

SSL/TLS troubleshooting with Wireshark

Sharkfest 2020 virtual pre-conference class



Sake Blok

Relational therapist for computer systems

sake.blok@SYN-bit.nl





Getting to know each other...





Time for a poll





Do you trust these companies?





- **Relational therapist for computer systems**
 - Solve {application,network,performance}-issues by looking at the communication between systems
- **Member Wireshark core-team since 2007**
- **Started SYN-bit in 2010**
 - Application and Network troubleshooting
 - Protocol and packet analysis
 - Training (Wireshark, TCP, TLS)





SYN-bit
deep traffic analysis

Application and network troubleshooting

Protocol and packet analysis

Training (Wireshark, TCP, SSL)

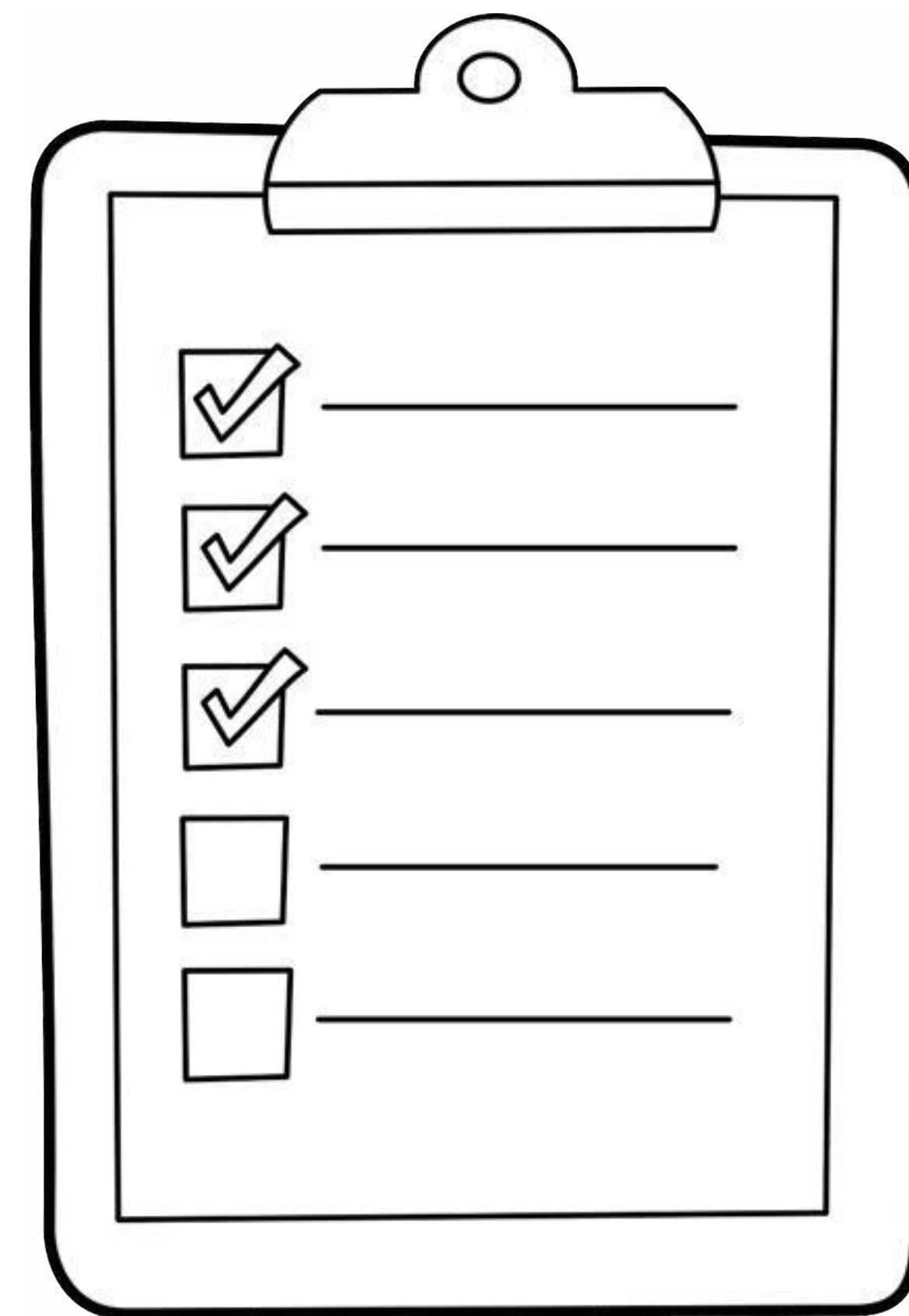
www.SYN-bit.nl



Agenda



- TLS fundamentals
 - The need for TLS (and what happens to SSL?)
 - Cryptology 101 & PKI
 - Troubleshoot the TLS handshake (SSLv3 - TLSv1.2)
 - Troubleshoot the TLS handshake (TLSv1.3)
- Analysing Application Data
 - Without decryption
 - With decryption using the server's private key
 - With decryption using TLS session keys
- Creating configuration profiles for TLS analysis
- SSL/TLS vulnerabilities & privacy concerns
- Summary and Q&A



<https://www.needpix.com/photo/download/94588/checklist-action-check-list-verification-tick-approved-okay-check-mark>



Day planning(-ish)



	San Francisco (PDT, -7)	New York (EDT, -4)	London (BST, +1)	Amsterdam (CEST, +2)	Perth (AWST, +8)	Sydney (AEDT, +11)	Wellington (NZDT, +13)
Part I	8:00	11:00	16:00	17:00	23:00	2:00	4:00
	-	-	-	-	-	-	-
	11:00	14:00	19:00	20:00	2:00	5:00	7:00

Long Break (lunch/diner/snack/breakfast?)

Part II	12:00	15:00	20:00	21:00	3:00	6:00	8:00
	-	-	-	-	-	-	-
	15:00	18:00	23:00	0:00	6:00	9:00	11:00

Short Break (tea/diner/breakfast/coffee?)

Part III	15:30	18:30	23:30	0:30	6:30	9:30	11:30
	-	-	-	-	-	-	-
	17:00	20:00	1:00	2:00	8:00	11:00	13:00



Some Zoom Guidelines



- Please mute yourself
 - if I hear too much noise, I will just mute everyone :-)
- Share your video for a more personal touch
 - But feel free to stop video too of course
- Ask questions in the chat, I'll try to read it often
 - Please don't start chatting, then I can't see the questions
 - Use discord for chatting with each other, I'll read that later
- Raise your hand if you want to say something
 - And wait for me to unmute you please
- If I ask you a question by mentioning your name, please unmute yourself to answer
 - Please don't forget to mute yourself after the discussion :-)





Agenda



- **TLS fundamentals**

- The need for TLS (and what happens to SSL?)
- Cryptology 101 & PKI
- Troubleshoot the TLS handshake (SSLv3 - TLSv1.2)
- Troubleshoot the TLS handshake (TLSv1.3)

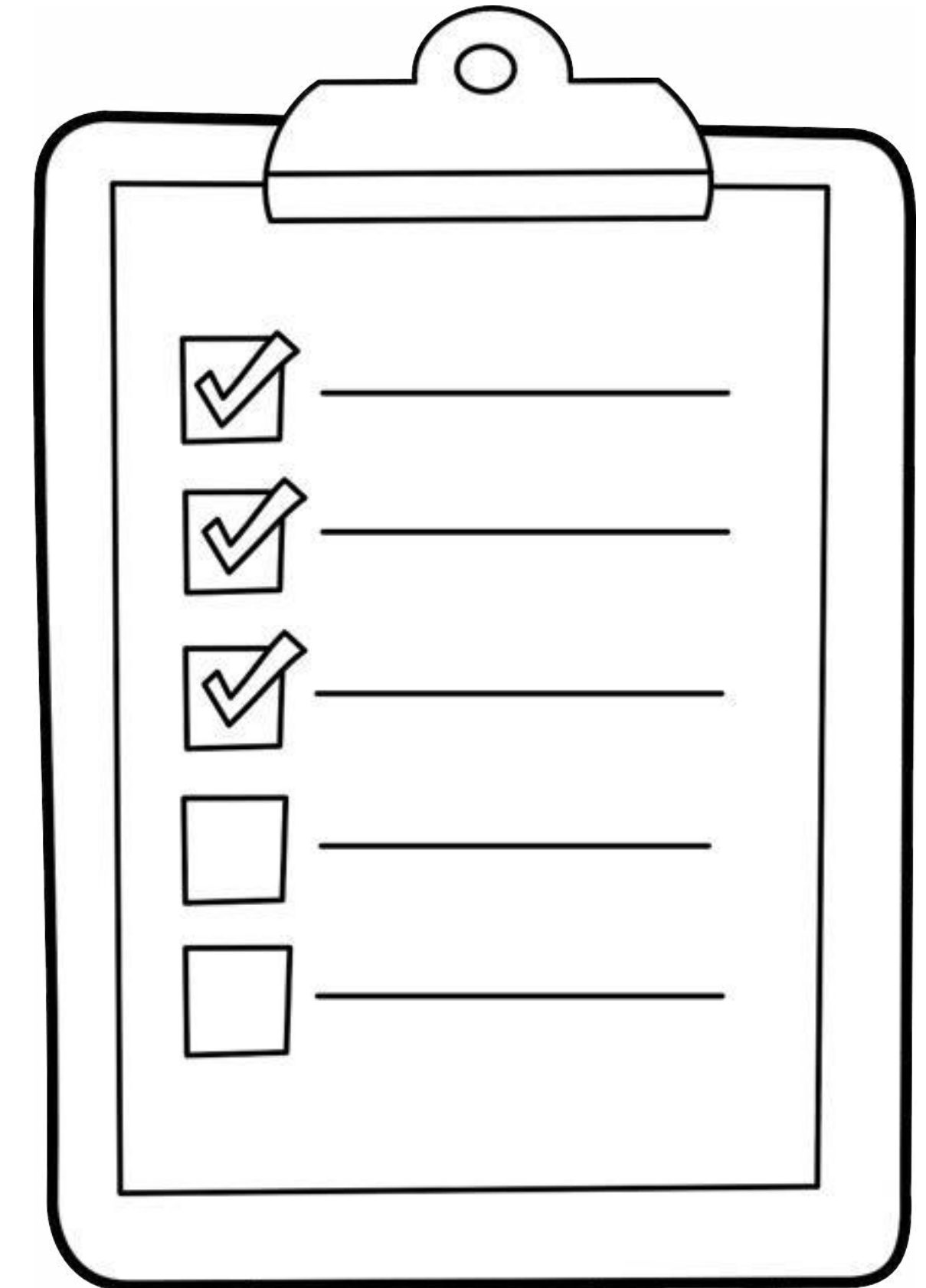
- **Analysing Application Data**

- Without decryption
- With decryption using the server's private key
- With decryption using TLS session keys

- **Creating configuration profiles for TLS analysis**

- **SSL/TLS vulnerabilities & privacy concerns**

- **Summary and Q&A**



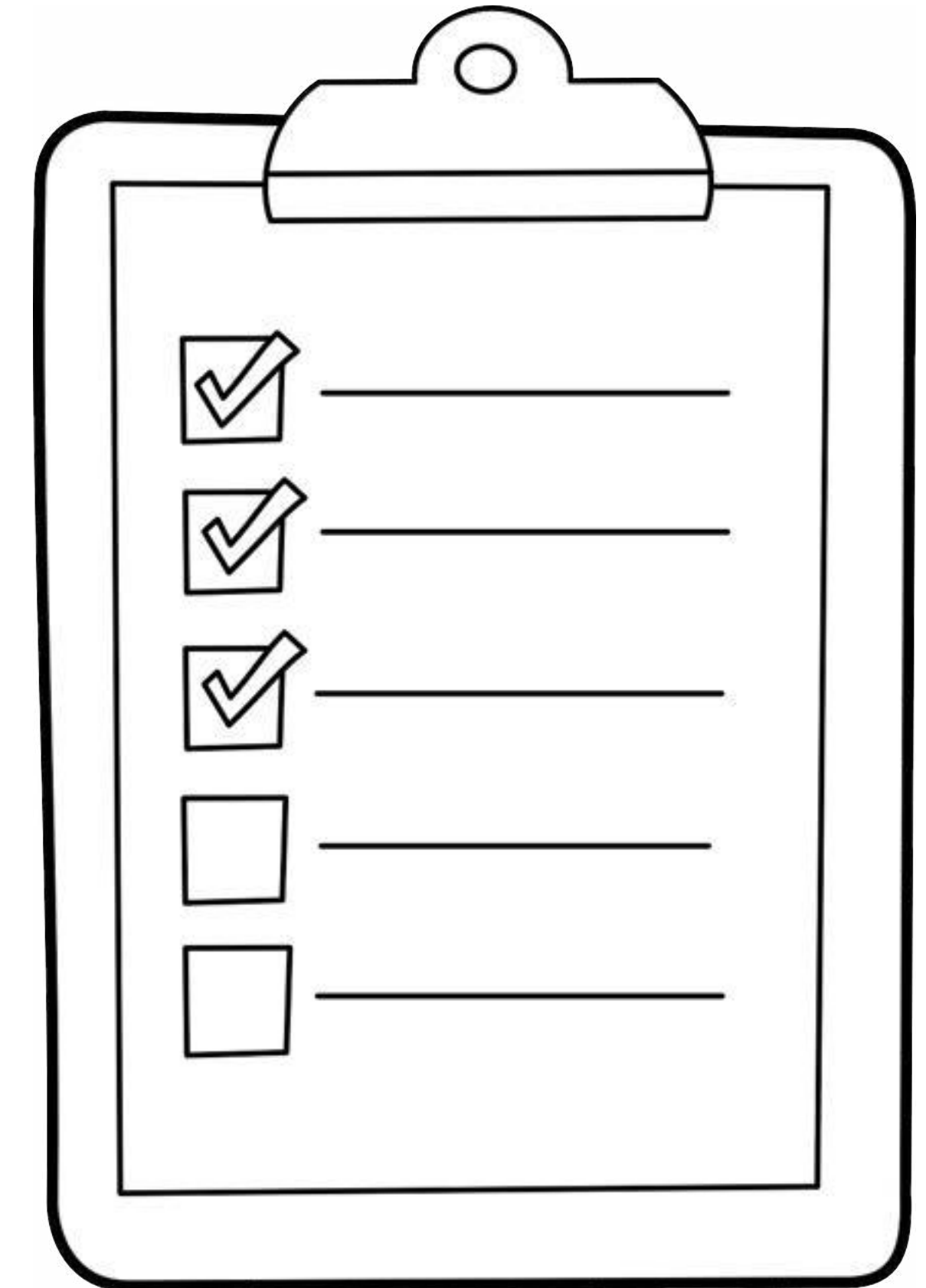
<https://www.needpix.com/photo/download/94588/checklist-action-check-list-verification-tick-approved-okay-check-mark>



Agenda



- TLS fundamentals
 - **The need for TLS (and what happens to SSL?)**
 - Cryptology 101 & PKI
 - Troubleshoot the TLS handshake (SSLv3 - TLSv1.2)
 - Troubleshoot the TLS handshake (TLSv1.3)
- Analysing Application Data
 - Without decryption
 - With decryption using the servers private key
 - With decryption using TLS session keys
- Creating configuration profiles for TLS analysis
- SSL/TLS vulnerabilities & privacy concerns
- Summary and Q&A



<https://www.needpix.com/photo/download/94588/checklist-action-check-list-verification-tick-approved-okay-check-mark>



Birth of HTTP and SSL/TLS



- 1989 invention of HTML, HTTP and URLs by Tim Berners-Lee (and team)
- Mid 90's, Internet made its way to the public
 - NCSA Mosaic (1993)
 - Netscape (1994)
 - Internet Explorer (1995)
- Start of e-commerce
 - but how to safely use credit card numbers?
- Secure Socket Layer & Transport Layer Security
 - Makes use of Public Key Infrastructure (PKI)



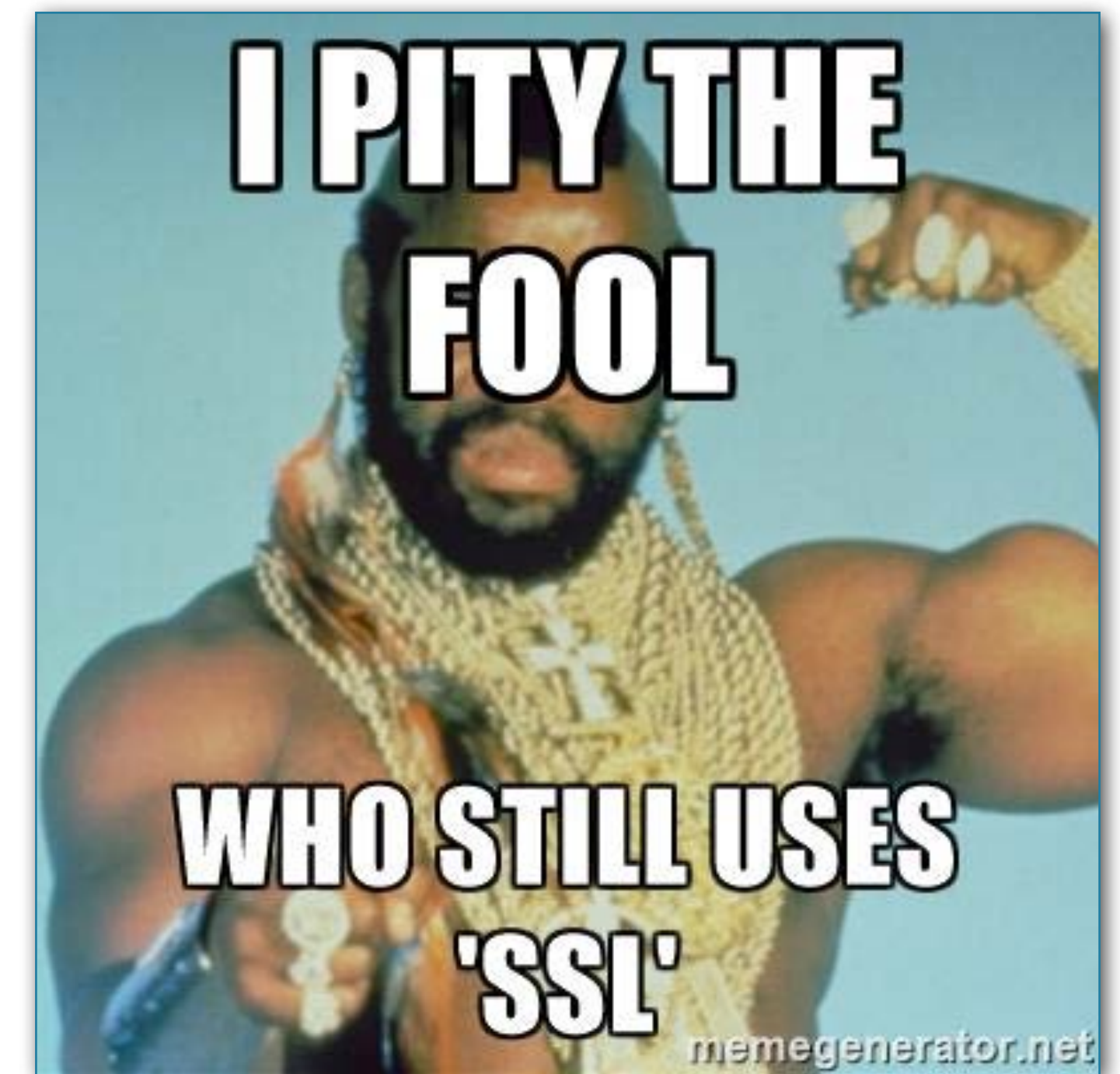
<https://cds.cern.ch/record/39437>



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)
 - Deprecated March 31, 2020
- TLSv1.1, IETF (RFC 4346, 2006)
 - Deprecated March 31, 2020
- **TLSv1.2, IETF (RFC 5246, 2008)**
- **TLSv1.3, IETF (RFC 8446, 2018)**





Public Key Infrastructure?



A public key infrastructure (PKI) is a **set of roles, policies, and procedures** needed to create, manage, distribute, use, store, and revoke **digital certificates** and manage **public-key encryption**.

The purpose of a PKI is to facilitate the **secure electronic transfer of information** for a range of network activities such as **e-commerce, internet banking and confidential email**.

https://en.wikipedia.org/wiki/Public_key_infrastructure



Some methods of PKI certification



- Certificate Authorities
 - X.509
- Web of Trust
 - PGP, OpenPGP, GnuPG
- Simple Public Key Infrastructure
- Blockchain based





Methods of PKI certification



- **Certificate Authorities**

- X.509

- Web of Trust

- PGP, OpenPGP, GnuPG

- Simple Public Key Infrastructure

- Blockchain based





Why TLS and Certificates?

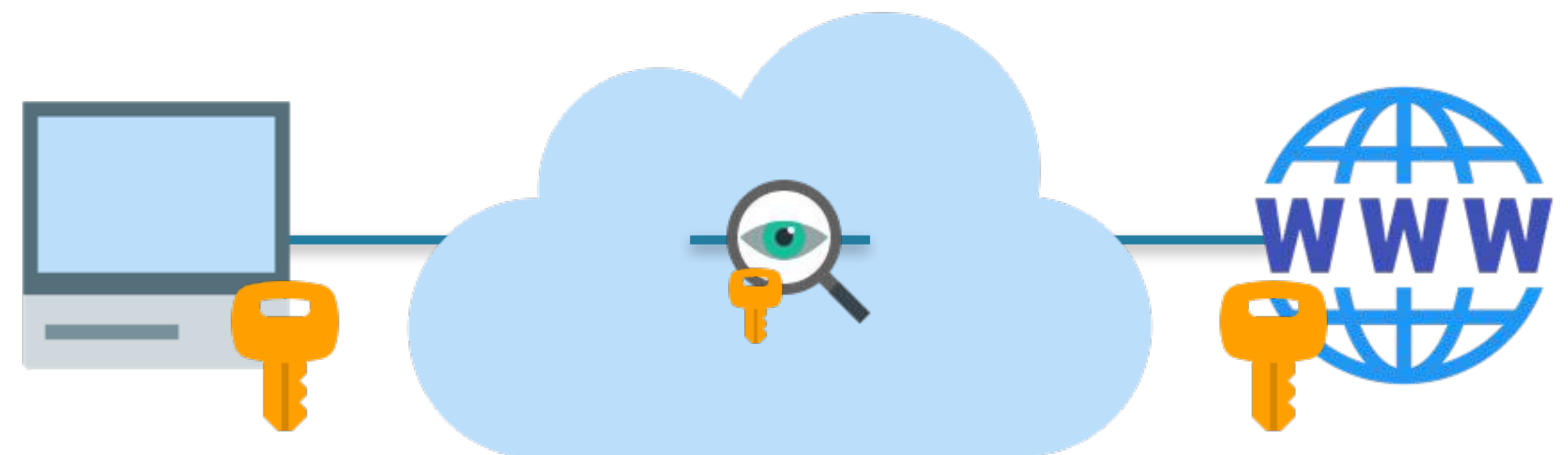




Step 1: symmetric keys



- Server creates a symmetric key
- Sends key to client through secure channel
- Encrypt traffic with symmetric key
- Keys need to be unique to prevent evesdropping
- **Need a solution for large scale key distribution**

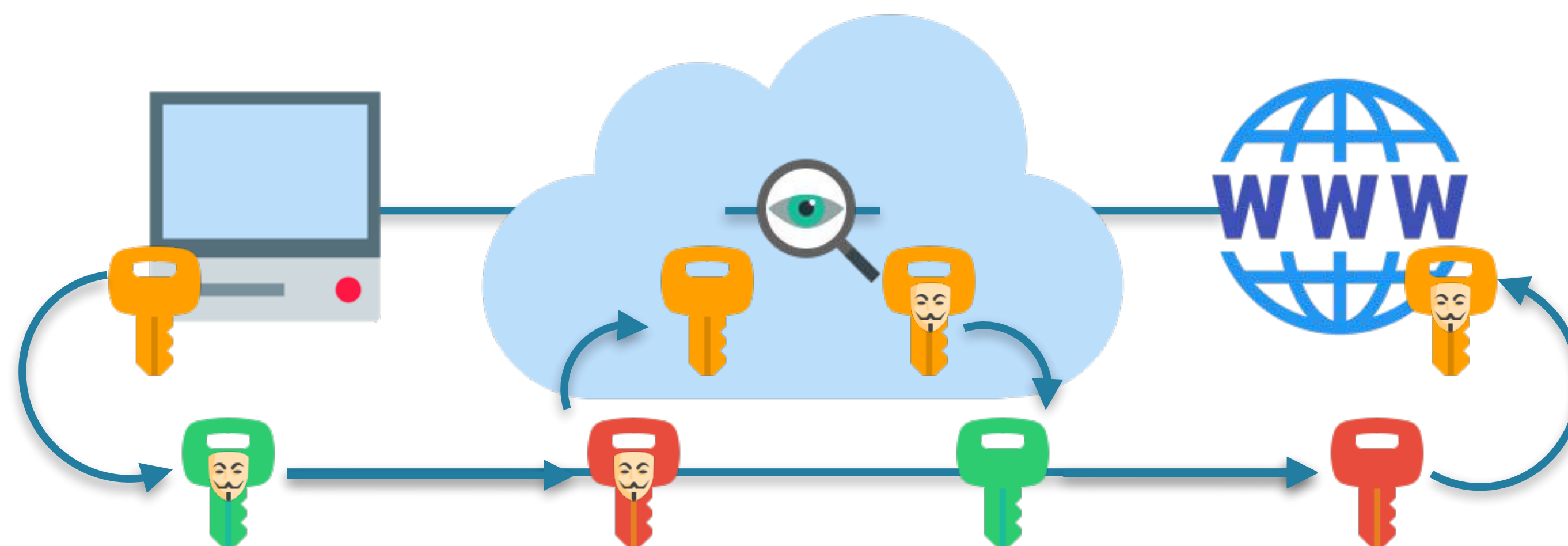




Step 2: asymmetric keys



- Server creates public/private key-pair
- ...and sends public key to client
- Client creates (symmetric) session key, encrypts with public key
- Client sends encrypted session key to server
- Server decrypts session key with private key
- **But how to prevent MITM?**





Step 3: certificates



- Agree on a trusted 3rd party (the certificate authority or CA)
- CA signs ID+PUB of the server with its private key => certificate
- Server sends certificate to client, client checks with public key of CA
- Now client knows who it is talking to and can trust the public key
- Client creates session key and encrypts with public key
- Client sends encrypted session key

✓ Confidentiality
✓ Integrity
✓ Authentication



How can we accomplish these goals?

- Authentication

- We want to know who we are talking to
- By using: certificates and certificate authorities (the PKI)

- Confidentiality

- We want to make sure nobody can read our messages
- By using: symmetric encryption and decryption

- Integrity & non-repudiation

- We want to be sure that our messages are not changed
- By using: message digests and signing

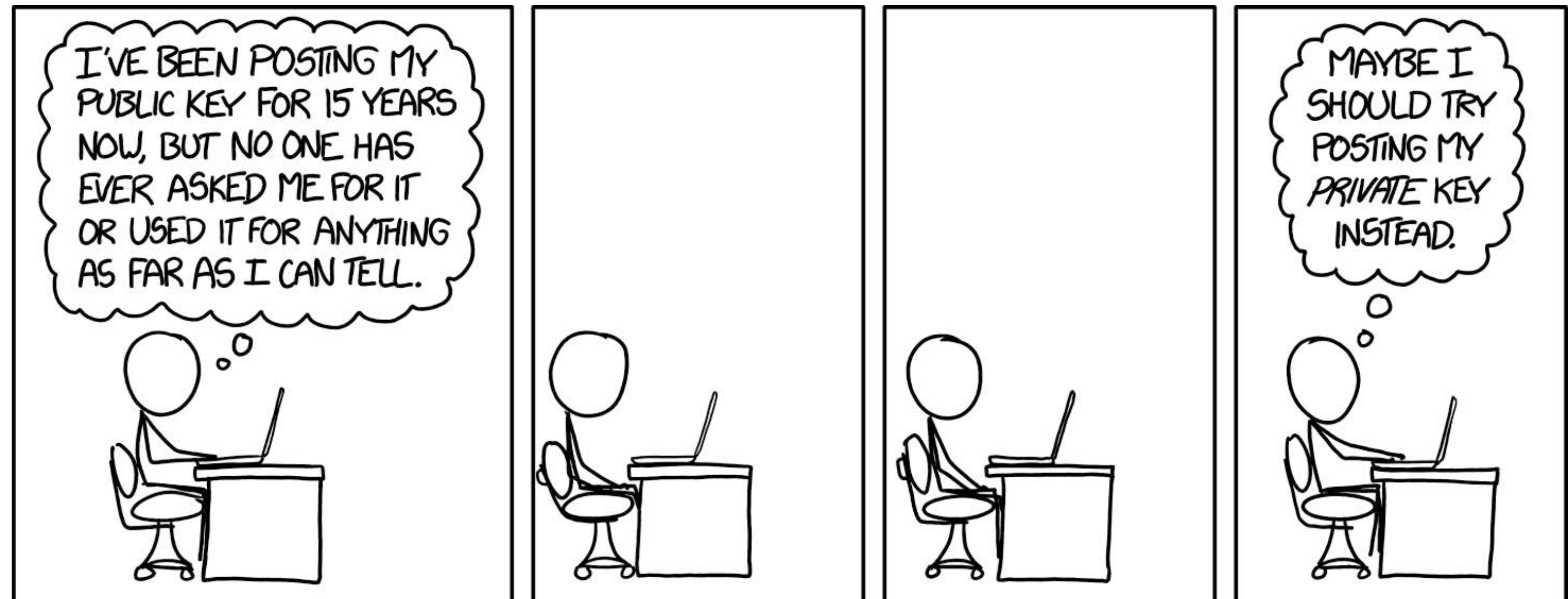




Agenda



- TLS fundamentals
 - The need for TLS (and what happens to SSL?)
 - **Cryptology 101 & PKI**
 - Troubleshoot the TLS handshake
 - Troubleshoot the TLS handshake
- Analysing Application Data
 - Without decryption
 - With decryption using the session key
 - With decryption using TLS session key
- Creating configuration profiles
- SSL/TLS vulnerabilities & mitigations
- Summary and Q&A



<https://xkcd.com/1553/>



History of cryptography



- **It's not new!**
 - 1900 BC: Non-standard hieroglyphs
 - 1500 BC: Recipe for pottery glaze
 - 400 BC to 200 AD: Communication between lovers
 - Ancient Greeks: Used by Spartan Military
- **Techniques**
 - Pen & paper (or simple mechanics)
 - Complex (electro)mechanical devices
 - Computing
- **From governments to private persons**
 - Big leap caused by e-commerce
- **Cryptoanalysis**



CC BY-SA 3.0, <https://commons.wikimedia.org/w/index.php?curid=23821>



The weakest link



“Much of the German cipher traffic was encrypted on the Enigma machine. Used properly, the German military Enigma would have been virtually unbreakable; in practice, shortcomings in operation allowed it to be broken.”

[<https://en.wikipedia.org/wiki/Ultra>]

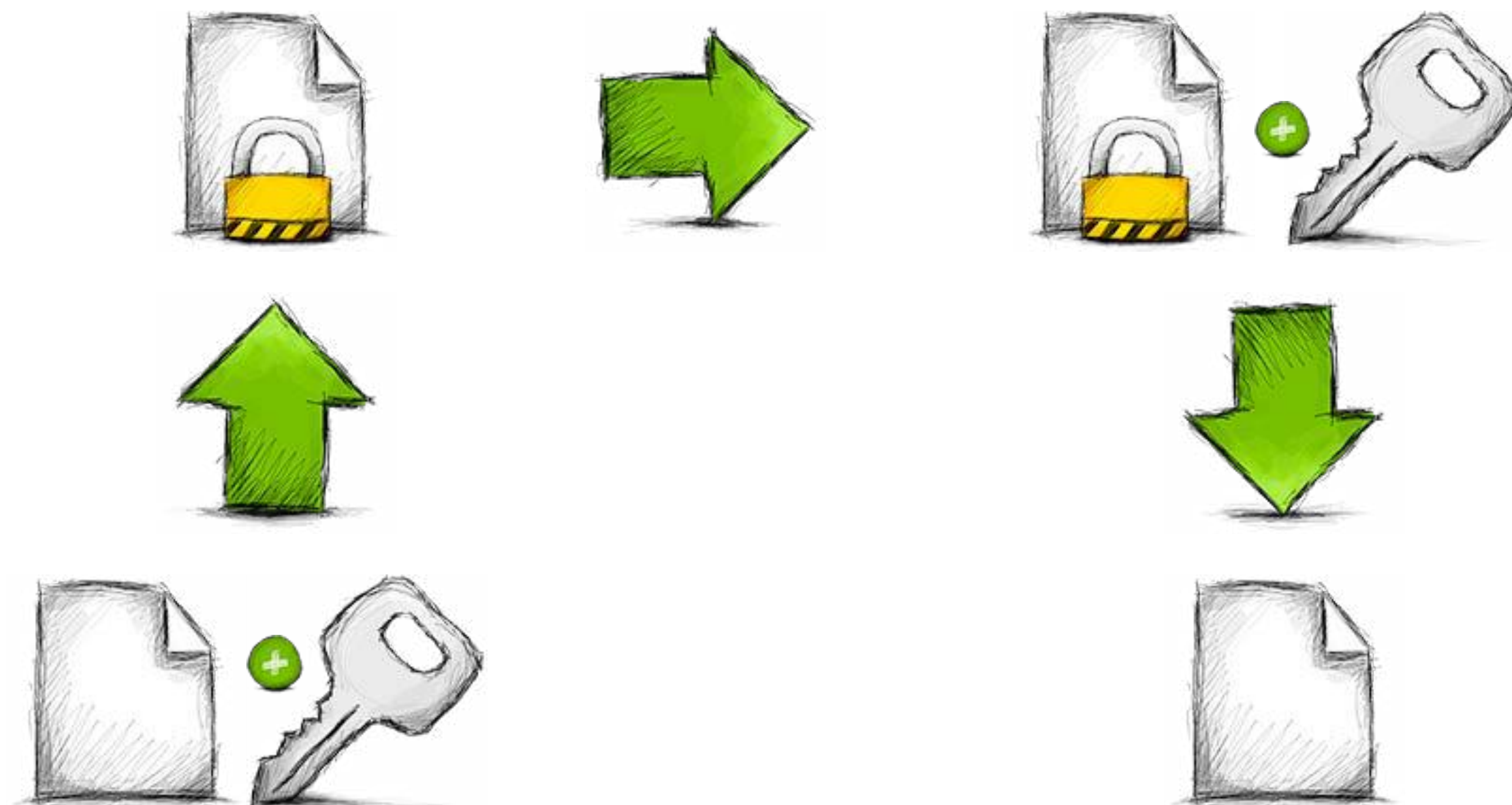




Symmetrical Encryption

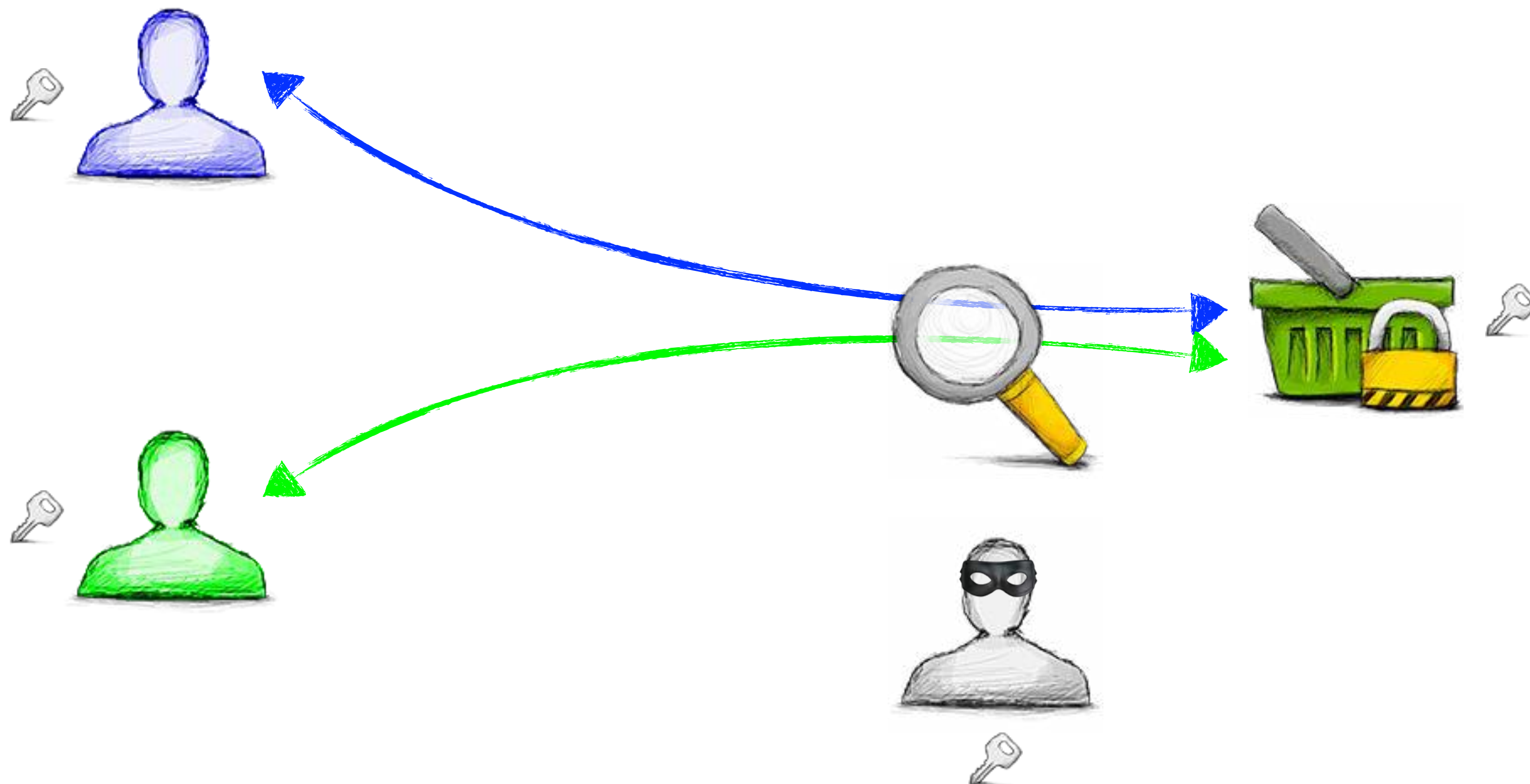


- Same key for encryption and decryption
- Computative "cheap"
- Short key lengths (typically 40-256 bits)
- ~~DES, 3DES, RC4~~, AESxxx





Symmetrical Session Key

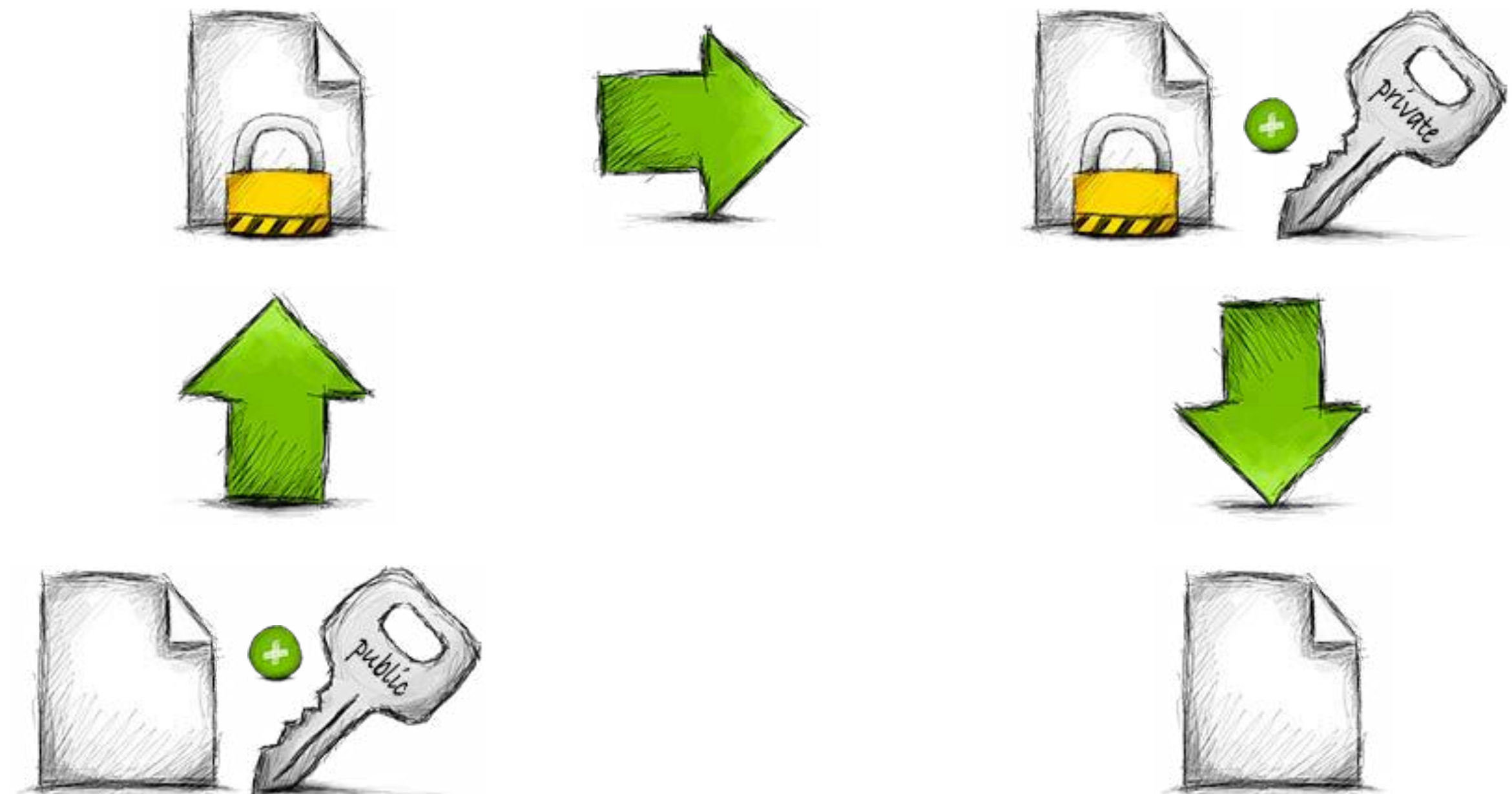




Asymmetrical Encryption



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

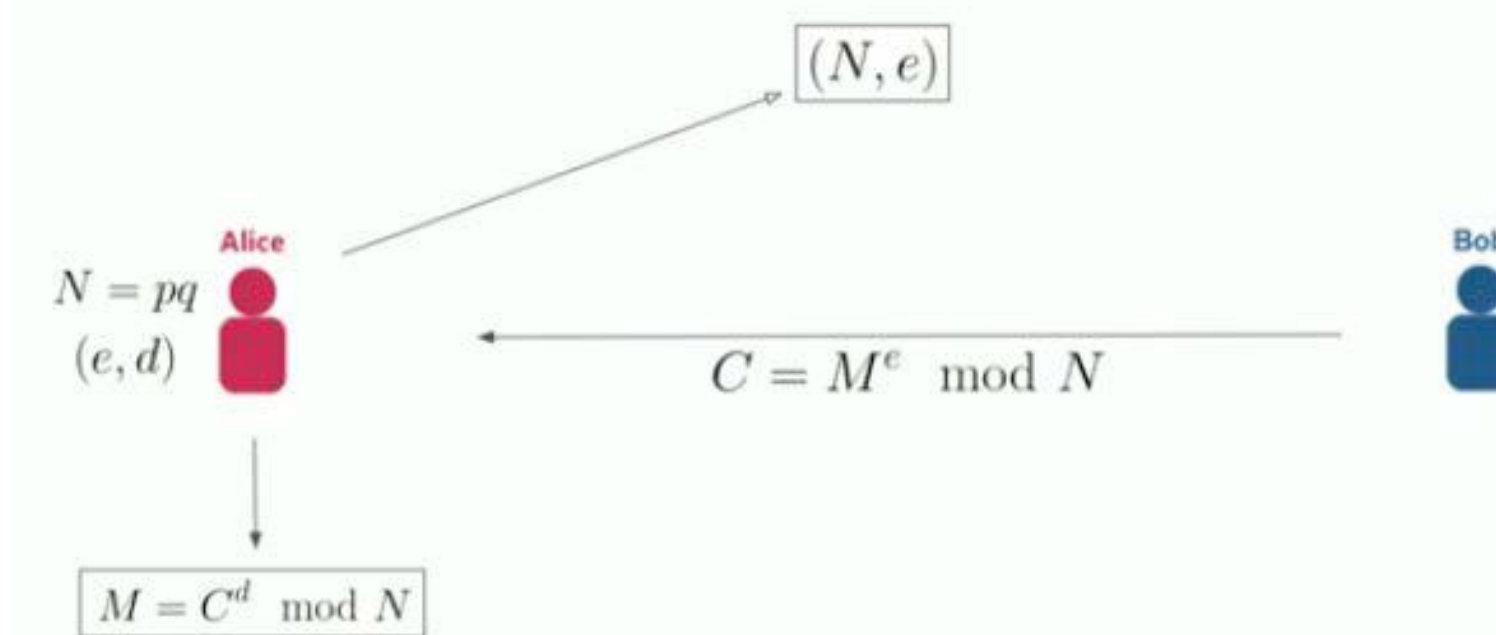




RSA



- Ron **R**ivest, Adi **S**hamir, and Leonard **A**dleman
- Publicly described the algorithm in 1977
- Public key is product of two large primes
- Keep primes private
- Relies on "factoring problem"
- In ~25 years(?), a quantum computer could break 2048-bit RSA encryption in ~8 hours???
- "Seriously, stop using RSA" (unless done right)
 - <https://blog.trailofbits.com/2019/07/08/fuck-rsa/>



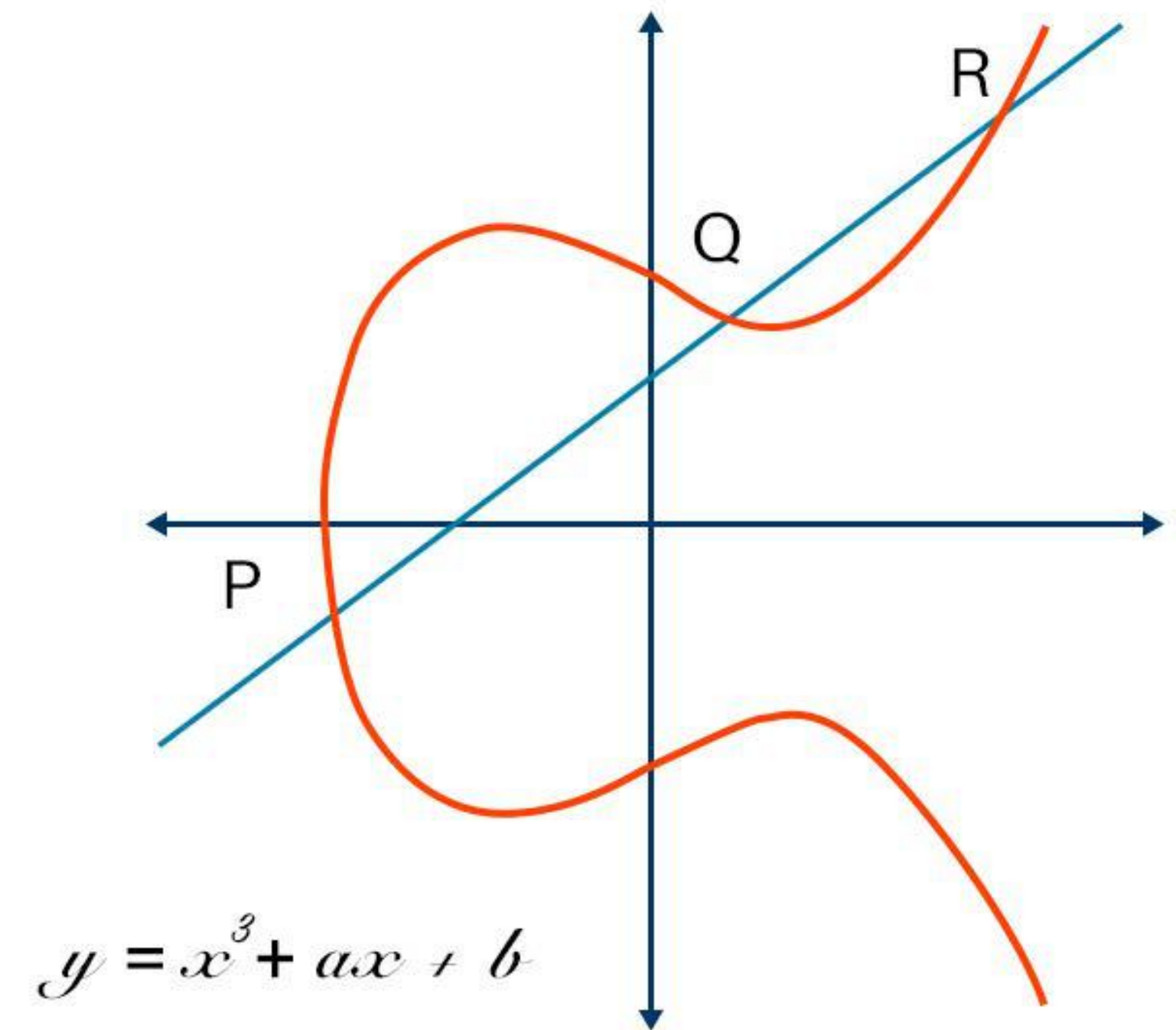
<https://blog.trailofbits.com/2019/07/08/fuck-rsa/>



Elliptic-curve cryptography (ECC)



- suggested independently by Neal Koblitz and Victor S. Miller in 1985
- Based on the algebraic structure of elliptic curves over finite fields
- Wide in use since 2004/2005 (NSA Suite B)
- Good for mobile devices and IoT
 - Smaller key size, reducing storage and transmission
 - Less computing needed
- Can be broken with quantum computing (little easier even than RSA)



<https://avinetworks.com/glossary/elliptic-curve-cryptography/>

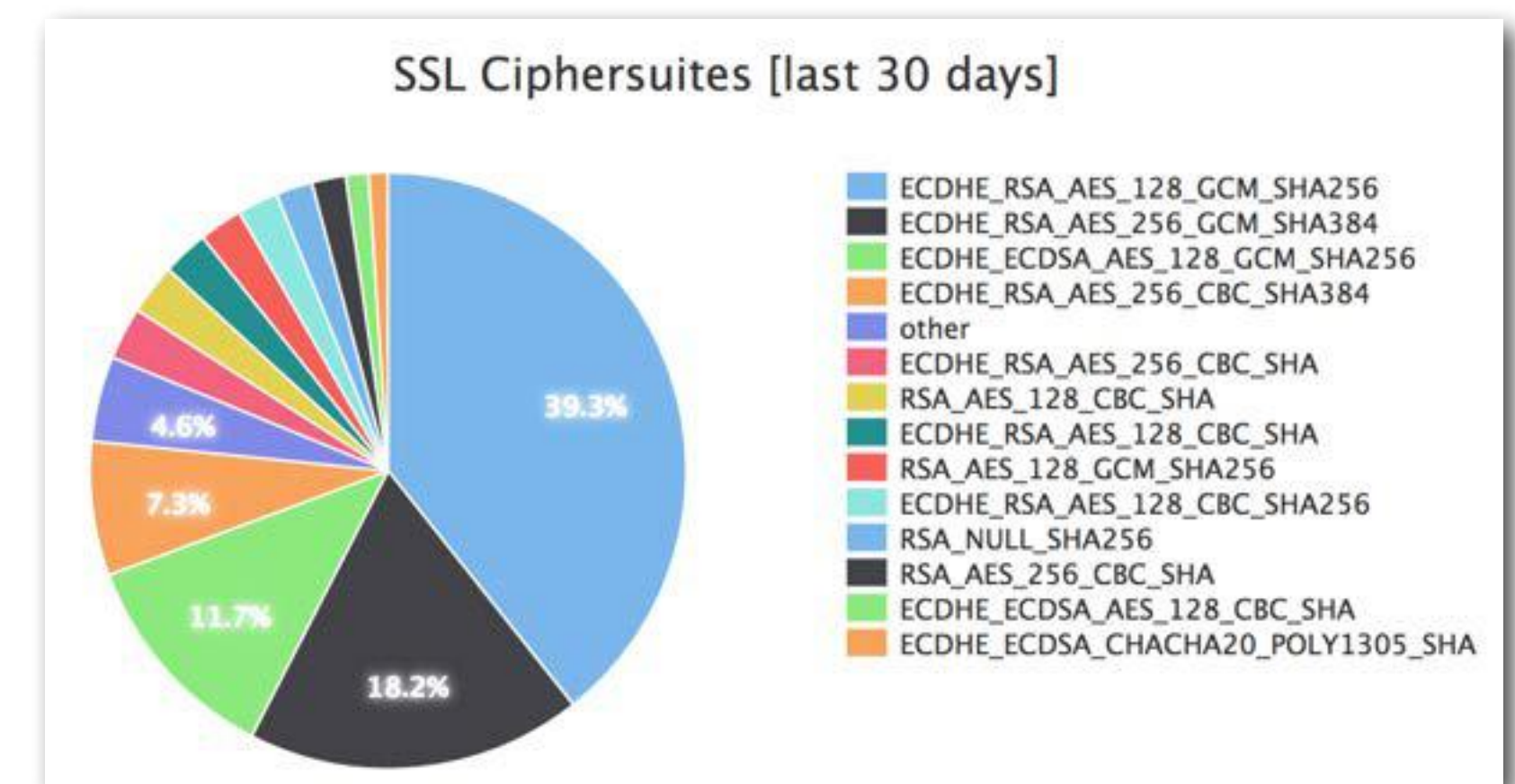


RSA versus ECC



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

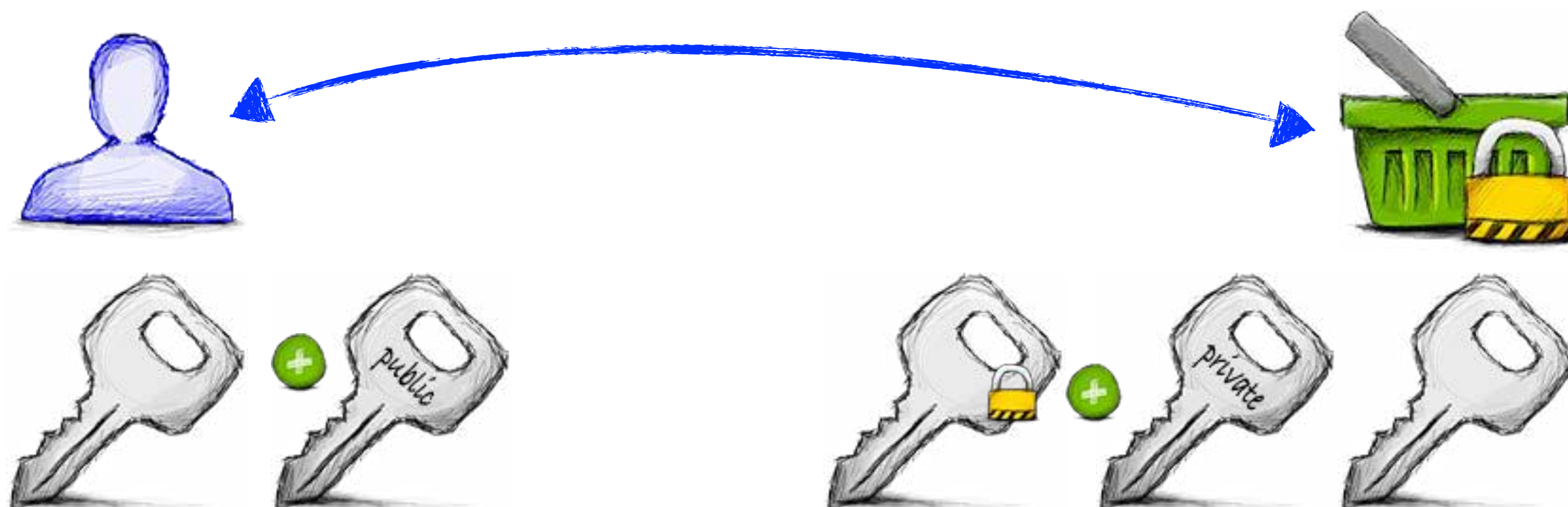
Symmetric	ECC	DH/DSA/RSA
80	163	1024
112	233	2048
128	283	3072
192	409	7680
256	571	15360



<https://notary.icsi.berkeley.edu/> (from 2017)

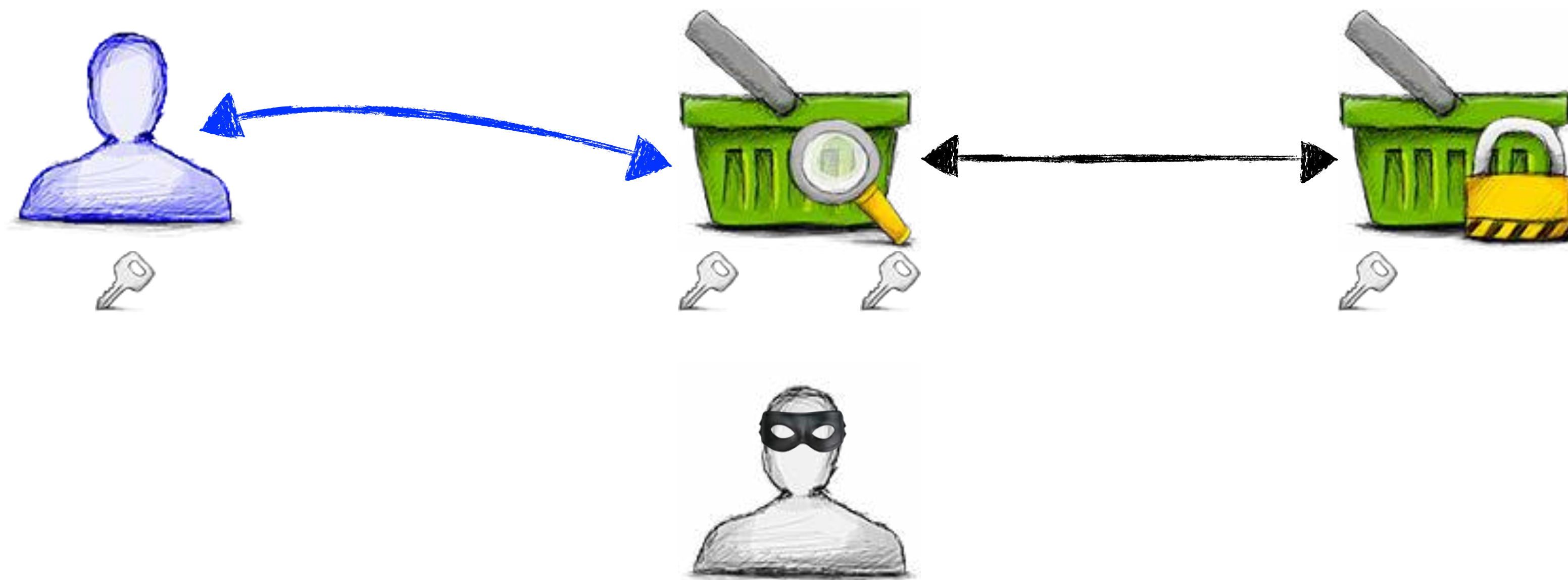


Secure(?) Key Exchange





Man-in-the-Middle



How can we solve this?



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**."

http://en.wikipedia.org/wiki/Digital_certificate

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



Message Digest



- Irreversible
 - original text not reproducible 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

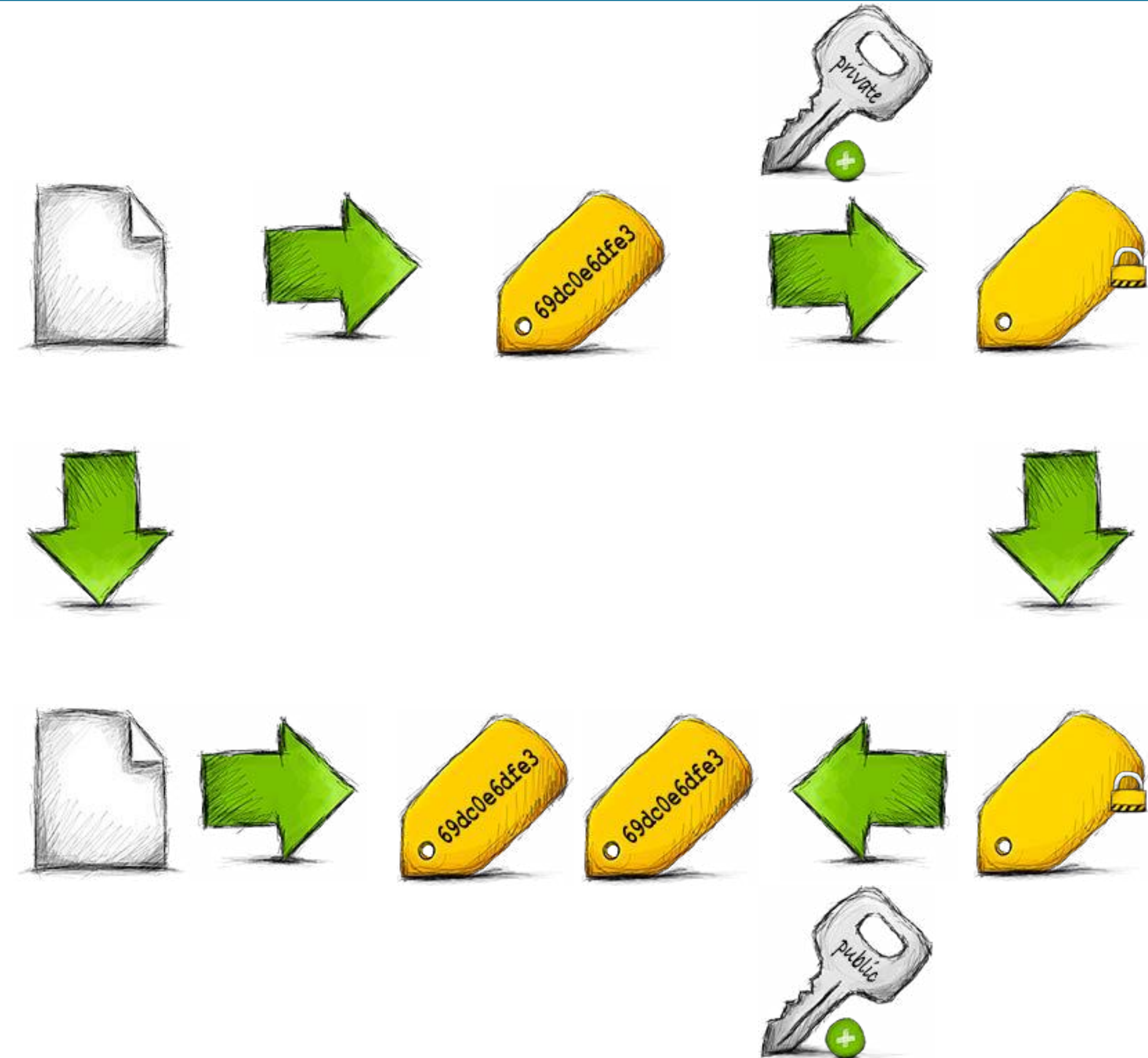




Signing

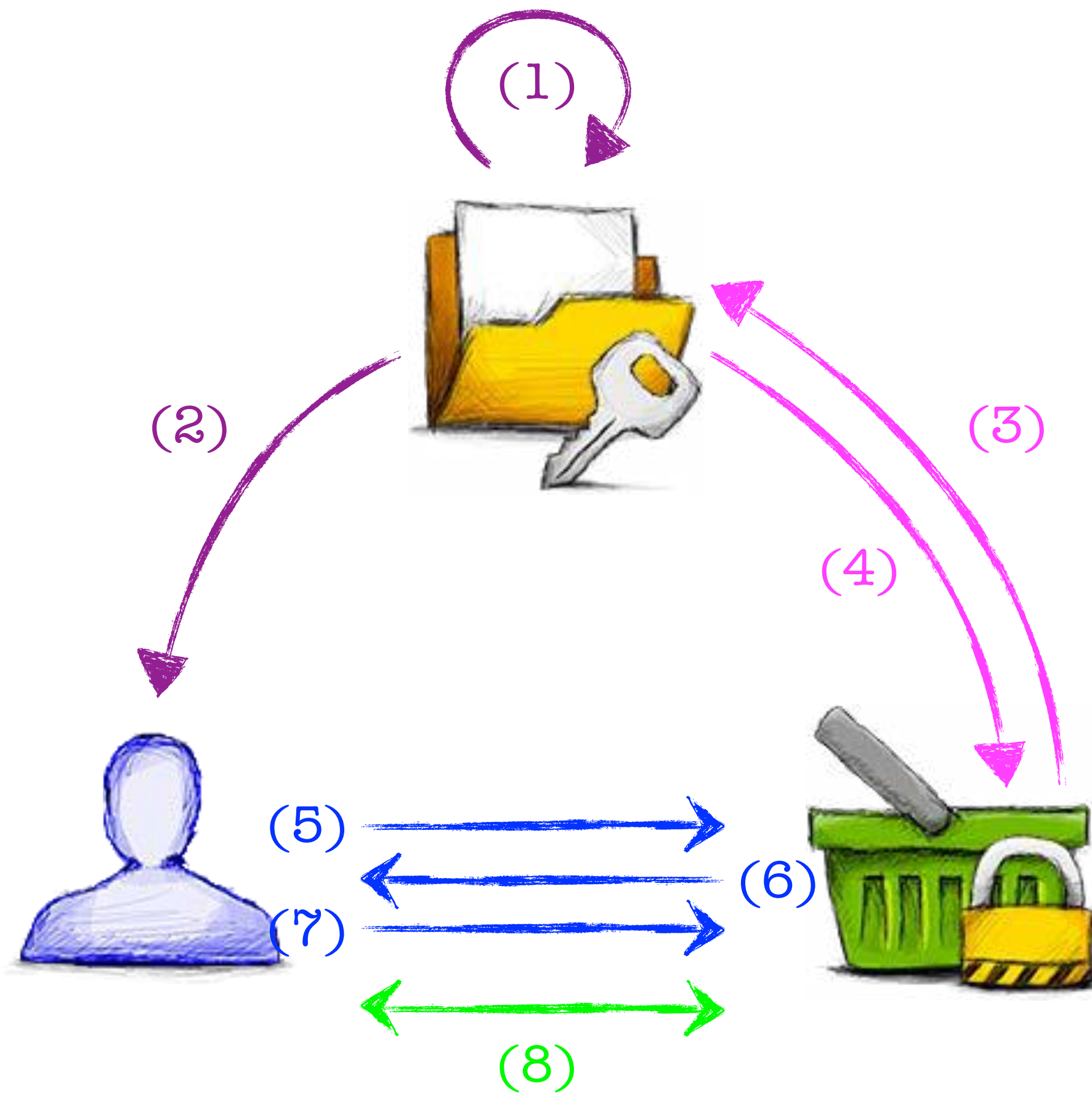


- Create digest of message
- Encrypt digest with private key
- Authenticity and sender of message can be checked with public key





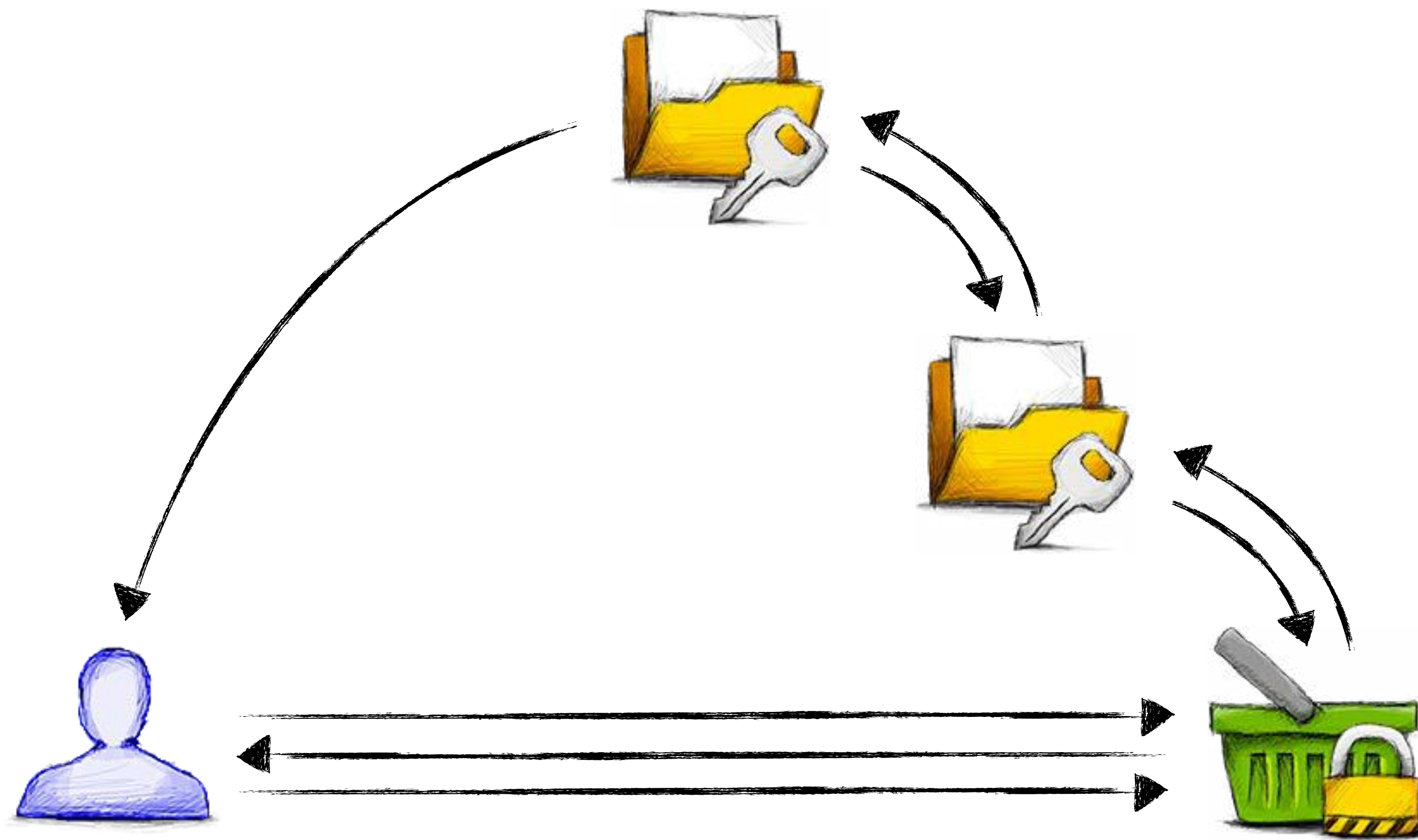
Certificate Authority



- 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 **(in a secure way!?!)** (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

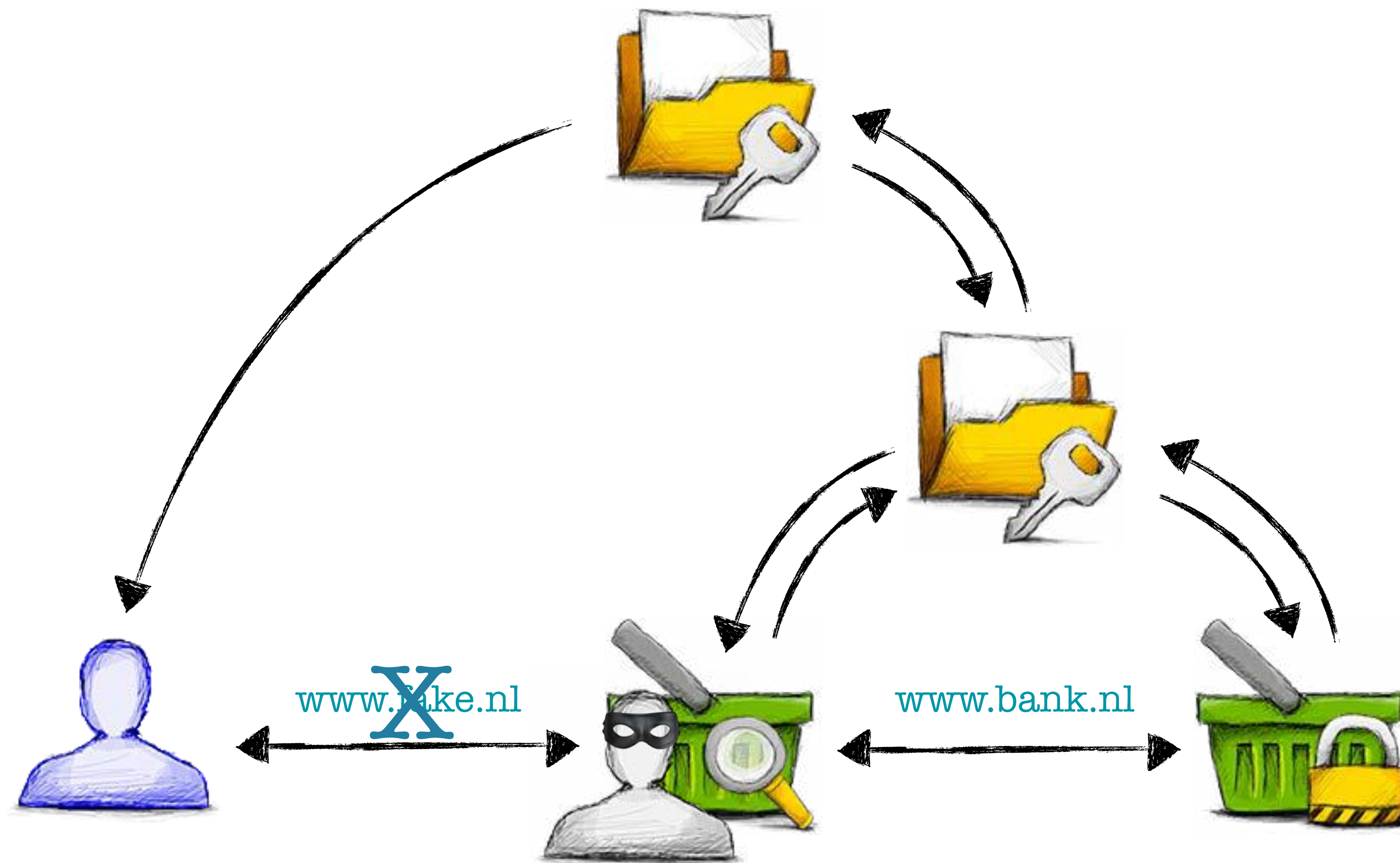


Intermediate CA's



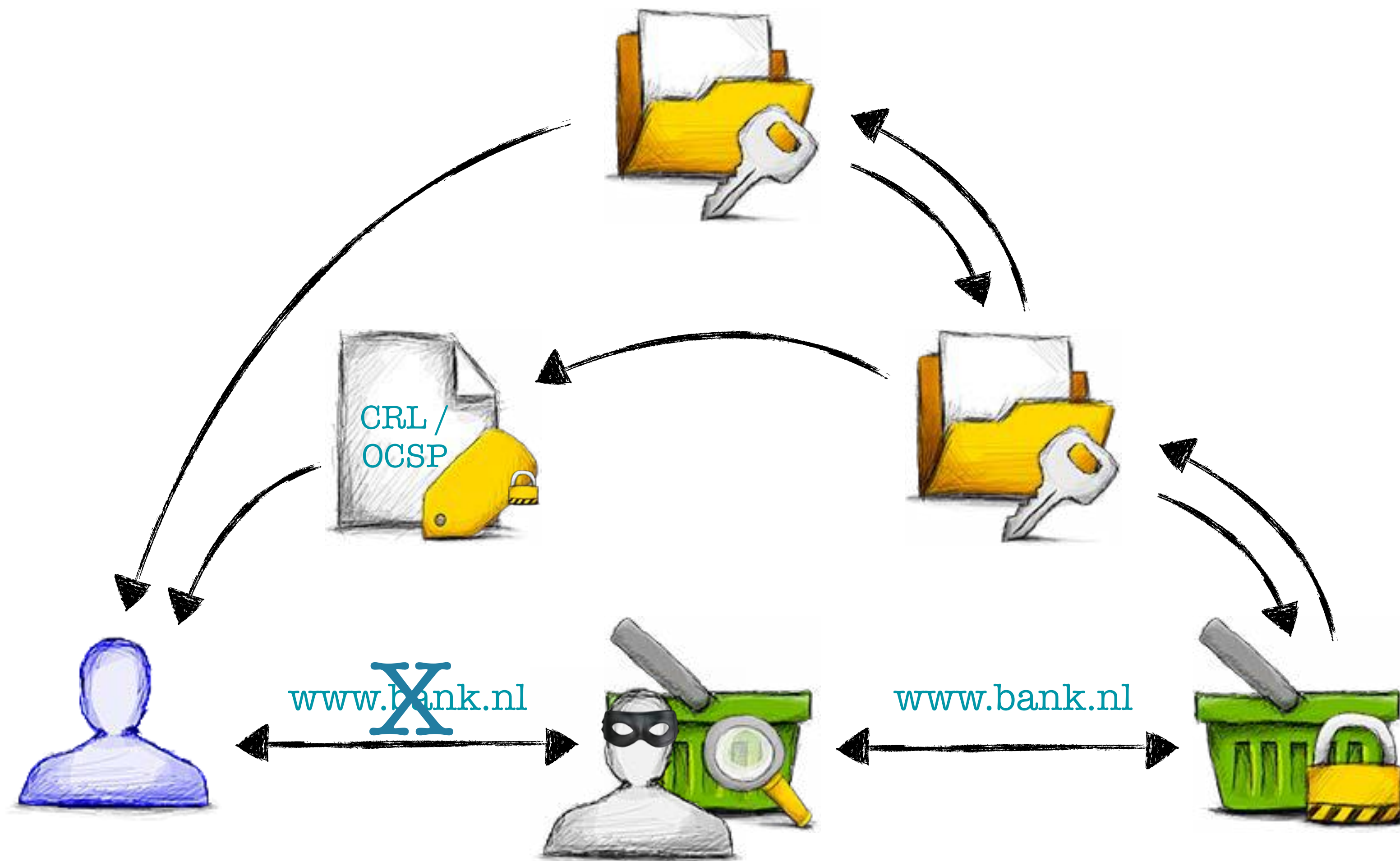


Man-in-the-Middle II





Compromised Private Key





Certificate check on the client



- Check if the **ID** in the certificate matches what was requested
- Check if certificate chain is correct and ends in a **trusted CA**
- Check if the certificate is **valid**
- Check if the certificate has not been **revoked**





Certificate check on the client



- Check if the **ID** in the certificate matches what was requested
<https://wrong.host.badssl.com/>
- Check if certificate chain is correct and ends in a **trusted CA**
<https://untrusted-root.badssl.com/>
- Check if the certificate is **valid**
<https://expired.badssl.com/>
- Check if the certificate has not been **revoked**
<https://revoked.badssl.com/>



Certificate Revocation

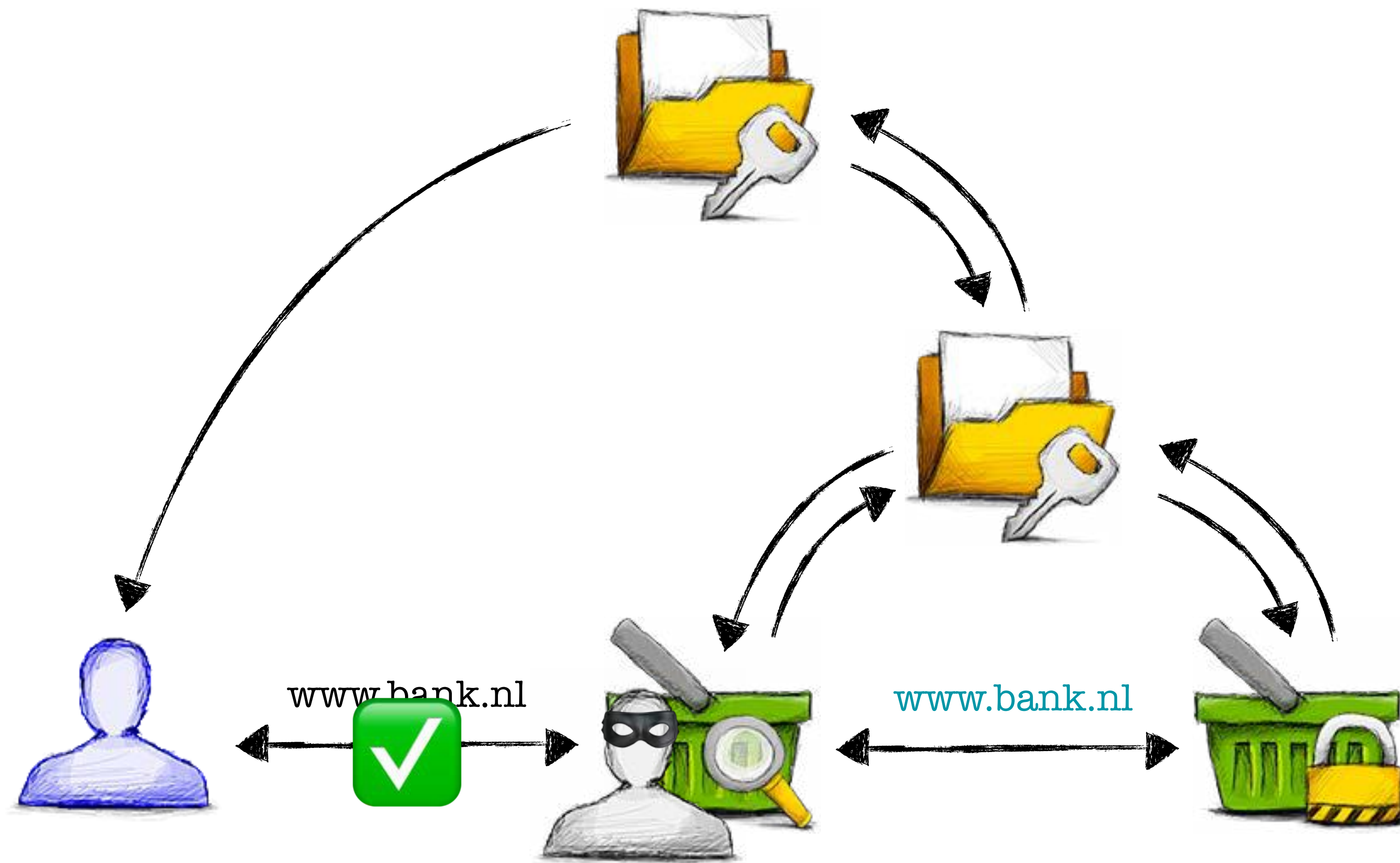


- Certificate Revocation List (CRL)
 - Signed by CA
 - Can grow large, periodically retrieved
 - “deprecated”
- Online Certificate Status Protocol (OCSP)
 - Checked per connection
 - large load on CA
 - exposes user behaviour to CA
- OCSP Stapling
 - The server provides a time stamped, signed OCSP response
 - Must-staple extension





Compromised CA





Diginotar 2011



- 6-19 Jun: Hacker gained access to systems
- 10+19 Jul: Hacker created certificates
- 27 Jul: *.google.com certificate used for M-i-t-M attack in Iran
- 4 Aug: M-i-t-M attack now on large scale
- 27 Aug: Discovery of the falsified certificate
- 29 Aug: Revocation of *.google.com certificate
- 30 Aug: First removal of DigiNotar CA

```
Certificate:
Data:
  Version: 3 (0x2)
  Serial Number:
    05:e2:e6:a4:cd:09:ea:54:d6:66:b0:75:fe:22:a2:56
  Signature Algorithm: sha1WithRSAEncryption
  Issuer: C=NL, O=DigiNotar, CN=DigiNotar Public CA 2025/emailAddress=info@diginotar.nl
  Validity
    Not Before: Jul 10 19:06:30 2011 GMT
    Not After : Jul  9 19:06:30 2013 GMT
  Subject: C=US, O=Google Inc, L=Mountain View/serialNumber=PK000229200002, CN=*.google.com
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    Public-Key: (2048 bit)
  X509v3 extensions:
    Authority Information Access:
      OCSP - URI:http://validation.diginotar.nl
    X509v3 CRL Distribution Points:
      URI:http://service.diginotar.nl/crl/public2025/latestCRL.crl
  Signature Algorithm: sha1WithRSAEncryption
  02:ce:5d:2f:b3:7d:c3:34:49:90:8e:00:ab:89:42:e6:df:23:
  [...]
  6b:2c:d9:1a
```



Trust???



- Security measures not OK
- No virus scanners used on critical systems
- No separation of critical systems
- Weak passwords
- Unpatched software on public systems

OSCP queries for fake Google Certificate



<http://www.youtube.com/watch?v=wZsWoSxxwVY>

- **No (open) communication about the hack**
- **Misleading information (PKloverheid CA was indeed hacked contrary to earlier statements)**



Certificate Identities (Common Name)



- Normal

- www.example.com

- Wildcard

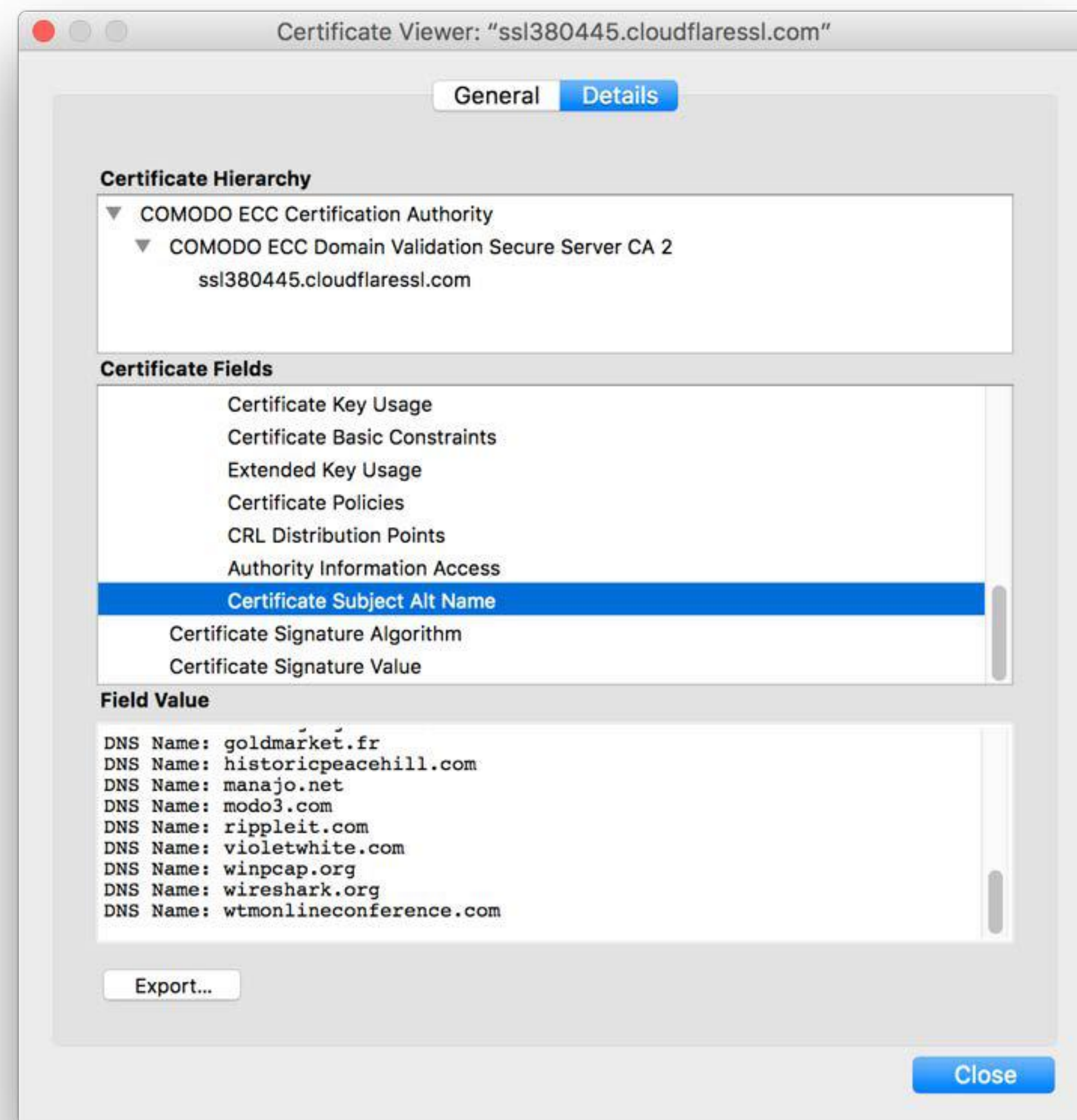
- *.example.com

- SAN

- www.example.com

- mail.example.com

- www.another.org

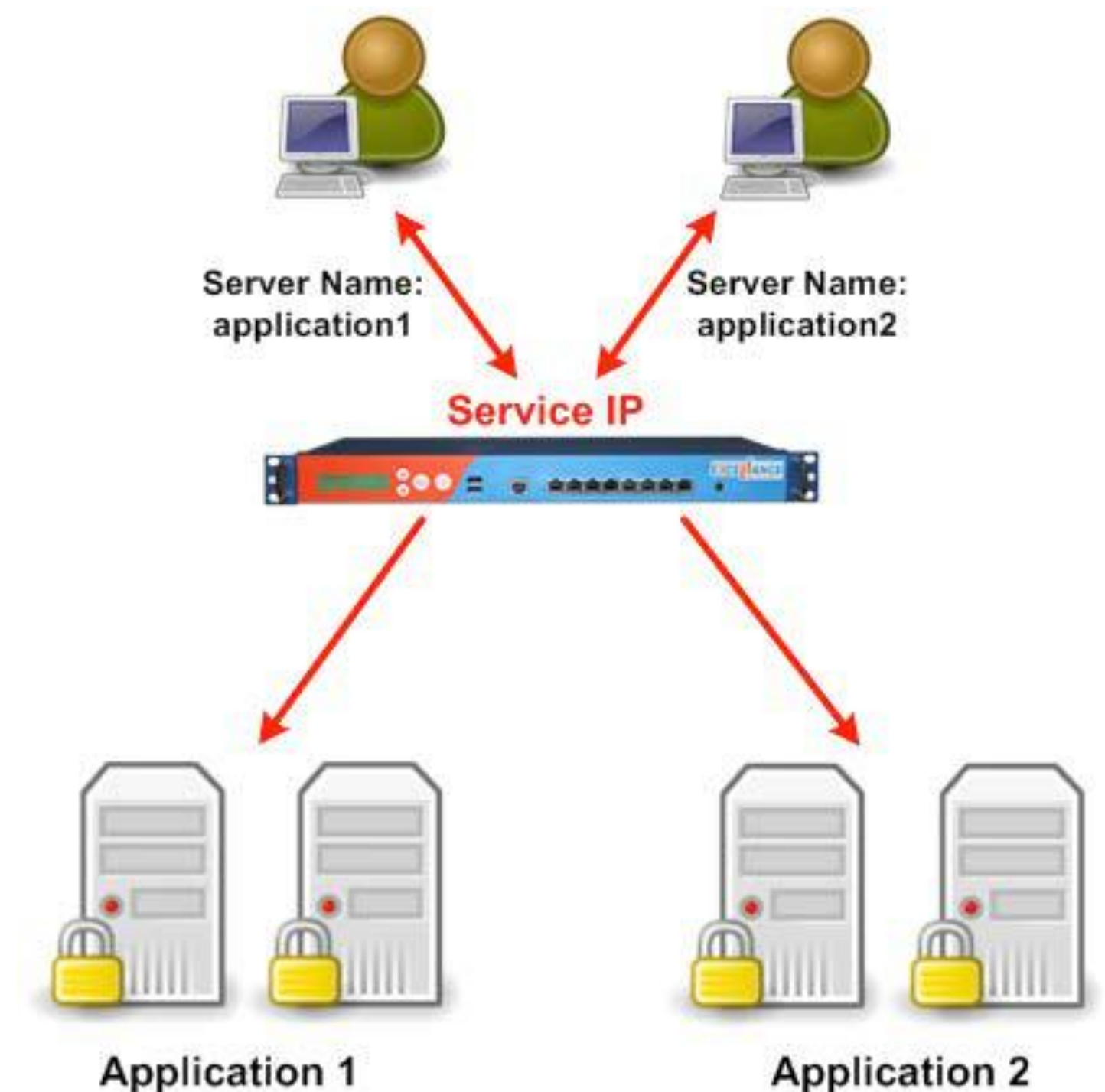




Server Name Indication (SNI)



- Multiple virtual hosts on one IP
- Host header (with FQDN) only available after SSLhandshake
- So, which certificate to use?
- Client provides servername in TLS extension
- Server (or ADC) can send specific certificate





Certificate types



- Self-singed
 - No identity verification (Of course you can trust me!!!)
- Domain Validated
 - Just ownership of the domain (automatically) checked
- Organization Validated (High Assurance)
 - Also (manual) organization check
- Extended Validation
 - Only audited CAs may provide EV certificates
 - Establish the legal identity as well as the operational and physical presence of website owner.
 - Establish that the applicant is the domain name owner or has exclusive control
 - Confirm the identity and authority of the individuals acting for the website owner
 - No wildcard certificates





Free Certificates



- Self-Signed certificates
- Create your own CA
- CAcert
 - 6 month expiration (24 with validation)
 - Community validation (face-to-face)
 - Root CA currently not trusted by many browsers
- Let's Encrypt
 - 90 day expiration
 - Automatic domain validation, no OV or EV certificates
 - No mail or code-signing certificates
- Some CA's offer free certificates for Open Source projects
- Free trials



FREE (AS IN) BEER.

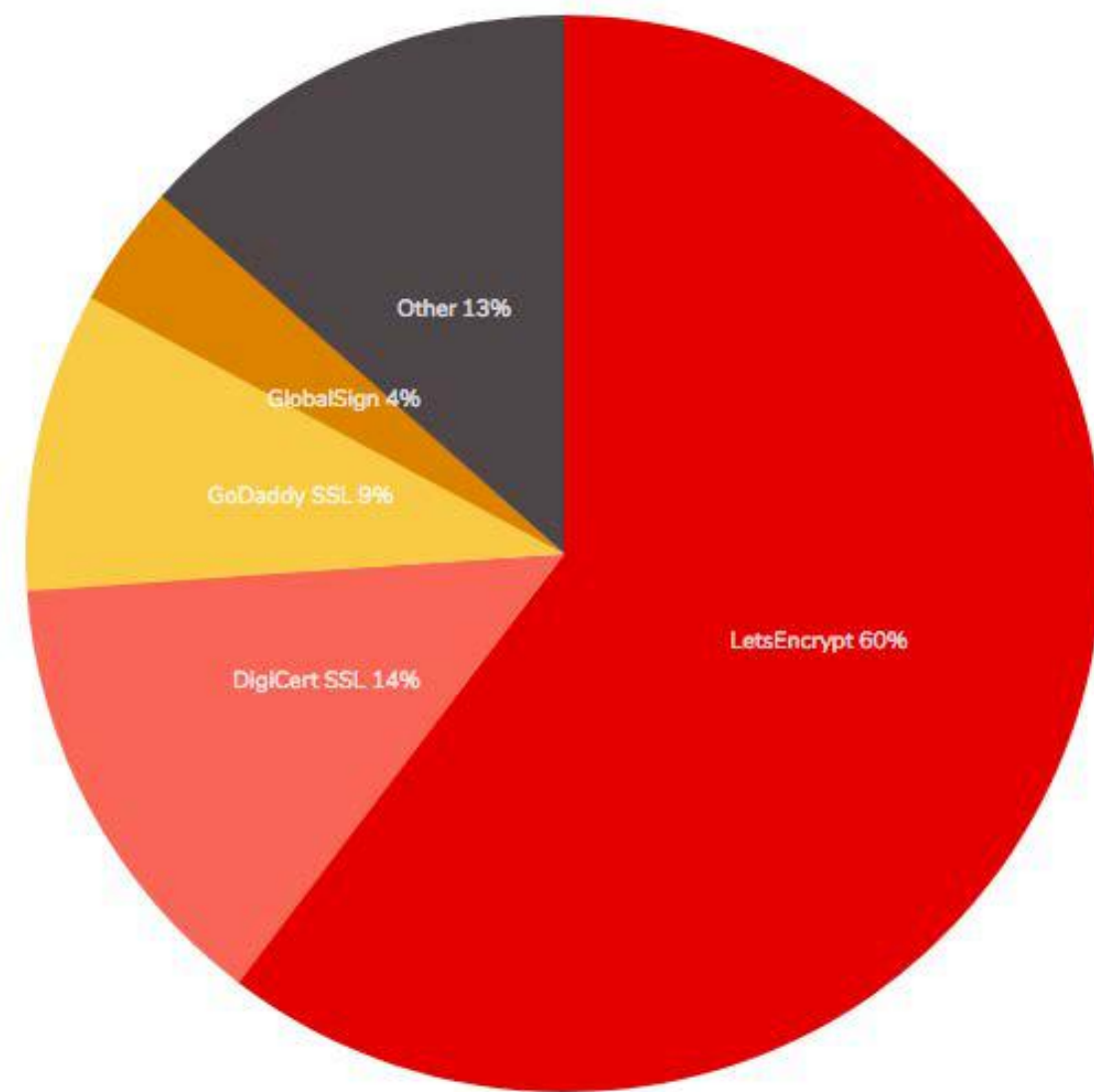


Who do we trust the most?



Root Authority Usage Distribution in the Top 1 Million Sites

Statistics for websites using Root Authority providers

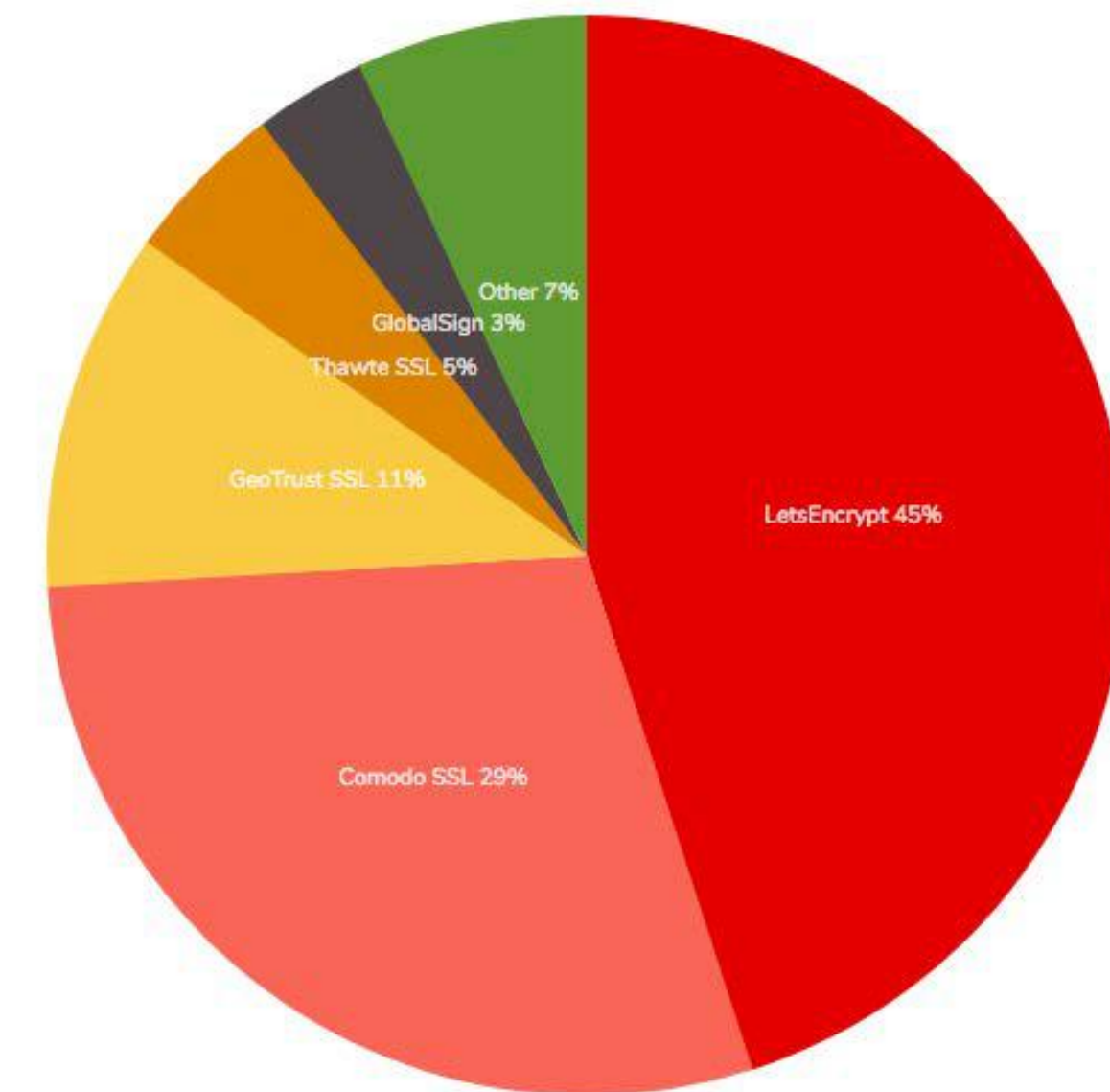


Top In Root Authority Usage Distribution in the Top 1 Million Sites

Technology	Websites	%
LetsEncrypt	401,152	40.12
DigiCert SSL	90,198	9.02
GoDaddy SSL	59,762	5.98
GlobalSign	24,434	2.44
GeoTrust SSL	18,144	1.81

Root Authority Usage Distribution in the Netherlands

Statistics for websites using Root Authority providers



Top In Root Authority Usage Distribution in the Netherlands

Technology	Websites	%
LetsEncrypt	119,253 *	45.08
Comodo SSL	76,847	29.05
GeoTrust SSL	28,533	10.79
Thawte SSL	12,926	4.89
GlobalSign	8,815	3.33

<https://trends.builtwith.com/ssl/>



Trust!



Demo & Exercise

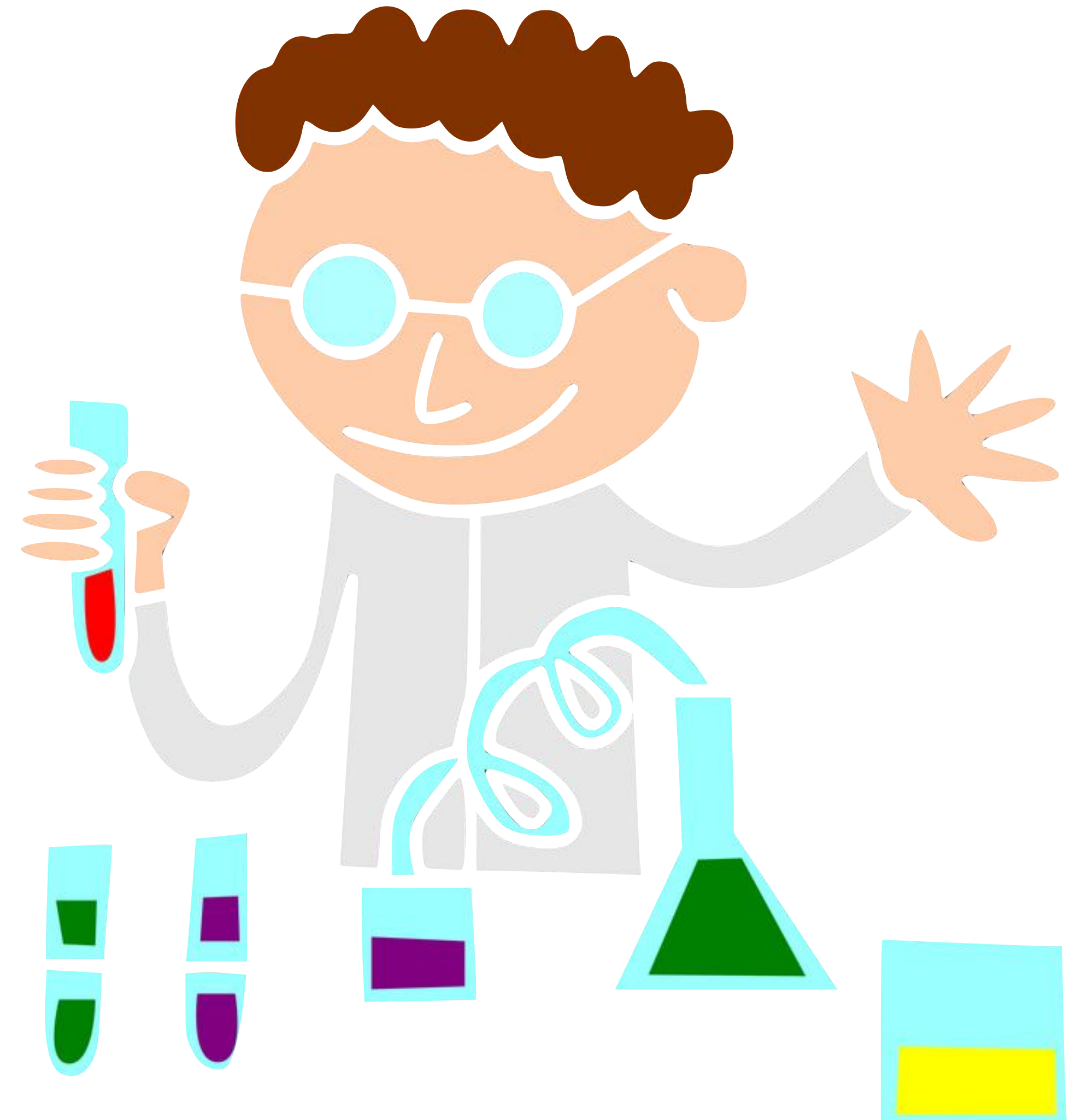


- Demo

- <https://nu.nl>
- <https://tweakers.net>

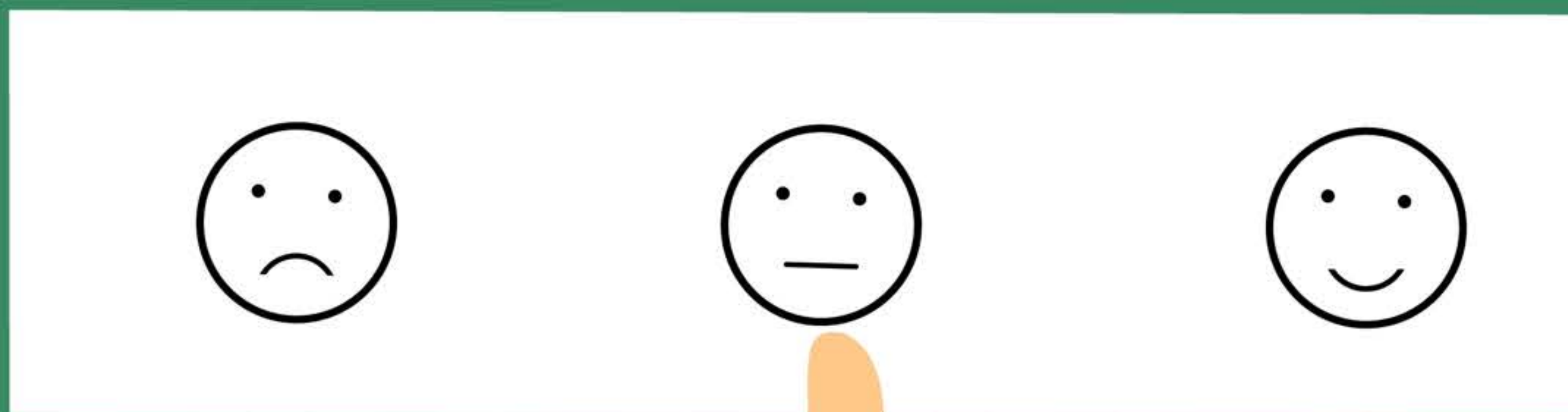
- Exercise

- Visit 5 to 10 of your favorite websites
 - Are they redirecting to HTTPS?
 - What is the common name of the certificate?
 - Are there any alternative names?
 - In which period is the certificate valid?
 - What kind of public key is used? How many bits?
 - Which signig method was used?
 - Is it an DV, OV or EV certificate?
 - How many intermediate certificates? Who is the Root CA?
- Write the details down for the 5th site you visit





Time for a poll

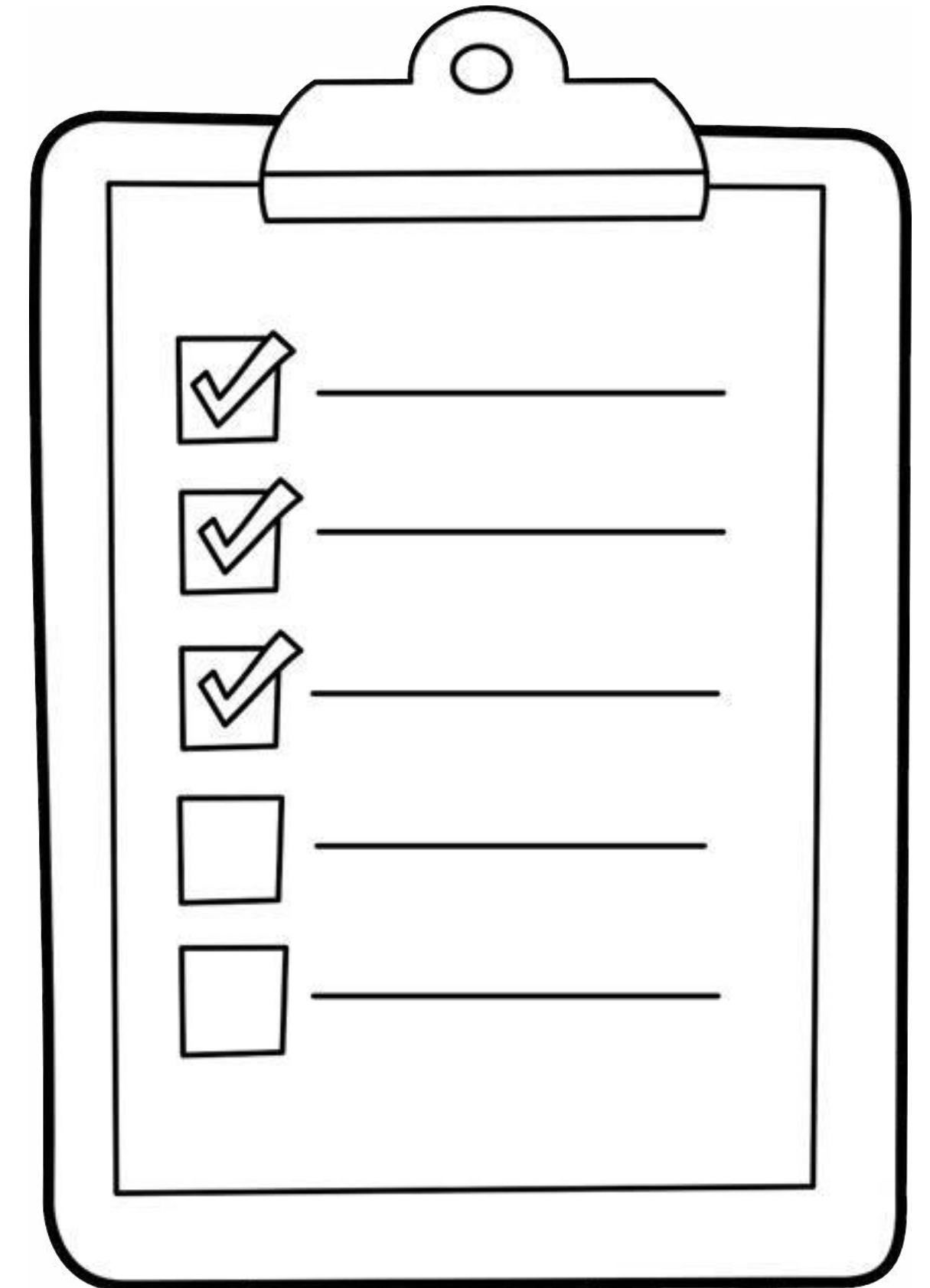




Agenda



- TLS fundamentals
 - The need for TLS (and what happens to SSL?)
 - Cryptology 101 & PKI
 - **Troubleshoot the TLS handshake (SSLv3 - TLSv1.2)**
 - Troubleshoot the TLS handshake (TLSv1.3)
- Analysing Application Data
 - Without decryption
 - With decryption using the server's private key
 - With decryption using TLS session keys
- Creating configuration profiles for TLS analysis
- SSL/TLS vulnerabilities & privacy concerns
- Summary and Q&A



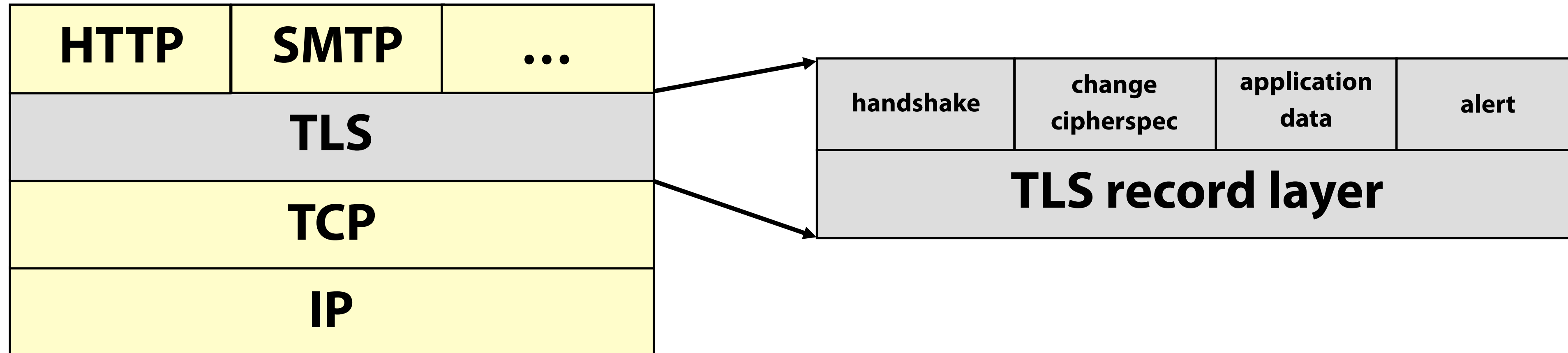
<https://www.needpix.com/photo/download/94588/checklist-action-check-list-verification-tick-approved-okay-check-mark>



Place in TCP/IP stack



- Between transport and application layer
- Protocol independent

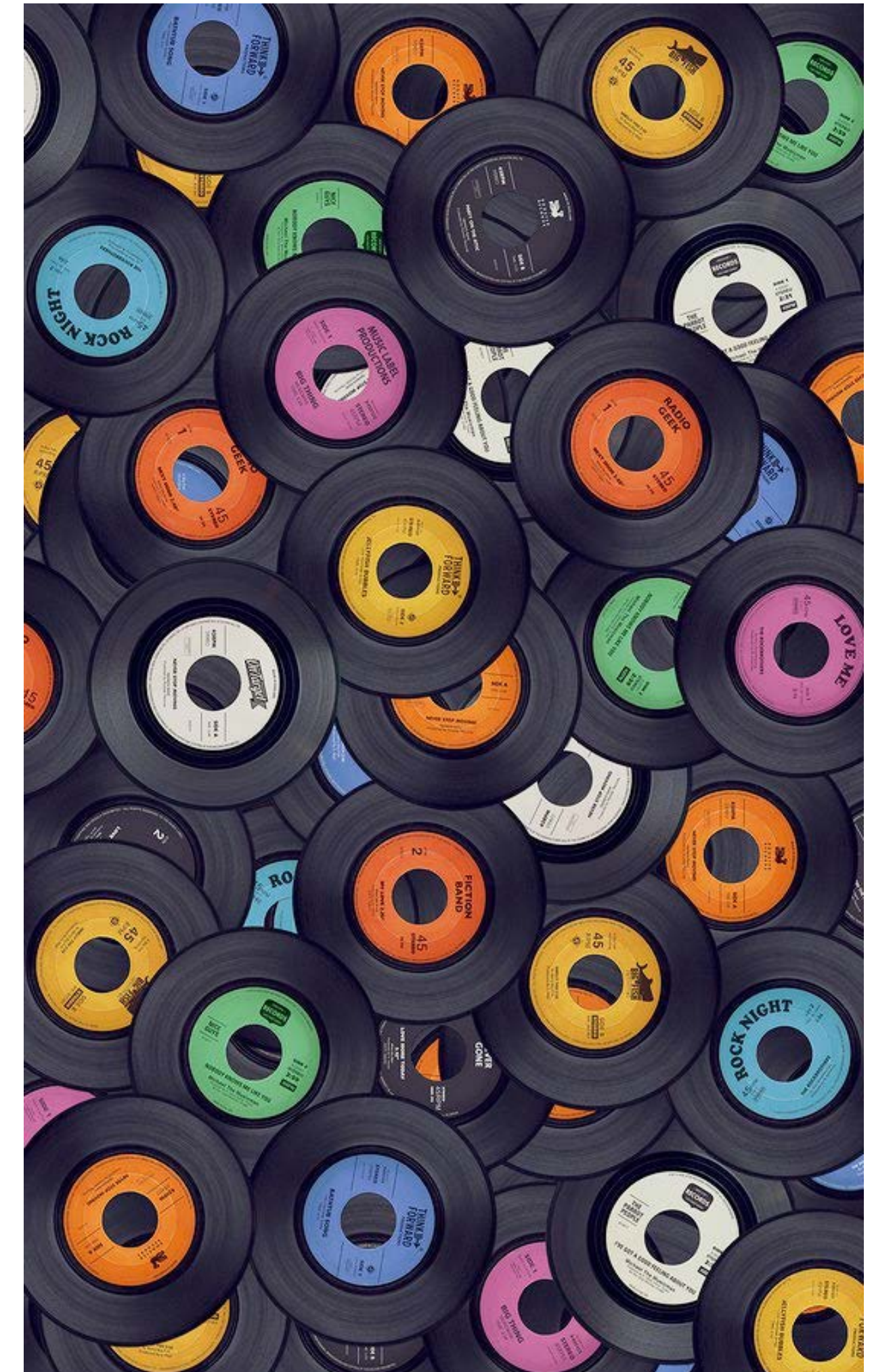




SSL Record Layer



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





SSL Content Types

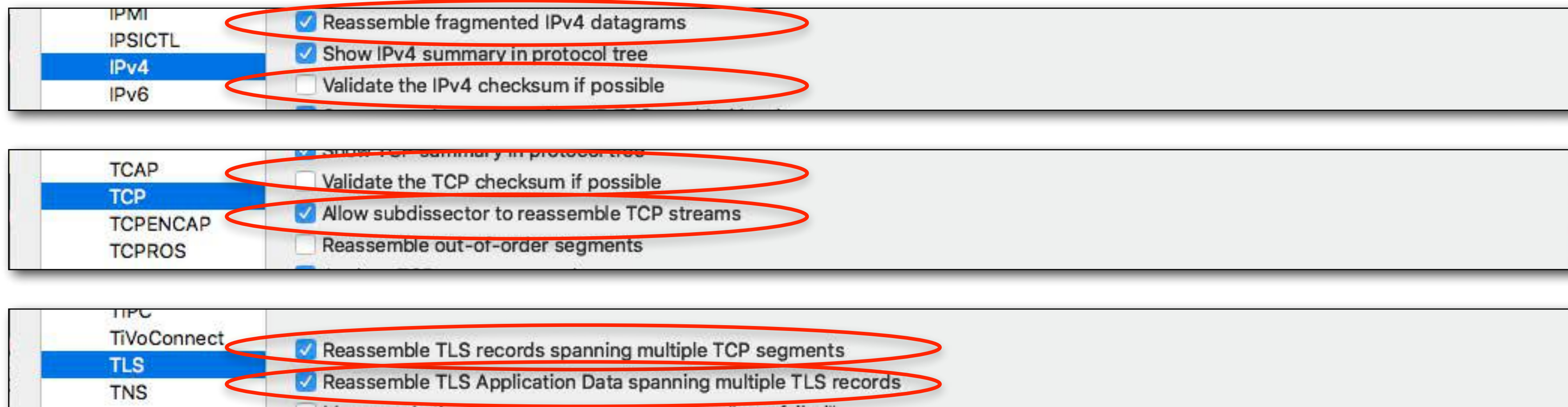


- Handshake Protocol (0x16)
 - responsible for authentication and session key setup
- ChangeCipherSpec Protocol (0x14)
 - Notify start of encryption
- Alert Protocol (0x15)
 - Reporting of warnings and fatal errors
- Application Protocol (0x17)
 - Actual encryption and transport of data





Use the right protocol preferences



```
sake@MacSake:~$ tshark -G currentprefs | egrep "^#?(ip|tcp|tls).(defrag|deseg|check)"
#ip.defragment: TRUE
#ip.check_checksum: FALSE
#tcp.check_checksum: FALSE
#tcp.desegment_tcp_streams: TRUE
#tls.desegment_ssl_records: TRUE
#tls.desegment_ssl_application_data: TRUE
sake@MacSake:~$
```

Luckily those are all default settings



Analyzing the TLS record layer (1)



The image shows a Wireshark packet capture analysis of a TLS Client Hello message. The packet list pane shows packet 4 as an SSL Client Hello. The packet details pane is expanded to show the TLSv1 Record Layer, which contains a Handshake Protocol: client hello. The content type is Handshake (22), the version is TLS 1.0 (0x0301), and the length is 65 bytes. The handshake protocol: client hello is also expanded. The packet bytes pane shows the raw data in hexadecimal and ASCII.

No.	Time	Source	Destination	Protocol	Info
4	0.011511	192.168.3.1	192.168.3.3	SSL	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

Frame 4 (124 bytes on wire, 124 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: 18736 (18736), dst port: https (443), seq: 1, ack: 1, len: 70
- 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

```
0000  00 0c 29 5d c5 66 00 50 56 c0 00 01 08 00 45 00  ..)].f.P V.....E.
0010  00 6e 42 29 40 00 80 06 31 0c c0 a8 03 01 c0 a8  .nB)@... 1.....
0020  03 03 49 30 01 bb 21 62 08 73 02 3e 54 89 50 18  ..I0..!b .s.>T.P.
0030  fa 00 67 eb 00 00 16 03 01 00 41 01 00 00 3d 03  ..g..... ..A...=.
0040  01 49 eb 46 9e dd 81 95 16 fc 5d dd d0 97 42 8d  .I.F.... ..]...B.
0050  41 0d 78 52 e2 57 9e 2e 89 03 cd b3 31 c7 63 dc  A.XR.W.. ....1.c.
0060  a9 00 00 10 00 84 00 35 00 41 00 04 00 05 00 2f  .....5 .A...../
0070  fe ff 00 0a 01 00 00 04 00 23 00 00  ..#..
```

File: "C:\cygwin\home\sablo\sharkfest\2009\traces\session-reuse.cap" 13 KB 00:02:17 Packets: 56 Displayed: 56 ... Profile: ...



Analyzing the SSL record layer (2)



No.	Time	Source	Destination	Protocol	Info
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 Message
10	0.038327	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: 18736 (18736), seq: 1, ack: 71, len: 1460
- Secure Socket Layer
 - TLSv1 Record Layer: Handshake Protocol: Server Hello

0x091c = 2332 bytes

Record layer (ssl.record), 79 bytes

Packets: 56 Displayed: 56 ... Profile: ...



Analyzing the SSL record layer (3)



Packet list:

No.	Time	Source	Destination	Protocol	Info
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

Packet 7 details:

- Frame 7 (1019 bytes on wire, 1019 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: 18736 (18736), Seq: 1461, Ack: 71, Len: 965
- [Reassembled TCP segments (2346 bytes): #6(1381), #7(965)]
 - [Frame: 6, payload: 0-1380 (1381 bytes)]
 - [Frame: 7, payload: 1381-2345 (965 bytes)]
- Secure socket Layer
 - TLSv1 Record Layer: Handshake Protocol: Certificate
 - Content Type: Handshake (22)
 - Version: TLS 1.0 (0x0301)
 - Length: 2332
 - 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

Calculation: $(5+2332) + (5+4) = 2346$

Hex dump:

```
0000  00 50 56 c0 00 01 00 0c 29 5d c5 66 08 00 45 00  .PV.....)].f..E.
0010  03 ed 6b e5 40 00 40 06 43 d1 c0 a8 03 03 c0 a8  ..k.@.@.C.....
0020  03 01 01 bb 49 30 02 3e 5a 3d 21 62 08 b9 50 18  ....IO.>Z=!b..P.
0030  01 6d 02 00 00 00 02 ff 04 02 12 12 52 68 61 77  w  u  char
```

Frame (1019 bytes) | Reassembled TCP (2346 bytes)

File: "C:\cygwin\home\sablo\sharkfest\2009\traces\session-reuse.cap" 13 KB 00:02:17 | Packets: 56 Displayed: 56 ... | Profile: ...



Troubleshooting case



- Upgrade F5 loadbalancer
- Site stops working
- F5 does not see response from backend server
- Trace in the backend:

ion	Protocol	Length	Record Length	Info
16.118	TLSv1.2	552	448	Application Data
64.37	TCP	99		443 → 25298 [ACK] Seq=4074 Ack=773 Win=64133 Len=0
64.37	TCP	1507		443 → 25298 [ACK] Seq=4074 Ack=773 Win=64133 Len=1408 [TCP
64.37	TCP	1507		443 → 25298 [ACK] Seq=5482 Ack=773 Win=64133 Len=1408 [TCP
64.37	TCP	1507		443 → 25298 [ACK] Seq=6890 Ack=773 Win=64133 Len=1408 [TCP
64.37	TCP	1507		443 → 25298 [ACK] Seq=8298 Ack=773 Win=64133 Len=1408 [TCP
64.37	TCP	1507		443 → 25298 [ACK] Seq=9706 Ack=773 Win=64133 Len=1408 [TCP
16.118	TCP	99		25298 → 443 [ACK] Seq=773 Ack=11114 Win=15373 Len=0
64.37	TCP	1507		443 → 25298 [ACK] Seq=11114 Ack=773 Win=64133 Len=1408 [TCP
64.37	TCP	1507		443 → 25298 [ACK] Seq=12522 Ack=773 Win=64133 Len=1408 [TCP
64.37	TCP	1507		443 → 25298 [ACK] Seq=13930 Ack=773 Win=64133 Len=1408 [TCP
64.37	TCP	1507		443 → 25298 [ACK] Seq=15338 Ack=773 Win=64133 Len=1408 [TCP
64.37	TCP	1507		443 → 25298 [ACK] Seq=16746 Ack=773 Win=64133 Len=1408 [TCP
64.37	TCP	1507		443 → 25298 [ACK] Seq=18154 Ack=773 Win=64133 Len=1408 [TCP
64.37	TLSv1.2	1507	16464	Application Data [TCP segment of a reassembled PDU]
64.37	TCP	1507		443 → 25298 [ACK] Seq=20970 Ack=773 Win=64133 Len=1408 [TCP
64.37	TCP	1507		443 → 25298 [ACK] Seq=22378 Ack=773 Win=64133 Len=1408 [TCP
16.118	TCP	99		25298 → 443 [ACK] Seq=773 Ack=23786 Win=28045 Len=0
16.118	TCP	87		25298 → 443 [RST, ACK] Seq=773 Ack=23786 Win=0 Len=0

```
▶ Transmission Control Protocol, Src Port: 443, Dst Port: 25298, Seq: 19562, Ack: 773, Len
▶ [12 Reassembled TCP Segments (16469 bytes): #19(1408), #20(1408), #21(1408), #22(1408),
▼ Transport Layer Security
  ▼ TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
    Content Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 16464
    Encrypted Application Data: 2ca1cbf1fa261fdb359ae7f6dbd2a0f6ac2149b489b346c8..
```

Too Large???



Clientside trace of working version



No.	Time	Delta	#stream	Source	Destination	Protocol	Length	Record Length	Info
35	12:55:57.787979	0.000000	0	10.0.2.104	137.224.9.101	TLSv1.2	647	576	Application Data
41	12:55:58.838822	1.050843	0	137.224.9.101	10.0.2.104	TLSv1.2	215	3040	Application Data
52	12:55:58.839278	0.000456	0	137.224.9.101	10.0.2.104	TLSv1.2	1514	7280	Application Data [TCP segment of a reassembled PDU]
56	12:55:58.839282	0.000004	0	137.224.9.101	10.0.2.104	TLSv1.2	1514	6320	Application Data [TCP segment of a reassembled PDU]
74	12:55:58.859782	0.020500	0	137.224.9.101	10.0.2.104	TLSv1.2	1514	16432	Application Data [TCP segment of a reassembled PDU]
91	12:55:58.875755	0.015973	0	137.224.9.101	10.0.2.104	TLSv1.2	1514	16432	Application Data [TCP segment of a reassembled PDU]
107	12:55:58.899097	0.023342	0	137.224.9.101	10.0.2.104	TLSv1.2	171	8640	Application Data

```

▶ Frame 74: 1514 bytes on wire (1514 bytes captured on interface en0, id 0)
▶ Ethernet II, Src: PaloAlto_b9:7a:31 (e8:98:6d:b9:7a:31), Dst: Apple_cb:26:45 (ac:bc:62:26:45:00)
▶ Internet Protocol Version 4, Src: 137.224.9.101, Dst: 10.0.2.104
▶ Transmission Control Protocol, Src Port: 443, Dst Port: 60572, Seq: 37058, Ack: 128
▼ [12 Reassembled TCP Segments (16437 bytes): #56(870), #57(1448), #61(1448), #62(1448), #63(1448), #64(1448), #67(1448), #68(1448), #69(1448), #70(1448), #73(1448), #74(1087)]
  [Frame: 56, payload: 0-869 (870 bytes)]
  [Frame: 57, payload: 870-2317 (1448 bytes)]
  [Frame: 61, payload: 2318-3765 (1448 bytes)]
  [Frame: 62, payload: 3766-5213 (1448 bytes)]
  [Frame: 63, payload: 5214-6661 (1448 bytes)]
  [Frame: 64, payload: 6662-8109 (1448 bytes)]
  [Frame: 67, payload: 8110-9557 (1448 bytes)]
  [Frame: 68, payload: 9558-11005 (1448 bytes)]
  [Frame: 69, payload: 11006-12453 (1448 bytes)]
  [Frame: 70, payload: 12454-13901 (1448 bytes)]
  [Frame: 73, payload: 13902-15349 (1448 bytes)]
  [Frame: 74, payload: 15350-16436 (1087 bytes)]
  [Segment count: 12]
  [Reassembled TCP length: 16437]
  [Reassembled TCP Data: 1703034030652ef030572d54baf78cc5347f94c0b852f61b...]
▼ Transport Layer Security
  ▼ TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
    Content Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 16432
    Encrypted Application Data: 652ef030572d54baf78cc5347f94c0b852f61b82f59aaf44...
  
```

```

Transport Layer Security
▼ TLSv1.2 Record Layer: Application Data Protocol: http-over-tls
  Content Type: Application Data (23)
  Version: TLS 1.2 (0x0303)
  Length: 16432
  Encrypted Application Data: 652ef030572d54baf78cc5347f94c0b852f61b82f59aaf44...
  [Reassembled PDU in frame: 107]
  TLS segment data (16384 bytes)
  
```

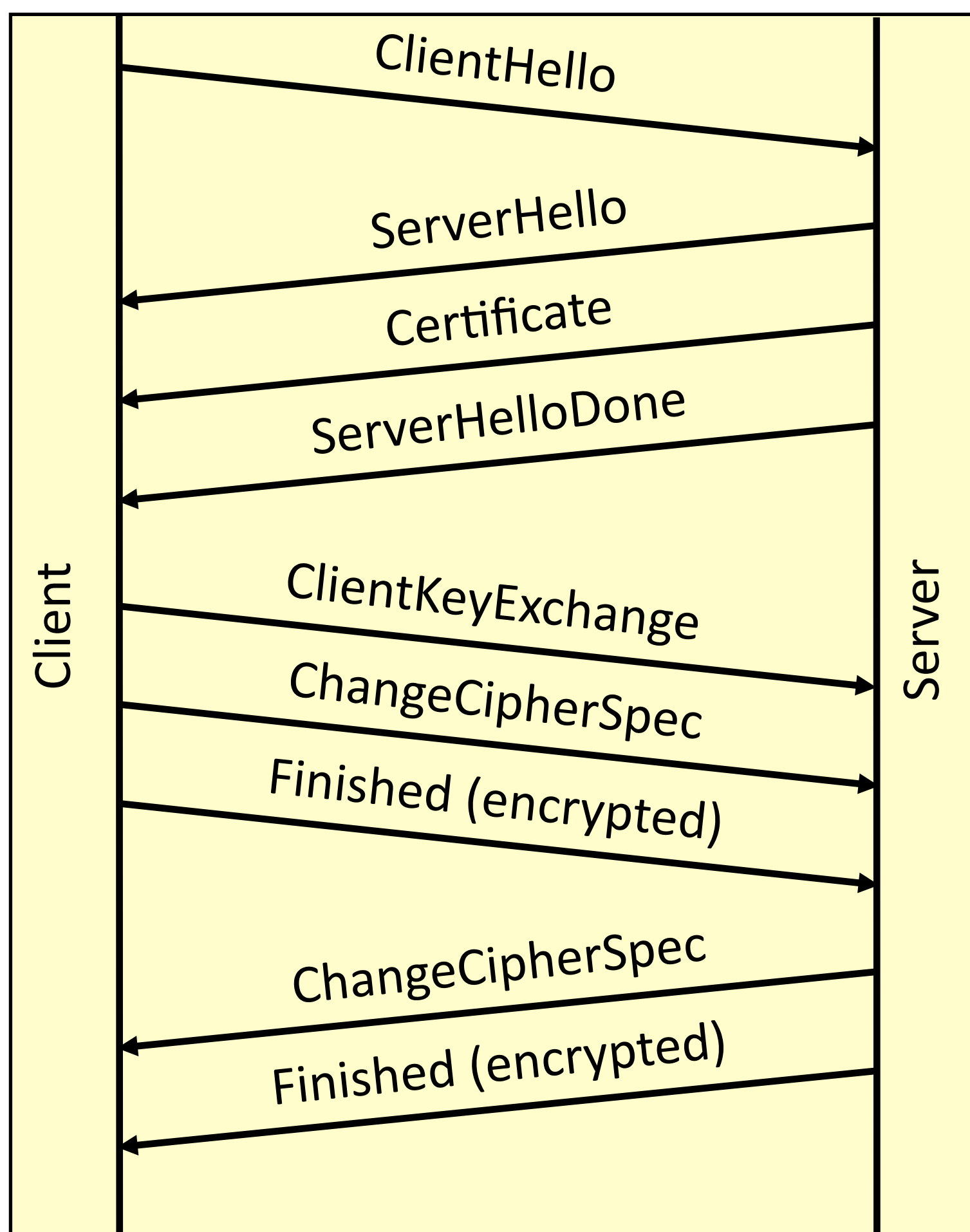
Space in header name!!!

```

[6 Reassembled TLS segments (57882 bytes): #41(3003), #52(7240), #56(6269), #74(16384), #91(16384), #107(8602)]
  [Frame: 41, payload: 0-3002 (3003 bytes)]
  [Frame: 52, payload: 3003-10242 (7240 bytes)]
  [Frame: 56, payload: 10243-16511 (6269 bytes)]
  [Frame: 74, payload: 16512-32895 (16384 bytes)]
  [Frame: 91, payload: 32896-49279 (16384 bytes)]
  [Frame: 107, payload: 49280-57881 (8602 bytes)]
  [Segment count: 6]
  [Reassembled PDU length: 57882]
  [Reassembled PDU data: 485454502f312e3120323030204f4b0d0a43616368652d43...]
Hypertext Transfer Protocol
▶ HTTP/1.1 200 OK\r\n
  Cache-Control: private\r\n
  Content-Type: text/html; charset=utf-8\r\n
  Server: \r\n
  Set-Cookie: ASP.NET_SessionId=0qkerxhtzgsr5ijspqs3uio4; path=/; HttpOnly;
  X-AspNet-Version: \r\n
  X-Xss-Protection: 1; mode=block\r\n
  X-Content-Type-Options: nosniff\r\n
  Referrer-Policy: no-referrer\r\n
  Access-Control-Allow-Origin : http://MetaBaseOpenServices.wecr.wur.nl\r\n
  Date: Sat, 10 Oct 2020 10:55:58 GMT\r\n
  Content-Length: 57326\r\n
  
```



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
│           gmtime_unix_time: Apr 19, 2009 17:43:26.0000000000
│           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
│           gmtime_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)



```
[-] Certificate ()
  [-] signedCertificate
    version: v3 (2)
    serialNumber: 2
    [-] signature (shawithRSAEncryption)
    [-] issuer: rdnSequence (0)
      [-] rdnSequence: 5 items ()
        [-] RDNSequence: 1 item ()
        [-] RDNSequence: 1 item ()
        [-] RDNSequence: 1 item ()
        [-] RDNSequence: 1 item ()
        [-] RelativeDistinguishedName
          Id: 2.5.4.3 (id-at-commonName)
          [-] Directorystring: printablestring (1)
            printablestring: Sharkfest Lab Server CA
          [-] RDNSequence: 1 item ()
      [-] validity
      [-] subject: rdnSequence (0)
      [-] subjectPublicKeyInfo
      [-] extensions: 4 items
    [-] algorithmIdentifier (shawithRSAEncryption)
      Padding: 0
      encrypted: 739D20C79873ADD406549E824AE1304525EEA1A5E185FB0B...
    Certificate Length: 1240
  [-] Certificate ()
    [-] signedCertificate
      version: v3 (2)
      serialNumber: 1
      [-] signature (shawithRSAEncryption)
      [-] issuer: rdnSequence (0)
      [-] validity
      [-] subject: rdnSequence (0)
        [-] rdnSequence: 5 items ()
          [-] RDNSequence: 1 item ()
          [-] RDNSequence: 1 item ()
          [-] RDNSequence: 1 item ()
          [-] RDNSequence: 1 item ()
          [-] RelativeDistinguishedName
            Id: 2.5.4.3 (id-at-commonName)
            [-] Directorystring: printablestring (1)
              printablestring: Sharkfest Lab Server CA
```



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
```



Analyzing TLS 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...
```

Later today: More on analysing Application Data



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)



Reusing SSL sessions

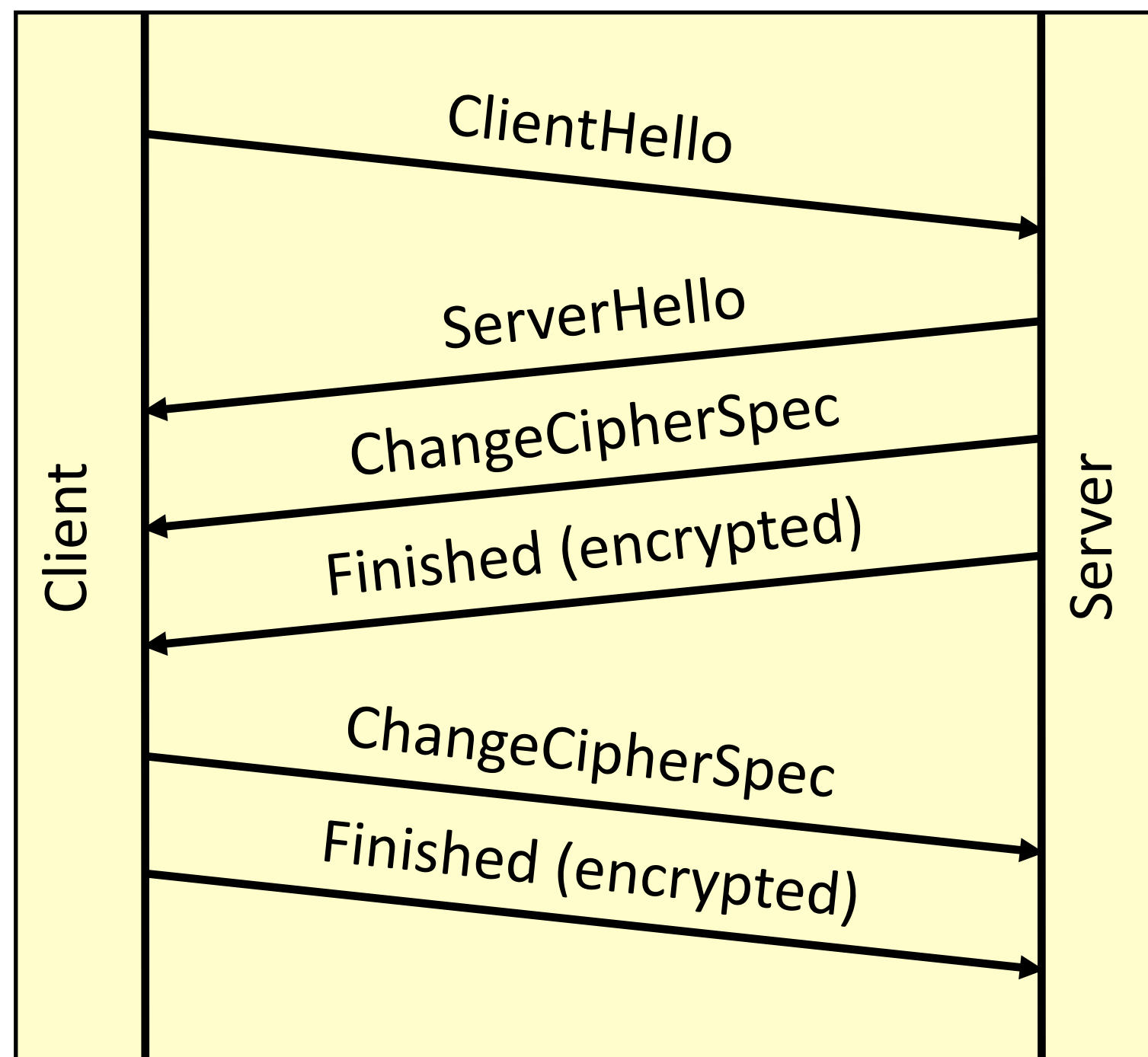


- Key negotiation "expensive"
- Cache session keys and re-use for new TCP sessions
- SSL session ID is used as Index
- Timeout on SSL session ID is an "absolute timeout" not an "idle timeout"
 - Old IE: 2 minutes, now 10 hours





Handshake of a Reused Session



No.	Time	Source	Destination	Protocol	Info
23	39.687726	192.168.3.1	192.168.3.3	TCP	18774 > https [SYN] seq=0 win=65535 Len=0 MSS=1460 WS=1
24	39.688101	192.168.3.3	192.168.3.1	TCP	https > 18774 [SYN, ACK] seq=0 Ack=1 win=5840 Len=0 MSS=1460 WS=1
25	39.688149	192.168.3.1	192.168.3.3	TCP	18774 > https [ACK] seq=1 Ack=1 win=128000 Len=0
26	39.688711	192.168.3.1	192.168.3.3	TLSv1	Client Hello
27	39.688988	192.168.3.3	192.168.3.1	TCP	https > 18774 [ACK] seq=1 Ack=103 win=5840 Len=0
28	39.694301	192.168.3.3	192.168.3.1	TLSv1	Server Hello, Change Cipher Spec, Encrypted Handshake Message
29	39.717354	192.168.3.1	192.168.3.3	TLSv1	change cipher spec, Encrypted Handshake Message, Application Data



Adding a column for SSL session ID



Filter: `ssl.handshake` Expression... Clear Apply

No.	Time	Source	Destination	Protocol	ssl-id len	ssl-id	Info
4	0.011511	192.168.3.1	192.168.3.3	SSL	0		Client Hello
6	0.017431	192.168.3.3	192.168.3.1	TLSv1	32	DB00C2	Server Hello,
7	0.017782	192.168.3.3	192.168.3.1	TLSv1			Certificate
9	0.026711	192.168.3.1	192.168.3.3	TLSv1			Client Key Exchange, Change Cipher Spec, Encry
10	0.038327	192.168.3.3	192.168.3.1	TLSv1			Change Cipher Spec, Encrypted Handshake Messag
26	39.688711	192.168.3.1	192.168.3.3	SSL	32	DB00C2	Client Hello
28	39.694301	192.168.3.3	192.168.3.1	TLSv1	32	DB00C2	server Hello, Change cipher spec, Encrypted Ha
29	39.717354	192.168.3.1	192.168.3.3	TLSv1			Change Cipher Spec, Encrypted Handshake Messag
41	111.192869	192.168.3.1	192.168.3.3	SSL	32	DB00C2	Client Hello
43	111.197758	192.168.3.3	192.168.3.1	TLSv1	32	FBCF32	Server Hello,
44	111.197984	192.168.3.3	192.168.3.1	TLSv1			Certificate
46	111.205534	192.168.3.1	192.168.3.3	TLSv1			Client Key Exchange, Change Cipher Spec, Encry
47	111.217564	192.168.3.3	192.168.3.1	TLSv1			Change Cipher Spec, Encrypted Handshake Messag

Destination	Destination address
Protocol	Protocol
ssl-id len	Custom (ssl.handshake.session_id_length)
ssl-id	Custom (ssl.handshake.session_id)
Info	Information

Properties

Add Remove

Format: Custom Field name: `ssl.handshake.session_id`



SSL session reuse I



Filter: ssl.handshake Expression... Clear Apply

No. -	Time	Source	Destination	Protocol	ssl-id len	ssl id	Info
4	0.011511	192.168.3.1	192.168.3.3	SSL	0		Client Hello
6	0.017431	192.168.3.3	192.168.3.1	TLSv1	32	DB00Cz	Server Hello, Certificate
7	0.017782	192.168.3.3	192.168.3.1	TLSv1			Client Key Exchange, Change Cipher Spec, Encry
9	0.026711	192.168.3.1	192.168.3.3	TLSv1			Change Cipher Spec, Encrypted Handshake Messag
10	0.038327	192.168.3.3	192.168.3.1	TLSv1			
26	39.688711	192.168.3.1	192.168.3.3	SSL	32	DB00Cz	Client Hello
28	39.694301	192.168.3.3	192.168.3.1	TLSv1	32	DB00Cz	server Hello, Change Cipher spec, Encrypted Ha
29	39.717354	192.168.3.1	192.168.3.3	TLSv1			Change Cipher Spec, Encrypted Handshake Messag
41	111.192869	192.168.3.1	192.168.3.3	SSL	32	DB00Cz	Client Hello
43	111.197758	192.168.3.3	192.168.3.1	TLSv1	32	FBCF3z	server Hello, Certificate
44	111.197984	192.168.3.3	192.168.3.1	TLSv1			Client Key Exchange, Change Cipher Spec, Encry
46	111.205534	192.168.3.1	192.168.3.3	TLSv1			Change Cipher Spec, Encrypted Handshake Messag
47	111.217564	192.168.3.3	192.168.3.1	TLSv1			

Full Handshake



SSL session reuse II



Filter: ssl.handshake Expression... Clear Apply

No. -	Time	Source	Destination	Protocol	ssl-id len	ssl-id	Info
4	0.011511	192.168.3.1	192.168.3.3	SSL	0		Client Hello
6	0.017431	192.168.3.3	192.168.3.1	TLSv1	32	DB00Cz	Server Hello,
7	0.017782	192.168.3.3	192.168.3.1	TLSv1			Certificate
9	0.026711	192.168.3.1	192.168.3.3	TLSv1			Client Key Exchange, Change Cipher Spec, Encry
10	0.038327	192.168.3.3	192.168.3.1	TLSv1			Change Cipher Spec, Encrypted Handshake Messag
26	39.688711	192.168.3.1	192.168.3.3	SSL	32	DB00Cz	Client Hello
28	39.694301	192.168.3.3	192.168.3.1	TLSv1	32	DB00Cz	Server Hello, Change Cipher Spec, Encrypted Ha
29	39.717354	192.168.3.1	192.168.3.3	TLSv1			Change Cipher Spec, Encrypted Handshake Messag
41	111.192869	192.168.3.1	192.168.3.3	SSL	32	DB00Cz	Client Hello
43	111.197758	192.168.3.3	192.168.3.1	TLSv1	32	FBCF3z	server Hello,
44	111.197984	192.168.3.3	192.168.3.1	TLSv1			Certificate
46	111.205534	192.168.3.1	192.168.3.3	TLSv1			Client Key Exchange, Change Cipher Spec, Encry
47	111.217564	192.168.3.3	192.168.3.1	TLSv1			Change Cipher Spec, Encrypted Handshake Messag

Partial Handshake



SSL session reuse III



Filter: ssl.handshake Expression... Clear Apply

No. -	Time	Source	Destination	Protocol	ssl-id len	ssl-id	Info
4	0.011511	192.168.3.1	192.168.3.3	SSL	0		Client Hello
6	0.017431	192.168.3.3	192.168.3.1	TLSv1	32	DB00C2	Server Hello,
7	0.017782	192.168.3.3	192.168.3.1	TLSv1			Certificate
9	0.026711	192.168.3.1	192.168.3.3	TLSv1			Client Key Exchange, Change Cipher Spec, Encry
10	0.038327	192.168.3.3	192.168.3.1	TLSv1			Change Cipher Spec, Encrypted Handshake Messag
26	39.688711	192.168.3.1	192.168.3.3	SSL	32	DB00C2	Client Hello
28	39.694301	192.168.3.3	192.168.3.1	TLSv1	32	DB00C2	Server Hello, Change Cipher spec, Encrypted Ha
29	39.717354	192.168.3.1	192.168.3.3	TLSv1			Change Cipher Spec, Encrypted Handshake Messag
41	111.192869	192.168.3.1	192.168.3.3	SSL	32	DB00C2	Client Hello
43	111.197758	192.168.3.3	192.168.3.1	TLSv1	32	FBCF32	Server Hello,
44	111.197984	192.168.3.3	192.168.3.1	TLSv1			Certificate
46	111.205534	192.168.3.1	192.168.3.3	TLSv1			Client Key Exchange, Change Cipher Spec, Encry
47	111.217564	192.168.3.3	192.168.3.1	TLSv1			Change Cipher Spec, Encrypted Handshake Messag

Expired session

```
# Inter-Process Session Cache:
# Configure the SSL Session Cache: First the mechanism
# to use and second the expiring timeout (in seconds).
#SSLSessionCache dbm:/var/run/apache2/ssl_scache
SSLSessionCache shmcb:/var/run/apache2/ssl_scache(512000)
SSLSessionCacheTimeout 60
```



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

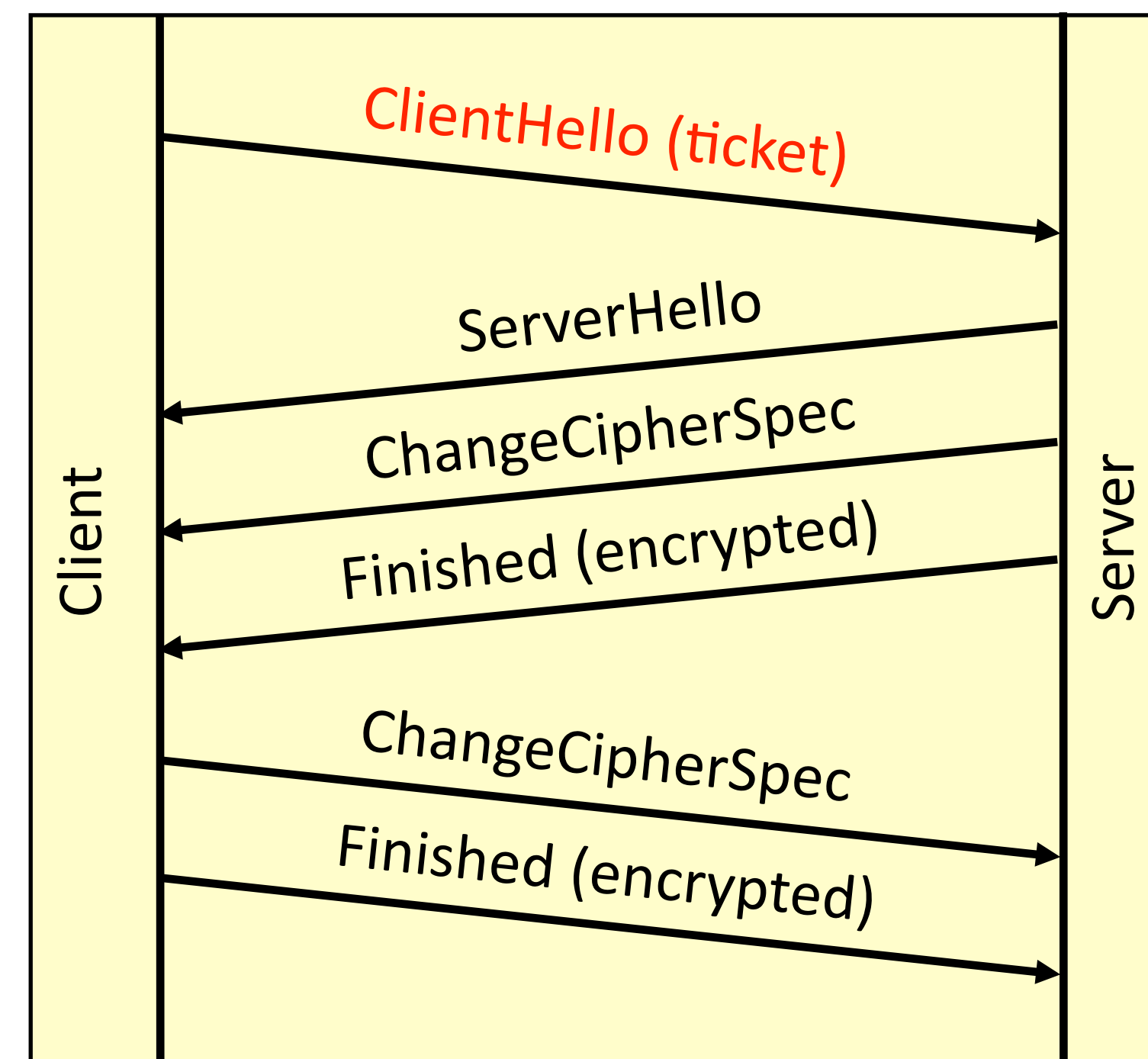
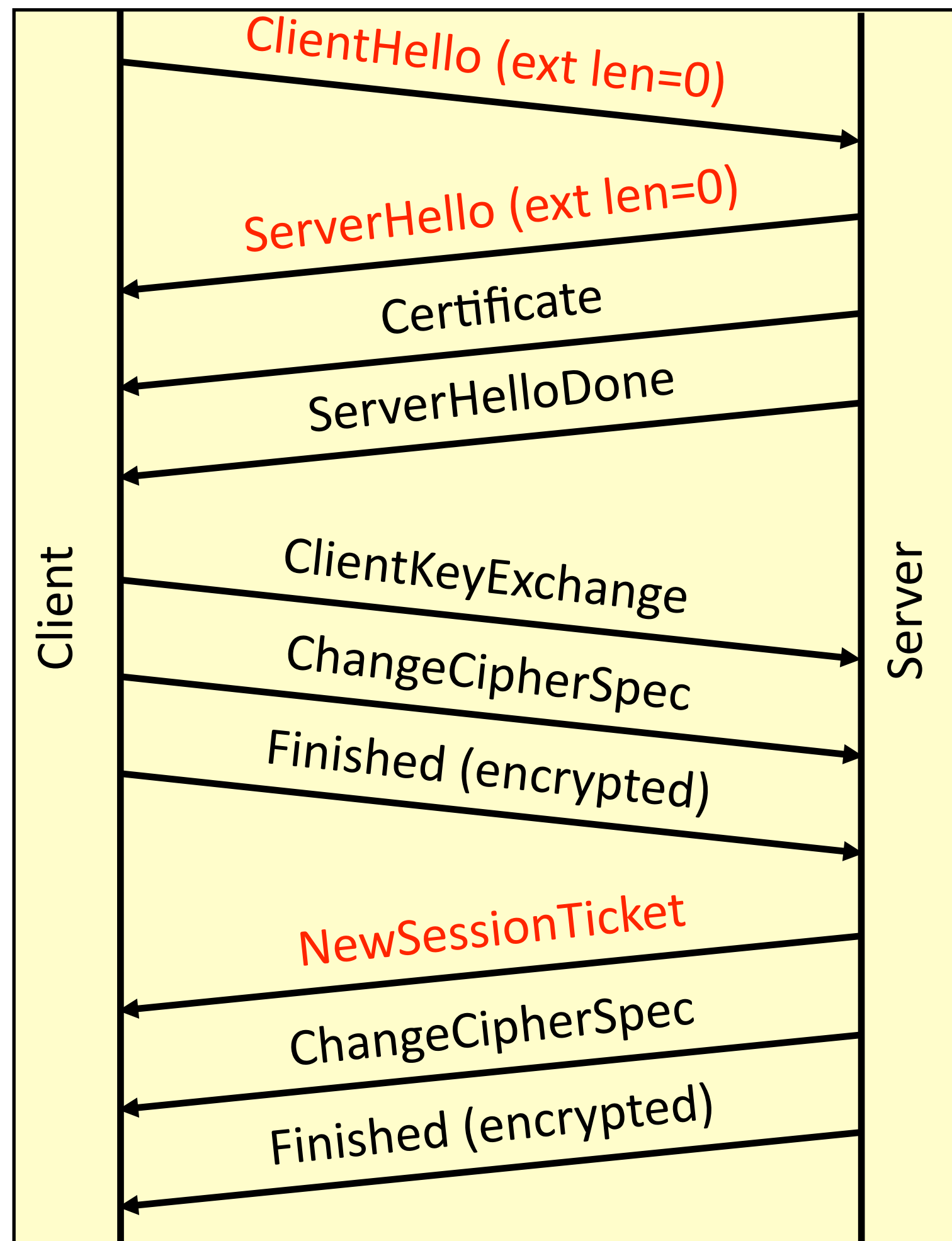


- TLS session tickets (RFC 5077)
- Do not keep state on server, only on client
- TLS extension in ClientHello and ServerHello
- New TLS HandshakeType: **NewSessionTicket**





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 (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



Common TLS handshake problems I



- Session stops right after “ClientHello”
 - No mutually supported TLS version
 - No mutually accepted cipher
 - No certificate for SNI name

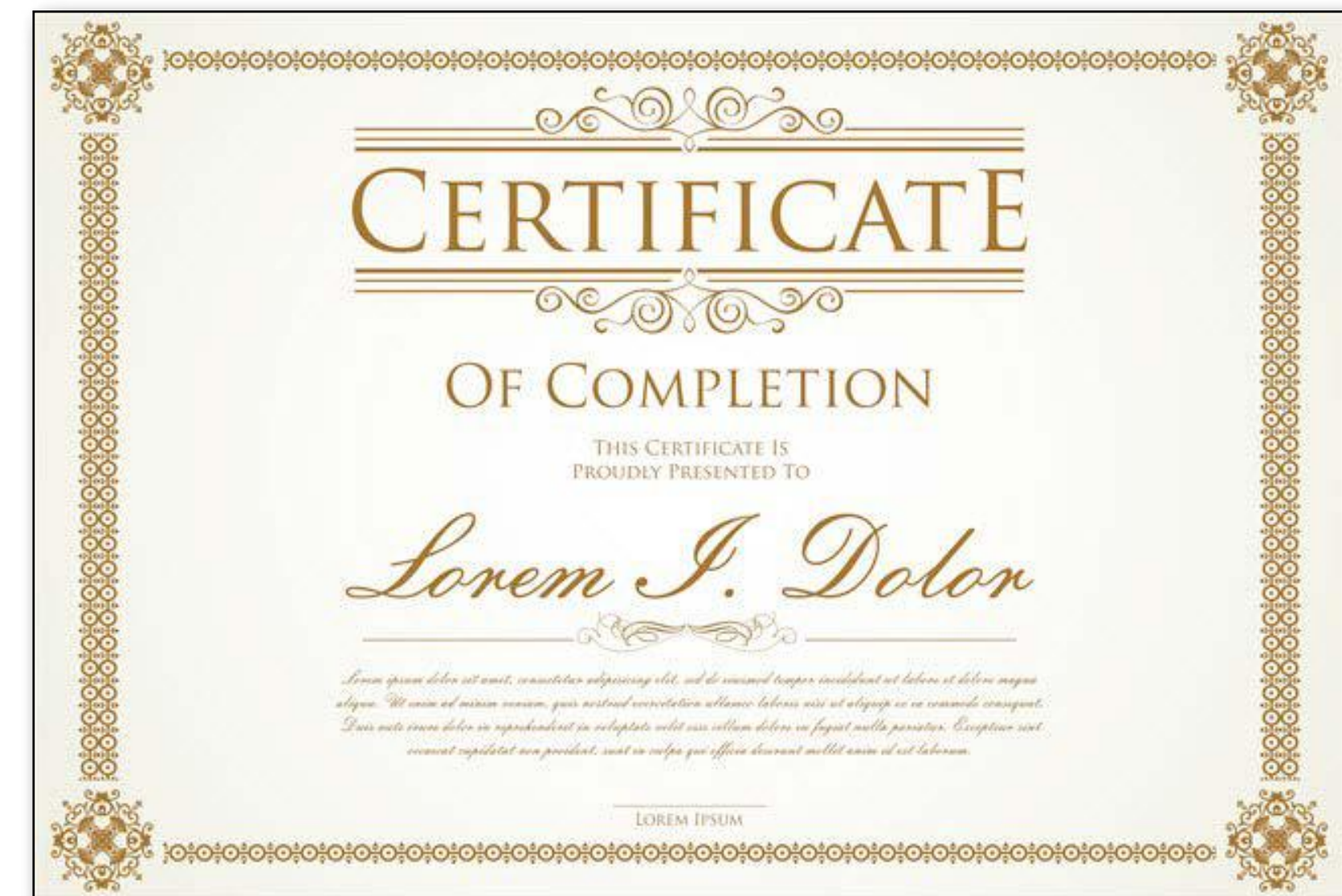




Common TLS handshake problems II



- 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

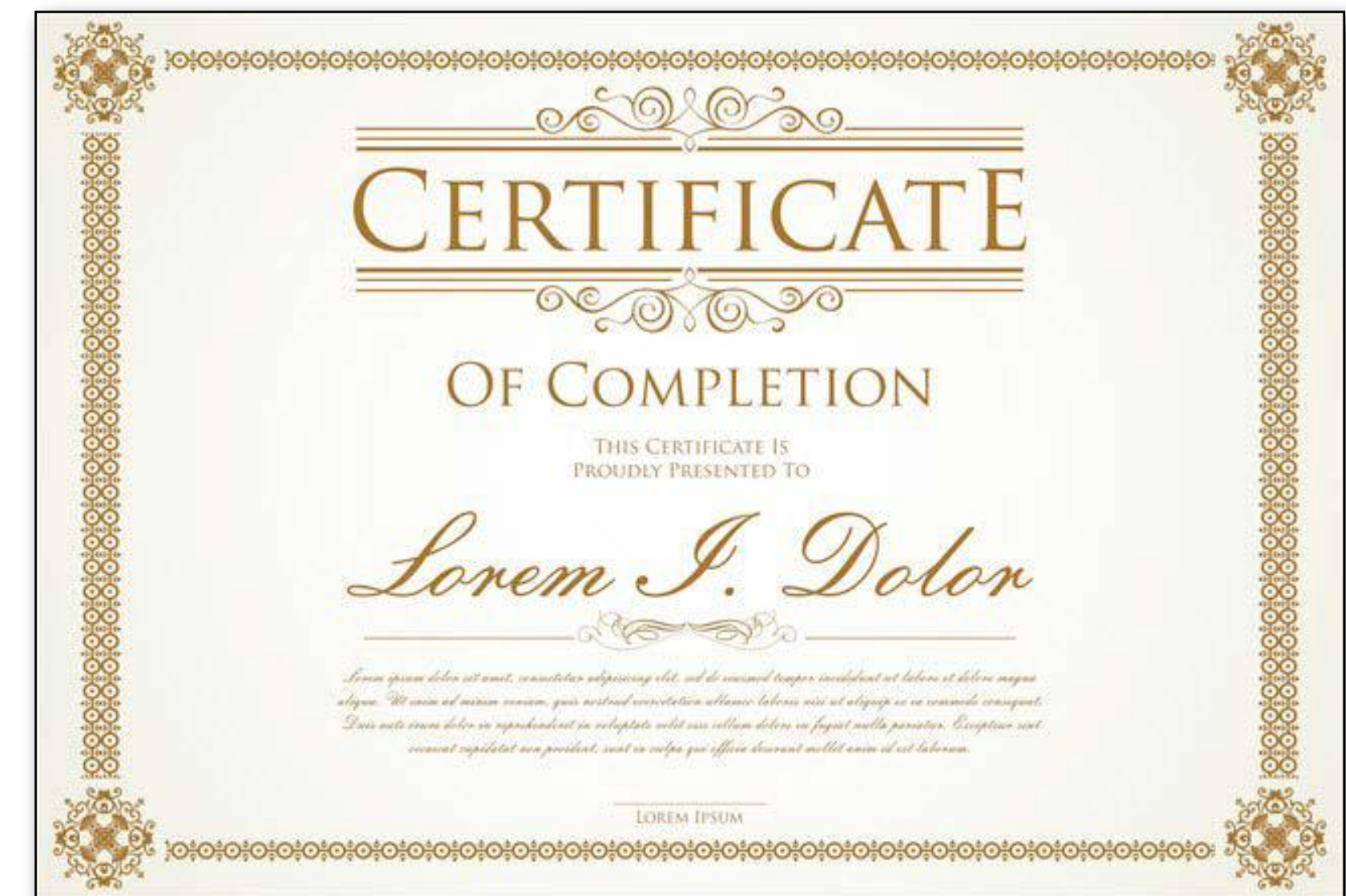




Common TLS handshake problems III

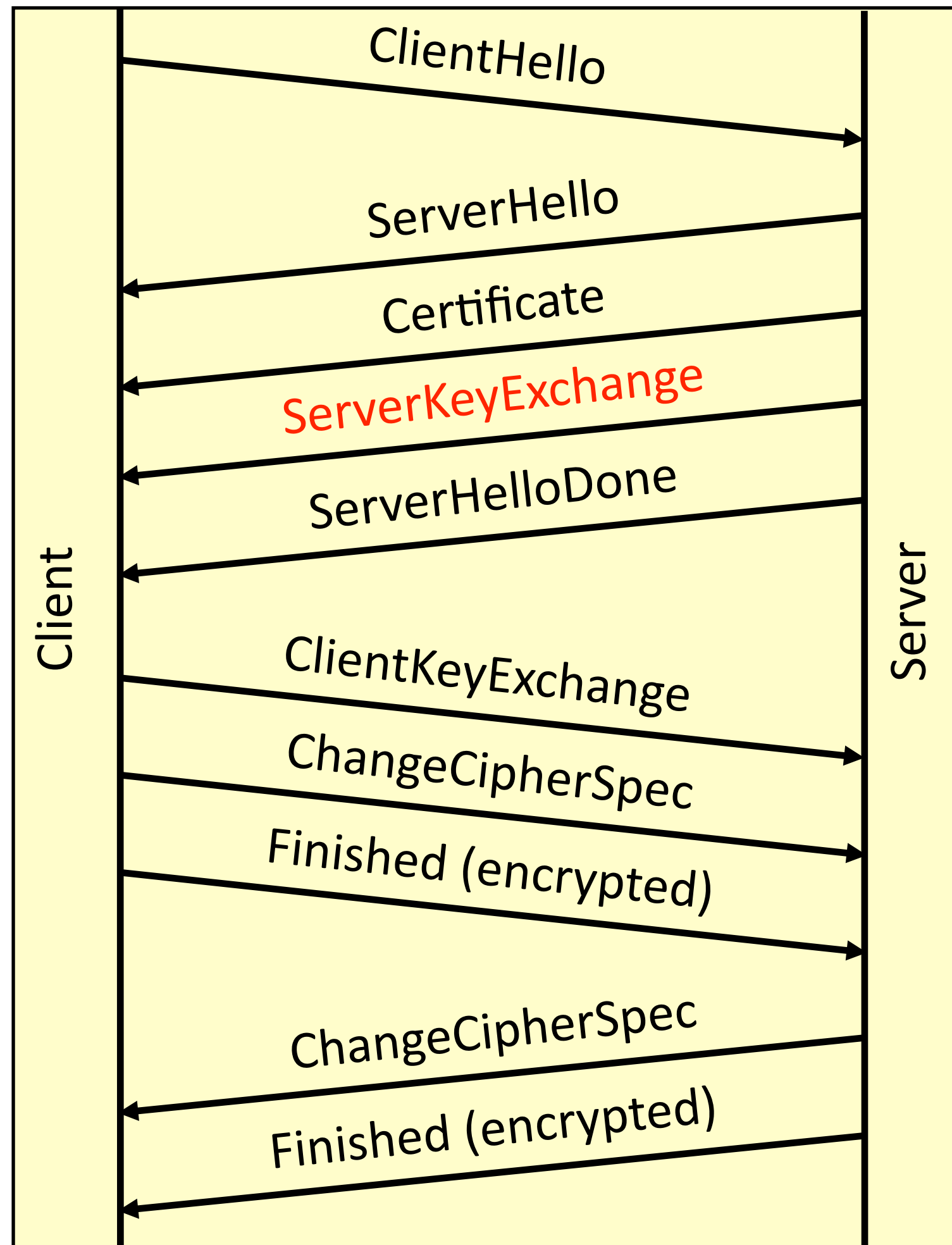


- Session stops right after “Certificate” (from the client)
 - The root certificate for the client certificate is not in the server’s trust store
 - Server has root certificate, but client does not send the intermediate CA certificate
 - Client did not send a certificate
 - perhaps it did not have a certificate signed by one of the CA’s in the DN list
 - Client certificate has expired
 - Client certificate has been revoked
 - The CRL has expired
 - Client certificate chain is too long





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 / Mutual TLS

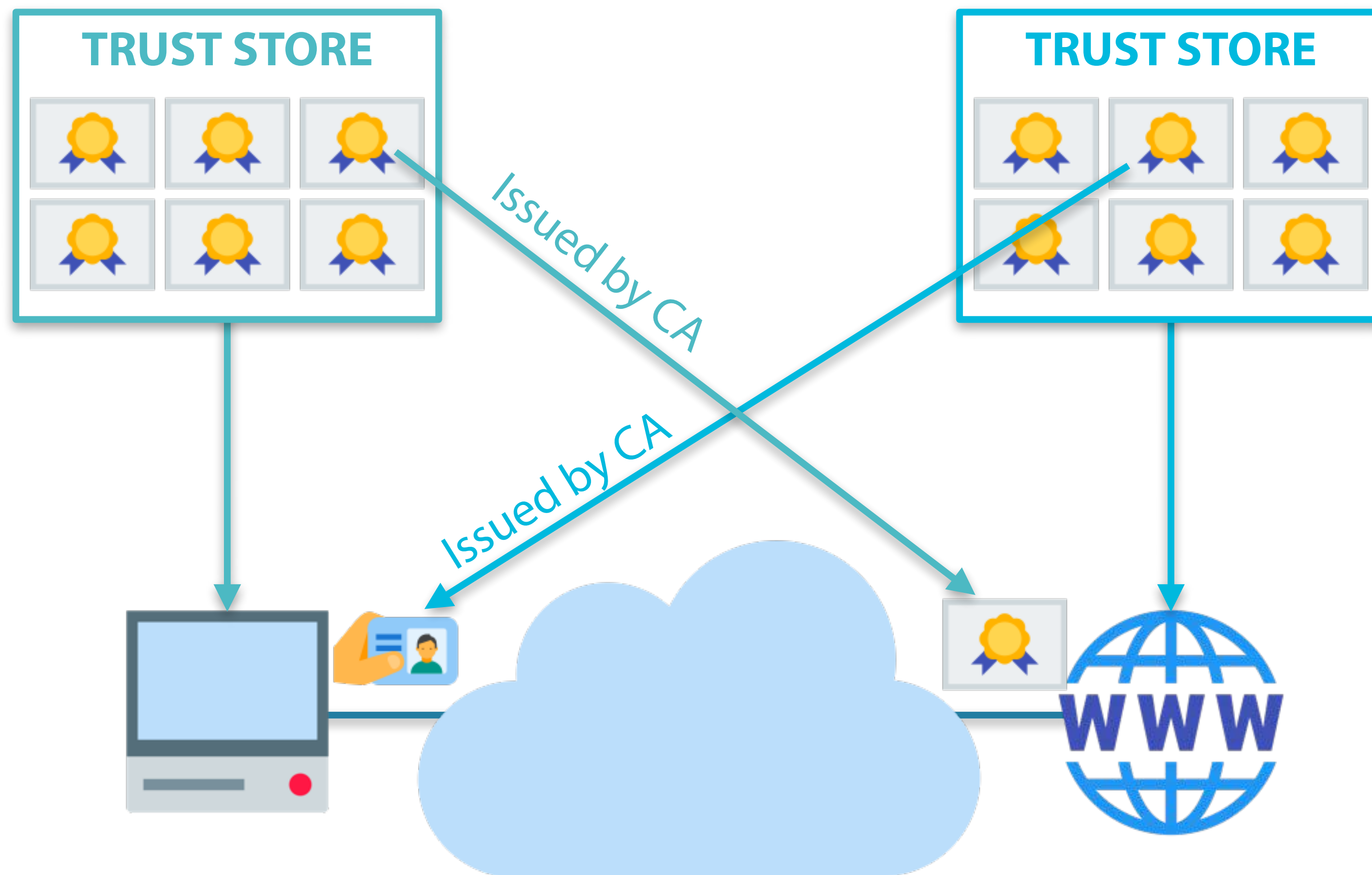


- Server asks certificate from the client
- Client presents its certificate
- Server performs the checks
- ID in the client certificate is used for authentication / authorization





Mutual TLS trust relationships





First configure regular TLS

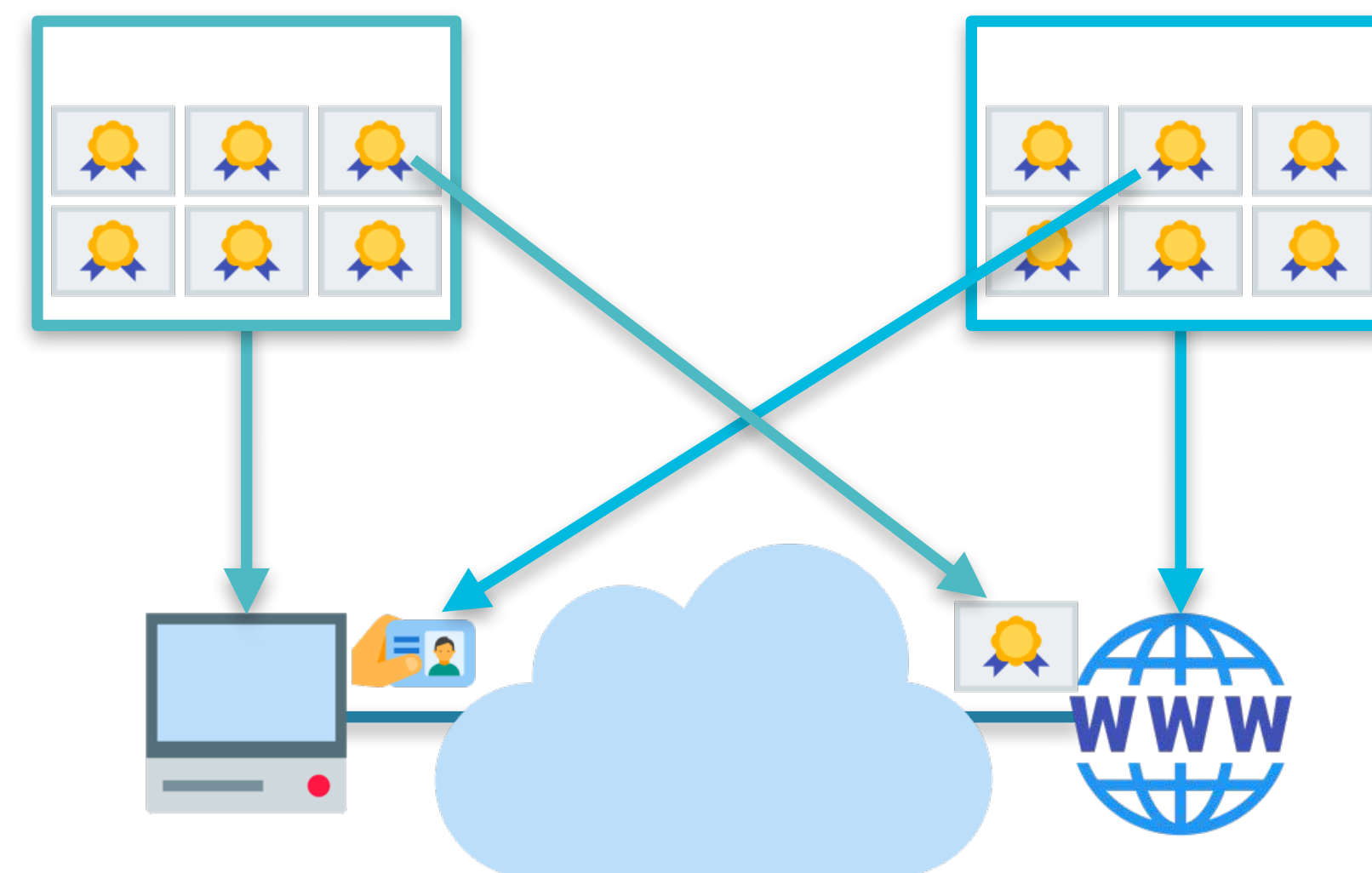


- Server side

- Add certificate
- Add Intermediate CA's (don't include the RootCA)
- Configure TLS version (TLS 1.2)
- Limit cipher suites (check with SSL-labs)

- Client side

- Add RootCA (already done by browser vendor)





Then configure the client authentication

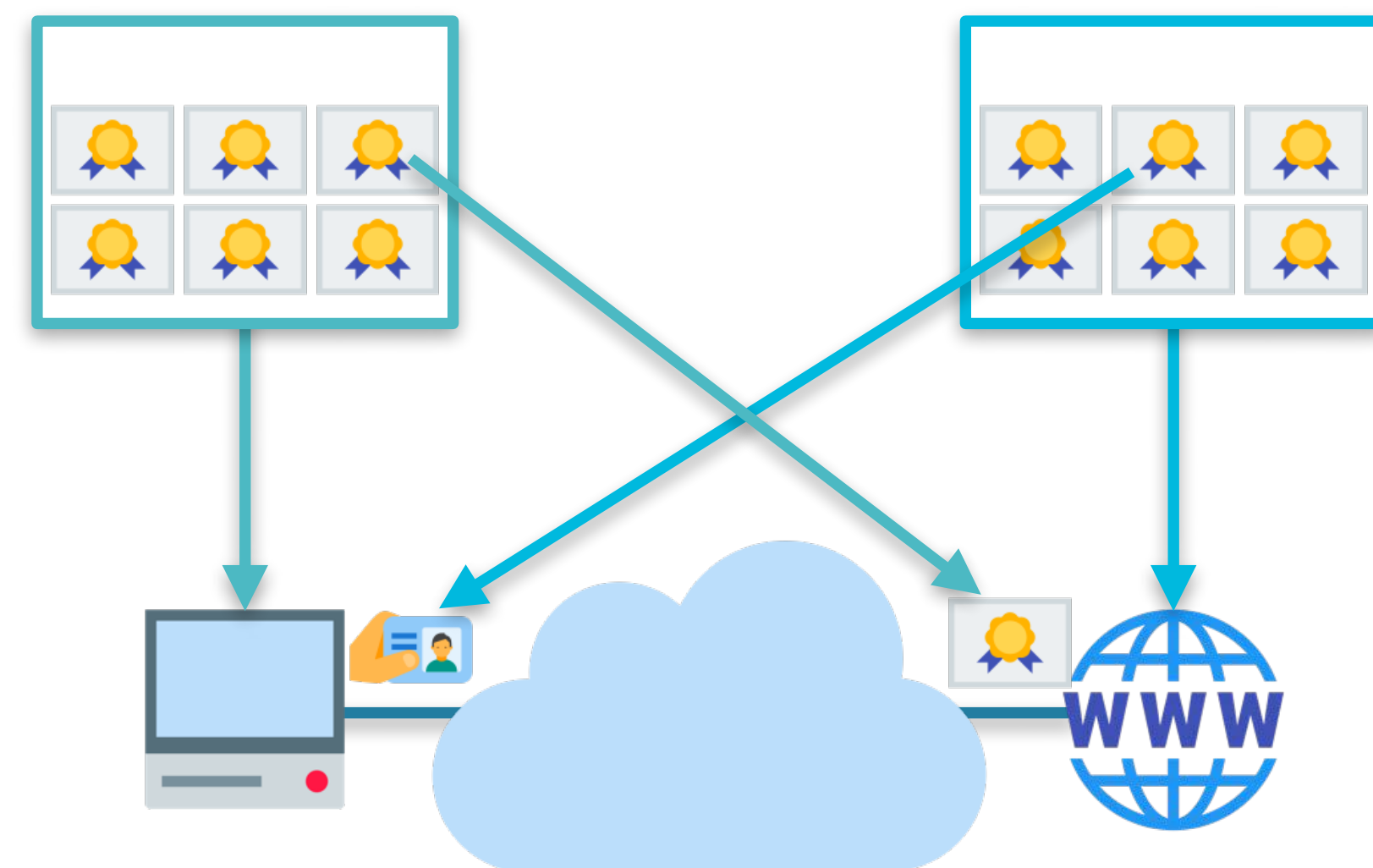


- Server side

- Add list of CA's that from which client certificates are trusted
 - Optional: some devices need you to provide a list of DN's separately
- Set "client certificates" to Request or Require
- Configure CA depth (optional)

- Client side

- Add client certificate (and private key), usually a PFX or P12 file
- Add intermediate CA's (don't include the RootCA)





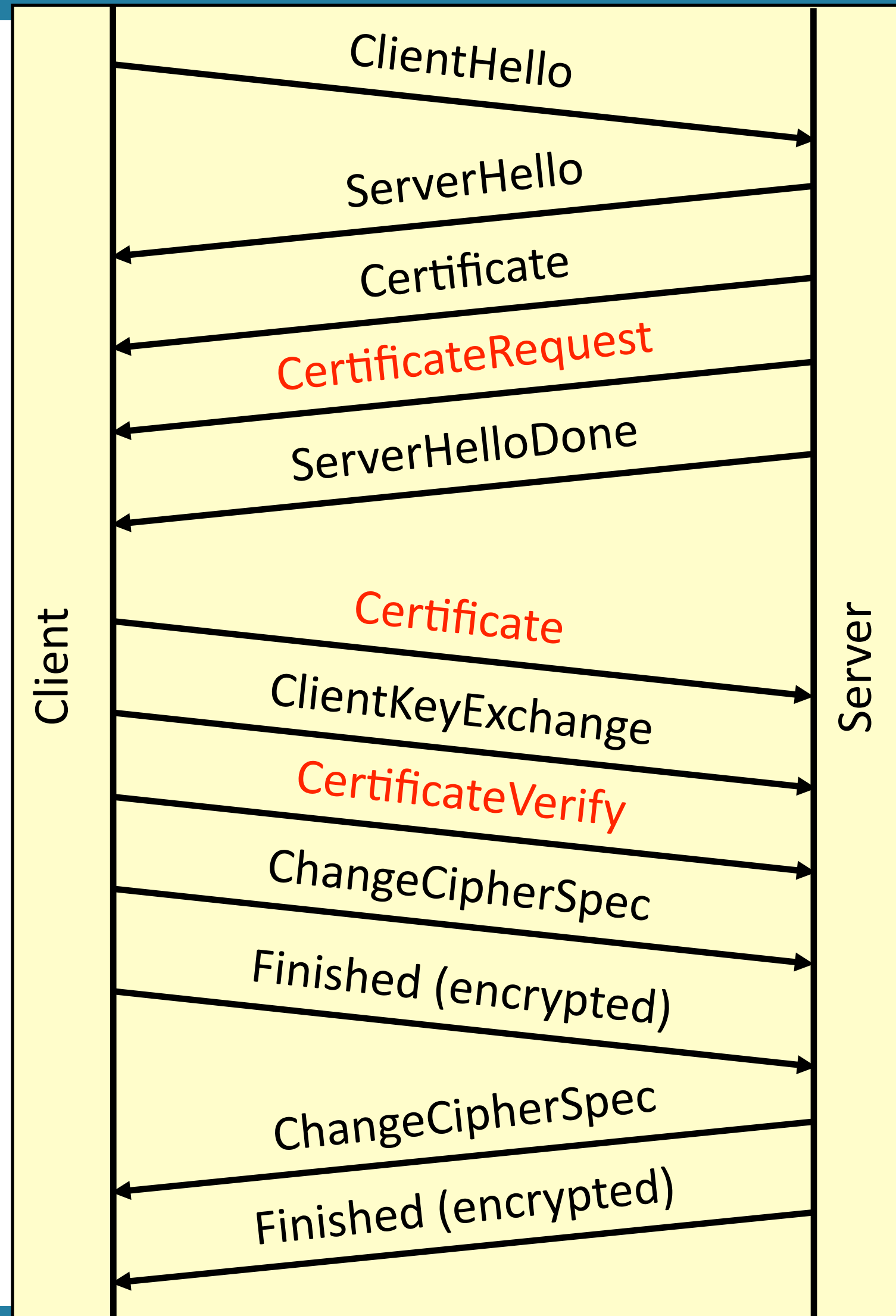
New Handshake Messages



- **CertificateRequest**
 - Ask the client to present a certificate
 - Provide a list of DN's of CA's so the client can select the correct client certificate
 - Does NOT mean these CA's are trusted, configure properly!
- **Certificate (client to server)**
 - The client offers the certificate and intermediate CA's
 - Always sent, even when not sending a certificate
 - Then the length will be 0
- **CertificateVerify**
 - Provide signature of previous handshake messages
 - Used to proof possession of the private key



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=1460
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 , Change Cipher Spec
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
```



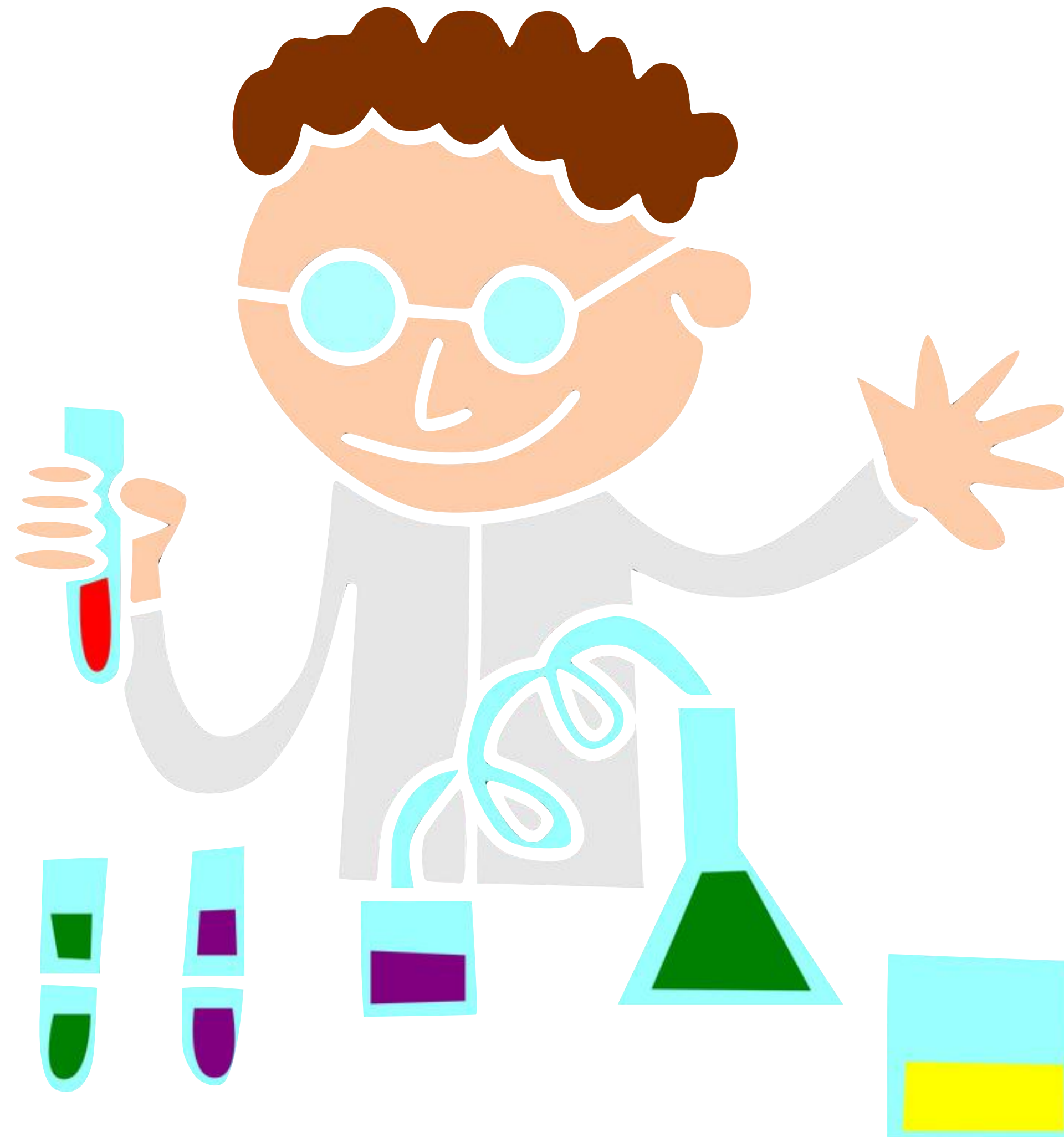
CertificateVerify (client)



```
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
```



LAB 1

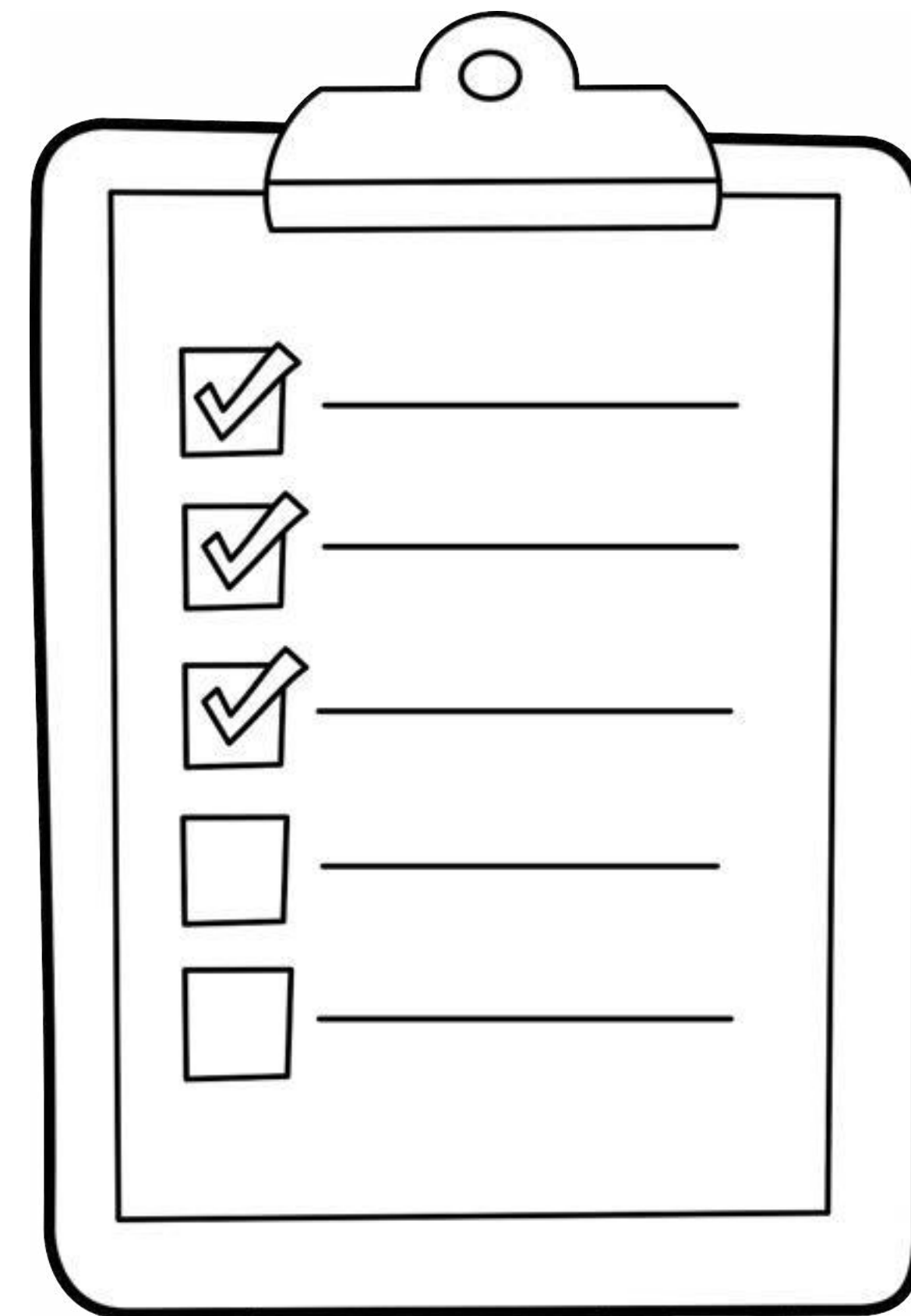




Agenda



- TLS fundamentals
 - The need for TLS (and what happens to SSL?)
 - Cryptology 101 & PKI
 - Troubleshoot the TLS handshake (SSLv3 - TLSv1.2)
 - **Troubleshoot the TLS handshake (TLSv1.3)**
- Analysing Application Data
 - Without decryption
 - With decryption using the server's private key
 - With decryption using TLS session keys
- Creating configuration profiles for TLS analysis
- SSL/TLS vulnerabilities & privacy concerns
- Summary and Q&A



<https://www.needpix.com/photo/download/94588/checklist-action-check-list-verification-tick-approved-okay-check-mark>



From TLSv1.2 to TLSv1.3

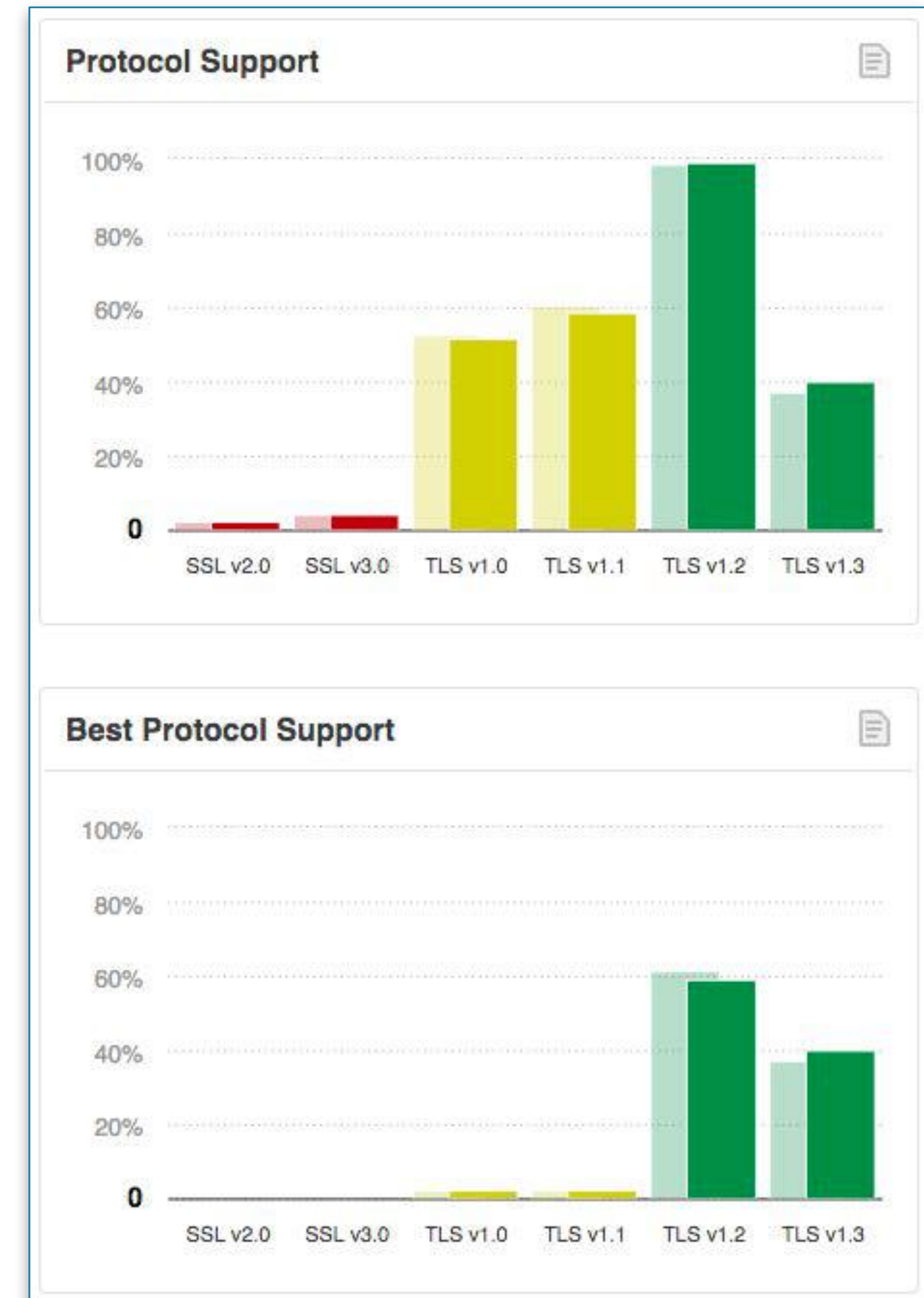


- Increased security by removal of support for:

- legacy bulk encryption protocols like RC4
- static RSA and DH cipher suites (now always PFS)
- Old hashing algorithms like MD5 and SHA1
- Compression
- Renegotiation

- Added features:

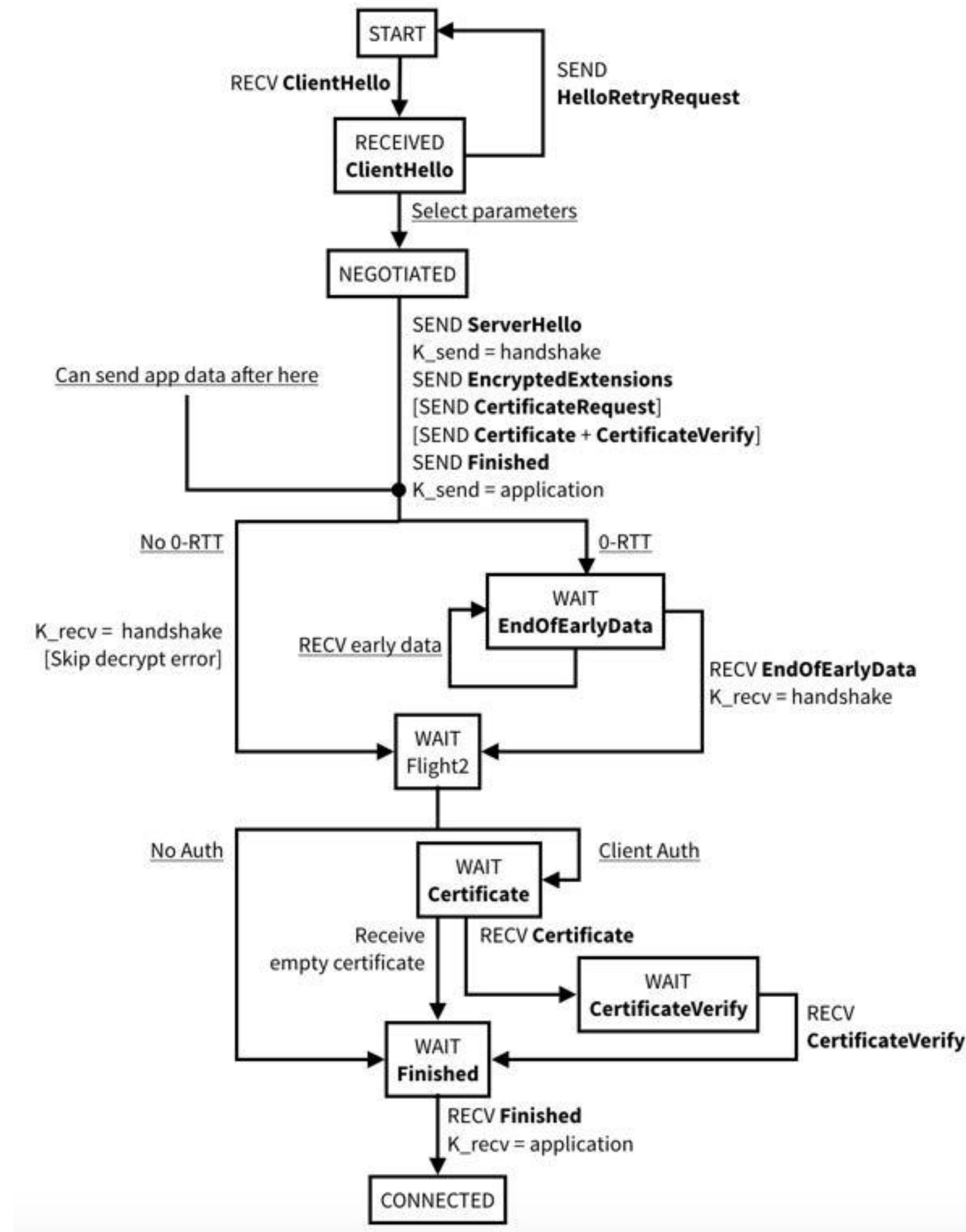
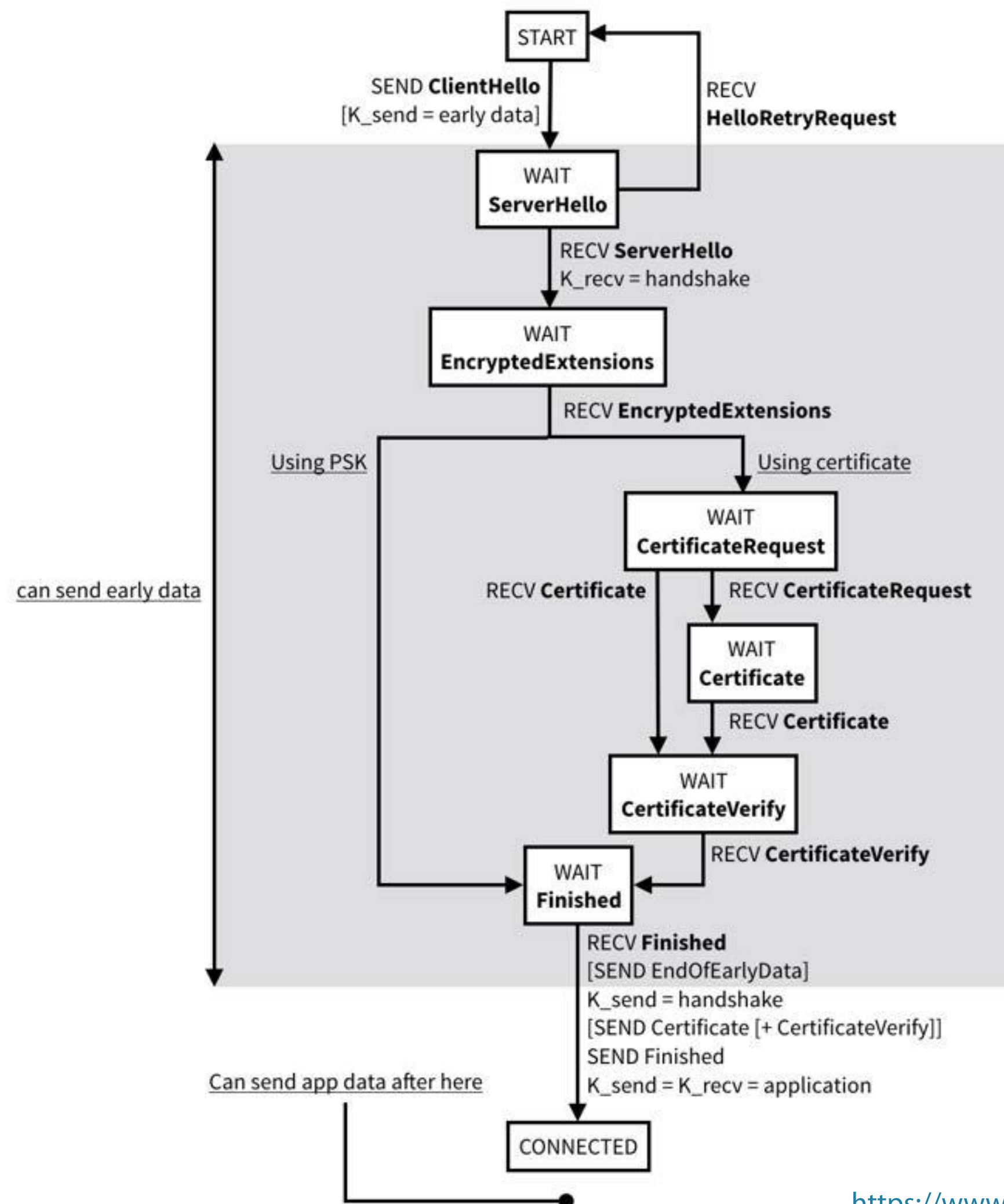
- Significantly restructured handshake
 - 1-RTT handshake by default, optional 0-RTT
 - Middlebox compatibility
- encryption starts after ServerHello
- Downgrade protection
- Support for ECC in base spec and new algorithms included
- TLS version negotiation extension
- Single PSK exchange for session resumption



<https://cds.cern.ch/record/39437>



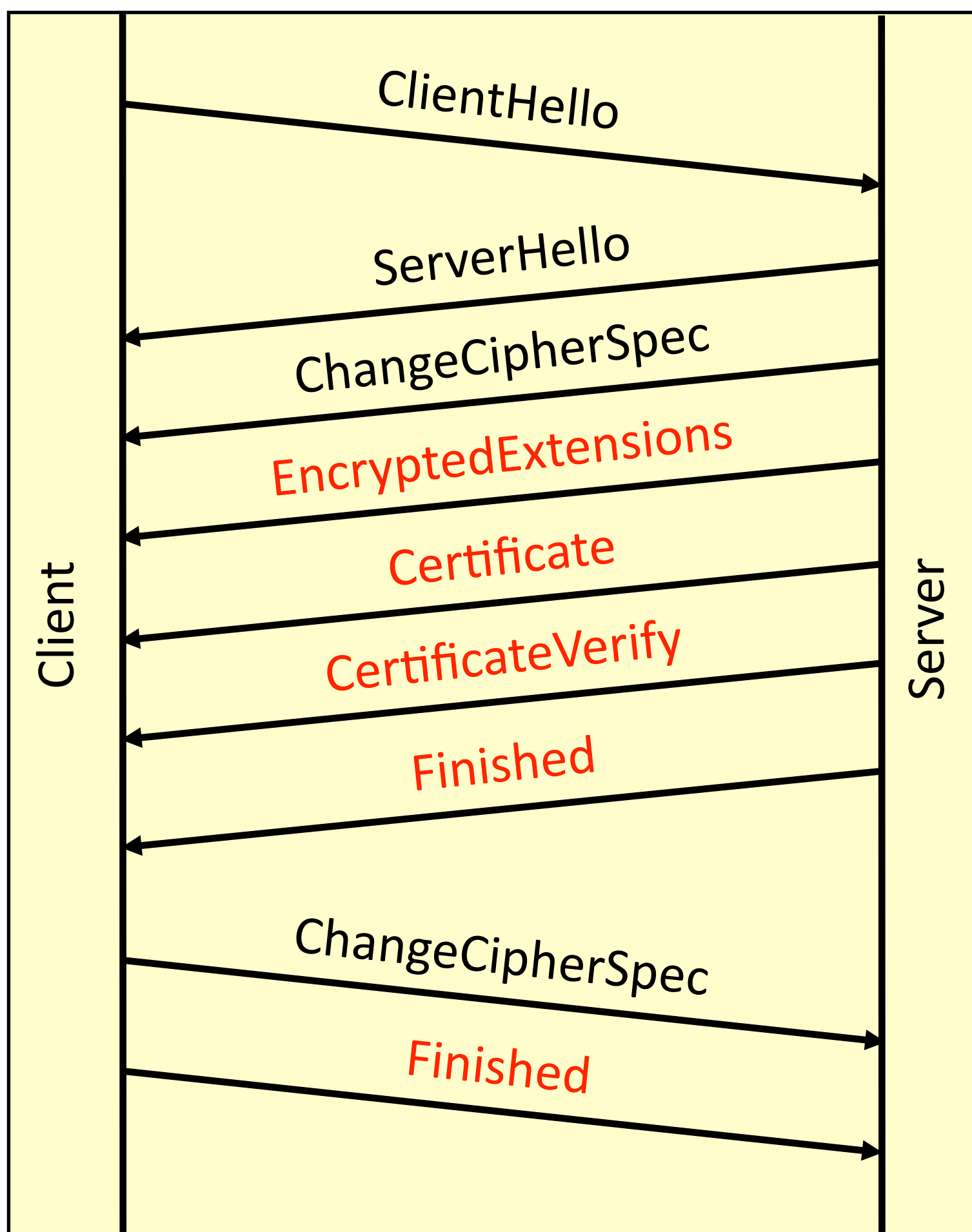
TLSv1.3 state diagrams



<https://www.davidwong.fr/tls13/>



TLSv1.3 Handshake



No.	Time	Delta	Source	Destination	Protocol	Length	Info
12	12:42:15.609765	0.000000	192.168.2.3	13.227.219.2	TCP	78	54463 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32
19	12:42:15.617106	0.007341	13.227.219.2	192.168.2.3	TCP	74	443 → 54463 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=
20	12:42:15.617142	0.000036	192.168.2.3	13.227.219.2	TCP	66	54463 → 443 [ACK] Seq=1 Ack=1 Win=131744 Len=0
23	12:42:15.619839	0.002697	192.168.2.3	13.227.219.2	TLSv1.3	583	Client Hello
25	12:42:15.628080	0.008241	13.227.219.2	192.168.2.3	TCP	66	443 → 54463 [ACK] Seq=1 Ack=518 Win=30208 Len=0
26	12:42:15.630108	0.002028	13.227.219.2	192.168.2.3	TLSv1.3	1514	Server Hello, Change Cipher Spec, Encrypted Extensions
27	12:42:15.630111	0.000003	13.227.219.2	192.168.2.3	TCP	1514	443 → 54463 [ACK] Seq=1449 Ack=518 Win=30208 Len=1448 [
28	12:42:15.630155	0.000044	192.168.2.3	13.227.219.2	TCP	66	54463 → 443 [ACK] Seq=518 Ack=2897 Win=128864 Len=0
32	12:42:15.633724	0.003569	13.227.219.2	192.168.2.3	TLSv1.3	1011	Certificate, Certificate Verify, Finished
33	12:42:15.633759	0.000035	192.168.2.3	13.227.219.2	TCP	66	54463 → 443 [ACK] Seq=518 Ack=3842 Win=130112 Len=0
40	12:42:15.641691	0.007932	192.168.2.3	13.227.219.2	TLSv1.3	130	Change Cipher Spec, Finished



ClientHello (client)



```
Transport Layer Security
  TLSv1.3 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 512
  Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 508
    Version: TLS 1.2 (0x0303)
    Random: 406185c1dcc4b0cfed8af676d5e51e384d786032324e32de...
    Session ID Length: 32
    Session ID: 9afb80411de191604dd7cd4318fd1b53a01878aad2a5c3a...
    Cipher Suites Length: 36
    Cipher Suites (18 suites)
    Compression Methods Length: 1
    Compression Methods (1 method)
    Extensions Length: 399
    Extension: server_name (len=40)
    Extension: extended_master_secret (len=0)
    Extension: renegotiation_info (len=1)
    Extension: supported_groups (len=14)
    Extension: ec_point_formats (len=2)
    Extension: session_ticket (len=0)
    Extension: application_layer_protocol_negotiation (len=14)
    Extension: status_request (len=5)
    Extension: key_share (len=107)
    Extension: supported_versions (len=5)
      Type: supported_versions (43)
      Length: 5
      Supported Versions length: 4
      Supported Version: TLS 1.3 (0x0304)
      Supported Version: TLS 1.2 (0x0303)
    Extension: signature_algorithms (len=24)
    Extension: psk_key_exchange_modes (len=2)
    Extension: record_size_limit (len=2)
    Extension: padding (len=127)
```

Minimum supported version

maximum supported version (TLSv1.2 compatibility)

supported versions (TLSv1.3 extension)

Not (or differently) used in TLSv1.3



ClientHello (client)



```
▼ Transport Layer Security
  ▼ TLSv1.3 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 512
  ▼ Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 508
    Version: TLS 1.2 (0x0303)
    Random: 406185c1dcc4b0cfed8af676d5e51e384d786032324e32de...
    Session ID Length: 32
    Session ID: 9afb80411de191604dd7cd4318fd1b53a01878aad2a5c3a...
    Cipher Suites Length: 36
  ▼ Cipher Suites (18 suites)
    Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
    Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
    Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xc030)
    Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xc031)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc032)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc033)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc034)
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc035)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc036)
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc037)
    Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0xc038)
    Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0xc039)
    Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0xc03a)
    Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0xc03b)
    Cipher Suite: TLS_RSA_WITH_3DES_EDE_CBC_SHA (0xc03c)
  Compression Methods Length: 1
  ► Compression Methods (1 method)
```

TLSv1.3 ciphersuites

TLSv1.2 ciphersuites



ClientHello (client)



```
▼ Handshake Protocol: Client Hello
  Handshake Type: Client Hello (1)
  Length: 508
  Version: TLS 1.2 (0x0303)
  Random: 406185c1dccc4b0cfed8af676d5e51e384d786032324e32de...
  Session ID Length: 32
  Session ID: 9afb80411de191604dd7cd4318fd1b53a01878aad2a5c3a...
  Cipher Suites Length: 36
  ▶ Cipher Suites (18 suites)
  Compression Methods Length: 1
  ▶ Compression Methods (1 method)
  Extensions Length: 399
  ▶ Extension: server_name (len=40)
  ▶ Extension: extended_master_secret (len=0)
  ▶ Extension: renegotiation_info (len=1)
  ▼ Extension: supported_groups (len=14)
    Type: supported_groups (10)
    Length: 14
    Supported Groups List Length: 12
    ▼ Supported Groups (6 groups)
      Supported Group: x25519 (0x001d)
      Supported Group: secp256r1 (0x0017)
      Supported Group: secp384r1 (0x0018)
      Supported Group: secp521r1 (0x0019)
      Supported Group: ffdhe2048 (0x0100)
      Supported Group: ffdhe3072 (0x0101)
  ▶ Extension: ec_point_formats (len=2)
  ▶ Extension: session_ticket (len=0)
  ▶ Extension: application_layer_protocol_negotiation (len=14)
  ▶ Extension: status_request (len=5)
  ▼ Extension: key_share (len=107)
    Type: key_share (51)
    Length: 107
    ▼ Key Share extension
      Client Key Share Length: 105
      ▶ Key Share Entry: Group: x25519, Key Exchange length: 32
      ▶ Key Share Entry: Group: secp256r1, Key Exchange length: 65
  ▶ Extension: supported_versions (len=5)
  ▼ Extension: signature_algorithms (len=24)
    Type: signature_algorithms (13)
    Length: 24
    Signature Hash Algorithms Length: 22
    ▶ Signature Hash Algorithms (11 algorithms)
  ▼ Extension: psk_key_exchange_modes (len=2)
    Type: psk_key_exchange_modes (45)
    Length: 2
    PSK Key Exchange Modes Length: 1
    PSK Key Exchange Mode: PSK with (EC)DHE key establishment (psk_dhe_ke) (1)
  ▶ Extension: record_size_limit (len=2)
  ▶ Extension: padding (len=127)
```

Supported Elliptic Curve Groups

Client Key Share (for some of the supported groups)

Also needed to derive the session keys



ServerHello (server)



```
▼ Transport Layer Security
  ▼ TLSv1.3 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 122
  ▼ Handshake Protocol: Server Hello
    Handshake Type: Server Hello (2)
    Length: 118
    Version: TLS 1.2 (0x0303)
    Random: 10e087ae42f59bfa5e8fde564c7d27f0da2936936e8509f7...
    Session ID Length: 32
    Session ID: 9afb80411de191604dd7cd4318fd1b53a01878aadc2a5c3a...
    Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
    Compression Method: null (0)
    Extensions Length: 46
  ▼ Extension: supported_versions (len=2)
    Type: supported_versions (43)
    Length: 2
    Supported Version: TLS 1.3 (0x0304)
  ▼ Extension: key_share (len=36)
    Type: key_share (51)
    Length: 36
  ▼ Key Share extension
  ▼ Key Share Entry: Group: x25519, Key Exchange length: 32
    Group: x25519 (29)
    Key Exchange Length: 32
    Key Exchange: 713c6634f3afa36009dd9e77014dfc1392eb9ddea91d91a3...
  ▼ TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
    Content Type: Change Cipher Spec (20)
    Version: TLS 1.2 (0x0303)
    Length: 1
    Change Cipher Spec Message
  ▶ TLSv1.3 Record Layer: Handshake Protocol: Encrypted Extensions
```

Dummy protocol version for middleboxes

Chosen cipher suite

Chosen protocol version

Chosen Group and server key share



Change Cipher Spec (server)



```
▼ Transport Layer Security
  ▼ TLSv1.3 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 122
  ▼ Handshake Protocol: Server Hello
    Handshake Type: Server Hello (2)
    Length: 118
    Version: TLS 1.2 (0x0303)
    Random: 10e087ae42f59bfa5e8fde564c7d27f0da2936936e8509f7...
    Session ID Length: 32
    Session ID: 9afb80411de191604dd7cd4318fd1b53a01878aad2a5c3a...
    Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
    Compression Method: null (0)
    Extensions Length: 46
  ▼ Extension: supported_versions (len=2)
    Type: supported_versions (43)
    Length: 2
    Supported Version: TLS 1.3 (0x0304)
  ▼ Extension: key_share (len=36)
    Type: key_share (51)
    Length: 36
  ▼ Key Share extension
    ▼ Key Share Entry: Group: x25519, Key Exchange length: 32
      Group: x25519 (29)
      Key Exchange Length: 32
      Key Exchange: 713c6634f3afa36009dd9e77014dfc1392eb9ddea91d91a3...
  ▼ TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
    Content Type: Change Cipher Spec (20)
    Version: TLS 1.2 (0x0303)
    Length: 1
    Change Cipher Spec Message
  ▶ TLSv1.3 Record Layer: Handshake Protocol: Encrypted Extensions
```

Dummy ChangeCipherSpec for middleboxes



EncryptedExtensions (server)



```
▼ Transport Layer Security
  ▶ TLSv1.3 Record Layer: Handshake Protocol: Server Hello
  ▶ TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
  ▼ TLSv1.3 Record Layer: Handshake Protocol: Encrypted Extensions
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 36
    [Content Type: Handshake (22)]
  ▼ Handshake Protocol: Encrypted Extensions
    Handshake Type: Encrypted Extensions (8)
    Length: 15
    Extensions Length: 13
    ▼ Extension: server_name (len=0)
      Type: server_name (0)
      Length: 0
    ▼ Extension: application_layer_protocol_negotiation (len=5)
      Type: application_layer_protocol_negotiation (16)
      Length: 5
      ALPN Extension Length: 3
      ▼ ALPN Protocol
        ALPN string length: 2
        ALPN Next Protocol: h2
```

Application Layer Protocol Negotiation



Certificate (server)



```
▼ Handshake Protocol: Certificate
  Handshake Type: Certificate (11)
  Length: 3297
  Certificate Request Context Length: 0
  Certificates Length: 3293
  ▼ Certificates (3293 bytes)
    Certificate Length: 1628
    ► Certificate: 3082065830820540a00302010202100fa257b8180b19dd18... (id-at-commonName=*.cdn.mozilla.net,id
    Extensions Length: 479
  ▼ Extension: status_request (len=475)
    Type: status_request (5)
    Length: 475
    Certificate Status Type: OCSP (1)
    OCSP Response Length: 471
    ▼ OCSP Response
      responseStatus: successful (0)
      ▼ responseBytes
        ResponseType Id: 1.3.6.1.5.5.7.48.1.1 (id-pkix-ocsp-basic)
        ▼ BasicOCSPResponse
          ▼ tbsResponseData
            ► responderID: byKey (2)
            producedAt: 2020-09-28 05:09:01 (UTC)
            ▼ responses: 1 item
              ▼ SingleResponse
                ► certID
                ► certStatus: good (0)
                thisUpdate: 2020-09-28 05:09:01 (UTC)
                nextUpdate: 2020-10-05 04:24:01 (UTC)
            ► signatureAlgorithm (sha256WithRSAEncryption)
            Padding: 0
            signature: 6b3a6f7bdf21436199037c2c51b696e8935defd1b58816c9...
    Certificate Length: 1176
    ► Certificate: 308204943082037ca003020102021001fda3eb6eca75c888... (id-at-commonName=DigiCert SHA2 Secure
    Extensions Length: 0
```

Sever Certificate

Certificate extensions, in this case OCSP

(Intermediate) CA Certificate



Certificate Verify (server)



```
▶ Transmission Control Protocol, Src Port: 443, Dst Port: 54463, Seq: 2897, Ack: 518, Len: 945
▶ [3 Reassembled TCP Segments (3323 bytes): #26(1274), #27(1448), #32(601)]
▼ Transport Layer Security
  ▶ TLSv1.3 Record Layer: Handshake Protocol: Certificate
▼ Transport Layer Security
  ▼ TLSv1.3 Record Layer: Handshake Protocol: Certificate Verify
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 281
    [Content Type: Handshake (22)]
    ▼ Handshake Protocol: Certificate Verify
      Handshake Type: Certificate Verify (15)
      Length: 260
      ▶ Signature Algorithm: rsa_pss_rsae_sha256 (0x0804)
        Signature length: 256
        Signature: 01d1b1949bd7b6118749b8a46310d693139683a9c6e7ba5e...
    ▶ TLSv1.3 Record Layer: Handshake Protocol: Finished
```

Certificate Verify proves possession of the private key of the server certificate



Finished (server)



```
▶ Transmission Control Protocol, Src Port: 443, Dst Port: 54463, Seq: 2897, Ack: 518, Len: 945
▶ [3 Reassembled TCP Segments (3323 bytes): #26(1274), #27(1448), #32(601)]
▼ Transport Layer Security
▶ TLSv1.3 Record Layer: Handshake Protocol: Certificate
▼ Transport Layer Security
▶ TLSv1.3 Record Layer: Handshake Protocol: Certificate Verify
▼ TLSv1.3 Record Layer: Handshake Protocol: Finished
  Opaque Type: Application Data (23)
  Version: TLS 1.2 (0x0303)
  Length: 53
  [Content Type: Handshake (22)]
  ▼ Handshake Protocol: Finished
    Handshake Type: Finished (20)
    Length: 32
    Verify Data
```

Finished message proves nothing in the handshake has been altered (mitm protection)



Change Cipher Spec (client)



```
▶ Frame 40: 130 bytes on wire (104 bytes captured) on interface en0, id 0
▶ Ethernet II, Src: MacSake.local (ac:bc:32:cb:26:45), Dst: router.home (24:00:11:11:11:11)
▶ Internet Protocol Version 4, Src: macsake.home (192.168.2.3), Dst: d2nxq2...
▶ Transmission Control Protocol, Src Port: 54463, Dst Port: 443, Seq: 518,
▼ Transport Layer Security
  ▼ TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
    Content Type: Change Cipher Spec (20)
    Version: TLS 1.2 (0x0303)
    Length: 1
    Change Cipher Spec Message
  ▼ TLSv1.3 Record Layer: Handshake Protocol: Finished
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 53
    [Content Type: Handshake (22)]
  ▼ Handshake Protocol: Finished
    Handshake Type: Finished (20)
    Length: 32
    Verify Data
```

Dummy ChangeCipherSpec for middleboxes



Finished (client)



```
▶ Frame 40: 130 bytes on wire (104 bytes captured) on interface en0, id 0
▶ Ethernet II, Src: MacSake.local (ac:bc:32:cb:26:45), Dst: router.home (08:00:27:00:00:00)
▶ Internet Protocol Version 4, Src: macsake.home (192.168.2.3), Dst: d2nx.com (192.168.2.1)
▶ Transmission Control Protocol, Src Port: 54463, Dst Port: 443, Seq: 518414488, Len: 130
▼ Transport Layer Security
  ▼ TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
    Content Type: Change Cipher Spec (20)
    Version: TLS 1.2 (0x0303)
    Length: 1
    Change Cipher Spec Message
  ▼ TLSv1.3 Record Layer: Handshake Protocol: Finished
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 53
    [Content Type: Handshake (22)]
    ▼ Handshake Protocol: Finished
      Handshake Type: Finished (20)
      Length: 32
      Verify Data
```

Application Data instead of Handshake to please the middle boxes

Finished message proves nothing in the handshake has been altered (mitm protection)



TLSv1.3 handshake partially encrypted



No.	Time	Delta	Source	Destination	Protocol	Length	Info
12	12:42:15.609765	0.000000	192.168.2.3	13.227.219.2	TCP	78	54463 → 443 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 SACK_PERM=1
19	12:42:15.617106	0.007341	13.227.219.2	192.168.2.3	TCP	74	443 → 54463 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 WS=256
20	12:42:15.617142	0.000036	192.168.2.3	13.227.219.2	TCP	66	54463 → 443 [ACK] Seq=1 Ack=1 Win=131744 Len=0
23	12:42:15.619839	0.002697	192.168.2.3	13.227.219.2	TLSv1.3	583	Client Hello
25	12:42:15.628080	0.008241	13.227.219.2	192.168.2.3	TCP	66	443 → 54463 [ACK] Seq=1 Ack=518 Win=30208 Len=0
26	12:42:15.630108	0.002028	13.227.219.2	192.168.2.3	TLSv1.3	1514	Server Hello, Change Cipher Spec, Application Data
27	12:42:15.630111	0.000003	13.227.219.2	192.168.2.3	TCP	1514	443 → 54463 [ACK] Seq=1449 Ack=518 Win=30208 Len=1448 [TCP segment of a reassembled PDU]
28	12:42:15.630155	0.000044	192.168.2.3	13.227.219.2	TCP	66	54463 → 443 [ACK] Seq=518 Ack=2897 Win=128864 Len=0
32	12:42:15.633724	0.003569	13.227.219.2	192.168.2.3	TLSv1.3	1011	Application Data, Application Data, Application Data
33	12:42:15.633759	0.000035	192.168.2.3	13.227.219.2	TCP	66	54463 → 443 [ACK] Seq=518 Ack=3842 Win=130112 Len=0
40	12:42:15.641691	0.007932	192.168.2.3	13.227.219.2	TLSv1.3	130	Change Cipher Spec, Application Data
41	12:42:15.642542	0.000851	192.168.2.3	13.227.219.2	TLSv1.3	236	Application Data
42	12:42:15.642595	0.000053	192.168.2.3	13.227.219.2	TLSv1.3	308	Application Data
48	12:42:15.650773	0.008178	13.227.219.2	192.168.2.3	TCP	66	443 → 54463 [ACK] Server Hello, Change Cipher Spec, Encrypted Extensions
49	12:42:15.650776	0.000003	13.227.219.2	192.168.2.3	TLSv1.3	137	Application Data
50	12:42:15.650817	0.000041	192.168.2.3	13.227.219.2	TCP	66	54463 → 443 [ACK] 443 → 54463 [ACK] Seq=1449 Ack=518 Win=30208 Len=1448 [TCP segment of a reassembled PDU]
51	12:42:15.650960	0.000143	192.168.2.3	13.227.219.2	TLSv1.3	97	Application Data Certificate, Certificate Verify, Finished
54	12:42:15.653997	0.003037	13.227.219.2	192.168.2.3	TCP	1514	443 → 54463 [ACK] 54463 → 443 [ACK] Seq=518 Ack=2897 Win=128864 Len=0
55	12:42:15.654001	0.000004	13.227.219.2	192.168.2.3	TCP	1514	443 → 54463 [ACK] Change Cipher Spec, Finished
56	12:42:15.654002	0.000001	13.227.219.2	192.168.2.3	TCP	1514	443 → 54463 [ACK] Seq=6809 Ack=994 Win=32256 Len=1448 [TCP segment of a reassembled PDU]
57	12:42:15.654047	0.000045	192.168.2.3	13.227.219.2	TCP	66	54463 → 443 [ACK] Seq=1025 Ack=6809 Win=128160 Len=0
58	12:42:15.654185	0.000138	192.168.2.3	13.227.219.2	TCP	66	54463 → 443 [ACK] Seq=1025 Ack=8257 Win=131072 Len=0
59	12:42:15.654717	0.000532	13.227.219.2	192.168.2.3	TLSv1.3	1430	Application Data
60	12:42:15.654754	0.000037					

Dummy "Change Cipher Spec" messages for middlebox compatibility



TLSv1.3 session resumption

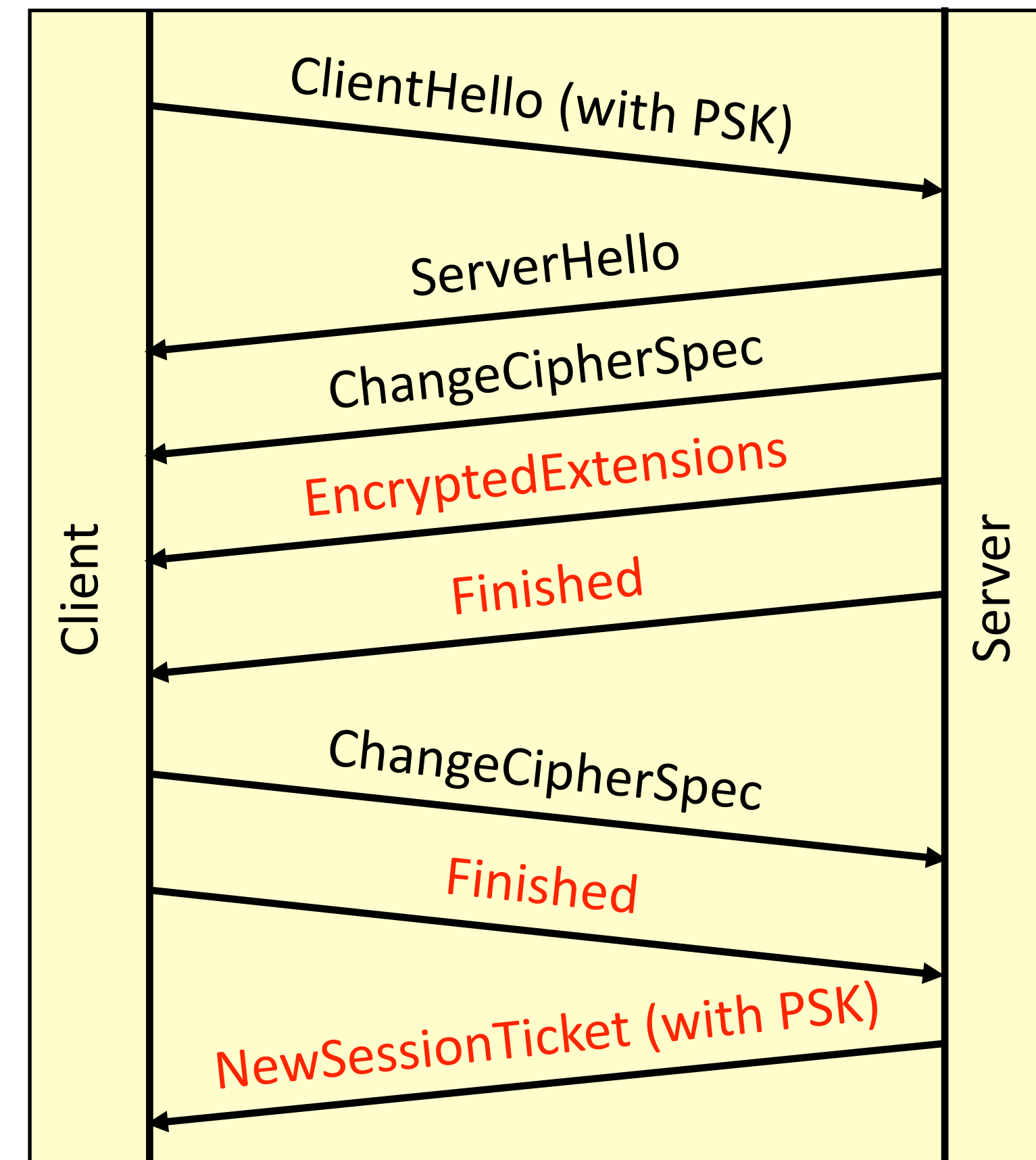
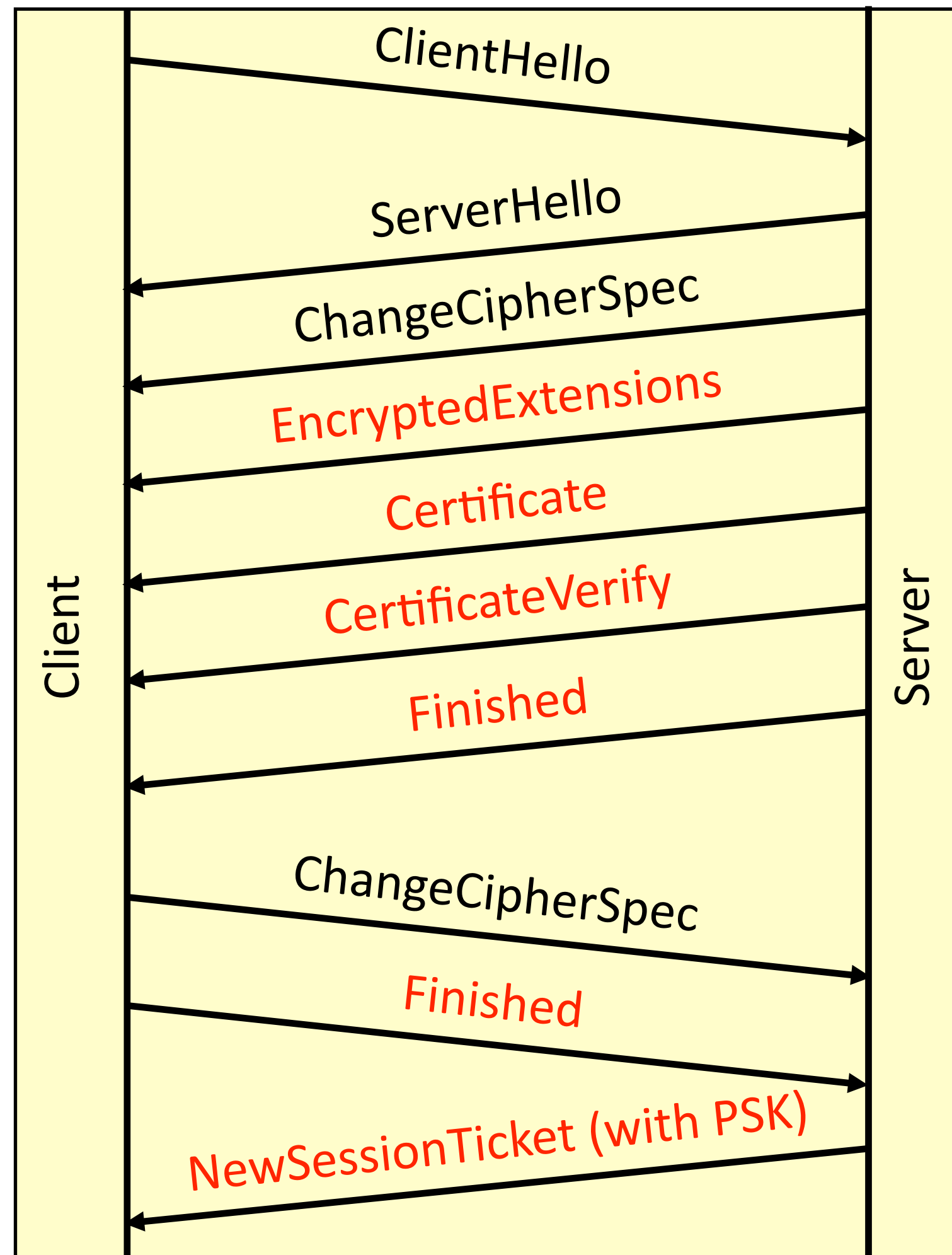


- Server creates PSK and either
 - stores it and sends the pointer as the PSK Identity
 - encrypts and sends the PSK as the PSK Identity
- TLS extension in ClientHello and ServerHello
- Reuse TLS HandshakeType "NewSessionTicket"
 - to send the PSK Identity
 - More than one can be sent
 - Can be implemented as one-time-tickets





TLS session Tickets (2)





New Session Ticket (server)



```
▼ Transport Layer Security
  ▼ TLSv1.3 Record Layer: Handshake Protocol: New Session Ticket
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 282
    [Content Type: Handshake (22)]
  ▼ Handshake Protocol: New Session Ticket
    Handshake Type: New Session Ticket (4)
    Length: 261
  ▼ TLS Session Ticket
    Session Ticket Lifetime Hint: 7300 seconds (2 hours, 1 minute, 40 seconds)
    Session Ticket Age Add: 3273930522
    Session Ticket Nonce Length: 8
    Session Ticket Nonce: 0000000000000000
    Session Ticket Length: 240
    Session Ticket: 00003f8af43053355eb2ed50fa6c4a14291b0d29edcdbc64...
    Extensions Length: 0
```

Lifetime for the PSK (and offset)

PSK identity for session resumption



ServerHello (server)



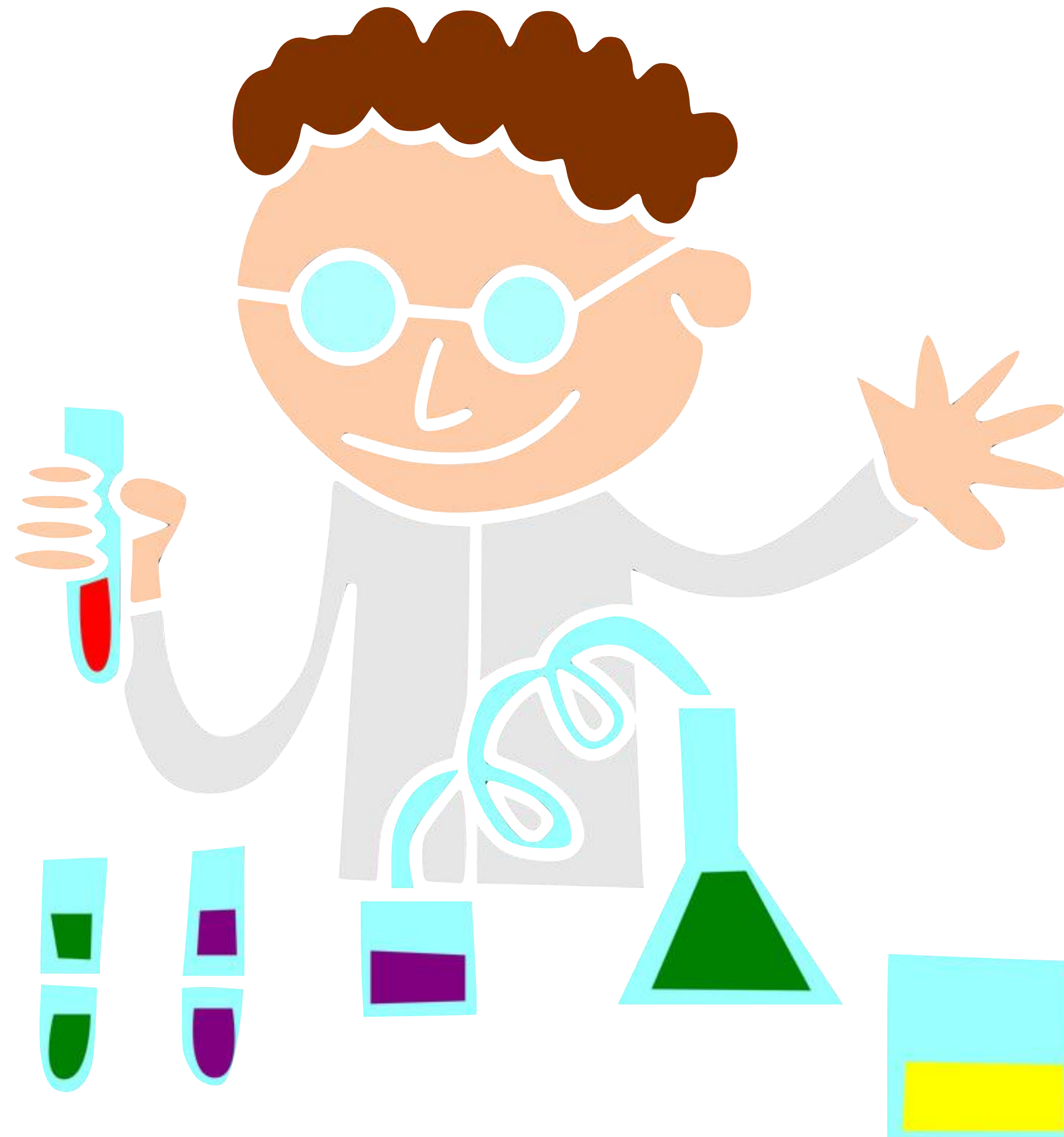
```
▼ Transport Layer Security
  ▼ TLSv1.3 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 128
  ▼ Handshake Protocol: Server Hello
    Handshake Type: Server Hello (2)
    Length: 124
    Version: TLS 1.2 (0x0303)
    Random: a943f707bd16f6b0cc3fb4a6fb820e37e82edc45fcd3c37e...
    Session ID Length: 32
    Session ID: 27e54dcc61155c0df9e206ddea929279126d1c2c06b4262b...
    Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
    Compression Method: null (0)
    Extensions Length: 52
    ▶ Extension: supported_versions (len=2)
    ▶ Extension: key_share (len=36)
    ▼ Extension: pre_shared_key (len=2)
      Type: pre_shared_key (41)
      Length: 2
      ▼ Pre-Shared Key extension
        Selected Identity: 0
    ▶ TLSv1.3 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec
    ▶ TLSv1.3 Record Layer: Handshake Protocol: Encrypted Extensions
    ▶ TLSv1.3 Record Layer: Handshake Protocol: Finished
```

Selected PSK Identity

No more Certificate and Certificate Verify



LAB 2

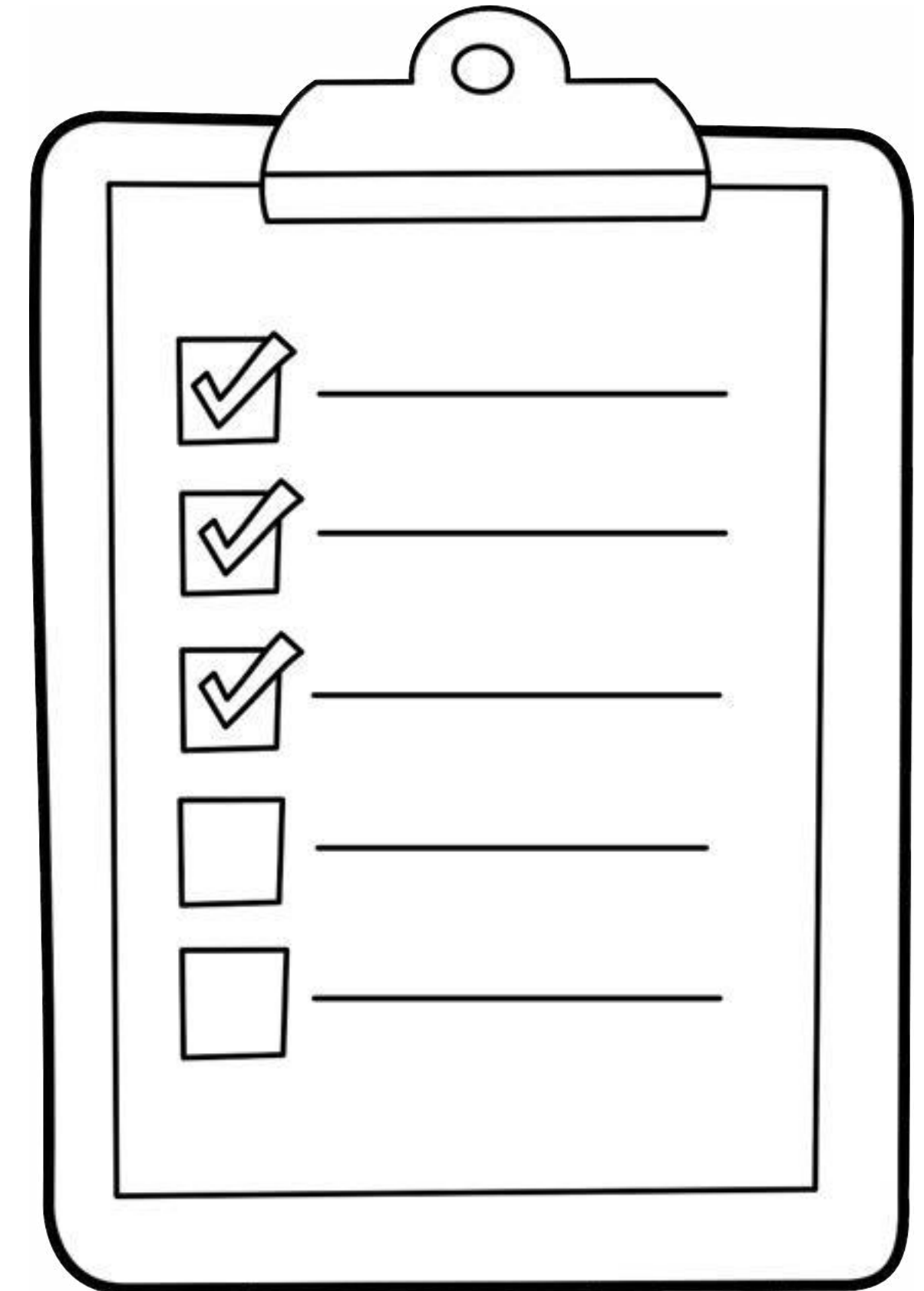




Agenda



- TLS fundamentals
 - The need for TLS (and what happens to SSL?)
 - Cryptology 101 & PKI
 - Troubleshoot the TLS handshake (SSLv3 - TLSv1.2)
 - Troubleshoot the TLS handshake (TLSv1.3)
- **Analysing Application Data**
 - Without decryption
 - With decryption using the server's private key
 - With decryption using TLS session keys
- Creating configuration profiles for TLS analysis
- SSL/TLS vulnerabilities & privacy concerns
- Summary and Q&A



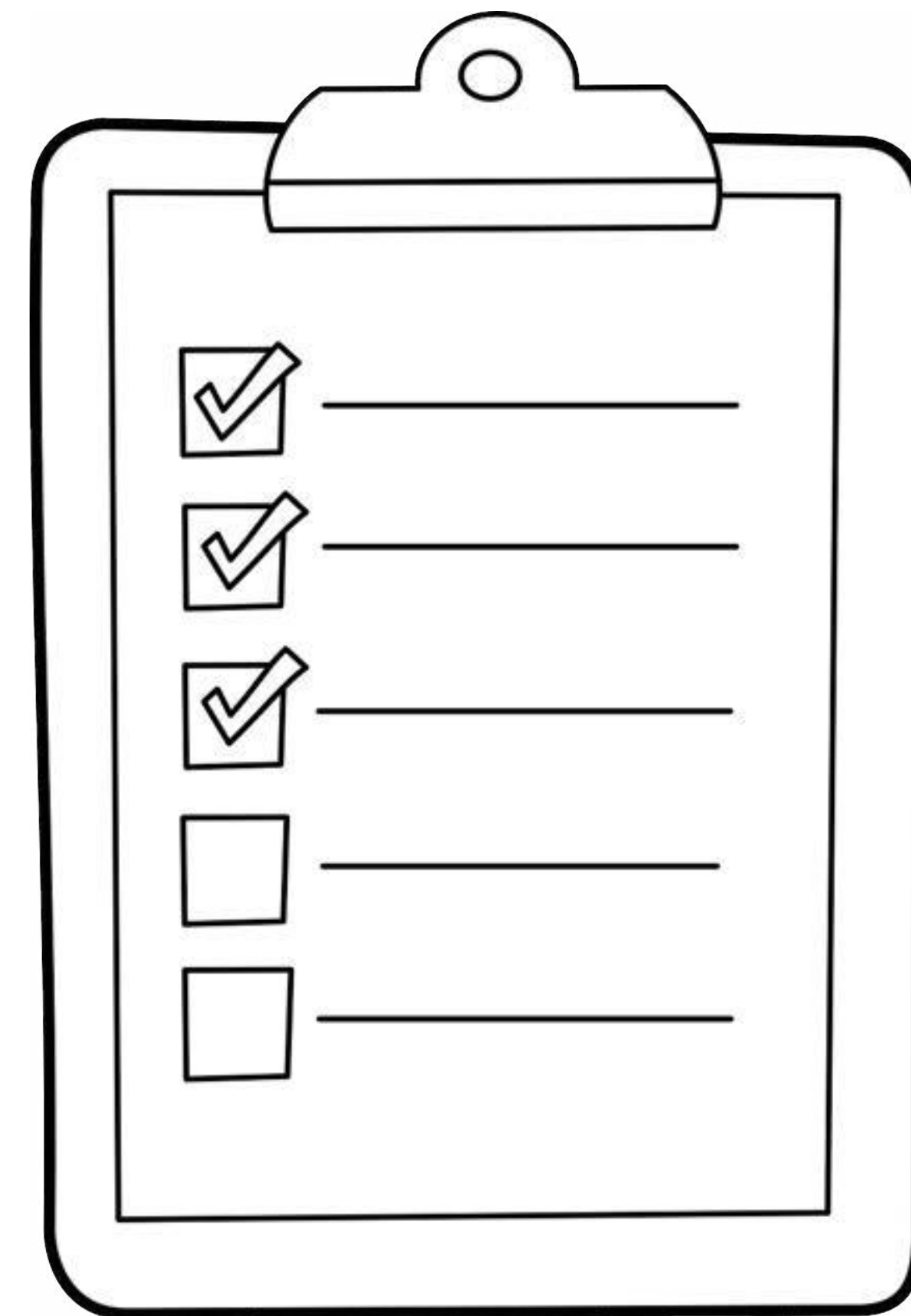
<https://www.needpix.com/photo/download/94588/checklist-action-check-list-verification-tick-approved-okay-check-mark>



Agenda



- TLS fundamentals
 - The need for TLS (and what happens to SSL?)
 - Cryptology 101 & PKI
 - Troubleshoot the TLS handshake (SSLv3 - TLSv1.2)
 - Troubleshoot the TLS handshake (TLSv1.3)
- Analysing Application Data
 - **Without decryption**
 - With decryption using the server's private key
 - With decryption using TLS session keys
- Creating configuration profiles for TLS analysis
- SSL/TLS vulnerabilities & privacy concerns
- Summary and Q&A



<https://www.needpix.com/photo/download/94588/checklist-action-check-list-verification-tick-approved-okay-check-mark>



Analyzing encrypted TLS data



- Filter on application data with:
`tls.record.content_type == 23`
- Add an extra custom column with:
`tls.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 TLS overhead)
- Use TRANSUM dissector





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



TRANSUM



tcp.stream==19 && frame.number>=2539 && frame.number<=2575 && tcp.len>0

Packet list Narrow & Wide Case sensitive Display filter transum.art >= 1

No.	Time	Delta	#strea	Source	Destination	Protocol	Length	FQD	tls.record.len	Service Time	Rsp Spread	Info
2539	17:53:36.079385000	0.000000000	19	192.168.2.3	178.250.0.165	TLSv1.3	130		1,53			Change Cipher Spec, Application Data
2540	17:53:36.080275000	0.000890000	19	192.168.2.3	178.250.0.165	TLSv1.3	236		165			Application Data
2541	17:53:36.080314000	0.000039000	19	192.168.2.3	178.250.0.165	TLSv1.3	340		269			Application Data
2542	17:53:36.080343000	0.000029000	19	192.168.2.3	178.250.0.165	TCP	1514					61859 → 443 [ACK] Seq=1026 Ack=2524 Win
2543	17:53:36.080344000	0.000001000	19	192.168.2.3	178.250.0.165	TLSv1.3	1350		2727			Application Data
2566	17:53:36.100486000	0.020142000	19	178.250.0.165	192.168.2.3	TLSv1.3	345		274			Application Data
2567	17:53:36.100488000	0.000002000	19	178.250.0.165	192.168.2.3	TLSv1.3	345		274			Application Data
2570	17:53:36.102079000	0.001591000	19	178.250.0.165	192.168.2.3	TLSv1.3	118		47			Application Data
2572	17:53:36.102082000	0.000003000	19	178.250.0.165	192.168.2.3	TLSv1.3	114		43			Application Data
2575	17:53:36.102397000	0.000315000	19	178.250.0.165	192.168.2.3	TLSv1.3	235		164	0.020142000	0.001911...	Application Data

▶ Frame 2575: 235 bytes on wire, 235 bytes captured on interface en0, id 0

▶ Ethernet II, Src: zte_be:f5:f0 (24:58:6e:be:f5:f0), Dst: Apple_cb:26:45 (ac:bc:32:cb:26:45)

▶ Internet Protocol Version 4, Src: 178.250.0.165, Dst: 192.168.2.3

▶ Transmission Control Protocol, Src Port: 443, Dst Port: 61859, Seq: 3182, Ack: 3758, Len: 169

▶ Transport Layer Security

▼ TRANSUM RTE Data

[RTE Status: OK]

[Req First Seg: 2539]

[Req Last Seg: 2543]

[Rsp First Seg: 2566]

[Rsp Last Seg: 2575]

[APDU Rsp Time: 0.023012000 seconds]

[Service Time: 0.020142000 seconds]

[Req Spread: 0.000959000 seconds]

[Rsp Spread: 0.001911000 seconds]

[Trace clip filter: tcp.stream==19 && frame.number>=2539 && frame.number<=2575 && tcp.len>0]

[Calculation: Generic TCP]

TPM2.0

TPNCP

TRANSUM

TSDNS

TSP

TTE

TURNCHANNEL

TUXEDO

TZSP

UA3G

UASIP

UAUDP

UBDP

UBERTOOTH

UCP

UDP

UDP-Lite

TRANSUM RTE Data

Capture position Client

Subdissector reassembly enabled

Output RTE data for these TCP service ports 25,80,443,1433,50001

Output RTE data for these UDP service ports 137-139

Discard orphaned TCP Keep-Alives

Add RTE data to the first request segment

Add RTE data to the last request segment

Add RTE data to the first response segment

Add RTE data to the last response segment

Enable debug info



Demo & Exercise

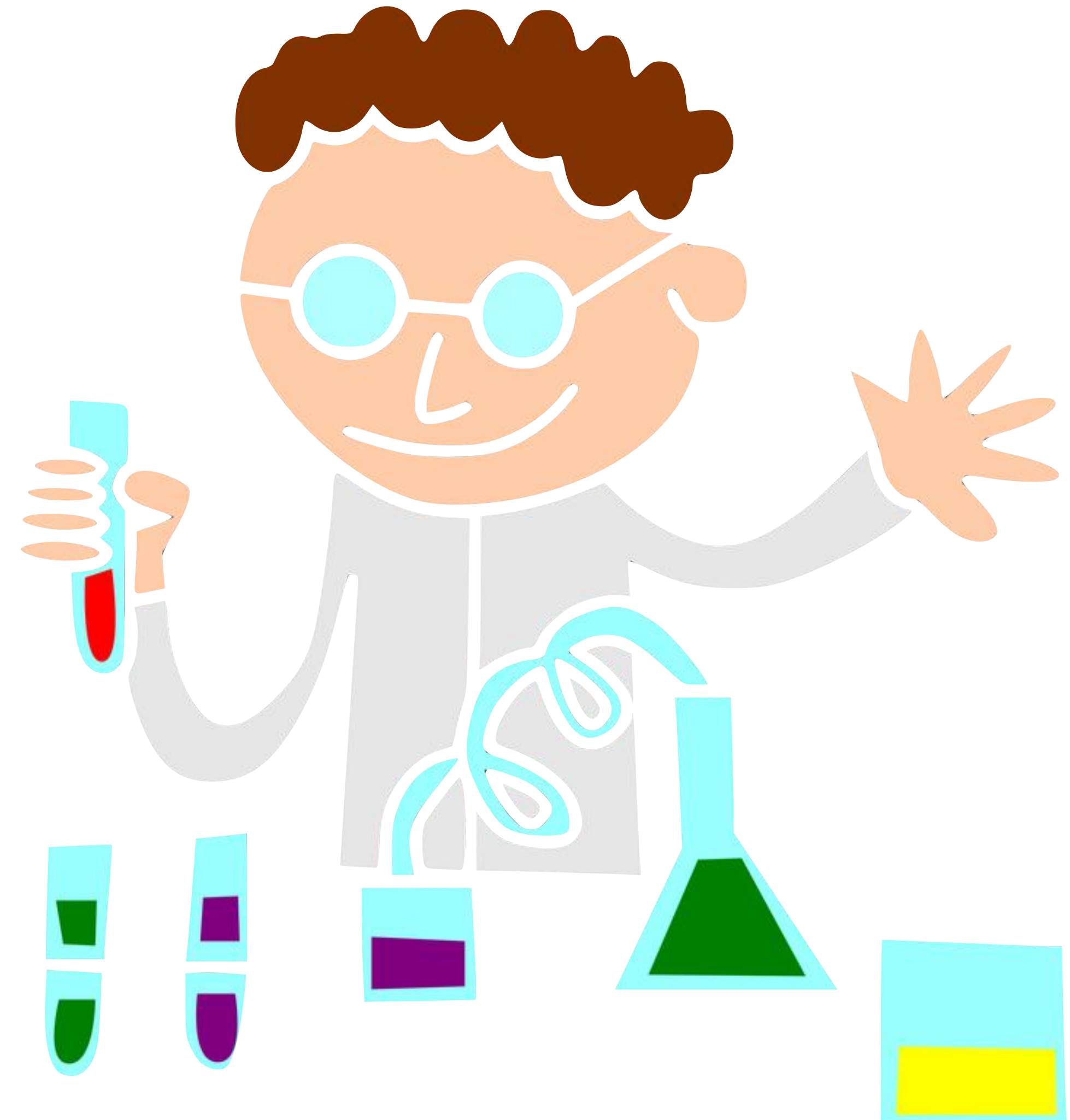


- Demo

- File: dv-ov-ev-wildcard.pcapng

- Exercise

- use filter "tcp.port==61945 && tcp.len>0"
- How many HTTP requests/responses?
- What was the "size" of the last request?
- What was the "size" of the last response?





Time for a poll

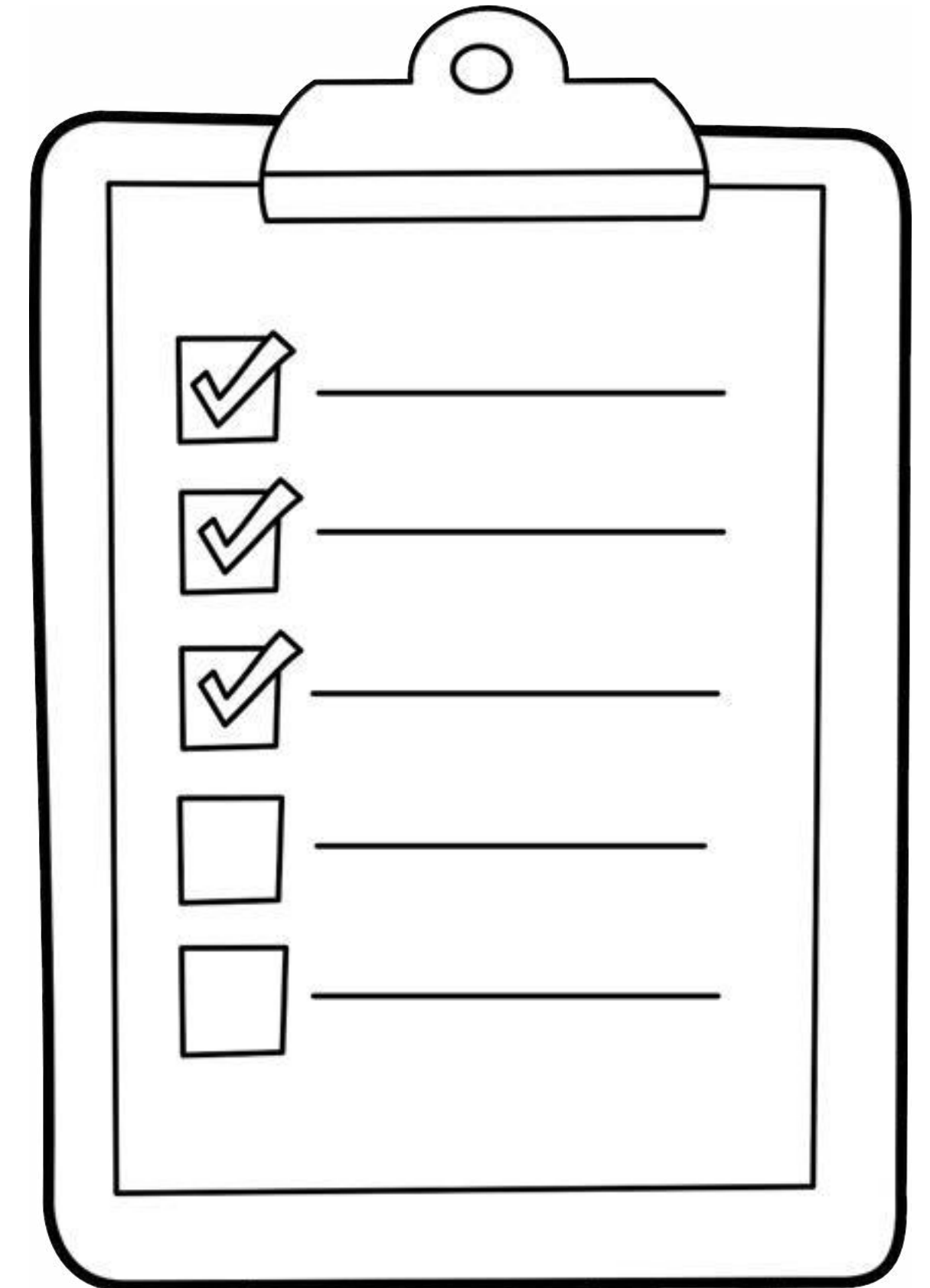




Agenda



- TLS fundamentals
 - The need for TLS (and what happens to SSL?)
 - Cryptology 101 & PKI
 - Troubleshoot the TLS handshake (SSLv3 - TLSv1.2)
 - Troubleshoot the TLS handshake (TLSv1.3)
- Analysing Application Data
 - Without decryption
 - **With decryption using the servers private key**
 - With decryption using TLS session keys
- Creating configuration profiles for TLS analysis
- SSL/TLS vulnerabilities & privacy concerns
- Summary and Q&A



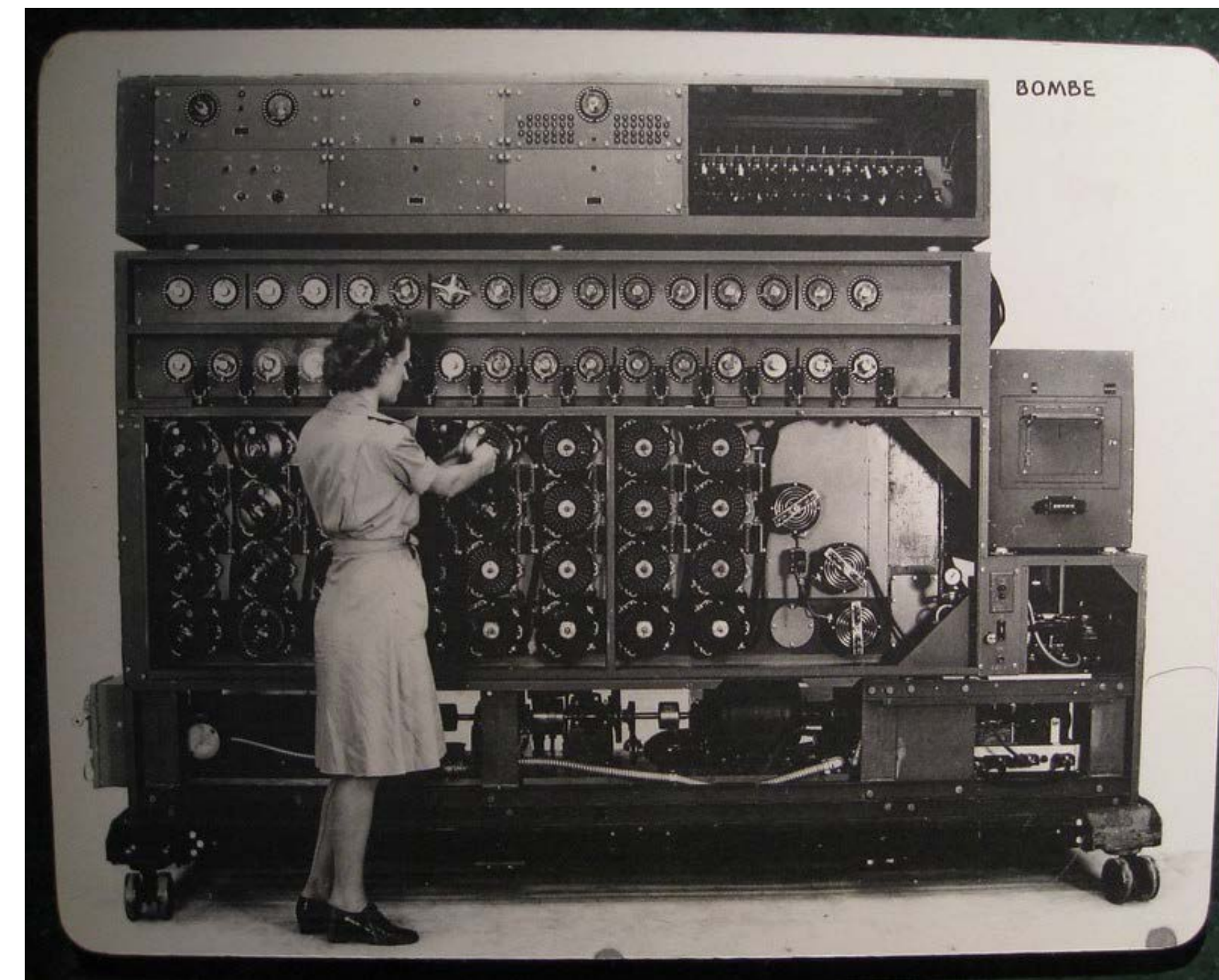
<https://www.needpix.com/photo/download/94588/checklist-action-check-list-verification-tick-approved-okay-check-mark>



Why TLS decryption?



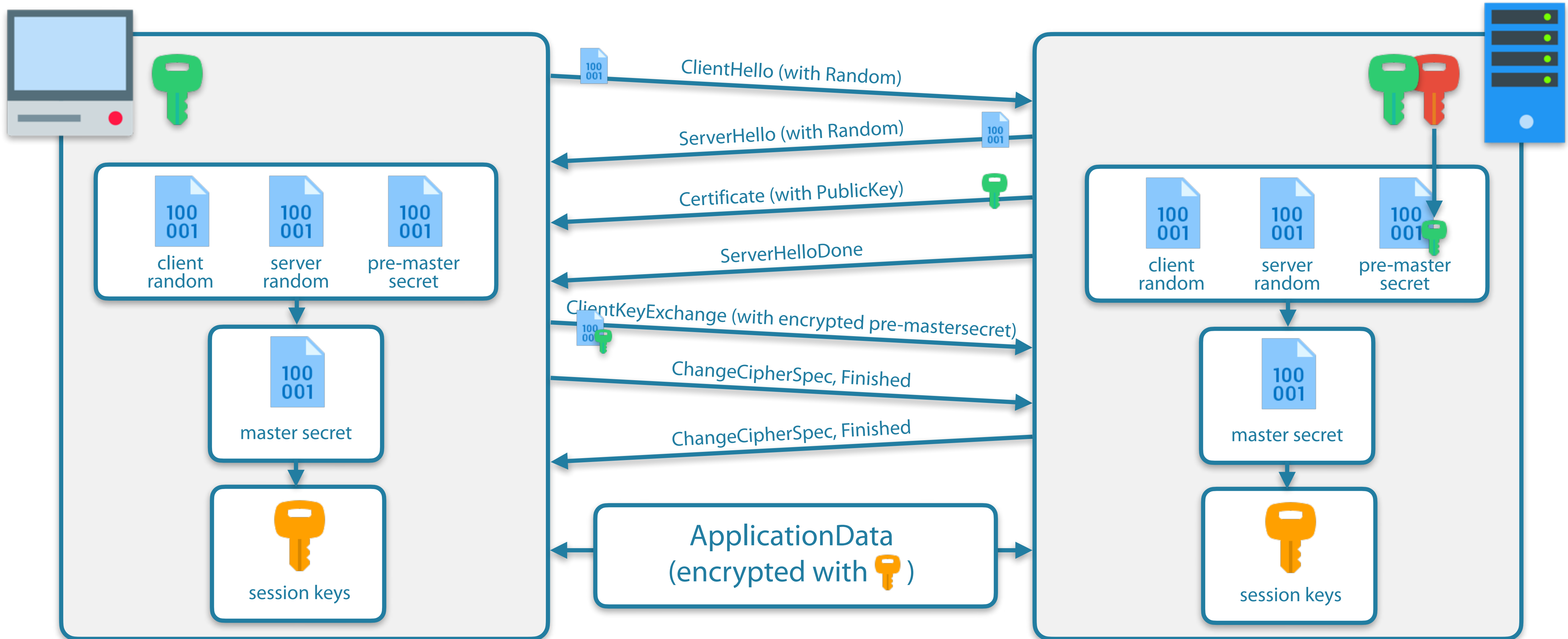
- Application Performance Monitoring
 - Which requests were slow???
- Troubleshooting
 - Why is this not working???
- Security
 - Malware/virus/etc. detection
 - Some level of detection by behavioural analysis or fingerprinting (JA3, JA3S, nDPIv3,etc)
 - Dataloss detection and prevention
 - GDPR
 - Forensics



<https://www.flickr.com/photos/brewbooks/3318600273>



RSA key exchange

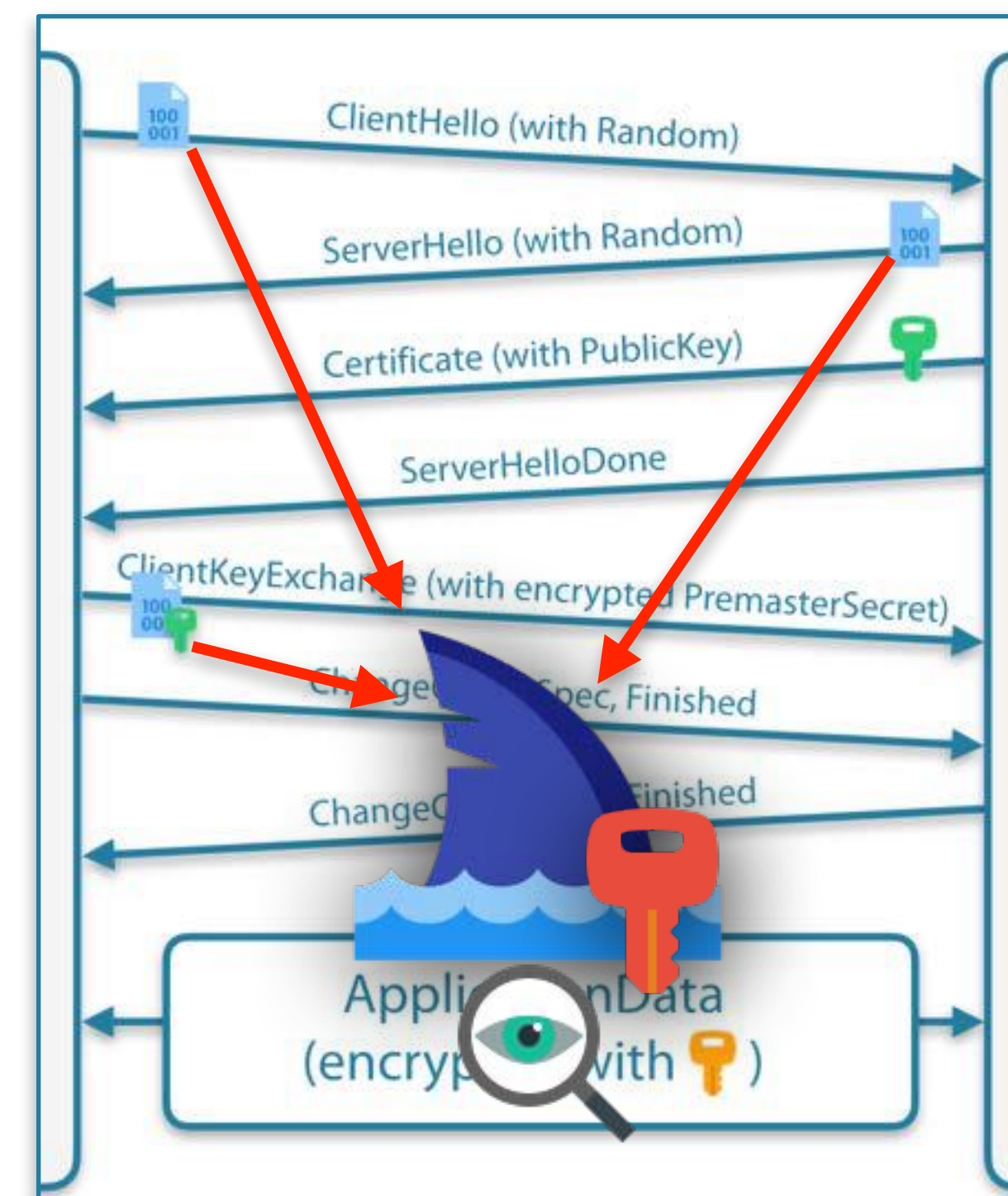




Decryption Recipe I



- Client Random
 - Captured in the **ClientHello**
- Server Random
 - Captured in the **ServerHello**
- Pre-master Secret
 - Captured in the **ClientKeyExchange** but encrypted with server public key
 - Use server **private key** to decrypt
- Calculate Master Secret
- Calculate Session Keys
- Decrypt application traffic with Session Keys





Decryption in Wireshark I

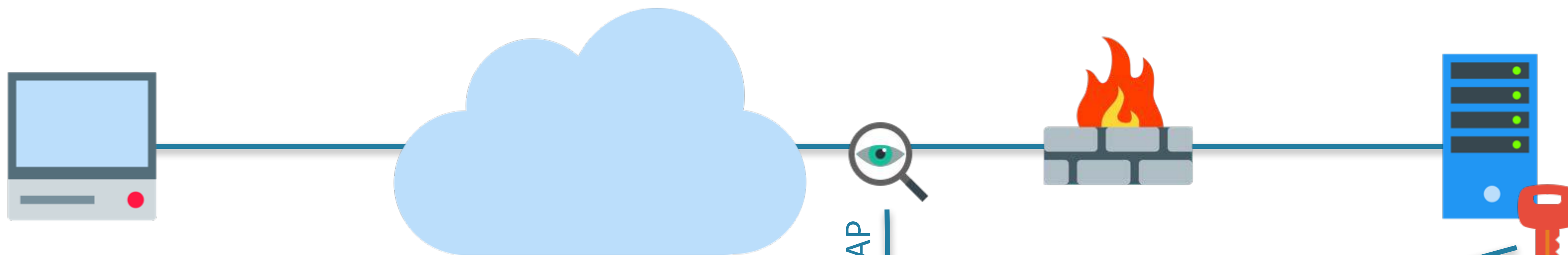


- Start capturing **before** opening the browser
 - Otherwise you will miss the full TLS handshake
- Provide the (right?!) server private key
 - PKCS#1 in PEM format without passphrase
 - Convert format and/or strip passphrase with OpenSSL if needed
 - PKCS#12 format (optionally with passphrase)
 - Connect to NetHSM (some support recently added)
- Want to try yourself?
 - Go to: <https://www.cloudshark.org/captures/a25364a5f672>
 - Click on More Info -> Download
 - And look at the capture file comments for instructions

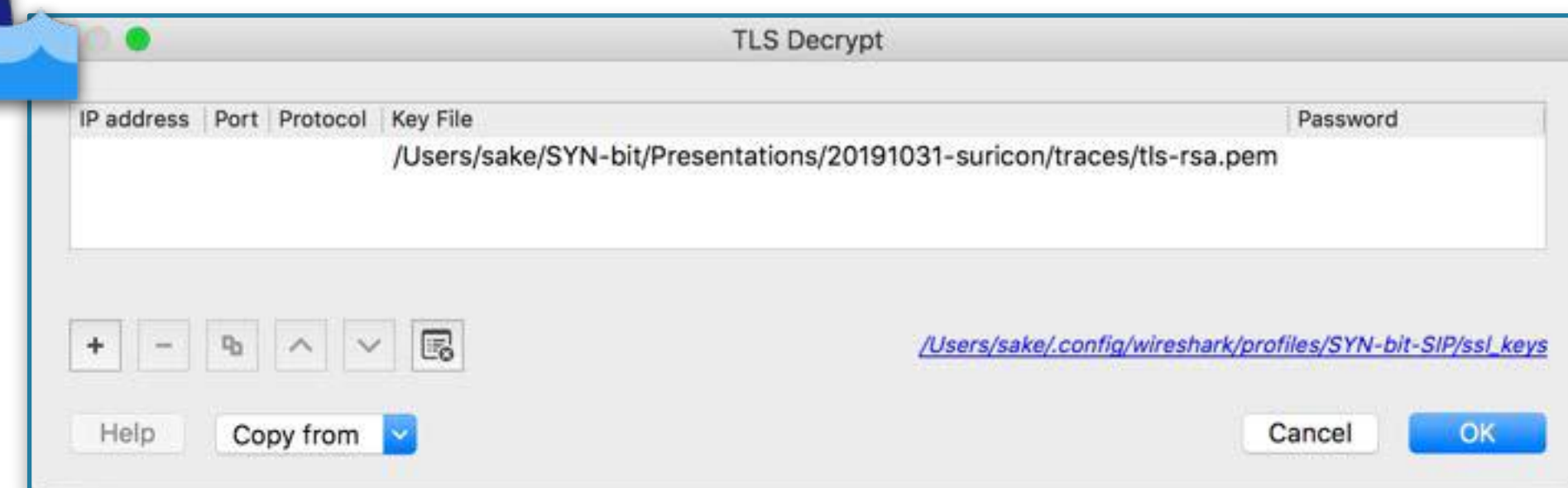
```
▶ Frame 11: 491 bytes on wire (3928 bytes captured) on interface 0:0:0:0:0:0  
▶ Ethernet II, Src: Vmware_08:00:27:00:00:00, Dst: 08:00:27:00:00:00  
▶ Internet Protocol Version 4, Src: 192.168.3.3, Dst: 192.168.3.3  
▶ Transmission Control Protocol, Src Port: 443, Dst Port: 80  
▶ Transport Layer Security  
▼ Hypertext Transfer Protocol  
  ▶ GET / HTTP/1.1\r\n    Host: 192.168.3.3\r\n    User-Agent: Mozilla/5.0 (Windows NT 6.0; rv:2.0; Firefox/3.0; .NET CLR 2.0.50727.303)\r\n    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n    Accept-Language: en-us\r\n    Accept-Encoding: gzip, deflate\r\n    Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3\r\n    Keep-Alive: 300\r\n    
```




TLS Decryption with private key



```
▶ Frame 11: 491 bytes on wire, 491 bytes captured on interface 0
▶ 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 Version 4, Src: 192.168.3.1, Dst: 192.168.3.3
▶ Transmission Control Protocol, Src Port: 18736, Dst Port: 443, Seq: 269, Ack: 2485, Len: 437
▶ Transport Layer Security
  ▼ Hypertext Transfer Protocol
    GET / HTTP/1.1\r\n
    Host: 192.168.3.3\r\n
    User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.8) Gecko/2009032609 Firefox/3.0.8\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-us,en;q=0.5\r\n
    Accept-Encoding: gzip,deflate\r\n
    Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
    Keep-Alive: 300\r\n
```





Accessing the private key



- You should **NOT** be able to get the private key
 - ...unless it is a dev/test/acc environment
- Out-of-luck with HSM cards
- You could interface with NetHSM devices
 - Limited support added to Wireshark added recently
- Protect the private key with your life!
 - Do you really want to have it on your laptop???

```
# cat private.pem
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQDrHdbb+yGE6m6EZ03bXURpZCjch2H6g97ZakJVGrjLZFfettBA
EYa8vYYxWsf8KBpEZeksSCsDA9MnU2H6QDjzqdOnaSWfeXMar40sCOpauStpreq7
q1hk8iOqy+f4KijRrhWplh1QW1A8gtSIg137pyUhW+WsfwxKwmzjGIC1SwIDAQAB
AoGBAMneA9U6KIxb+JUg/99c7h9W6wEvTYHNTXjf6psWA+hpuQ82E65/ZJdszL6
+8vfbrYdPfdcOznKdehE6lGgBIRjhb00e0xTxfbt1OWcVeqW17t52QizbaK/dt/j
Fn8RyixrU4NSSPWG7NCyLJwYU2lgstUbhHg2Hz9Mz59GgS2xAkEA+fxI/b65qF29
/OAz8iHrmGU/qmwrP2+I+gNB1ji2bIPfXLg/V3Q70golpeRhcORB1SbMYP8+PCU5
5NsC1iRGUQJBAPDMO/w8Rz14waw885c3DF1Iar81H3VCBoswUqPBDDQvtSatMq1c
BA27q5Re+XewBGx4AbP84lPPsDD7ax8eWiMCQQCEUfVdRgq4My+w3usAwa4bFYYX
fH2EbkG/v9upUIpZdZHXXn3BiP113hNB910RyvOCp7BHpLbIVhiIqtucisWZakAs
b6QKMh16r5wd6smQ+CmhOEnqyT5AIww12RIr9GbfIpTbtbRQw/EcQOCx9wFiEfo
tGSSEFi72rHK+DpJqRI9AkEA72gdyXRgPfgOS3rfQ3DBcImBQvDSCBa4cuU1XJ1/
MO93a8v9Vj87/yDm4xsBDsoz2PyBepawHV1IvZ6jDD0aXw==
-----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):

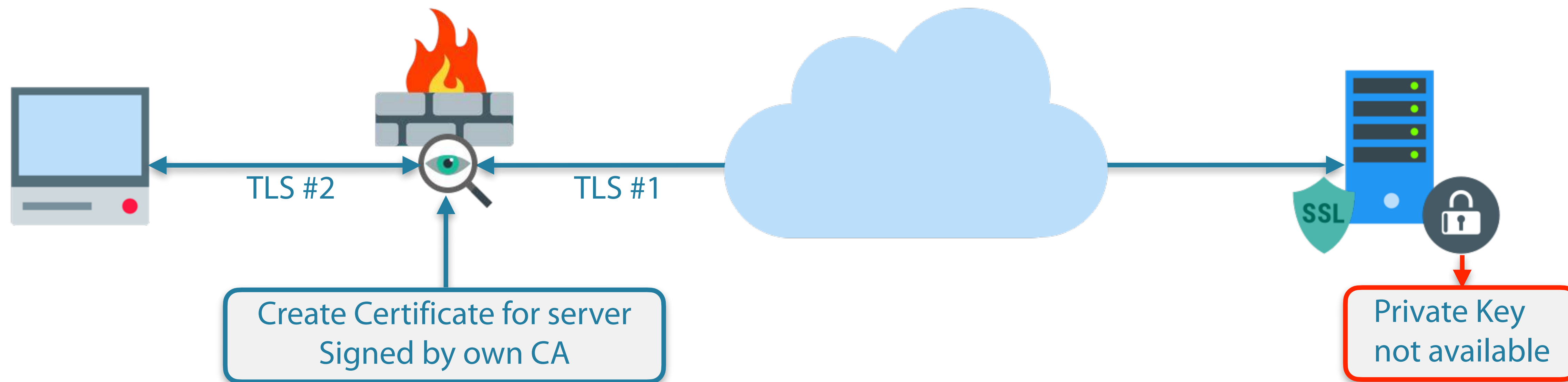
```
root@mgmt# openssl rsa -inform DER -in der.key -out pem.key
Enter pass phrase for encrypted.key: <passphrase>
writing RSA key
root@mgmt#
```

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#
```



Active (MITM) Decryption



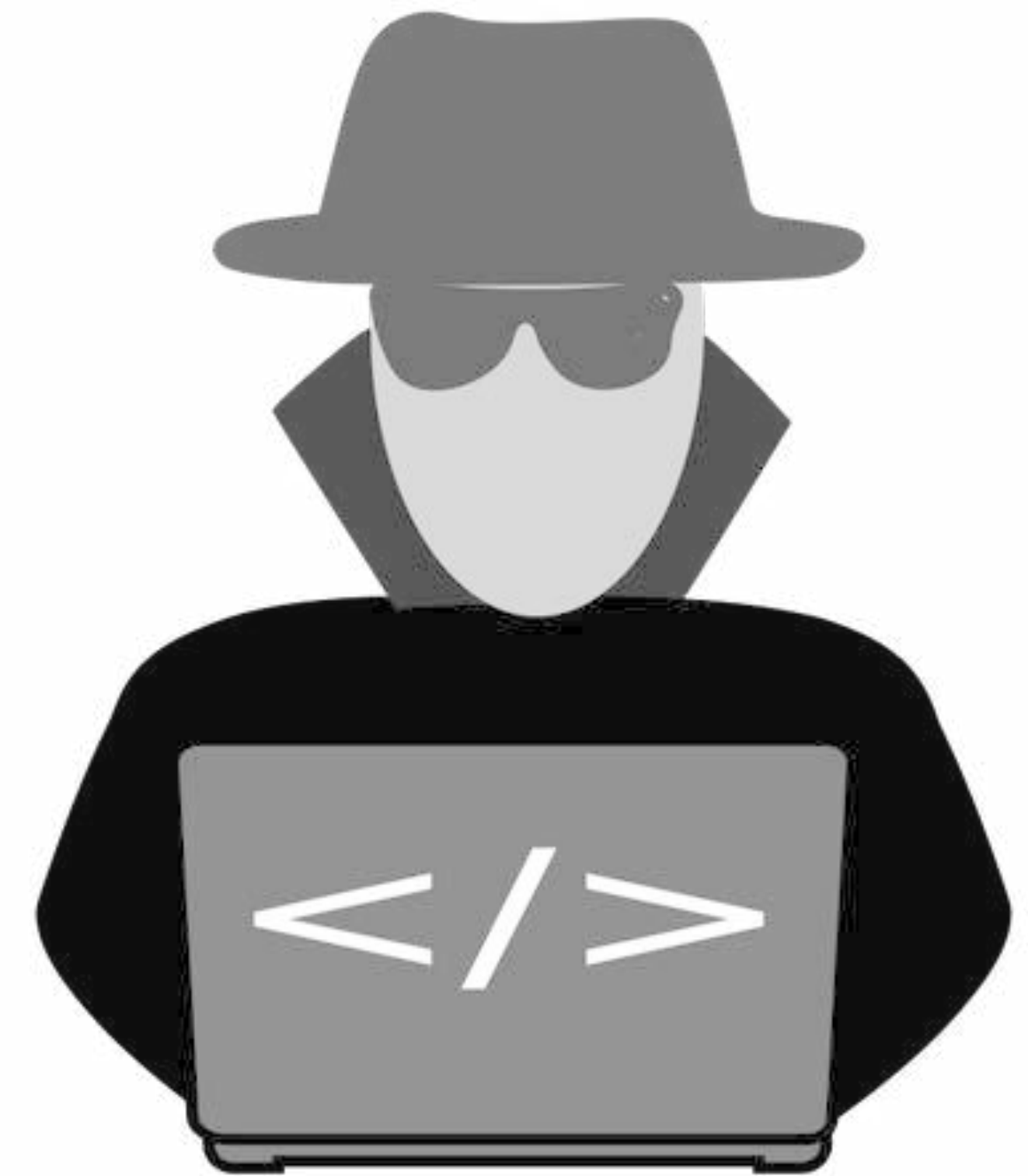
Caveats:
Client needs to have the RootCA of the Firewall in it's trust store
Certificate pinning will fail
Mutual TLS is not possible



Some MITM solutions



- Next-Gen firewalls
 - Palo Alto Networks (can send decrypted traffic)
 - Checkpoint
 - Fortigate
- Application Delivery Controllers
 - F5 (can export TLS session keys with an iRule)
 - NetScaler (can export TLS session keys when capturing)
- Proxy Applications
 - **mitmproxy (can export TLS session keys)**
 - Fiddler
 - Charles Proxy
 - BurpSuite
 - PolarProxy (can save decrypted traffic in pcap)



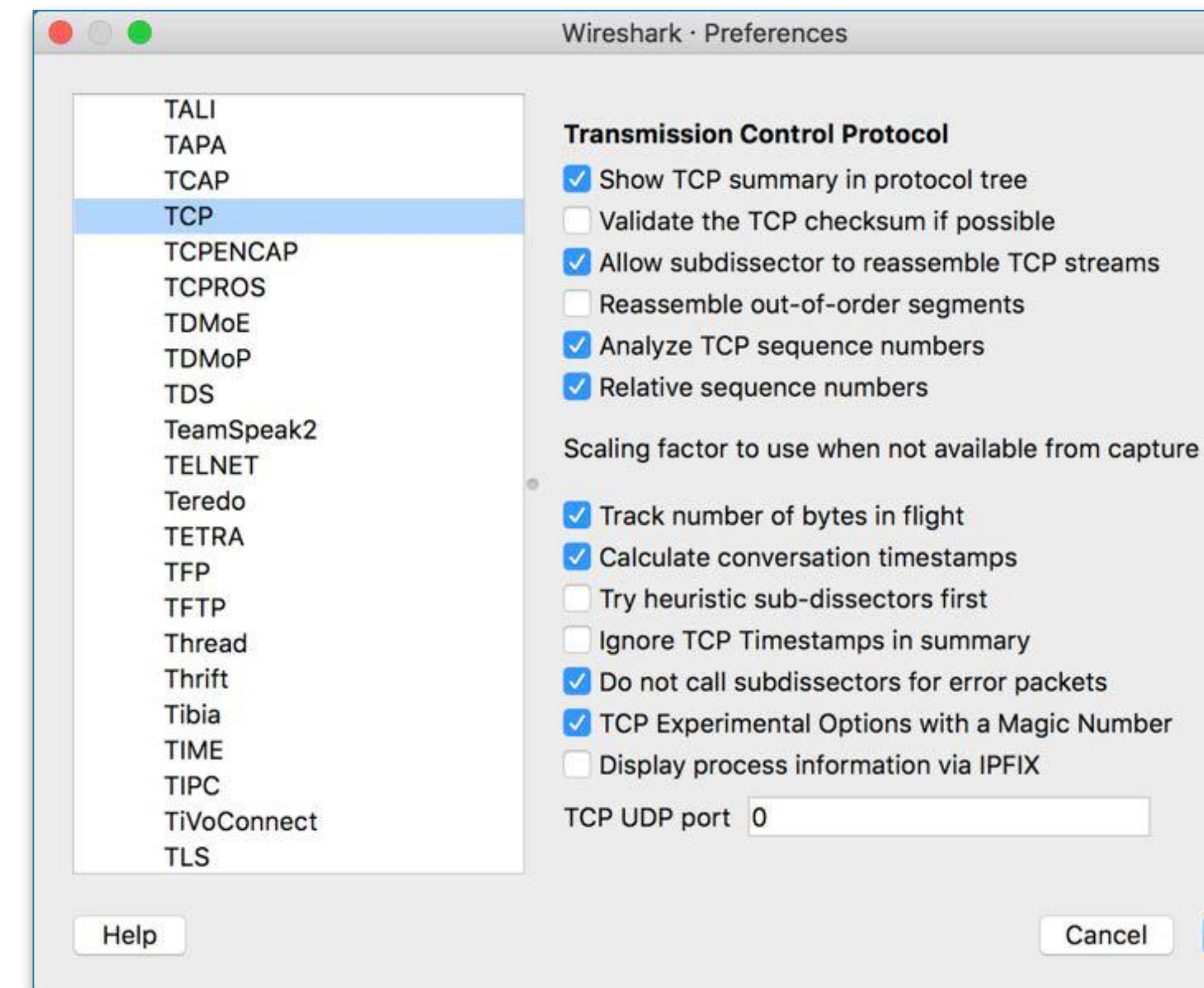
<https://publicdomainvectors.org/en/free-clipart/Spy-behind-computer/68287.html>



Wireshark requirements



- Enable defragmentation on IP layer
- Enable reassembly at TCP and TLS layer
 - And also on HTTP layer if traffic is HTTPS
- Disable checksum validation on IP and TCP
 - Checksum offloading results in bad checksums
 - Reassembly will be disabled with bad checksums
- Good news, these are all default settings!





TLS session resumption



- Master Secret is...
 - Cached on Client and Server (indexed by SessionID)
 - Sent in TLS session tickets
 - ▶ master secret encrypted by server, stored on client
- So no {Client,Server}KeyExchange in PCAP???
 - **No SessionKey, no decryption!**
- However, decryption will still work if you have captured the initial (full) TLS handshake

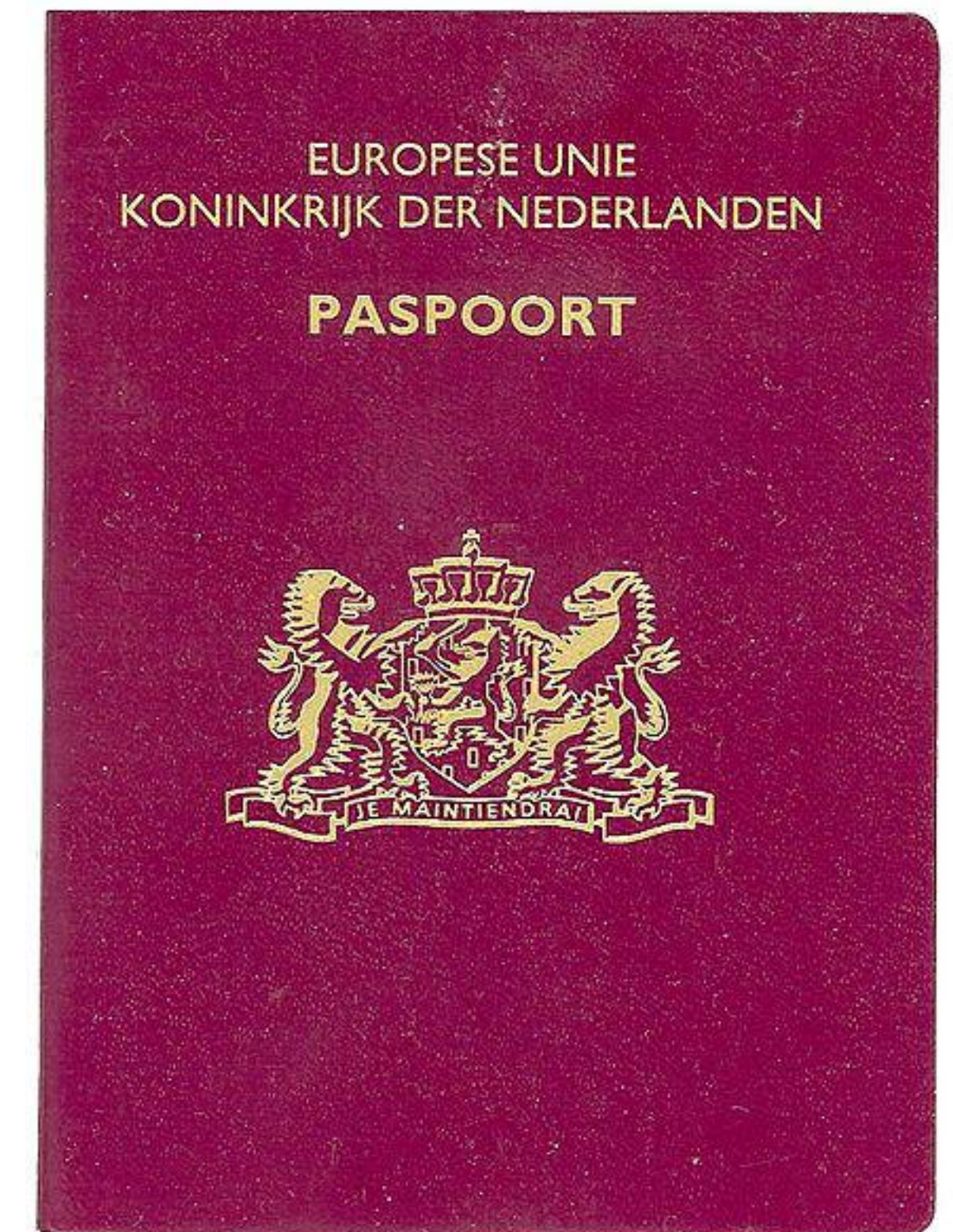




Mutual TLS



- Server requests certificate from client
- Client presents certificate to server
 - Contains the clients public key
- Client sends CertificateVerify message
 - Encrypted with clients private key
- Server can decrypt with public key
 - This proves the client has the private key
- Encryption/Decryption is not affected
- **So still only server private key needed for decryption**



https://nl.wikipedia.org/wiki/Bestand:Paspoort_NL.jpg



Check if decryption works



- Higher layer dissection might fail or packet-loss and re-ordering might break decryption later in the session so it might look like decryption failed
- Best check if decryption was successful:
 - Filter on `tls.handshake`
 - Check if "Encrypted handshake message" has changed to "Finished"

No.	Time	Delta	Source	Destination	Protocol	Length	Info
4	15:17:36.050028	0.000000	201.0.113.10	35.153.107.164	TLSv1.2	587	Client Hello
6	15:17:36.161459	0.111431	35.153.107.164	201.0.113.10	TLSv1.2	1518	Server Hello
12	15:17:36.162608	0.001149	35.153.107.164	201.0.113.10	TLSv1.2	1252	Certificate, Server Key Exchange, Server Hello Done
14	15:17:36.164351	0.001743	201.0.113.10	35.153.107.164	TLSv1.2	196	Client Key Exchange, Change Cipher Spec, Finished
15	15:17:36.276708	0.112357	35.153.107.164	201.0.113.10	TLSv1.2	121	Change Cipher Spec, Finished



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
AoGBAMneA9U6KIxb+JUg/99c7h9W6wEvTYHNTXjf6psWA+hpuQ82E65/ZJdszL6
...
b6QKMh16r5wd6smQ+CmhOEnqyT5AIwwl2RIr9GbfIpTbtbRQw/EcQOCx9wFiEfo
tGSsEFi72rHK+DpJqRI9AkeA72gdyXRgPfgOS3rfQ3DBcImBQvDSCBa4cuU1XJ1/
MO93a8v9Vj87/yDm4xsBDsoz2PyBepawHV1IvZ6jDD0aXw==
-----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
ERok2jPr2BzAaz15RBxS2TY/+7x0/dHD11sF3LnJUoNruo77TERxqgzOI0W1VDRA
...
ygw5JslxgiN18F36E/cEP5rKvVYvfEPMa6IsiRhFzk1jLAuZihVWc7JodDf+6RKV
yBXrK/bDtdEih+bOnYu+ZDvjAzVz9GhggCW4QHNboDpTxrrYPkj5Nw==
-----END RSA PRIVATE KEY-----
```



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_
packet_from_
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)



The screenshot shows the Wireshark interface with the following details:

- Packet List Pane:** Shows a selected packet of type 'Certificate' (length 2328 bytes) under the 'Handshake Protocol' layer.
- Packet Bytes Pane:** Displays the raw data of the selected packet in hexadecimal and ASCII format.
- Context Menu:** A right-click menu is open over the selected packet, with 'Export Selected Packet Bytes...' highlighted.
- Export Raw Data Dialog:** A dialog box titled 'Wireshark: Export Raw Data' is open, showing the file name 'cert.der' and 'Save as type' set to 'Raw data (*.bin, *.dat, *.raw)'. The dialog also indicates that 1079 bytes of raw binary data will be written.



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 Make sure that the private key matches the (server) certificate that
is used in the tracefile.
$
$ openssl
ce71158d3851a885314c264863142389
$
$ openssl rsa -noout -modulus -in private.sharkfest.local.key | openssl md5
a29682af822b4cd064d39d4ccd1e0e6c
$
```



Decrypt-problem III



Filter: `ssl.handshake` Expression... Clear Apply Save

No.	Time	Delta	Source	Destination	Protocol	Length	Info
4	0.012133000	0.000000000	192.168.0.133	172.217.17.35	TLSv1.2	262	Client Hello
6	0.043228000	0.031095000	172.217.17.35	192.168.0.133	TLSv1.2	1484	Server Hello
10	0.046051000	0.002823000	172.217.17.35	192.168.0.133	TLSv1.2	358	Certificate
13	0.047932000	0.001881000	192.168.0.133	172.217.17.35	TLSv1.2	324	Client Key Exchange, Change Cipher Spec, Hello Request, Hello
19	0.060356000	0.012424000	172.217.17.35	192.168.0.133	TLSv1.2	360	New Session Ticket, Change Cipher Spec, Hello Request, Hello

.....

▶ 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 (172.217.17.35), Dst: 192.168.0.133 (192.168.0.133)

▶ Transmission Control Protocol, Src Port: https (443), Dst Port: 64320 (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 (3)

Length: 511

▼ Handshake Protocol

Handshake Type: Server Hello

Length: 507

Version: TLS 1.2 (3)

▶ 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: Unknown 23

▶ Extension: SessionTicket TLS

▶ Extension: signed_certificate_timestamp

▶ Extension: Application Layer Protocol Negotiation

▶ Extension: Unknown 30032

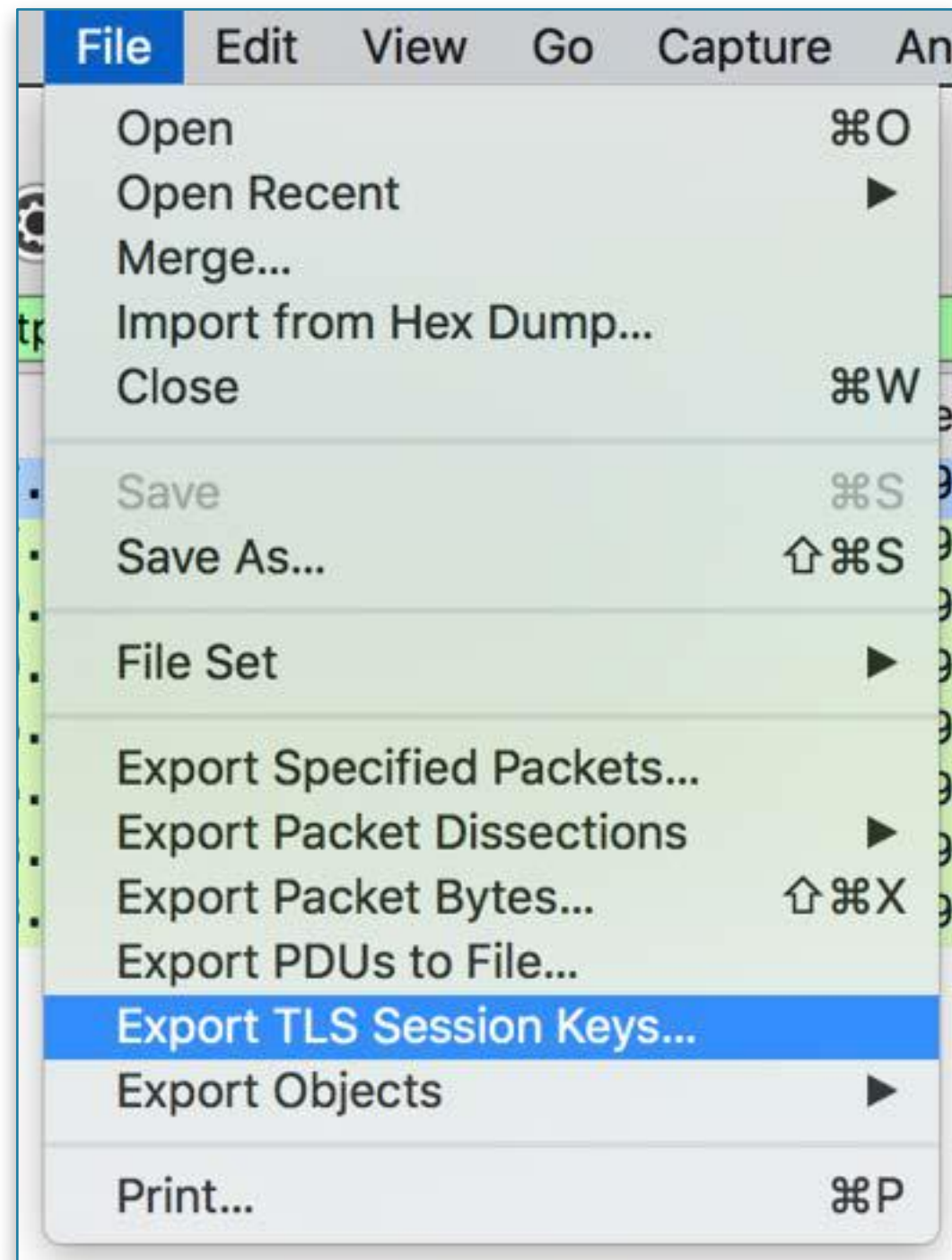
▶ Extension: ec_point_formats

Wireshark is only capable of decrypting sessions with the private key of the server if there was an RSA key exchange and not when Diffie Hellman key generation was used.

Make sure to use a cipher list restricted to RSA or ...



Exporting TLS Session Keys



The name of a file which contains a list of (pre-)master secrets in one of the following formats:

```

RSA <EPMS> <PMS>
RSA Session-ID:<SSLID> Master-Key:<MS>
CLIENT_RANDOM <CRAND> <MS>
PMS_CLIENT_RANDOM <CRAND> <PMS>

```

Where:

- <EPMS> = First 8 bytes of the Encrypted PMS
- <PMS> = The Pre-Master-Secret (PMS) used to derive the MS
- <SSLID> = The SSL Session ID
- <MS> = The Master-Secret (MS)
- <CRAND> = The Client's random number from the ClientHello message

(All fields are in hex notation)

```

$ cat session.keys
RSA Session-ID:fbcf322128ed0a00b272d6ac85843f50deccdd94ac33261523189639f5ba189a Master-Key:bda6ea472f6c39a9fcfd5dc79eb161d1a4cae5d924fdde800f276263fd6df1ee8ed246b5a6412e328eb85744c9bf7cf2
RSA Session-ID:db00c2aad79cfda109ce4f65a9801aa8d5f1bbeb9e1f848f6a2f7551f9de7577 Master-Key:92cdc769c670ba6f48cfe756992ad435401a26d0235900c0f67c846b5f360c108df167ca6b6f443f4d2b118de0ccadb8
CLIENT_RANDOM 49eb46c69e4c71ebfb757f67630e84ad2da1b0c2c1a9cbbald7a48744ed2e640 92cdc769c670ba6f48cfe756992ad435401a26d0235900c0f67c846b5f360c108df167ca6b6f443f4d2b118de0ccadb8
CLIENT_RANDOM 49eb470dd56239e9c8c4e7acb752614255177a3e77f47660a445385268335563 bda6ea472f6c39a9fcfd5dc79eb161d1a4cae5d924fdde800f276263fd6df1ee8ed246b5a6412e328eb85744c9bf7cf2
CLIENT_RANDOM 49eb469edd819516fc5ddd097428d410d7852e2579e2e8903cdb331c763dca9 92cdc769c670ba6f48cfe756992ad435401a26d0235900c0f67c846b5f360c108df167ca6b6f443f4d2b118de0ccadb8
CLIENT_RANDOM c8c82c81e65b2ba9df1234e660e5db4cc35b93720ba4eb2a1c01138b7ad56eaa 91788e318de3d91734c1e6eb34366f26709db19f56d9c8dde5248bb9a1560d2c9e5a2dc92d167c9df1fadf36fb395a09
$

```



Embedding TLS Session Keys



```
$ editcap -h
[...]
--inject-secrets <type>,<file>  Insert decryption secrets from <file>. List
                                supported secret types with "--inject-secrets help".
--discard-all-secrets          Discard all decryption secrets from the input file
                                when writing the output file. Does not discard
                                secrets added by "--inject-secrets" in the same
                                command line.
[...]
$ editcap --inject-secrets tls,session.keys tls-rsa.pcapng tls-embedded.pcapng
$
```

```
$ fgrep -f <(tshark -r tls-rsa.pcapng -Y tls.handshake.random -T fields -e tls.handshake.random) session.keys
CLIENT_RANDOM 49eb46c69e4c71ebfb757f67630e84ad2da1b0c2c1a9cbba1d7a48744ed2e640 92cdc769c670ba6f48cfe756992ad435401a26d0235900c0f67c846b5f360c108df167ca6b6f443f4d2b118de0ccadb8
CLIENT_RANDOM 49eb470dd56239e9c8c4e7acb752614255177a3e77f47660a445385268335563 bda6ea472f6c39a9fcfd5dc79eb161d1a4cae5d924fdde800f276263fd6df1ee8ed246b5a6412e328eb85744c9bf7cf2
CLIENT_RANDOM 49eb469edd819516fc5ddd097428d410d7852e2579e2e8903cdb331c763dca9 92cdc769c670ba6f48cfe756992ad435401a26d0235900c0f67c846b5f360c108df167ca6b6f443f4d2b118de0ccadb8
$
```

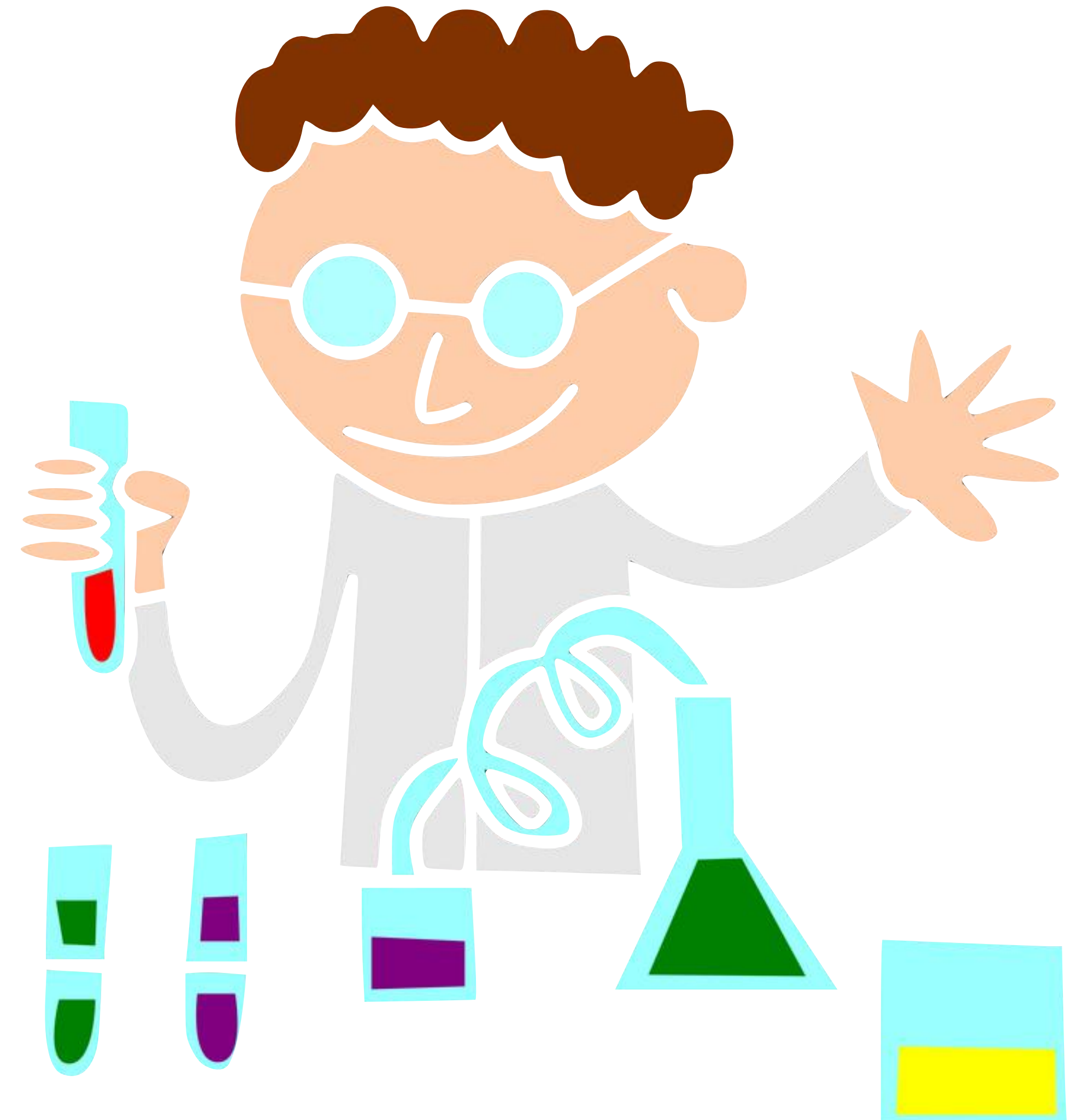


Exercise



- Exercise

- Open `tls-rsa.pcapng`
- Use `tls-rsa.pem` to decrypt
- Export session keys to `tls-rsa.keys`
- Edit file and remove one entry
- Change TLS protocol settings
 - Remove the RSA private key
 - Add (Pre)-master-secret log filename
- Is everything still encrypted?
- Good way to provide traces to 3rd parties

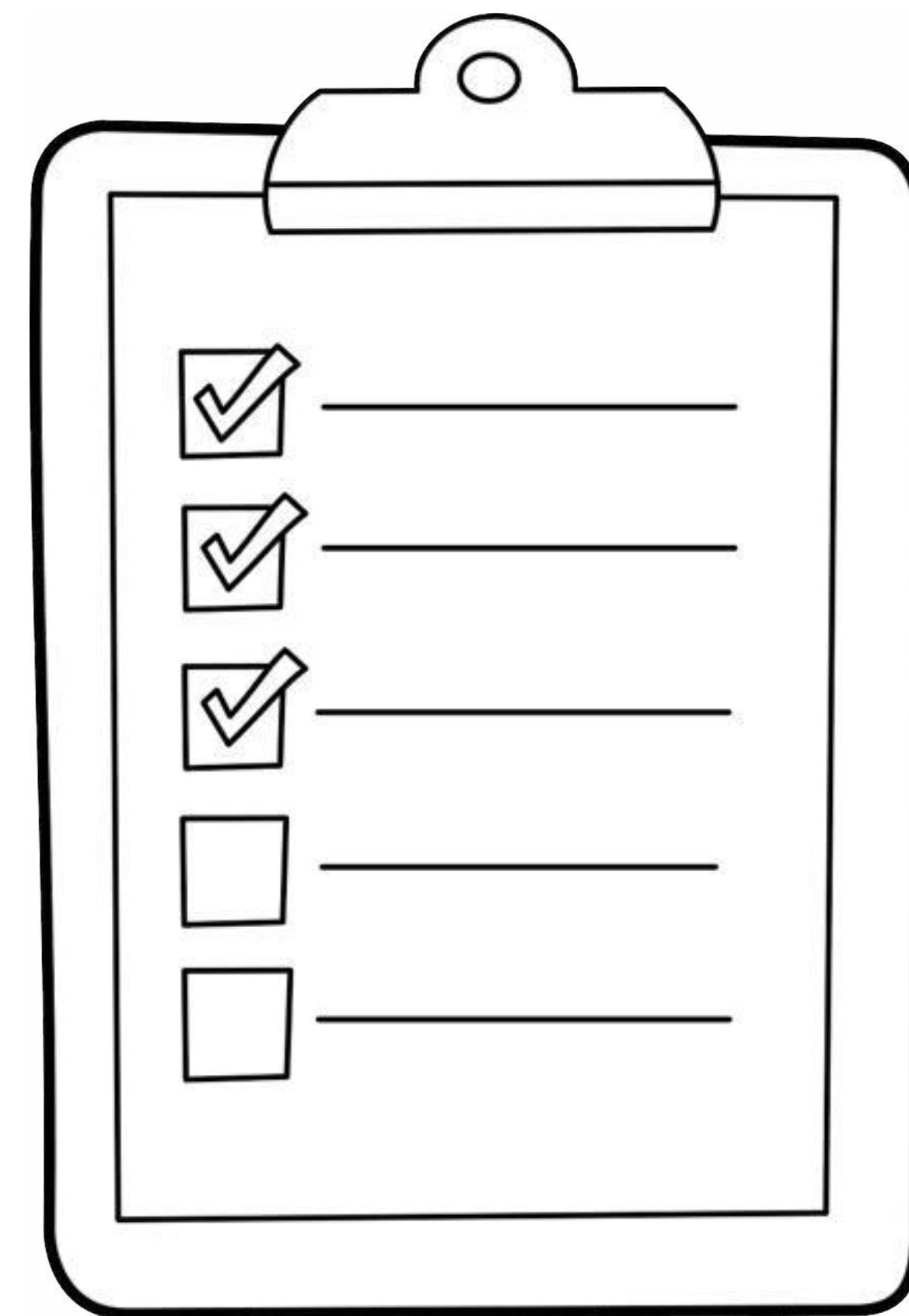




Agenda



- TLS fundamentals
 - The need for TLS (and what happens to SSL?)
 - Cryptology 101 & PKI
 - Troubleshoot the TLS handshake (SSLv3 - TLSv1.2)
 - Troubleshoot the TLS handshake (TLSv1.3)
- Analysing Application Data
 - Without decryption
 - With decryption using the servers private key
 - **With decryption using TLS session keys**
- Creating configuration profiles for TLS analysis
- SSL/TLS vulnerabilities & privacy concerns
- Summary and Q&A



<https://www.needpix.com/photo/download/94588/checklist-action-check-list-verification-tick-approved-okay-check-mark>



So long and thanks for all the fish



- Perfect Forwarding Secrecy

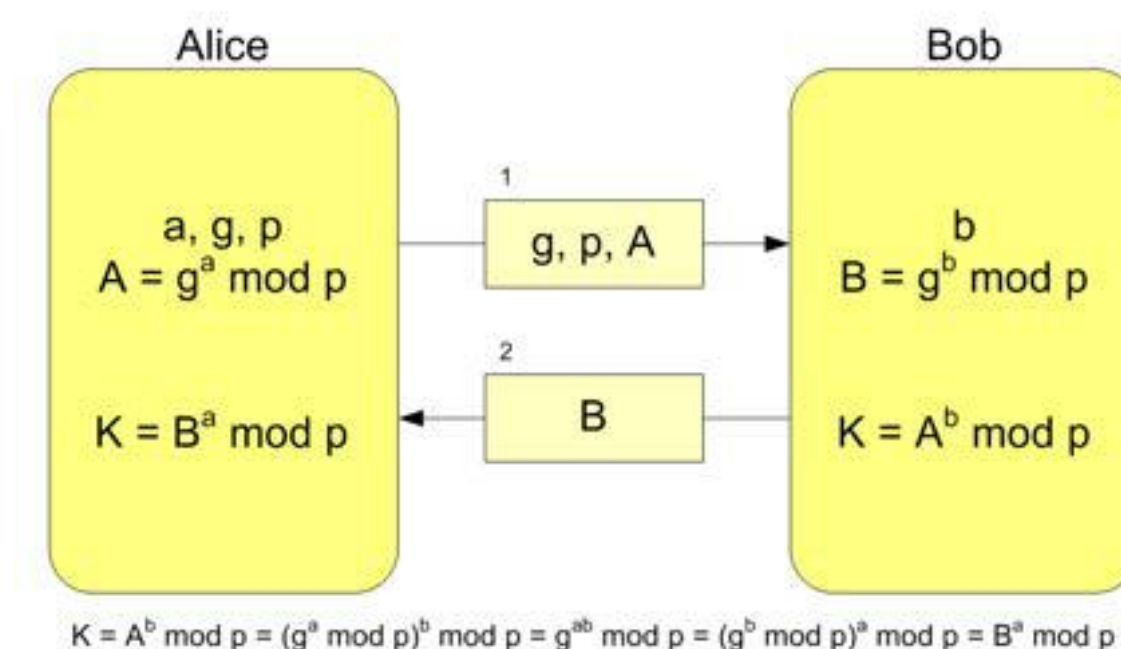
- **Diffie-Hellman algorithm uses ephemeral keys to create pre-master secret**
- Public key of the server only used for authentication
- Prevents decryption of current pcaps if private key is leaked in the future

- Today, >80% sessions use a cipher with DHE key negotiation

- See: <https://notary.icsi.berkeley.edu/#connection-cipher-details>

- TLSv1.3 finalised in august 2018 (RFC 8446)

- After 5 years and 28 versions!
- **No more RSA key exchanges!**

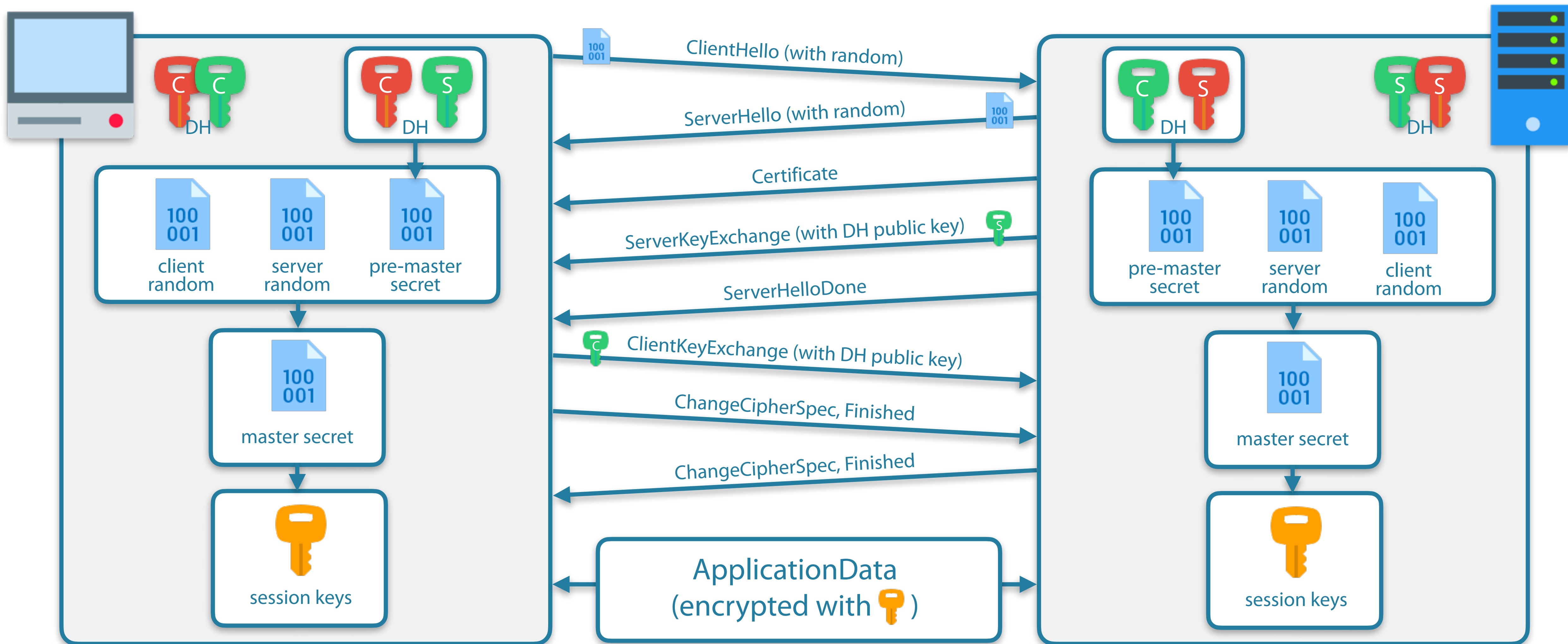


$$K = A^b \text{ mod } p = (g^a \text{ mod } p)^b \text{ mod } p = g^{ab} \text{ mod } p = (g^b \text{ mod } p)^a \text{ mod } p = B^a \text{ mod } p$$

<https://nl.wikipedia.org/wiki/Diffie-Hellman-sleuteluitwisselingsprotocol>



DHE key generation (TLSv1.2)

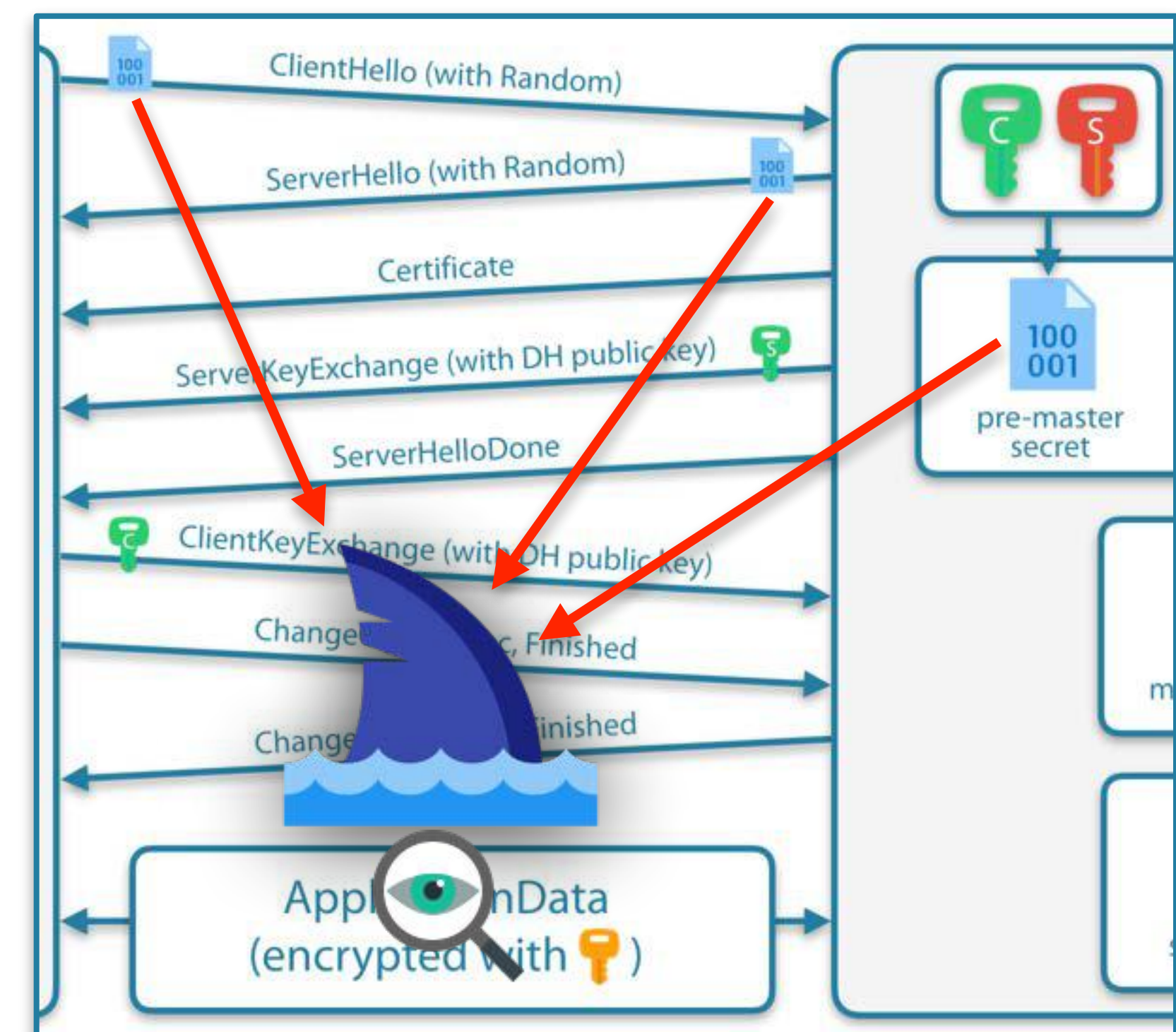




Decryption Recipe II



- Client random
 - Captured in the **ClientHello**
- Server random
 - Captured in the **ServerHello**
- Pre-master Secret
 - Can't be calculated, because DH private keys are ephemeral and only available in memory
 - **Pre-master Secret must be logged by either end-point**
- Calculate Master Secret
- Calculate Session keys
- Decrypt application traffic with Session keys





Decryption in Wireshark II

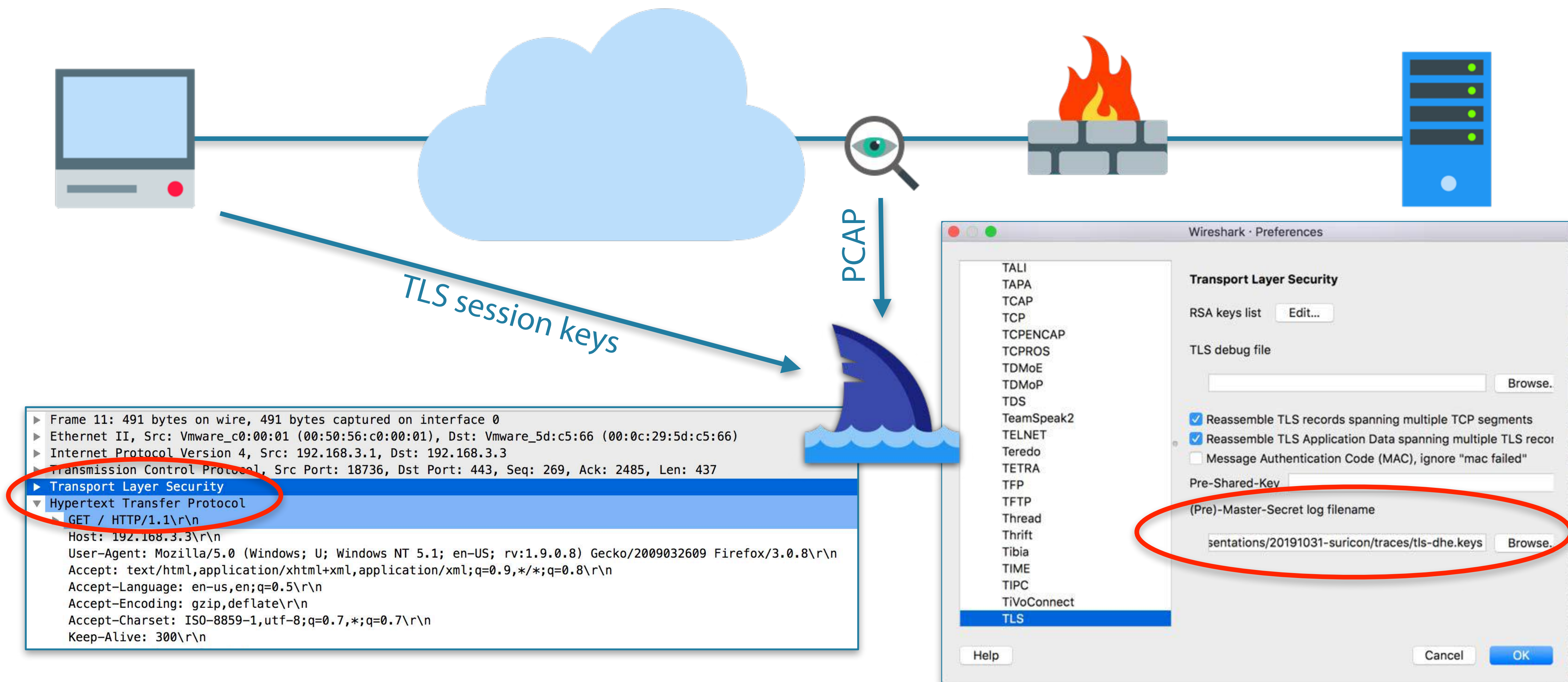


- Start capturing **before** opening the browser
 - Otherwise you will miss the full TLS handshake
- use `SSLKEYLOGFILE` environment variable
 - with cURL, Chrome, Firefox
- Point to the file with the TLS (Pre)-Master Secrets
- Want to try yourself?
 - Go to: <https://www.cloudshark.org/captures/1cd191698a5c>
 - Click on More Info -> Download
 - And look at the capture file comments for instructions

```
▶ Frame 11: 491 bytes on wire (400 bytes captured) on interface 0
▶ Ethernet II, Src: Vmware_08:00:27:00:00:00, Dst: 08:00:27:00:00:00
▶ Internet Protocol Version 4, Src: 192.168.3.3, Dst: 192.168.3.3
▶ Transmission Control Protocol, Src Port: 443, Dst Port: 80
▶ Transport Layer Security
▼ Hypertext Transfer Protocol
  ▶ GET / HTTP/1.1\r\n
    Host: 192.168.3.3\r\n
    User-Agent: Mozilla/5.0 (Windows NT 6.0; rv:2.0) Gecko/20100101 Firefox/4.0.1\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-us\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3\r\n
    Keep-Alive: 300\r\n
```




TLS Decryption with session keys



```
▶ Frame 11: 491 bytes on wire, 491 bytes captured on interface 0
▶ 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 Version 4, Src: 192.168.3.1, Dst: 192.168.3.3
▶ Transmission Control Protocol, Src Port: 18736, Dst Port: 443, Seq: 269, Ack: 2485, Len: 437
▶ Transport Layer Security
  ▼ Hypertext Transfer Protocol
    GET / HTTP/1.1\r\n
    Host: 192.168.3.3\r\n
    User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.8) Gecko/2009032609 Firefox/3.0.8\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-us,en;q=0.5\r\n
    Accept-Encoding: gzip,deflate\r\n
    Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
    Keep-Alive: 300\r\n
```

Wireshark - Preferences

- TALI
- TAPA
- TCAP
- TCP
- TCPCAP
- TCPROS
- TDMoE
- TDMoP
- TDS
- TeamSpeak2
- TELNET
- Teredo
- TETRA
- TFP
- TFTP
- Thread
- Thrift
- Tibia
- TIME
- TIPC
- TiVoConnect
- TLS**

Transport Layer Security

RSA keys list

TLS debug file

Reassemble TLS records spanning multiple TCP segments

Reassemble TLS Application Data spanning multiple TLS records

Message Authentication Code (MAC), ignore "mac failed"

Pre-Shared-Key

(Pre)-Master-Secret log filename



(Pre)-Master Secret logfile



- format of entries
- TLSv1.3 records not shown here

```
The name of a file which contains a list of
(pre-)master secrets in one of the following formats:

RSA <EPMS> <PMS>
RSA Session-ID:<SSLID> Master-Key:<MS>
CLIENT_RANDOM <CRAND> <MS>
PMS_CLIENT_RANDOM <CRAND> <PMS>

Where:
<EPMS> = First 8 bytes of the Encrypted PMS
<PMS> = The Pre-Master-Secret (PMS) used to derive the MS
<SSLID> = The SSL Session ID
<MS> = The Master-Secret (MS)
<CRAND> = The Client's random number from the ClientHello message

(All fields are in hex notation)
```



How to log TLS session keys I



- Applications using NSS or GnuTLS
 - Like Chrome and Firefox
 - use `SSLKEYLOGFILE` environment variable
- Applications using OpenSSL (>1.1.1) or BoringSSL (>d28f59c27bac 2015-11-19)
 - Use `SSL_CTX_set_keylog_callback` function
- cURL (> 7.58.0)
 - use `SSLKEYLOGFILE` environment variable
- `openssl s_client`
 - Logs Master-Key in output



How to log TLS session keys II



- Java Applications
 - use jSSLKeyLog (<http://jsslkeylog.sourceforge.net/>)
- From F5 Application Delivery Controllers
 - Can be done with a few iRule statements
- SChannel (TLS library in Windows)
 - Feature request is pending
- SecureTransport (TLS library in macOS)
 - No support yet



How to 'steal' TLS session keys



- Extract from memory (on the host/VM itself)
 - From a memory dump (for forensics)
 - From live memory (for monitoring)
 - Not just research, first commercial product already seen
- Extract from VM memory (on the hypervisor)
 - TeLeScope
 - TLSkex
- Use a proxy and log the keys
 - E.g. use mitmproxy (set SSLKEYLOGFILE)



Using SSLKEYLOGFILE



```
sake@MacSake:~$ tcpdump -nli en0 -w tls.pcapng -s0 host www.syn-bit.nl 2>/dev/null &
[1] 62366
sake@MacSake:~$ SSLKEYLOGFILE=./tls.keys curl https://www.SYN-bit.nl > /dev/null 2>&1
sake@MacSake:~$ kill %1
sake@MacSake:~$ cat tls.keys
CLIENT_RANDOM 8643368556C3DFBA4F25F5B8C14AC3AD47EE62A01BABDFF03E11AC384661547E BCDC163071B12AFE7B9FCC27A804137BA3AB4A4DDEF8B4476351D6499616CD0047B06CD51E8274569143ABABBDB1EB
[1]+  Done                  tcpdump -nli en0 -w tls.pcapng -s0 host www.syn-bit.nl 2> /dev/null
sake@MacSake:~$ tshark -r tls.pcapng -o tls.keylog_file:tls.keys -Y "tls.handshake"
  4  0.062445 0.000000  10.0.0.102 ? 77.111.240.149 TLSv1 299 0x0000 (0) Client Hello
  6  0.102868 0.040423  77.111.240.149 ? 10.0.0.102  TLSv1.2 1514 0xe95c (59740) Server Hello
  7  0.103027 0.000159  77.111.240.149 ? 10.0.0.102  TLSv1.2 1514 0xe95d (59741) Certificate [TCP segment of a reassembled PDU]
  8  0.103033 0.000006  77.111.240.149 ? 10.0.0.102  TLSv1.2 155 0xe95e (59742) Server Key Exchange, Server Hello Done
 11  0.105671 0.002638  10.0.0.102 ? 77.111.240.149 TLSv1.2 159 0x0000 (0) Client Key Exchange, Change Cipher Spec, Finished
 12  0.144614 0.038943  77.111.240.149 ? 10.0.0.102  TLSv1.2 117 0xe95f (59743) Change Cipher Spec, Finished
sake@MacSake:~$
```

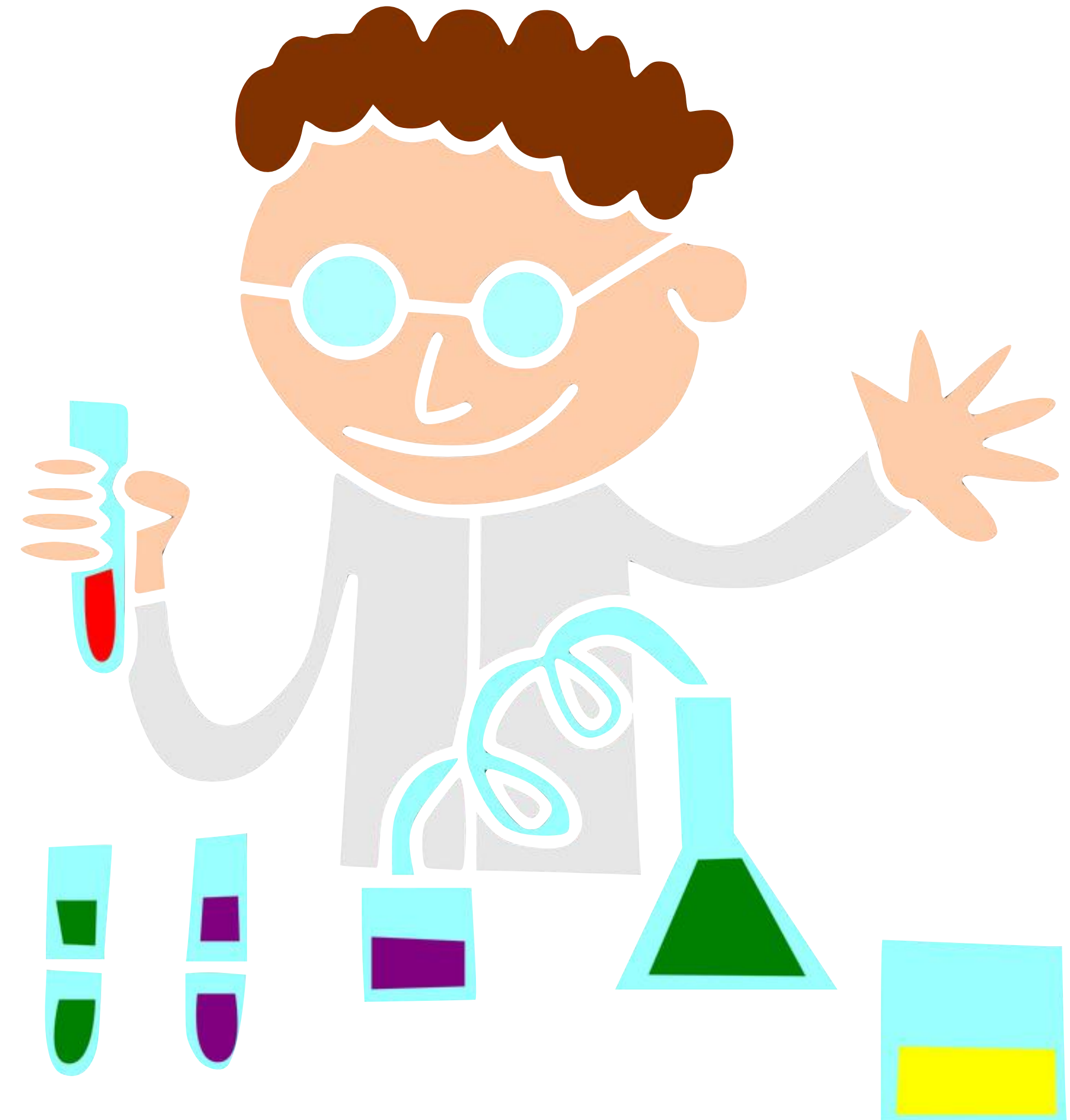
<https://redflagsecurity.net/2019/03/10/decrypting-tls-wireshark/>



Demo & Exercise

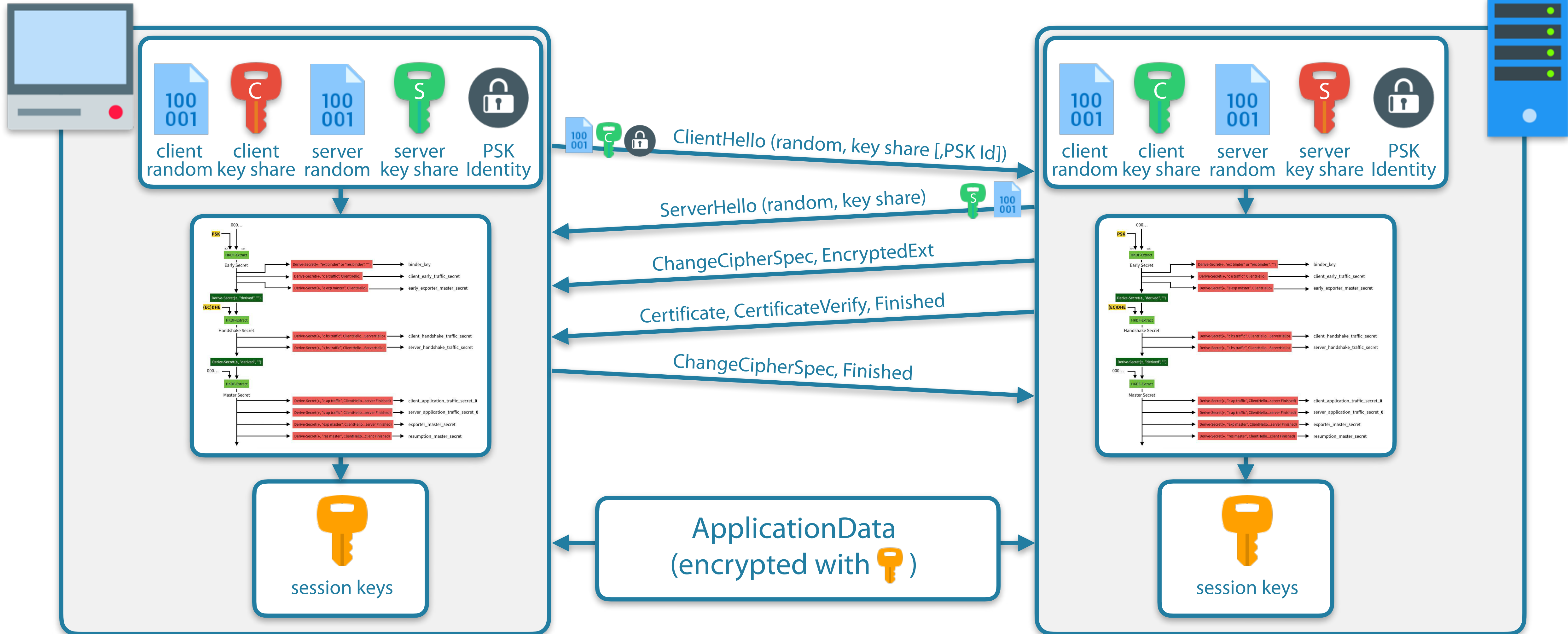


- Demo
 - Using SSLKEYLOGFILE
- Exercise
 - Use SSLKEYLOGFILE with Chrome or Firefox
 - Start Wireshark
 - Point to key file
 - Go to https:// site
 - Filter on "tls and (http || http2)"



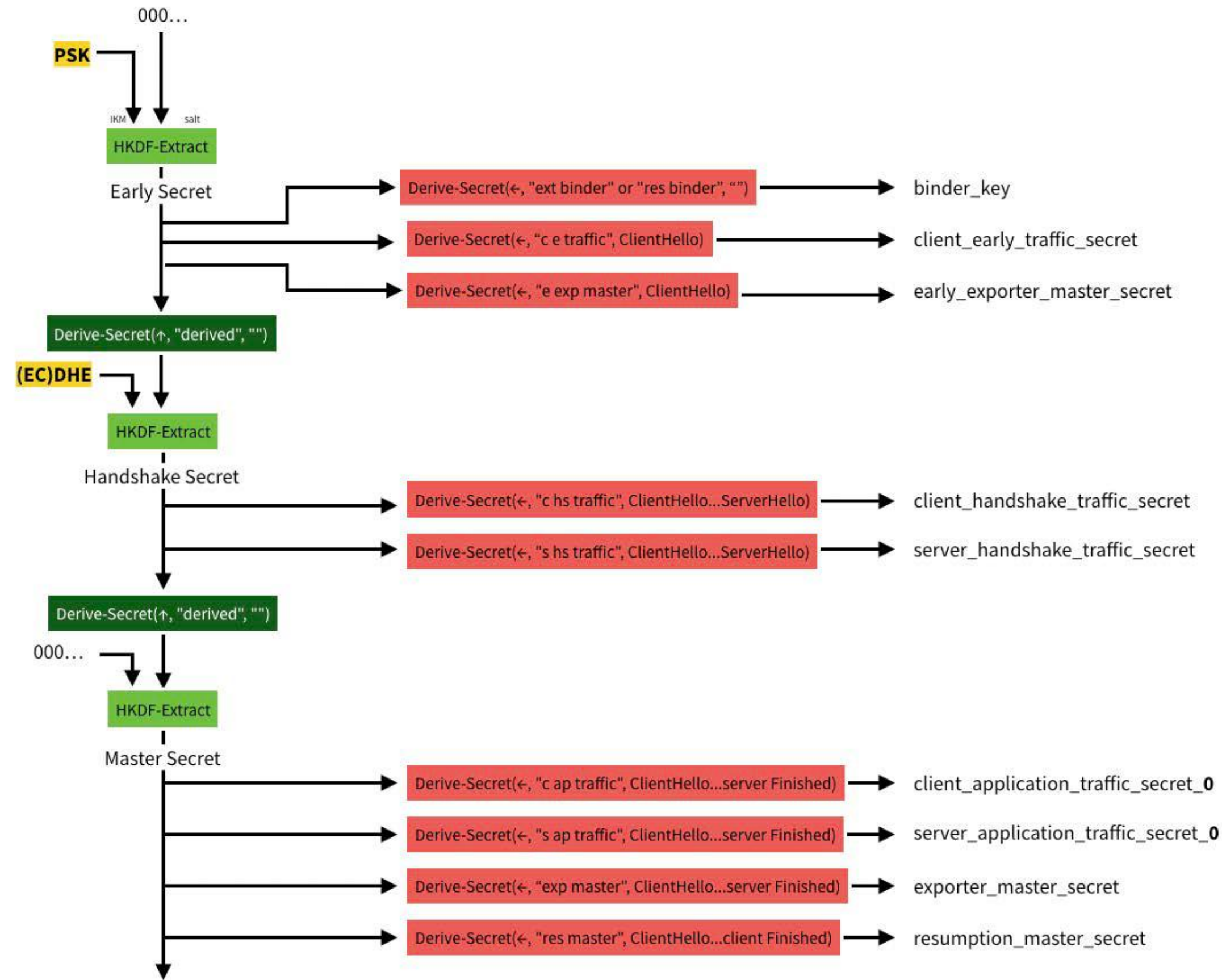


Key generation in TLSv1.3





Deriving TLSv1.3 keying material



<https://www.davidwong.fr/tls13/>

```

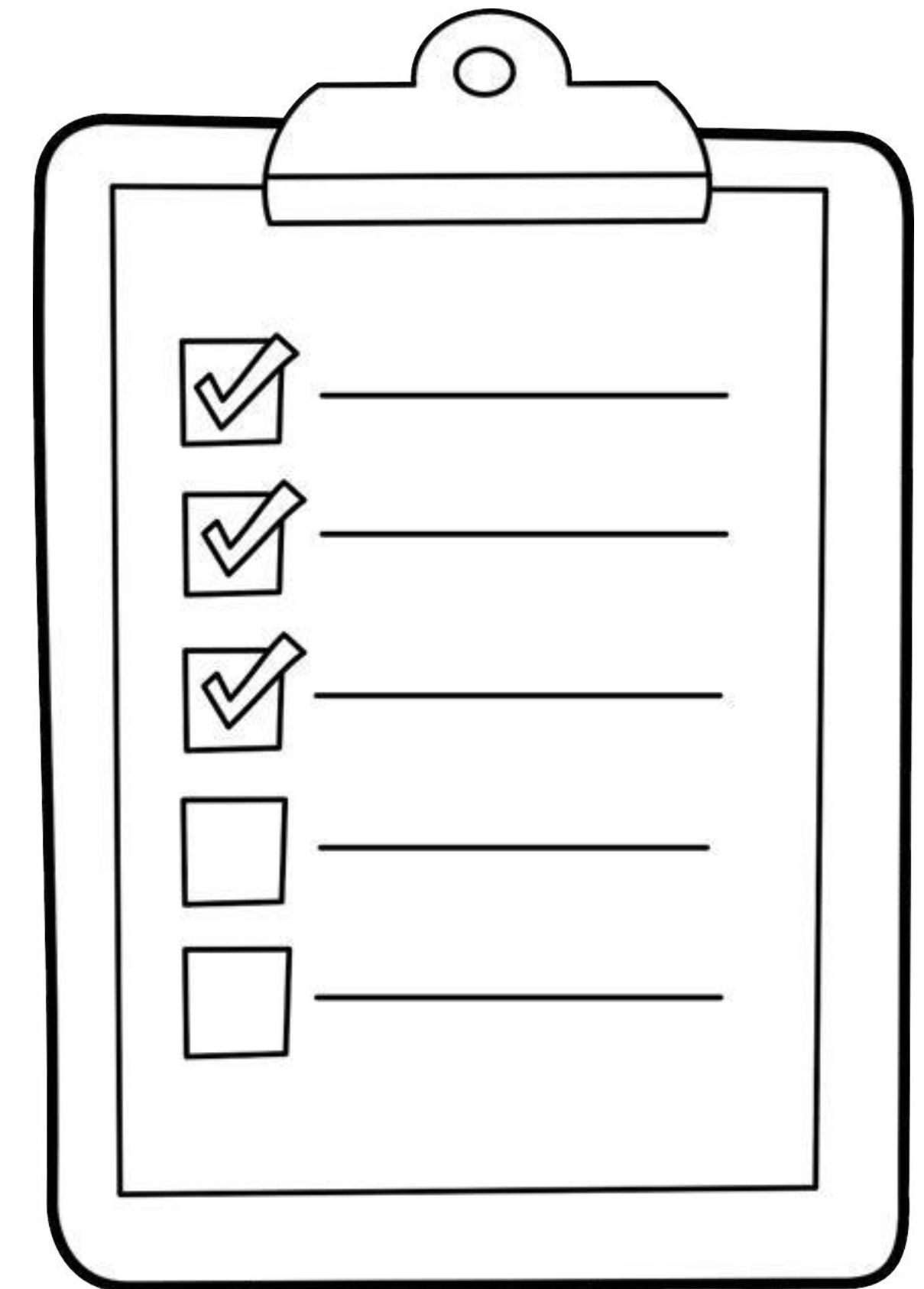
sake@MacSake:~$$ grep 042812bd0d8df6c71c5405789efdb77d06a076b94127492a4554e85e2f623f87 dv*keys
CLIENT_HANDSHAKE_TRAFFIC_SECRET 042812bd0d8df6c71c5405789efdb77d06a076b94127492a4554e85e2f623f87 984238a19484ce65e8395a23faad5117649e40533187ed7b5e883569412115cf3b9c59b156a164c3f86c31b8adea9af7
SERVER_HANDSHAKE_TRAFFIC_SECRET 042812bd0d8df6c71c5405789efdb77d06a076b94127492a4554e85e2f623f87 089af9caf6b42ec56793cfc91b95f833692e3807ce1f2a218bc7aaacfb0b7e4bf3ee0bbbbd8cc5eac2083626bf15b8e
CLIENT_TRAFFIC_SECRET_0 042812bd0d8df6c71c5405789efdb77d06a076b94127492a4554e85e2f623f87 530f5a1811f170bb957bc6a092c77ced8dd459697cfb5d8ec7f0a47ba63747386f394a8a0b824c91e656686876ee32a5
SERVER_TRAFFIC_SECRET_0 042812bd0d8df6c71c5405789efdb77d06a076b94127492a4554e85e2f623f87 9e426f3a086837024198448b077b1da63215b184b6b27dead509463c479a918fb84a98f673627574a3f554037a5e9f8
EXPORTER_SECRET 042812bd0d8df6c71c5405789efdb77d06a076b94127492a4554e85e2f623f87 6add3613f5fcc8679aa5f9313a683f03acfb02141d3897eb34ef16ae4a0f24872a9ae97fe7cea5fb1ea0c4a2fde41062
sake@MacSake:~$
  
```



Agenda



- TLS fundamentals
 - The need for TLS (and what happens to SSL?)
 - Cryptology 101 & PKI
 - Troubleshoot the TLS handshake (SSLv3 - TLSv1.2)
 - Troubleshoot the TLS handshake (TLSv1.3)
- Analysing Application Data
 - Without decryption
 - With decryption using the server's private key
 - With decryption using TLS session keys
- **Creating configuration profiles for TLS analysis**
- SSL/TLS vulnerabilities & privacy concerns
- Summary and Q&A



<https://www.needpix.com/photo/download/94588/checklist-action-check-list-verification-tick-approved-okay-check-mark>



What is a configuration profile?



- all Wireshark customizations in one package
- Easily switch between profiles
- Optimize workflow
- Share profiles with others

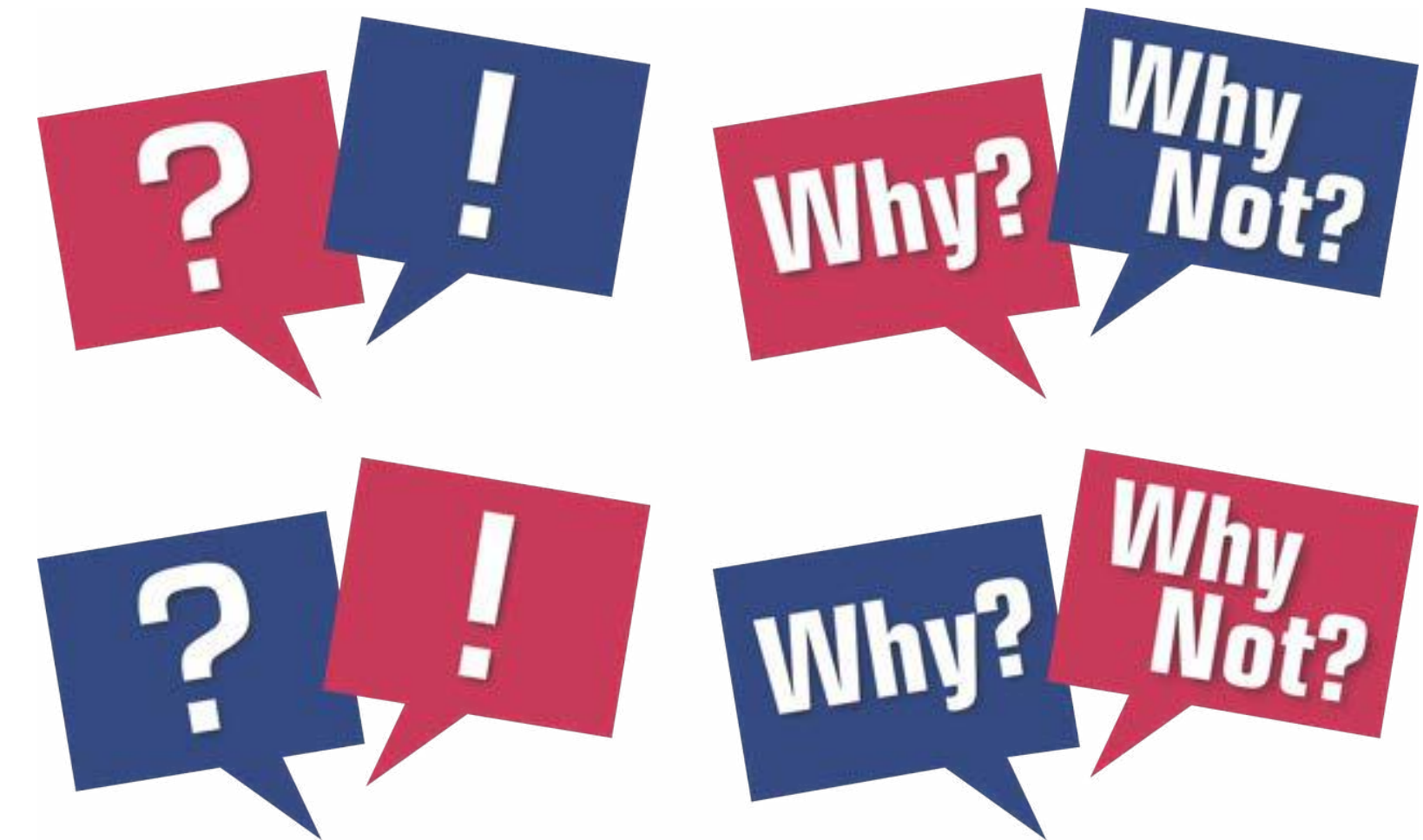




Why use configuration profiles?



- Differentiate your analysis workflow
 - Per protocol?
 - Per application?
 - Per customer?
- Increase performance
 - disable protocols
 - disable reassembly etc.
- Adapt to different setups (e.g. screen size)





Common Configuration Profile Elements



- GUI

- Layout
- Columns
- Colors

- Workflow

- filter buttons
- name resolution

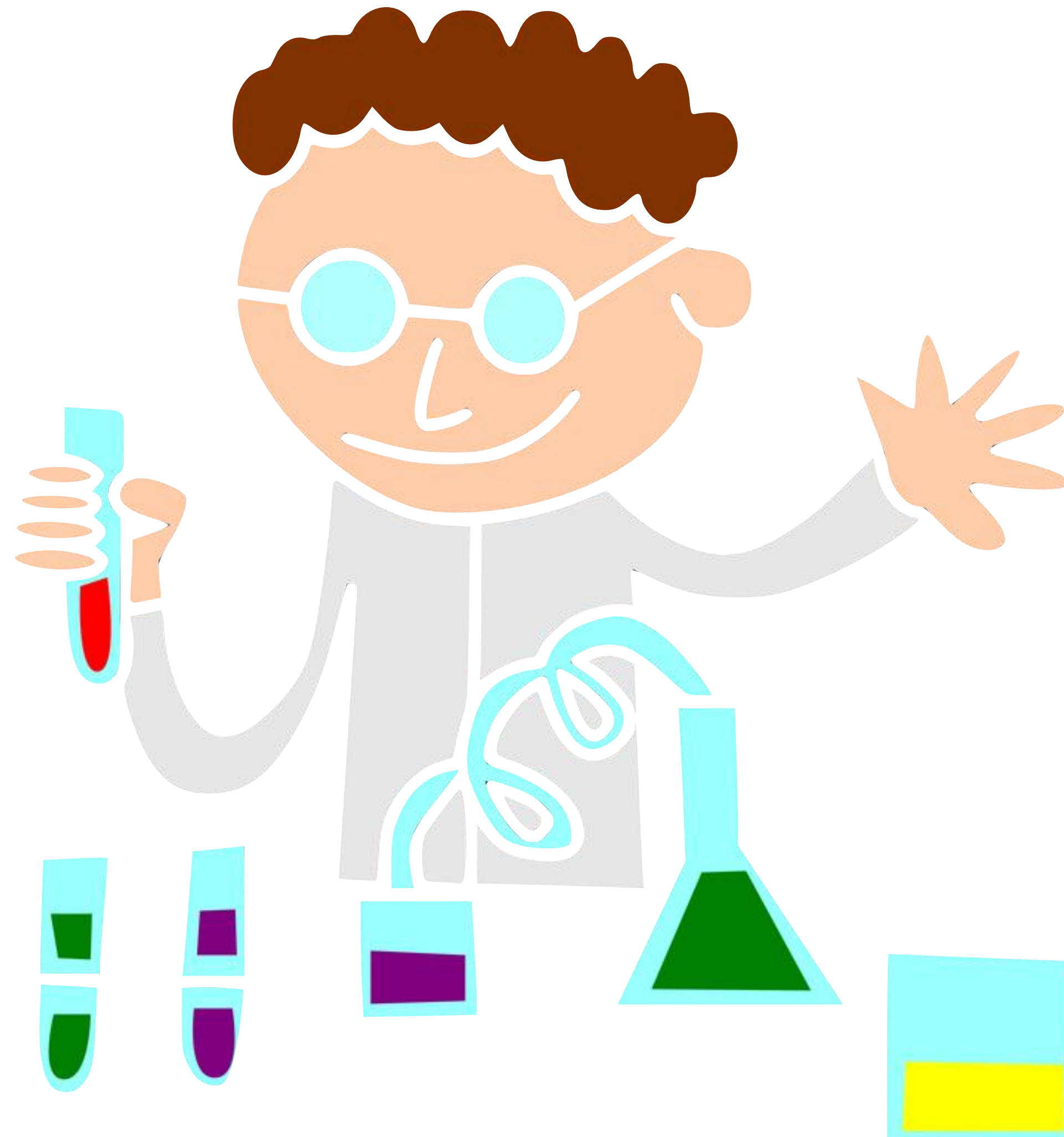
- Dissection

- Enabled protocols
- Protocol preferences





Demo





Agenda



- TLS fundamentals
 - The need for TLS (and what happened to SSL?)
 - Cryptology 101 & PKI
 - Troubleshoot the TLS handshake (SSLv3 - TLSv1.2)
 - Troubleshoot the TLS handshake (TLSv1.3)
- Analysing Application Data
 - Without decryption
 - With decryption using the server's private key
 - With decryption using TLS session keys
- Creating configuration profiles for TLS analysis
- **SSL/TLS vulnerabilities & privacy concerns**
- Summary and Q&A





Vulnerabilities



- SSLv2 : multiple vulnerabilities
- SSLv3 : POODLE
- Weak hashing :
 - MD5
 - SHA-1
- Weak keylengths :
 - Minimum safe keylength for RSA : 2048
 - Minimum safe keylength for ECDSA : 233
- Beast, Crime, Freak
- OpenSSL : Heartbleed
- Renegotiation
- RC4
- And many more...



<https://www.digicert.com/cert-inspector-vulnerabilities.htm>



TLS best practices



- Certificate management & monitoring
 - Use a reliable CA that matches your purpose
 - Use Certificate Authority Authorization (CAA)
 - Use Signed Certificate Timestamps (SCT)
 - Protect your private keys!
- TLS configuration
 - Use 2048 bit RSA and/or 256 bit ECDSA
 - Use SHA256 signatures
 - Use TLSv1.2 and/or TLSv1.3
 - Restrict ciphers to known safe ones
 - Use PFS (by using ECDHE)
 - Use OCSP stapling



<https://www.flickr.com/photos/barrydahl/6675297699>



Certificate Authority Authorization



- RFC 8659 (obsoletes 6844)
- Add CAA records to DNS to list accepted CA's
- Mandatory for CA's to check (sep 2017)
 - but allowed to sign certificates if no CAA record present
- Use of DNSSEC recommended

```
sake@MacSake:~$ nslookup -q=CAA belastingdienst.nl
Server:          1.1.1.1
Address:         1.1.1.1#53

Non-authoritative answer:
belastingdienst.nl      rdata_257 = 0 iodef "mailto:certificatenloket@belastingdienst.nl"
belastingdienst.nl      rdata_257 = 0 issue "apple.com"
belastingdienst.nl      rdata_257 = 0 issue "globalsign.com"
belastingdienst.nl      rdata_257 = 0 issue "kpn.com"
belastingdienst.nl      rdata_257 = 0 issue "pkioverheid.nl"
belastingdienst.nl      rdata_257 = 0 issue "quovadisglobal.com"

Authoritative answers can be found from:

sake@MacSake:~$
```



Signed Certificate Timestamps



- RFC 6962
- Log all certificates to an append-only log
- Monitor / Audit the logs for irregularities

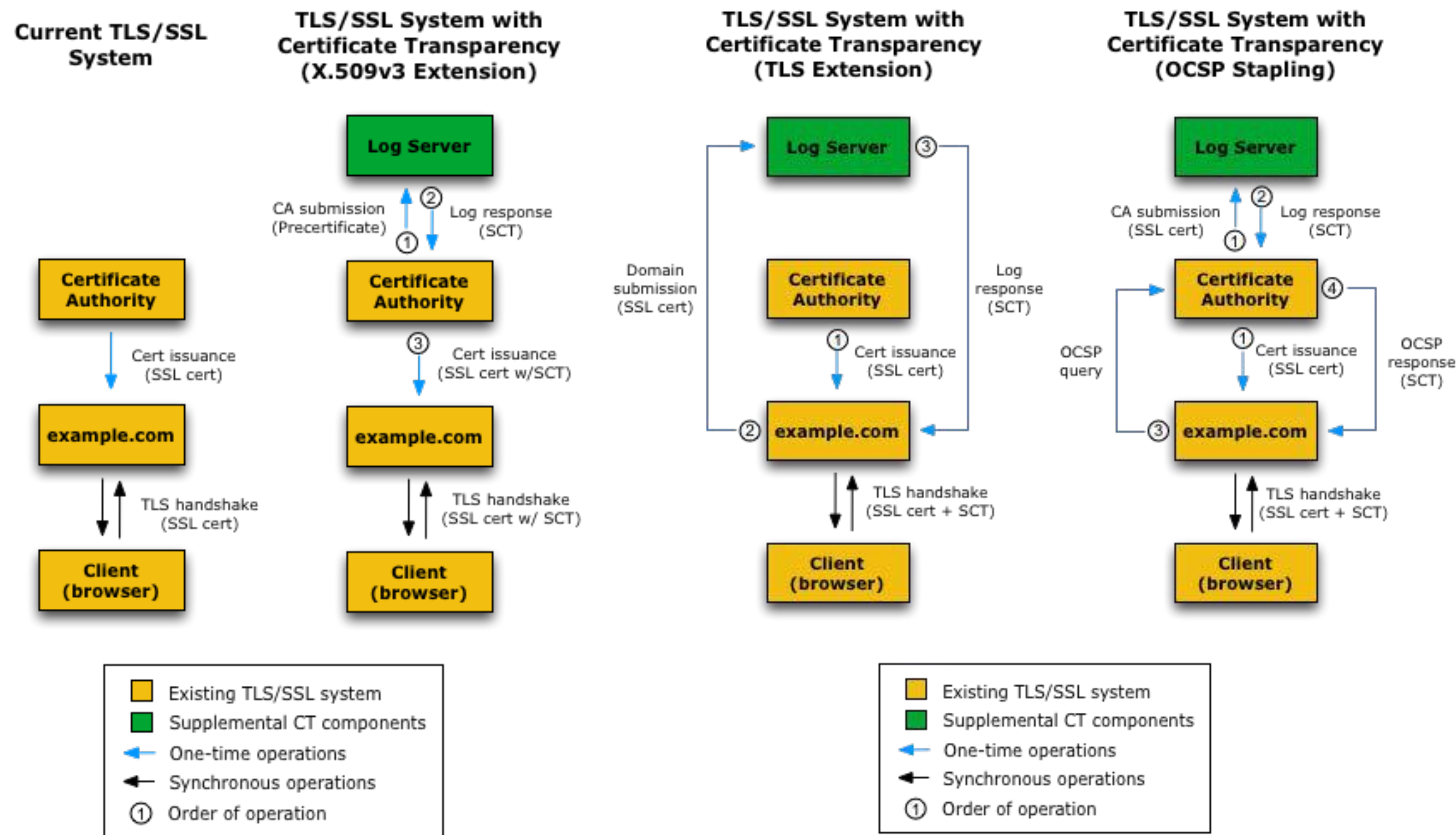


Figure 1

Figure 2

```

Embedded SCTs
Log ID 7D:3E:F2:F8:8F:FF:88:55:68:24:C2:C0:CA:9E:52:89:79:2B:C5:0E:78:09:7F:2E:6A:97:68:99:7E:22:F0:D7
Name Google "Xenon2021"
Signature Algorithm SHA-256 ECDSA
Version 1
Timestamp 9/2/2020, 8:28:03 AM (Central European Summer Time)
Log ID 44:94:65:2E:B0:EE:CE:AF:C4:40:07:D8:A8:FE:28:C0:DA:E6:82:BE:D8:CB:31:B5:3F:D3:33:96:B5:B6:81:A8
Name Cloudflare "Nimbus2021"
Signature Algorithm SHA-256 ECDSA
Version 1
Timestamp 9/2/2020, 8:28:03 AM (Central European Summer Time)

```



Test and monitor



Qualys. SSL Labs

Home Projects Qualys Free Trial Contact

You are here: [Home](#) > [Projects](#) > [SSL Server Test](#) > [www.belastingdienst.nl](#)

SSL Report: [www.belastingdienst.nl](#) (85.159.98.33)

Assessed on: Wed, 14 Oct 2020 09:35:41 UTC | [Hide](#) | [Clear cache](#) [Scan Another »](#)

Summary

Overall Rating: **A+**

Category	Score
Certificate	100
Protocol Support	100
Key Exchange	90
Cipher Strength	90

Visit our [documentation page](#) for more information, configuration guides, and books. Known issues are documented [here](#).

- This server's certificate is not trusted by Java trust store (see below for details).
- HTTP Strict Transport Security (HSTS) with long duration deployed on this server. [MORE INFO »](#)
- DNS Certification Authority Authorization (CAA) Policy found for this domain. [MORE INFO »](#)

<https://www.ssllabs.com/ssltest/analyze.html?d=www.belastingdienst.nl>

Qualys. SSL Labs

Home Projects Qualys Free Trial Contact

You are here: [Home](#) > [Projects](#) > [SSL Server Test](#) > [www.gmf.fr](#)

SSL Report: [www.gmf.fr](#) (81.80.208.15)

Assessed on: Wed, 14 Oct 2020 09:36:01 UTC | [Hide](#) | [Clear cache](#) [Scan Another »](#)

Summary

Overall Rating: **F**

Category	Score
Certificate	100
Protocol Support	100
Key Exchange	70
Cipher Strength	90

Visit our [documentation page](#) for more information, configuration guides, and books. Known issues are documented [here](#).

- This server is vulnerable to the [Return Of Bleichenbacher's Oracle Threat \(ROBOT\)](#) vulnerability. Grade set to F. [MORE INFO »](#)
- This server supports weak Diffie-Hellman (DH) key exchange parameters. Grade capped to B. [MORE INFO »](#)
- This server does not support Forward Secrecy with the reference browsers. Grade capped to B. [MORE INFO »](#)
- This server supports TLS 1.0 and TLS 1.1. Grade capped to B. [MORE INFO »](#)

<https://www.ssllabs.com/ssltest/analyze.html?d=www.gmf.fr>



Privacy: Encrypted SNI



- Not a standard yet
- Needs TLS 1.3
 - needs an encrypted Certificate handshake message
- Public key provided by DNS
 - Needs DoH or DoT, otherwise name is visible in DNS
- Watch out for IP addresses

- Protection for client, but now impossible to enforce rules based on server name
 - Chinese FW blocks all TLS v1.3 traffic that uses eSNI



<https://blog.cloudflare.com/encrypted-sni/>



Demo & Exercise



- Demo

- <https://www.ssllabs.com/ssltest/analyze.html?d=www.gmf.fr>

- Exercise

- Check a site you manage (or use a lot) at:

- ▶ <https://www.ssllabs.com/ssltest/>

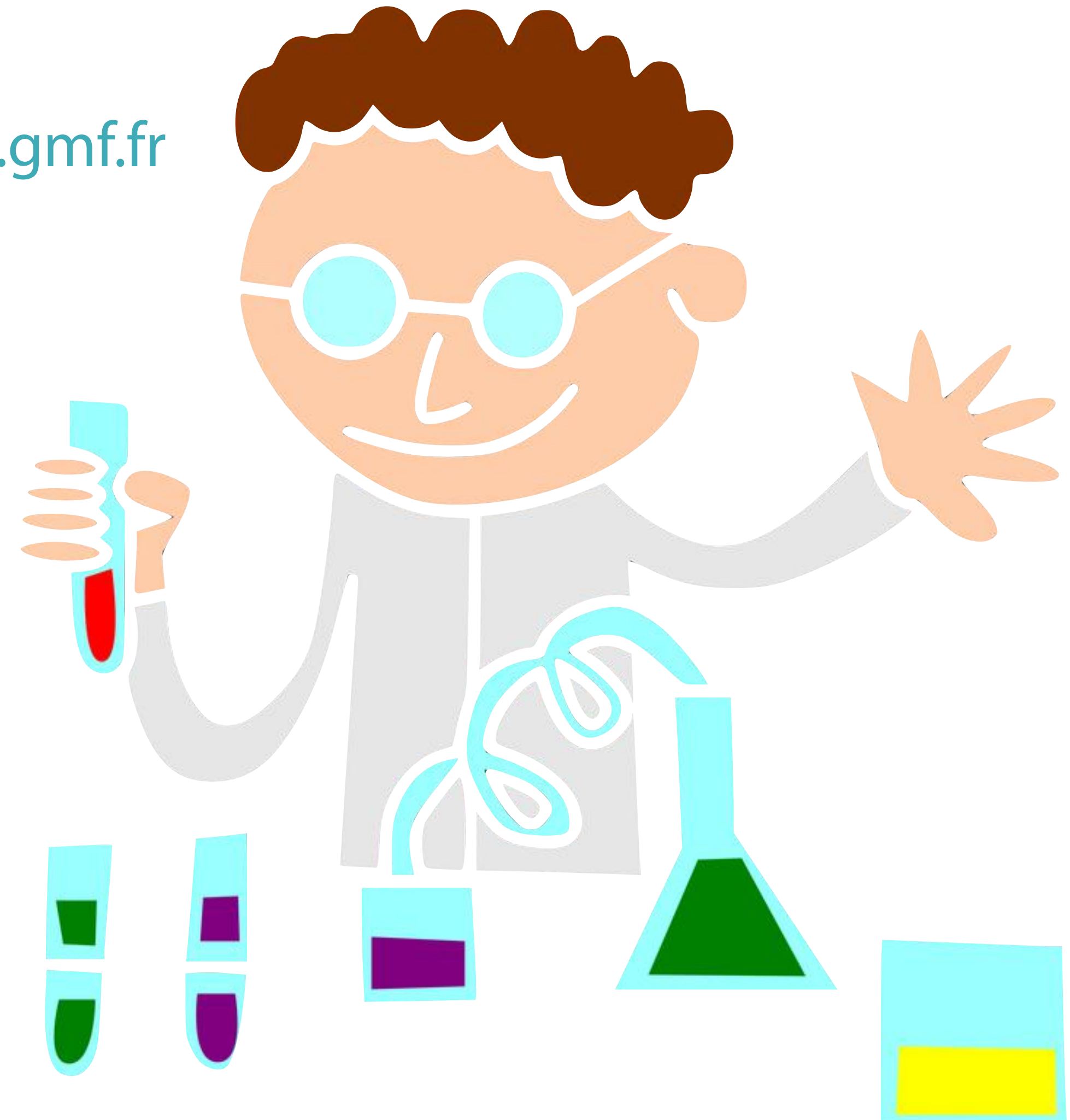
- Check whether it supports CAA with:

- ▶ `nslookup -q=CAA example.com`

- Check the certificate for SCT entries

- Check whether it supports ESNI with:

- ▶ `nslookup -q=TXT_esni.www.example.com`





Time for a poll

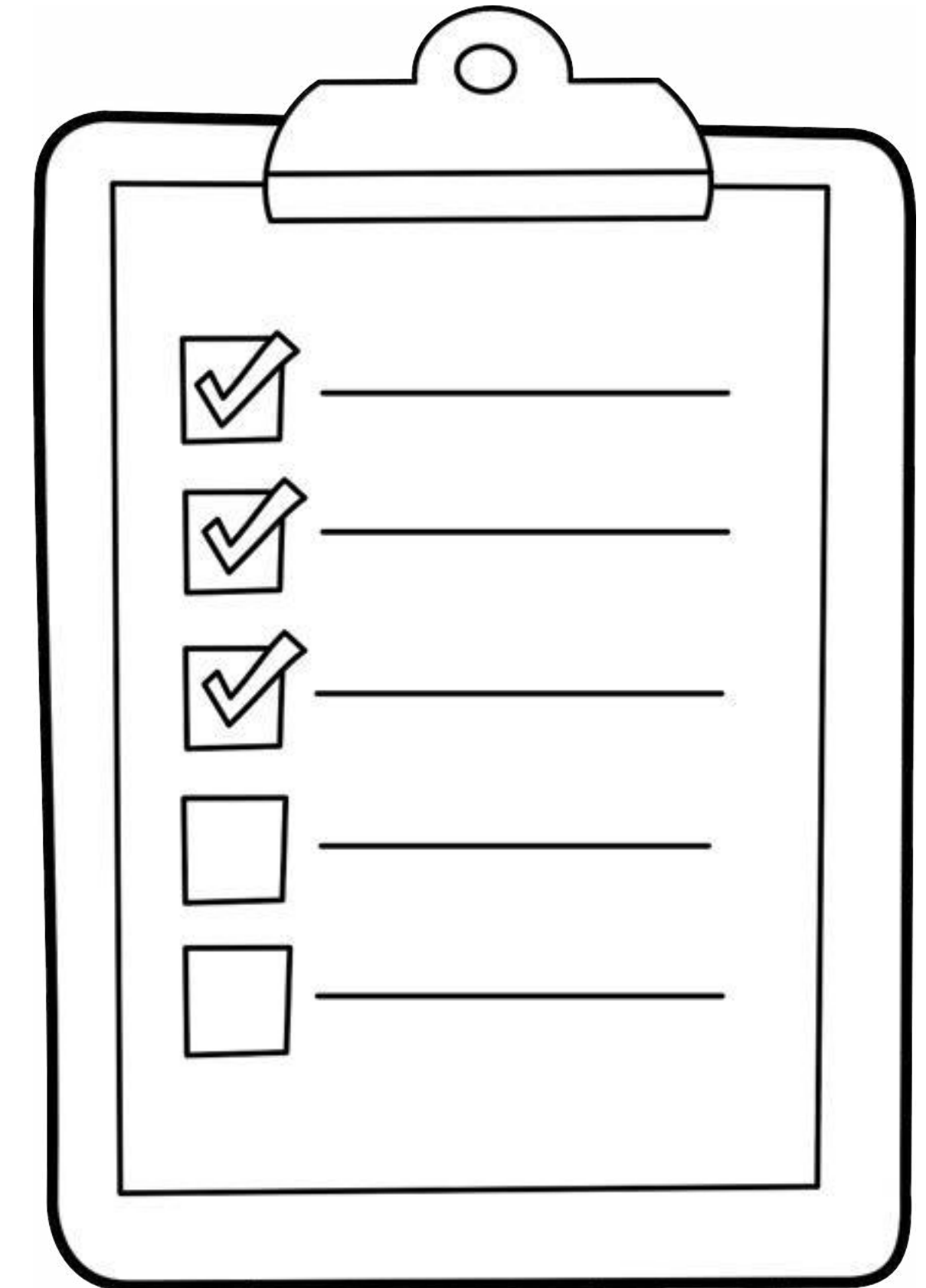




Agenda



- TLS fundamentals
 - The need for TLS (and what happens to SSL?)
 - Cryptology 101 & PKI
 - Troubleshoot the TLS handshake (SSLv3 - TLSv1.2)
 - Troubleshoot the TLS handshake (TLSv1.3)
- Analysing Application Data
 - Without decryption
 - With decryption using the server's private key
 - With decryption using TLS session keys
- Creating configuration profiles for TLS analysis
- SSL/TLS vulnerabilities & privacy concerns
- **Summary and Q&A**



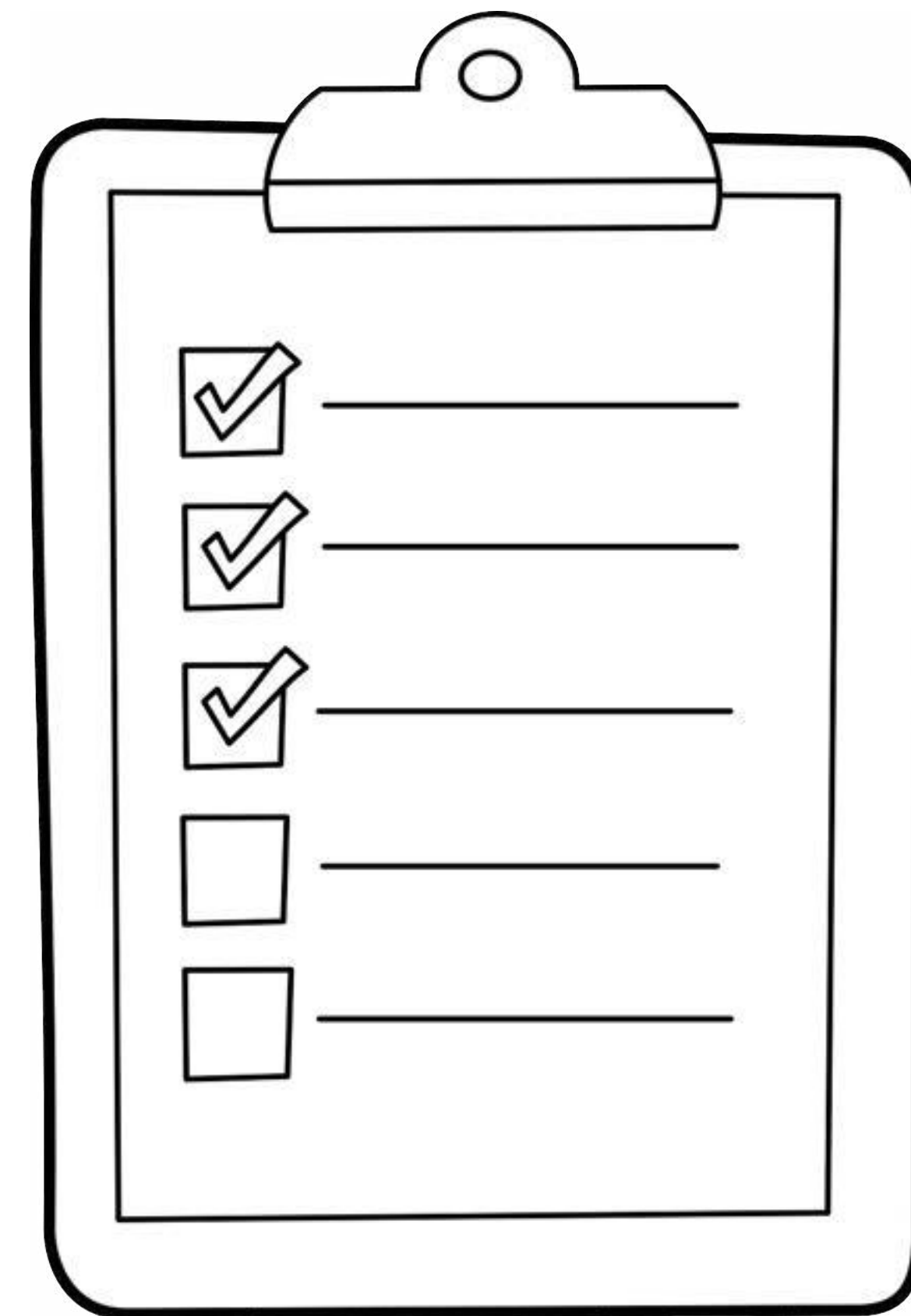
<https://www.needpix.com/photo/download/94588/checklist-action-check-list-verification-tick-approved-okay-check-mark>



Summary



- Get familiar with the basics of cryptography
- Analysing the TLS handshake
 - Pay attention to:
 - Root certificates in Trust Stores
 - Supported versions & ciphersuites
 - TLSv1.3 is a different beast (compared to SSLv3-TLSv1.2)
- Analysing application traffic
 - Without decryption it's limited and hard to do!
 - Decryption with private keys is possible, but fading away
 - Only decryption with TLS session keys will be possible in the (near?) future
 - Access to TLS session keys is new challenge



<https://www.needpix.com/photo/download/94588/checklist-action-check-list-verification-tick-approved-okay-check-mark>



Please take a minute to
fill in the survey at:

<https://SYN-b.it/survey>

... or scan the following QR-code to
go directly to the survey:





Links



- TLS illustrated:
<https://tls.ulfheim.net/> and <https://tls13.ulfheim.net/>
- <https://www.ssllabs.com/ssl-pulse/>
- Test your TLS infrastructure (Client, Server, best practices, how are others doing, etc):
<https://www.ssllabs.com/>
- Historic facts on SSL and TLS
<https://www.feistyduck.com/ssl-tls-and-pki-history/>
- TLSkex:
<https://www.sciencedirect.com/science/article/pii/S1742287616300081>
- TeLeScope:
<https://conference.hitb.org/hitbsecconf2016ams/wp-content/uploads/2015/11/D1T1-Radu-Caragea-Peering-into-the-Depths-of-TLS-Traffic-in-Real-Time.pdf>
- Using iRule on F5 to extract keys:
<https://support.f5.com/csp/article/K12783074>
- Sharkfest presentations on SSL & TLS
<https://sharkfesteurope.wireshark.org/assets/presentations16eu/07.pdf> & <https://youtu.be/oDaDY9QCnXk>
<https://lekensteyn.nl/files/wireshark-ssl-tls-decryption-secrets-sharkfest18eu.pdf>
<https://sharkfestus.wireshark.org/assets/presentations19/02-26.pptx> & https://www.youtube.com/watch?v=Fp_7g5as1VY
<https://lekensteyn.nl/files/wireshark-tls-debugging-sharkfest19us.pdf> & <https://www.youtube.com/watch?v=Ha4SLHceF6w>
- Some legal reflections [dutch]:
<https://www.security.nl/posting/516591/Juridische+vraag%3A+Mag+mijn+werkgever+ssl-verkeer+decrypten+en+inspecteren+om+datalekken+te+voorkomen%3F>
- From SSL to TLS, why the name change:
<http://tim.dierks.org/2014/05/security-standards-and-name-changes-in.html>

Credits:
Most icons by icons8
<https://icons8.com>



FIN/ACK/FIN/ACK



*If you still have any questions:
sake.blok@SYN-bit.nl*



SYN-bit
deep traffic analysis