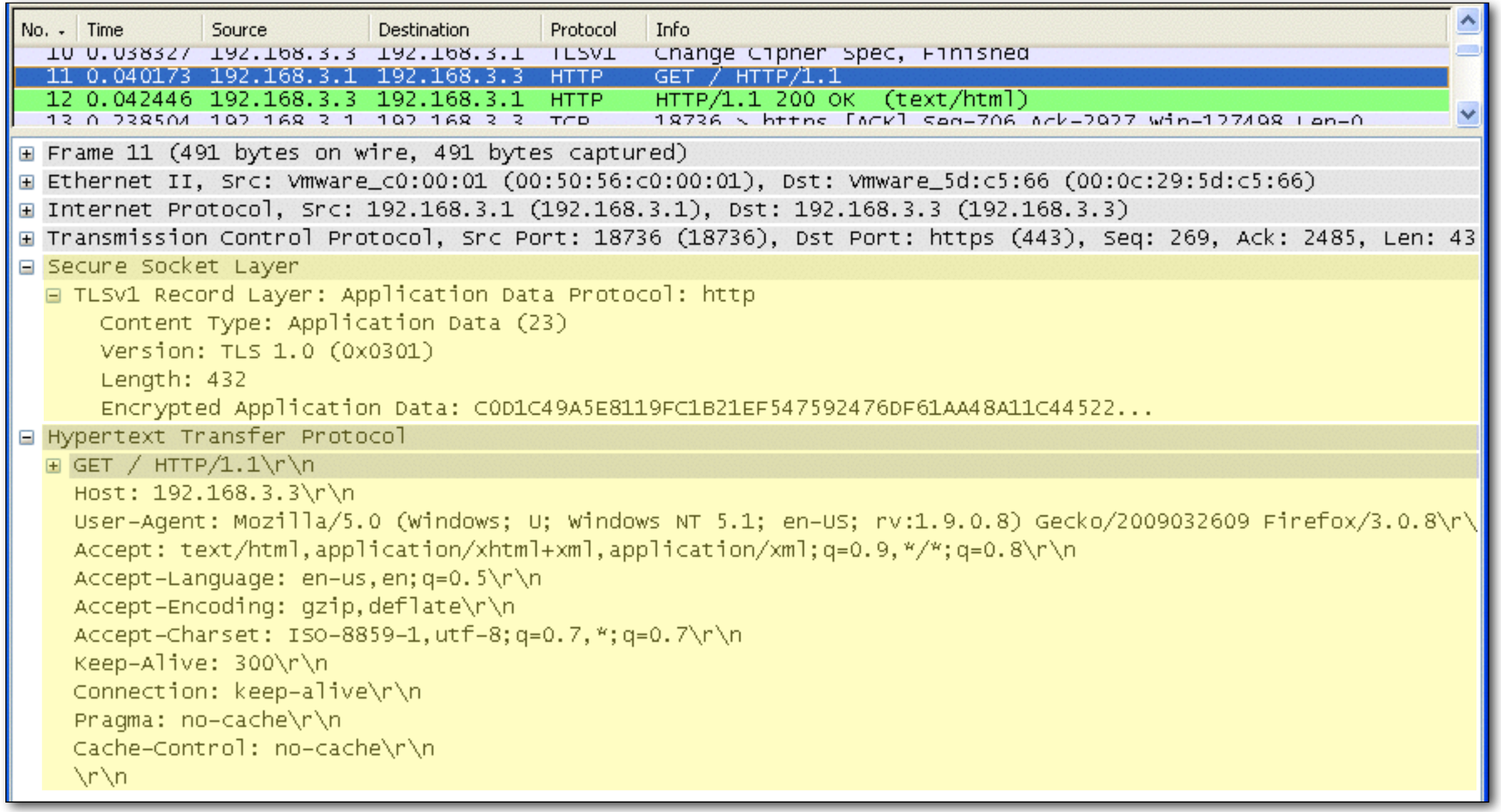
Converting from DER to PEM (and removing passphrase):

```
root@mgmt# openssl rsa -inform DER -in der.key -out pem.key
Enter pass phrase for encrypted.key: <passphrase>
writing RSA key
root@mgmt#
```

Converting from PEM to PKCS12 (and adding passphrase):

```
root@mgmt# openssl pkcs12 -in pem.cert -inkey pem.key -export -out cert.pkcs12
Enter Export Password: <new-passphrase>
Verifying - Enter Export Password: <new-passphrase>
root@mgmt#
```

Decryption in Action



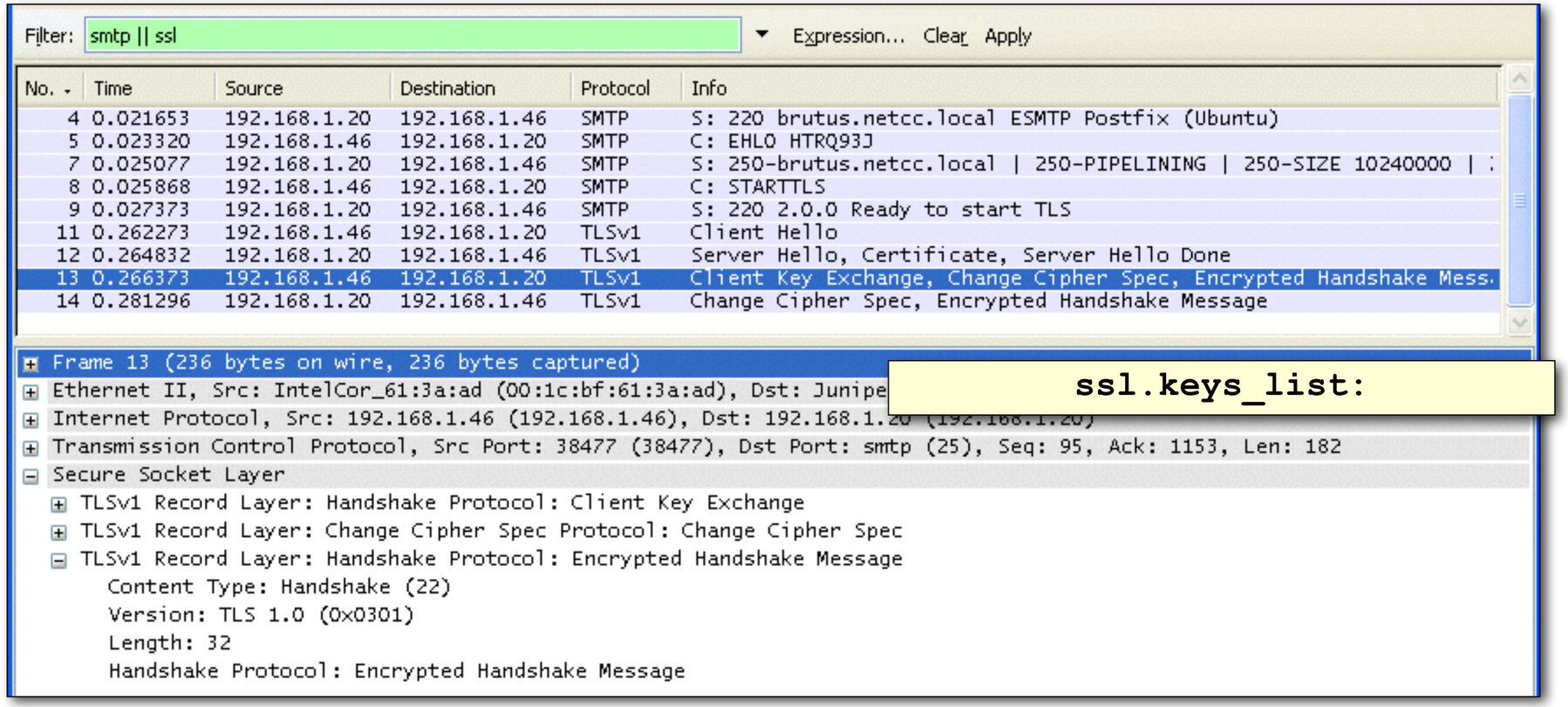
Decrypting IMAPS

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.1.46	192.168.1.20	TCP	22446 > imaps [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=1
2	0.001820	192.168.1.20	192.168.1.46	TCP	imaps > 22446 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 WS=7
3	0.001857	192.168.1.46	192.168.1.20	TCP	22446 > imaps [ACK] Seq=1 Ack=1 Win=128000 Len=0
4	0.010231	192.168.1.46	192.168.1.20	SSL	Client Hello
5	0.011625	192.168.1.20	192.168.1.46	TCP	imaps > 22446 [ACK] Seq=1 Ack=103 Win=5888 Len=0
6	0.012351	192.168.1.20	192.168.1.46	TLSv1	Server Hello, Certificate, Server Hello Done
7	0.013831	192.168.1.46	192.168.1.20	TLSv1	Client Key Exchange, Change Cipher Spec, Finished
8	0.019822	192.168.1.20	192.168.1.46	TLSv1	Change Cipher Spec, Finished
9	0.168748	192.168.1.46	192.168.1.20	TCP	22446 > imaps [ACK] Seq=285 Ack=978 Win=127022 Len=0
10	0.170301	192.168.1.20	192.168.1.46	IMAP	Response: * OK Dovecot ready.
11	0.172574	192.168.1.46	192.168.1.20	IMAP	Request: g7fg CAPABILITY

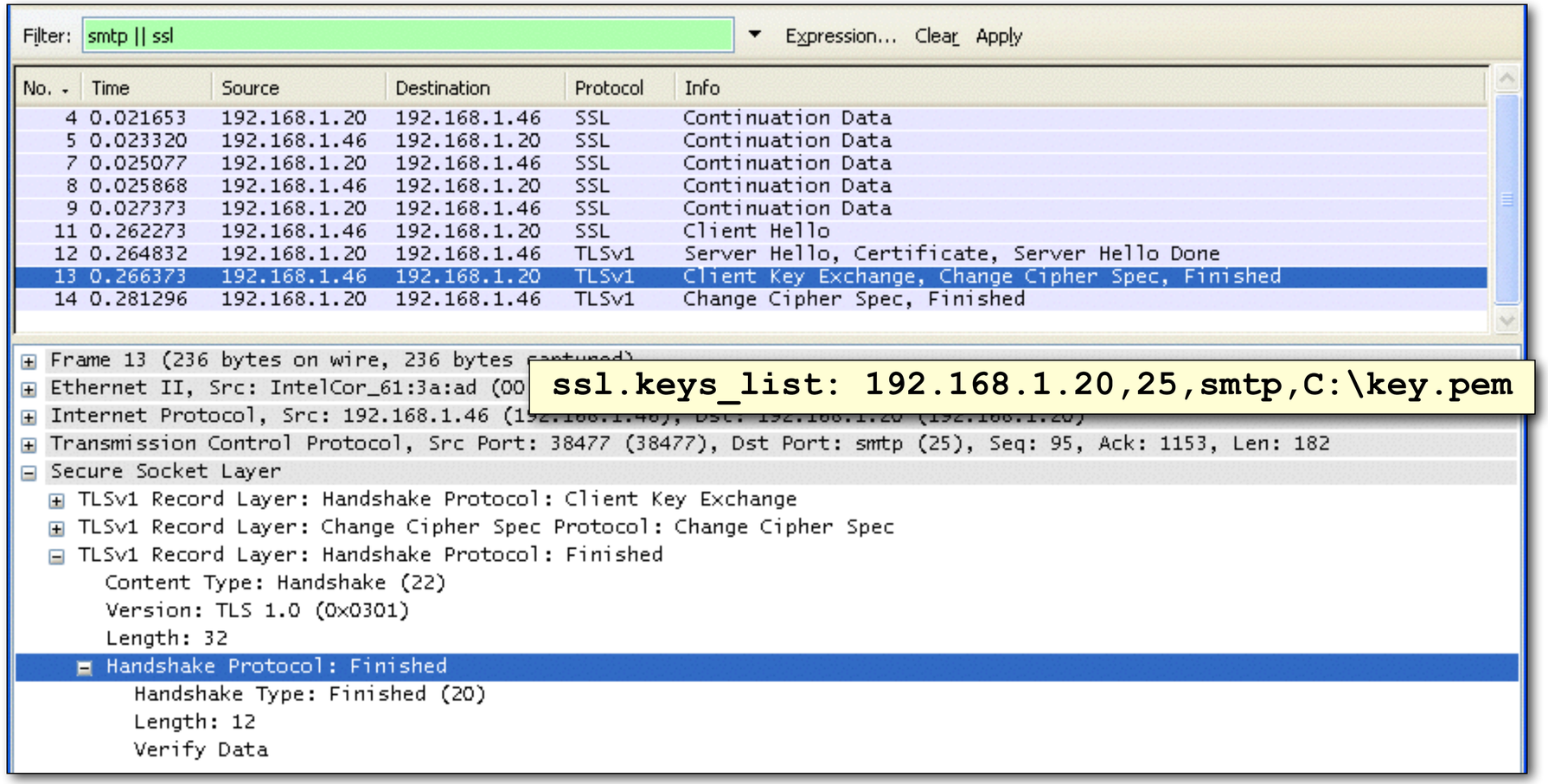
```
⊞ Frame 10 (96 bytes on wire, 96 bytes captured)
⊞ Ethernet II, Src: JuniperN_bb:d1:3b (0
⊞ Internet Protocol, Src: 192.168.1.20 (192.168.1.20), Dst: 192.168.1.46 (192.168.1.46)
⊞ Transmission Control Protocol, Src Port: imaps (993), Dst Port: 22446 (22446), Seq: 978, Ack: 285, Len: 42
⊟ Secure Socket Layer
  ⊟ TLSv1 Record Layer: Application Data Protocol: imap
      Content Type: Application Data (23)
      Version: TLS 1.0 (0x0301)
      Length: 37
      Encrypted Application Data: F8260B9E9D0597A3CE35E176BA3EDB28D588E004F6B57F74...
⊟ Internet Message Access Protocol
  ⊟ * OK Dovecot ready.\r\n
      Response Tag: *
      Response: OK Dovecot ready.
```

```
ssl.keys_list: 192.168.1.20,993,imap,C:\key.pem
```

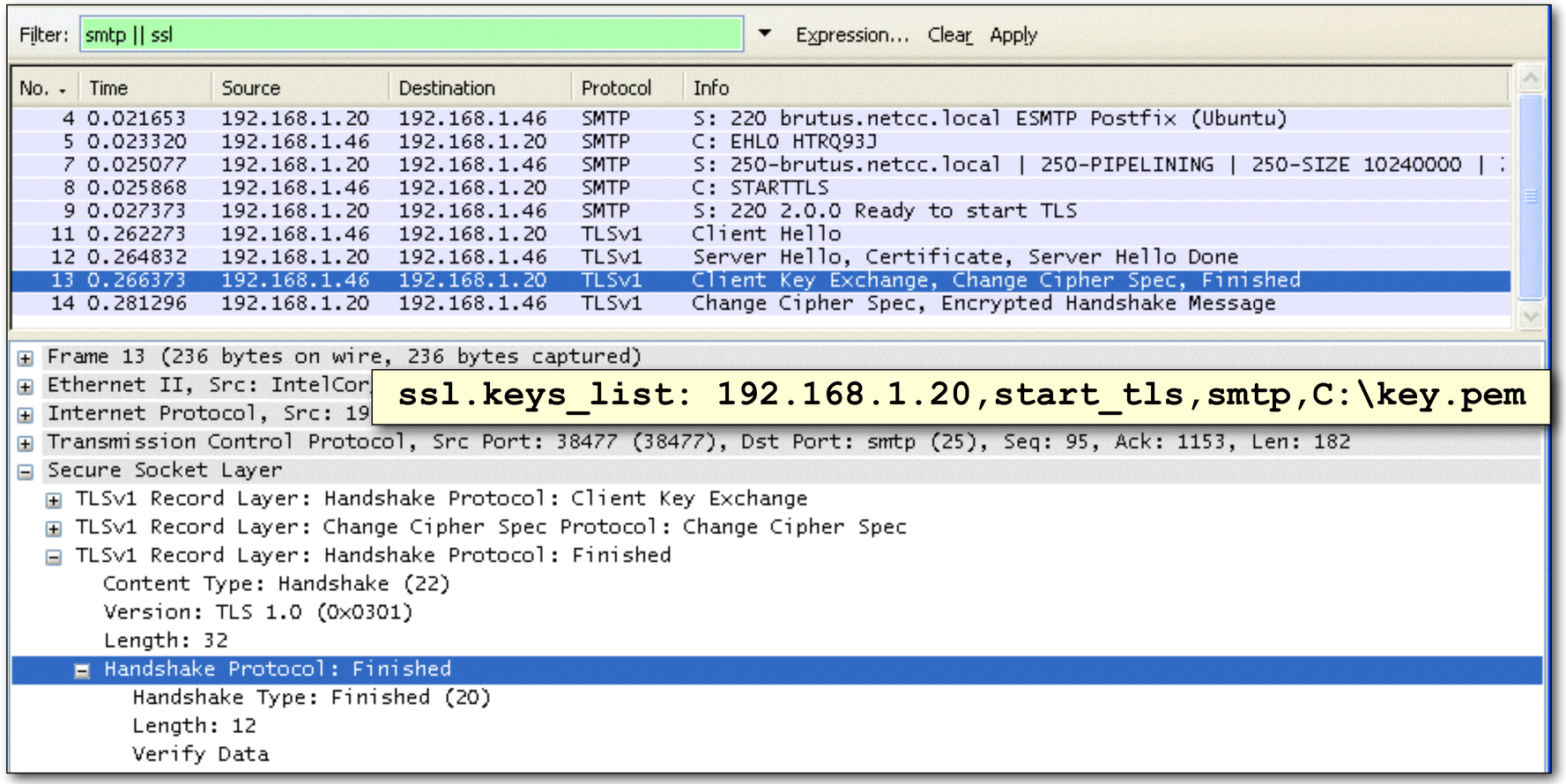
Decrypting "STARTTLS" (1)



Decrypting "STARTTLS" (2)



Decrypting "STARTTLS" (3)



Decrypt-problem I (1)

No.	Time	Source	Destination	Protocol	Info
1	0.000000	192.168.3.1	192.168.3.3	TCP	18774 > https [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=1
2	0.000375	192.168.3.3	192.168.3.1	TCP	https > 18774 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 WS=4
3	0.000423	192.168.3.1	192.168.3.3	TCP	18774 > https [ACK] Seq=1 Ack=1 Win=128000 Len=0
4	0.000985	192.168.3.1	192.168.3.3	SSL	Client Hello
5	0.001257	192.168.3.3	192.168.3.1	TCP	https > 18774 [ACK] Seq=1 Ack=103 Win=5840 Len=0
6	0.006575	192.168.3.3	192.168.3.1	TLSv1	Server Hello, Change Cipher Spec, Encrypted Handshake Message
7	0.029628	192.168.3.1	192.168.3.3	TLSv1	Change Cipher Spec, Encrypted Handshake Message, Application Data
8	0.032536	192.168.3.3	192.168.3.1	TLSv1	Application Data, Application Data

```
⊞ Frame 7 (550 bytes on wire, 550 bytes captured)
⊞ Ethernet II, Src: Vmware_c0:00:01 (00:50:56:c0:00:01), Dst: Vmware_5d:c5:66 (00:0c:29:5d:c5:66)
⊞ Internet Protocol, Src: 192.168.3.1 (192.168.3.1), Dst: 192.168.3.3 (192.168.3.3)
⊞ Transmission Control Protocol, Src Port: 18774 (18774), Dst Port: https (443), Seq: 103, Ack: 139, Len: 496
⊟ Secure Socket Layer
  ⊞ TLSv1 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  ⊟ TLSv1 Record Layer: Handshake Protocol: Encrypted Handshake Message
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 48
      Handshake Protocol: Encrypted Handshake Message
  ⊞ TLSv1 Record Layer: Application Data Protocol: http
```

```
ssl_init keys string:
192.168.3.3,443,http,c:\temp\public.sharkfest.local.key
ssl_init found host entry 192.168.3.3,443,http,c:\temp\public.sharkfest.local.key
ssl_init addr '192.168.3.3' port '443' filename 'c:\temp\public.sharkfest.local.key' password(only for p12 file)
'(null)'
Private key imported: KeyID B8:2B:EA:B8:F8:BD:62:50:E3:0C:2D:3D:06:09:91:64:...
ssl_init private key file c:\temp\public.sharkfest.local.key successfully loaded
association_add TCP port 443 protocol http handle 04086228
```

Decrypt-problem I (2)

Checking ssl debug log:

```
[...]
dissect_ssl enter frame #7 (first time)
  conversation = 07411870, ssl_session = 07411BC8
  record: offset = 0, reported_length_remaining = 496
dissect_ssl3_record: content_type 20
dissect_ssl3_change_cipher_spec
association_find: TCP port 18774 found 00000000
packet_from_server: is from server - FALSE
ssl_change_cipher CLIENT
  record: offset = 6, reported_length_remaining = 490
dissect_ssl3_record: content_type 22
decrypt_ssl3_record: app_data len 48 ssl, state 0x17
association_f
packet_from_s
decrypt_ssl3_
decrypt_ssl3_
dissect_ssl3_
  record: of
dissect_ssl3_
decrypt_ssl3_
association_find: TCP port 18774 found 00000000
packet_from_server: is from server - FALSE
decrypt_ssl3_record: using client decoder
decrypt_ssl3_record: no decoder available
association_find: TCP port 18774 found 00000000
association_find: TCP port 443 found 047AF518
[...]
```

Make sure that the whole SSL session (which can be made out of multiple TCP streams) is in the tracefile. Starting with the handshake and up to the current frame.

Decrypt-problem II (1)

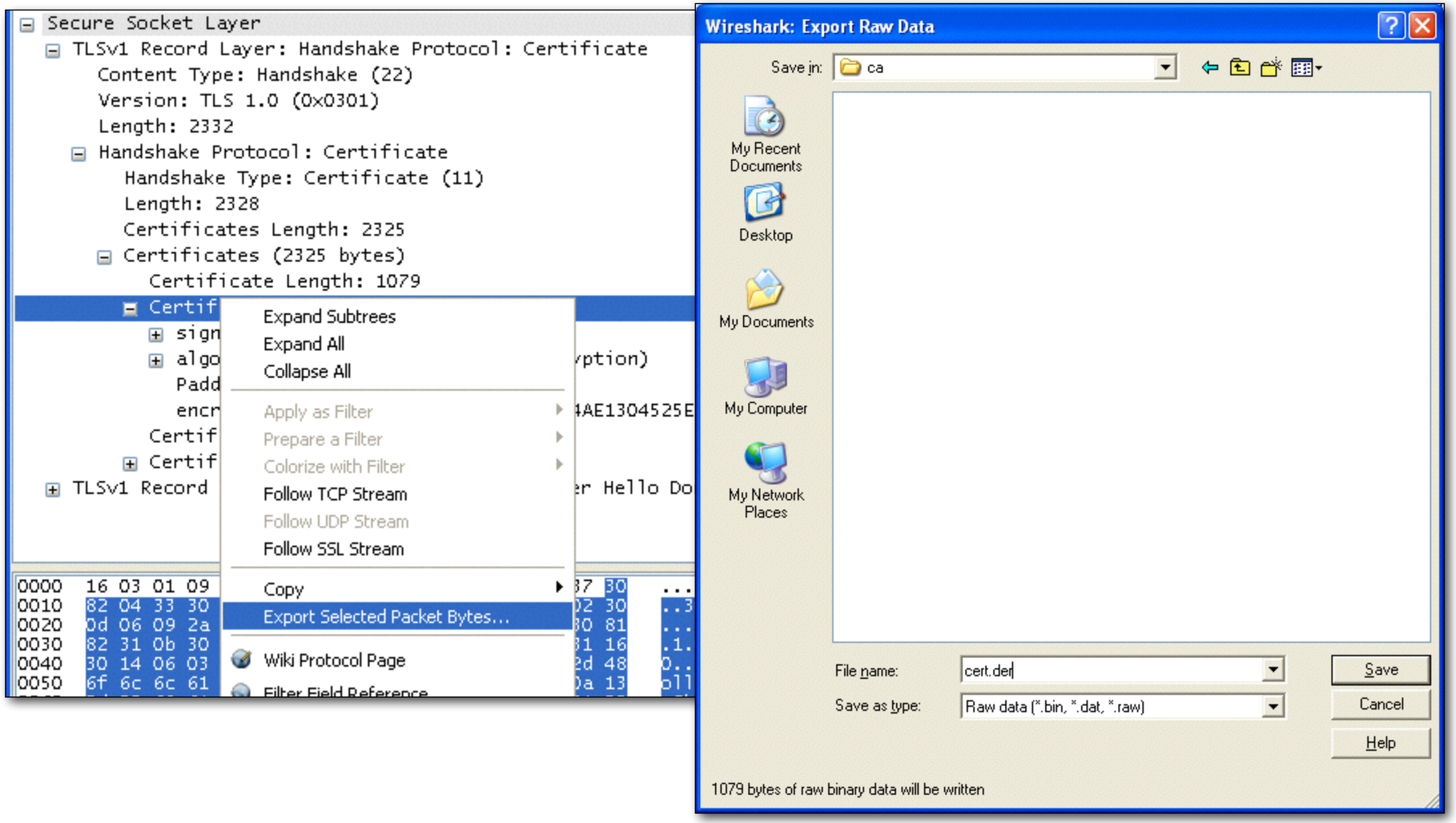
Checking ssl debug log:

```
ssl_association_remove removing TCP 443 - http handle 04086F30
ssl_init keys string:
192.168.3.3,443,http,c:\temp\public.sharkfest.local.key
ssl_init found host entry 192.168.3.3,443,http,c:\temp\public.sharkfest.local.key
ssl_init addr '192.168.3.3' port '443' filename 'c:\temp\public.sharkfest.local.key' password(only for p12 file)
'(null)'
Private key imported: KeyID FA:56:73:A4:38:9C:A1:4F:28:23:88:76:83:42:13:86:...
ssl_init private key file c:\temp\public.sharkfest.local.key successfully loaded
association_add TCP port 443 protocol http handle 04086F30

[...]

ssl_decrypt_pre_master_secret:RSA_private_decrypt
pcry_private_decrypt: stripping 0 bytes, decr_len zd
decrypted_unstrip_pre_master[128]:
6a f7 2a 4b 45 17 72 47 c2 11 d1 dd ad dc af b6
04 76 cb 3c 32 1c d1 01 57 4a 83 79 af d9 40 af
aa a8 71 1f bd 6f 70 d5 cc 49 e6 be 44 42 07 7c
45 b7 5b 5b 52 de 3e 58 d3 42 8d 5f bc 99 3e 13
f5 7d 27 a1 3e 7f b2 3f 8b 9d e5 fb 60 ec 40 26
87 8f 24 41 fb d4 ec f7 0e ea 04 46 c2 d7 5f 7b
4a d2 40 47 07 7b 0d 63 d8 d6 0f e6 9e 98 92 02
58 13 51 72 1b 85 69 04 52 42 74 12 40 e2 a5 bb
ssl_decrypt_pre_master_secret wrong pre_master_secret length (128, expected 48)
dissect_ssl3_handshake can't decrypt pre master secret
```

Decrypt-problem II (2)



Decrypt-problem II (3)

In wireshark preferences:

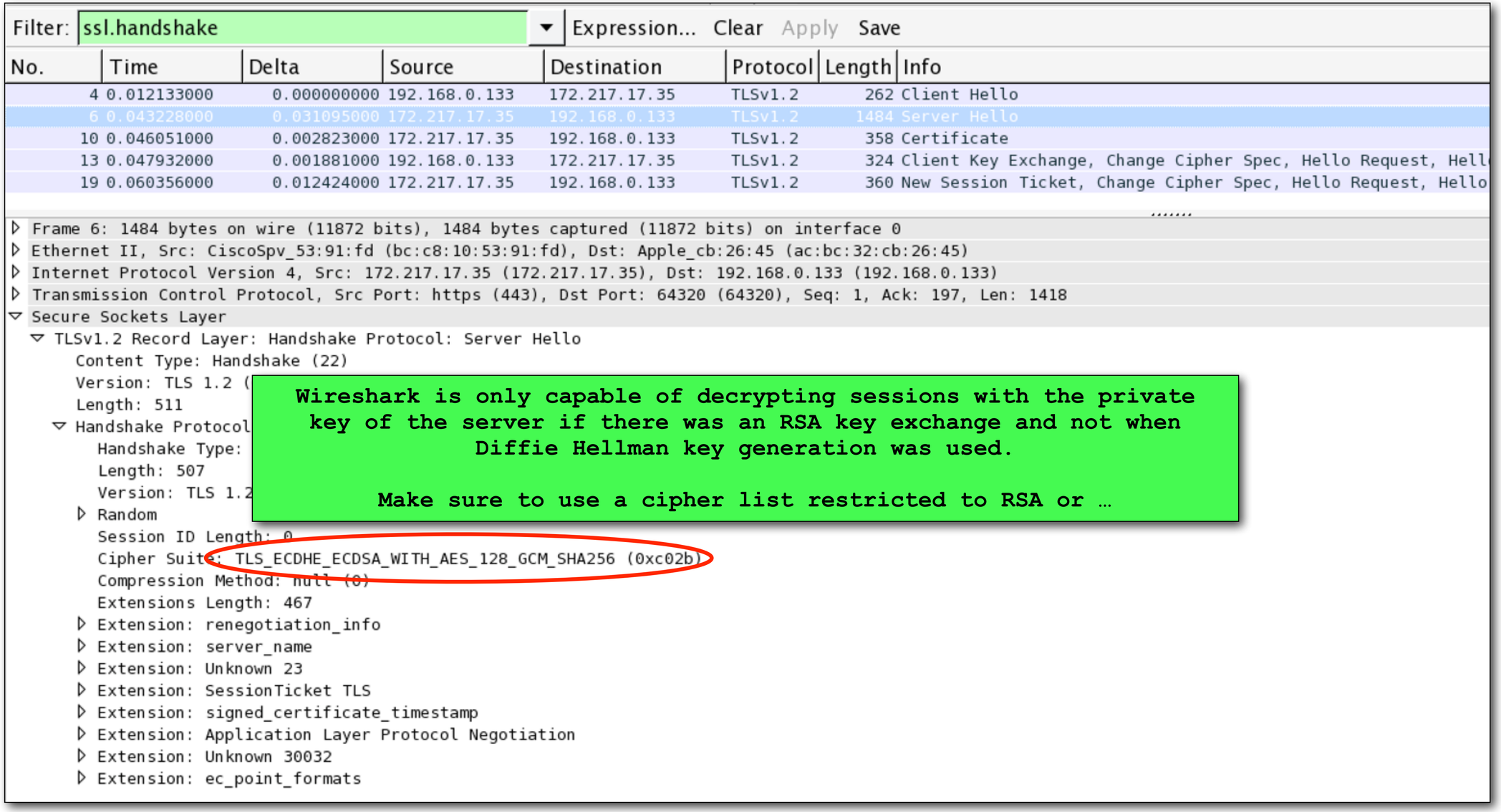
```
ssl.keys_list: 192.168.3.3,443,http,c:\temp\public.sharkfest.local.key
```

Checking whether certificate and key match:

```
$ openssl x509 -in cert.der -inform DER -noout -text | grep "Subject:"
        Subject: C=NL, ST=Noord-Holland, O=Sharkfest Lab, CN=public.sharkfest.local/
emailAddress=co@sharkfest.local
$
$ openssl
a29682af82
$
$ openssl
ce71158d3851a885314c264863142389
$
$ openssl rsa -noout -modulus -in private.sharkfest.local.key | openssl md5
a29682af822b4cd064d39d4ccd1e0e6c
$
```

Make sure that the private key matches the (server) certificate that is used in the tracefile.

Decrypt-problem III



Decryption Without The Private Key

- Using the unencrypted (pre-)Master-Secret
- Use SSLKEYLOGFILE environment variable
  (works for NSS based browsers like Chrome and Firefox)
- Or extract the info from client or server
  (like extracting it from openssl s_client output)
- SSL preferences: (Pre)-Master-Secret log filename

(Pre-)Master Secret log format


```
/* The format of the file is a series of records with one of the following formats:
 *   - "RSA xxxx yyyy"
 *     Where xxxx are the first 8 bytes of the encrypted pre-master secret (hex-encoded)
 *     Where yyyy is the cleartext pre-master secret (hex-encoded)
 *     (this is the original format introduced with bug 4349)
 *
 *   - "RSA Session-ID:xxxx Master-Key:yyyy"
 *     Where xxxx is the SSL session ID (hex-encoded)
 *     Where yyyy is the cleartext master secret (hex-encoded)
 *     (added to support openssl s_client Master-Key output)
 *     This is somewhat is a misnomer because there's nothing RSA specific
 *     about this.
 *
 *   - "PMS_CLIENT_RANDOM xxxx yyyy"
 *     Where xxxx is the client_random from the ClientHello (hex-encoded)
 *     Where yyyy is the cleartext pre-master secret (hex-encoded)
 *     (This format allows SSL connections to be decrypted, if a user can
 *     capture the PMS but could not recover the MS for a specific session
 *     with a SSL Server.)
 *
 *   - "CLIENT_RANDOM xxxx yyyy"
 *     Where xxxx is the client_random from the ClientHello (hex-encoded)
 *     Where yyyy is the cleartext master secret (hex-encoded)
 *     (This format allows non-RSA SSL connections to be decrypted, i.e.
 *     ECDHE-RSA.)
 */
```

SSLKEYLOGFILE


```
sake@MacSake:~/Dropbox/sharkfest/2016eu/src$ export SSLKEYLOGFILE=sslkeylogfile.log
sake@MacSake:~/Dropbox/sharkfest/2016eu/src$ /Applications/Google\ Chrome.app/Contents/MacOS/Google\ Chrome
2016-10-16 11:14:41.752 Google Chrome[35779:31787727] NSWindow warning: adding an unknown subview:
<FullSizeContentView: 0x7fc0d5c6cd10>. Break on NSLog to debug.
2016-10-16 11:14:41.752 Google Chrome[35779:31787727] Call stack:
(
    "+callStackSymbols disabled for performance reasons"
)
sake@MacSake:~/Dropbox/sharkfest/2016eu/src$ head -3 sslkeylogfile.log
CLIENT_RANDOM bd102f67590bfd2b7642872b6525443deb63fbed1584a27be33df503f514ef8e
19dc30ea26672d1d3386be1502c4390272604cf6ce9ba18790570a5136d43d7ec599f857dd1667e52982bfd2212060e4
CLIENT_RANDOM 6bd47de9893eb735d0f4ac7dbaf2627140d52fb82b681dae2ebb3a3f3bd185a4
e665465437ce91f336ec6fc47a3b96363192141e4a3145a292b67fc35b76c63a7ba16c3af796d4809e8e5ce39519b0f7
CLIENT_RANDOM a76f9ab9176ab74770cc77a9fc0cdeceb96c26a189ca36737f170039d78acd3d
6f9fea87caac265aad80be81a38b9763e1fc5e25290bbb654eda769bf97ecaac160d2587c803c1dd98f646fad9280c1c
sake@MacSake:~/Dropbox/sharkfest/2016eu/src$
```

SSLKEYLOGFILE

ssl

No.	Time	Source	Destination	Protocol	Length	Info
4	23:09:17.804825	192.168.0.133	172.217.17.35	TLSv1.2	262	Client Hello
6	23:09:17.835920	172.217.17.35	192.168.0.133	TLSv1.2	1484	Server Hello
10	23:09:17.838743	172.217.17.35	192.168.0.133	TLSv1.2	358	CertificateServer Key Exchange, Server Hello Done
13	23:09:17.840624	192.168.0.133	172.217.17.35	TLSv1.2	324	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message, Finished
14	23:09:17.844191	192.168.0.133	172.217.17.35	HTTP2	119	Magic
15	23:09:17.844192	192.168.0.133	172.217.17.35	HTTP2	116	SETTINGS
16	23:09:17.844231	192.168.0.133	172.217.17.35	HTTP2	108	WINDOW_UPDATE

```
▶ Frame 6: 1484 bytes on wire (11872 bits), 1484 bytes captured (11872 bits) on interface 0
▶ Ethernet II, Src: CiscoSpv_53:91:fd (bc:c8:10:53:91:fd), Dst: Apple_cb:26:45 (ac:bc:32:cb:26:45)
▶ Internet Protocol Version 4, Src: 172.217.17.35, Dst: 192.168.0.133
▶ Transmission Control Protocol, Src Port: 443, Dst Port: 64320, Seq: 1, Ack: 197, Len: 1418
▼ Secure Sockets Layer
  ▼ TLSv1.2 Record Layer: Handshake Protocol: Server Hello
      Content Type: Handshake (22)
      Version: TLS 1.2 (0x0303)
      Length: 511
    ▼ Handshake Protocol: Server Hello
        Handshake Type: Server Hello (2)
        Length: 507
        Version: TLS 1.2 (0x0303)
      ▶ Random
        Session ID Length: 0
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
        Compression Method: null (0)
        Extensions Length: 467
      ▶ Extension: renegotiation_info
      ▶ Extension: server_name
      ▶ Extension: Extended Master Secret
      ▶ Extension: SessionTicket TLS
      ▶ Extension: signed_certificate_timestamp
      ▶ Extension: Application Layer Protocol Negotiation
      ▶ Extension: channel_id
      ▶ Extension: ec_point_formats
```

openssl s_client

```
$ openssl s_client -cipher AES256-SHA -no_ticket -connect imap.syn-bit.nl:993 | tee openssl-s_client.txt
depth=1 C = GB, ST = Greater Manchester, L = Salford, O = COMODO CA Limited, CN = COMODO High-Assurance Secure Server CA
verify error:num=20:unable to get local issuer certificate
verify return:0
CONNECTED(00000003)
---
[...]
SSL-Se
    Pr
    Ci
    Se
    Se
    Ma
    Ke
    PS
    PS
    SR
    Co
    St
    Ti
    Ve
---
* OK I
HELP
* BAD
QUIT
DONE
```

15	0.180186	46.30.211.94	192.168.1.22	IMAP	119	Response: * OK IMAP4 ready
17	2.631302	192.168.1.22	46.30.211.94	IMAP	140	Request: HELP
18	2.669607	46.30.211.94	192.168.1.22	IMAP	135	Response: * BAD invalid command

```
▷ Frame 15: 119 bytes on wire (952 bits), 119 bytes captured (952 bits)
▷ Ethernet II, Src: JuniperN_bb:d1:32 (00:12:1e:bb:d1:32), Dst: Apple_d8:87:48 (f8:1e:df:d8:87:48)
▷ Internet Protocol Version 4, Src: 46.30.211.94 (46.30.211.94), Dst: 192.168.1.22 (192.168.1.22)
▷ Transmission Control Protocol, Src Port: imaps (993), Dst Port: 64400 (64400), Seq: 2965, Ack: 425, Len: 53
▽ Secure Sockets Layer
  ▽ TLSv1 Record Layer: Application Data Protocol: imap
      Content Type: Application Data (23)
      Version: TLS 1.0 (0x0301)
      Length: 48
      Encrypted Application Data: 74a980b1955b4f74c1be949df97da0f4a25f27704ed7b66a...
▽ Internet Message Access Protocol
  ▷ * OK IMAP4 ready\r\n
```

```
$ awk '$1~"Session-ID:" {printf("RSA %s%s ",$1,$2)} $1~"Master-Key:" {printf("%s%s\n",$1,$2)}' openssl-s_client.txt >
openssl-s_client.keys
$ cat openssl-s_client.keys
RSA Session-ID:5EF3E7EDCC46993E51935914ACC1CBE6723259121248F958BC223D54FA84CFA0 Master-Key:
0665121ADB266864CDEF89E32A6F1A39677D540DB5B362BC351D3B08EE3059800F9A218E6601710CE774AFB2CE3166C9
$
```

SSL

STANAG 5066

(Pre)-Master-Secret log filename: 012/traces/openssl-s_client.keys Browse...

Exporting SSL Session Keys

- Export:
  - File -> Export SSL Session Keys
- Import:
  - SSL preferences: (Pre)-Master-Secret log filename
- Provide SSL decryption in Wireshark to a 3rd party without having to share the private key!

Decrypting SSL with Tshark

- -V to show whole tree
  (and decrypted application data)

- Find the SSL preference keys:
  - tshark -G currentprefs | egrep "^#?ssl."

- tshark -o ssl.keys_list:<ip>,<port>,<proto>,<keyfile> \
  -o ssl.debug_file:<log-file>

Analyzing SSL with Tshark


```
sake@macsake:~/Dropbox/sharkfest/2016eu/src$ tshark -r google-ssl-stream-37.pcapng -o ssl.keylog_file:sslkeylogfile-stream-37.log -Y ssl
  4   0.012133 192.168.0.133 → 172.217.17.35 SSL 262 Client Hello
  6   0.043228 172.217.17.35 → 192.168.0.133 TLSv1.2 1484 Server Hello
 10   0.046051 172.217.17.35 → 192.168.0.133 TLSv1.2 358 CertificateServer Key Exchange, Server Hello Done
 13   0.047932 192.168.0.133 → 172.217.17.35 TLSv1.2 324 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message, Finished
 14   0.051499 192.168.0.133 → 172.217.17.35 HTTP2 119 Magic
 15   0.051500 192.168.0.133 → 172.217.17.35 HTTP2 116 SETTINGS
 16   0.051539 192.168.0.133 → 172.217.17.35 HTTP2 108 WINDOW_UPDATE
 17   0.051693 192.168.0.133 → 172.217.17.35 HTTP2 399 HEADERS
 18   0.051720 192.168.0.133 → 172.217.17.35 HTTP2 214 HEADERS
 19   0.060356 172.217.17.35 → 192.168.0.133 TLSv1.2 360 New Session Ticket, Change Cipher Spec, Finished
 21   0.061112 172.217.17.35 → 192.168.0.133 HTTP2 128 SETTINGS
[…]
sake@macsake:~/Dropbox/sharkfest/2016eu/src$ tshark -r google-ssl-stream-37.pcapng -o ssl.keylog_file:sslkeylogfile-stream-37.log \
                                             -Y frame.number==14  -V -O http2
Frame 14: 119 bytes on wire (952 bits), 119 bytes captured (952 bits) on interface 0
Ethernet II, Src: Apple_cb:26:45 (ac:bc:32:cb:26:45), Dst: CiscoSpv_53:91:fd (bc:c8:10:53:91:fd)
Internet Protocol Version 4, Src: 192.168.0.133, Dst: 172.217.17.35
Transmission Control Protocol, Src Port: 64320, Dst Port: 443, Seq: 455, Ack: 4389, Len: 53
Secure Sockets Layer
HyperText Transfer Protocol 2
    Stream: Magic
        Magic: PRI * HTTP/2.0\r\n\r\nSM\r\n\r\n

sake@macsake:~/Dropbox/sharkfest/2016eu/src$
```

Analyzing application data without decrypting

Analyzing encrypted SSL data

- Filter on application data with:
  ssl.record.content_type == 23

- Add an extra custom column with:
  ssl.record.length

- Look at the request / response pattern
  - check timestamps between c->s and s->c packets
  - check the ssl record lengths for an indication of the request/response sizes (actual data is less because of SSL overhead)

Analyzing encrypted SSL data

Filter: ssl.record.content_type == 23 Expression... Clear Apply Save

No.	Time	Delta	Source	Destination	Protocol	Length	ssl.len	Info
11	0.040173	0.000000	192.168.3.1	192.168.3.3	TLSv1	491	432	Application Data
12	0.042446	0.002273	192.168.3.3	192.168.3.1	TLSv1	496	384,48	Application Data, Application Data
14	12.494568	12.452122	192.168.3.1	192.168.3.3	TLSv1	491	432	Application Data
15	12.495834	0.001266	192.168.3.3	192.168.3.1	TLSv1	496	384,48	Application Data, Application Data
29	39.717354	27.221520	192.168.3.1	192.168.3.3	TLSv1	550	1,48,432	Change Cipher Spec, Encrypted Handshake Message, Application Data
30	39.720262	0.002908	192.168.3.3	192.168.3.1	TLSv1	496	384,48	Application Data, Application Data
48	111.230987	71.510725	192.168.3.1	192.168.3.3	TLSv1	491	432	Application Data
49	111.233419	0.002432	192.168.3.3	192.168.3.1	TLSv1	496	384,48	Application Data, Application Data

Common SSL handshake problems

Session stops right after “ClientHello”

- No mutually supported SSL version
- No mutually accepted cipher
- No certificate for SNI name

Session stops right after "Certificate"

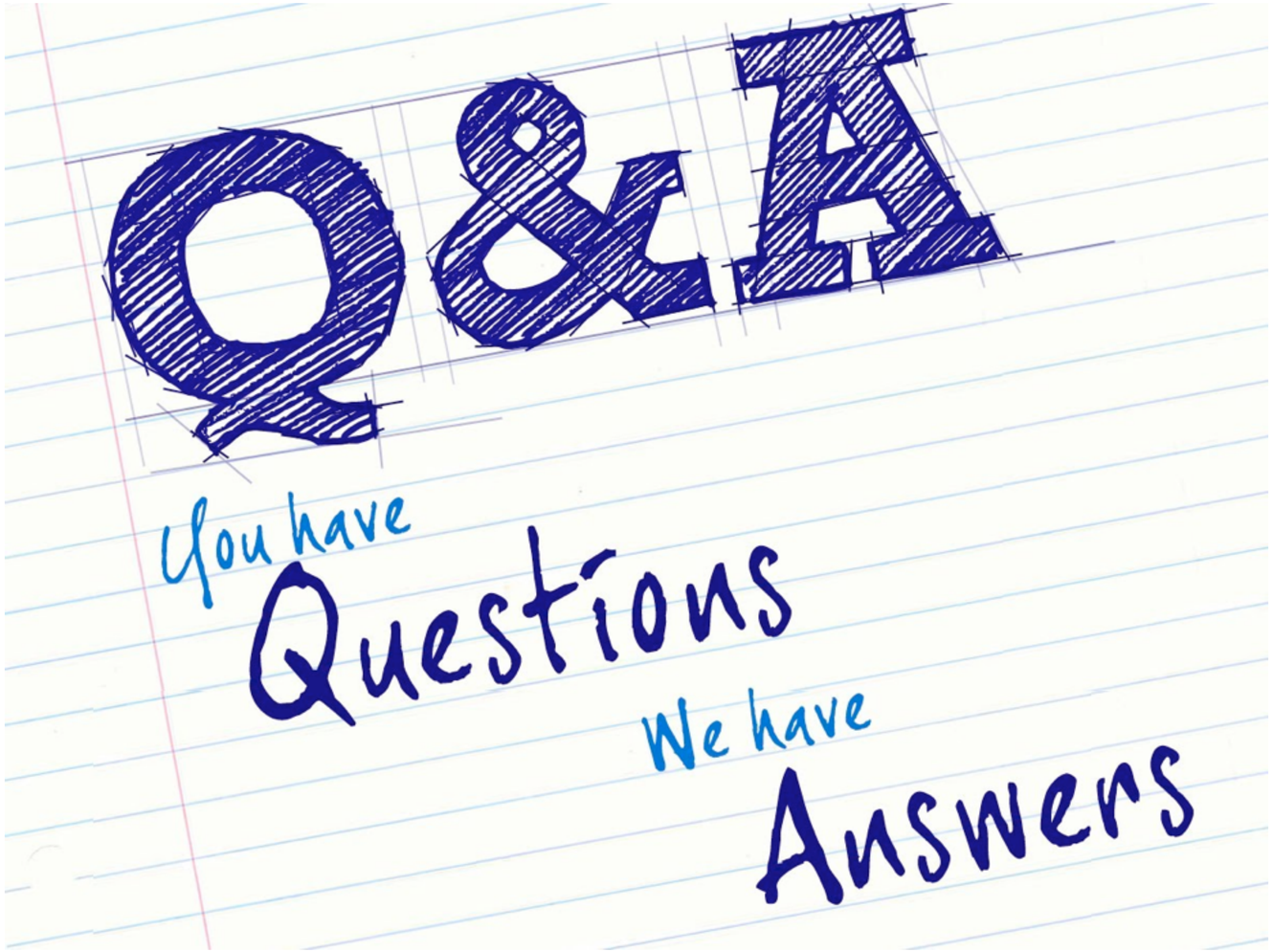
- The Common Name in the certificate does not match with the requested hostname
- The root certificate for the server certificate is not in the client's trust store
- Client has root certificate, but server does not send the intermediate CA certificate
- Server certificate has expired
- Server certificate has been revoked

Session stops right after "Certificate" (from the client)

- The root certificate for the client certificate is not in the server's trust store
- Server has root certificate, but client does not send the intermediate CA certificate
- Client certificate has expired
- Client certificate has been revoked
- The CRL has expired
- Client certificate chain is too long
- Client did not send a certificate as it did not have a certificate signed by one of the CA's in the DN list

Summary

- SSL is a trust model based on cryptology
- Analyze SSL handshake to solve connection problems
- Decrypt traffic to solve application issues
  - use the private key for RSA key exchanges
  - use logging of (pre-)master secrets for all others


Q&A
You have
Questions
We have
Answers

FIN/ACK, ACK, FIN/ACK, ACK

Any questions?

sake.blok@SYN-bit.nl

SYN-bit

deep traffic analysis