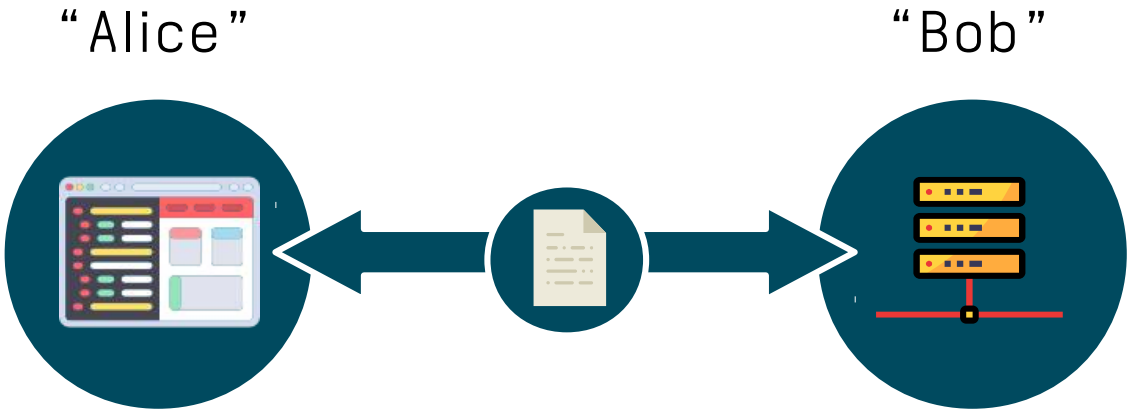
SSL and Certificates

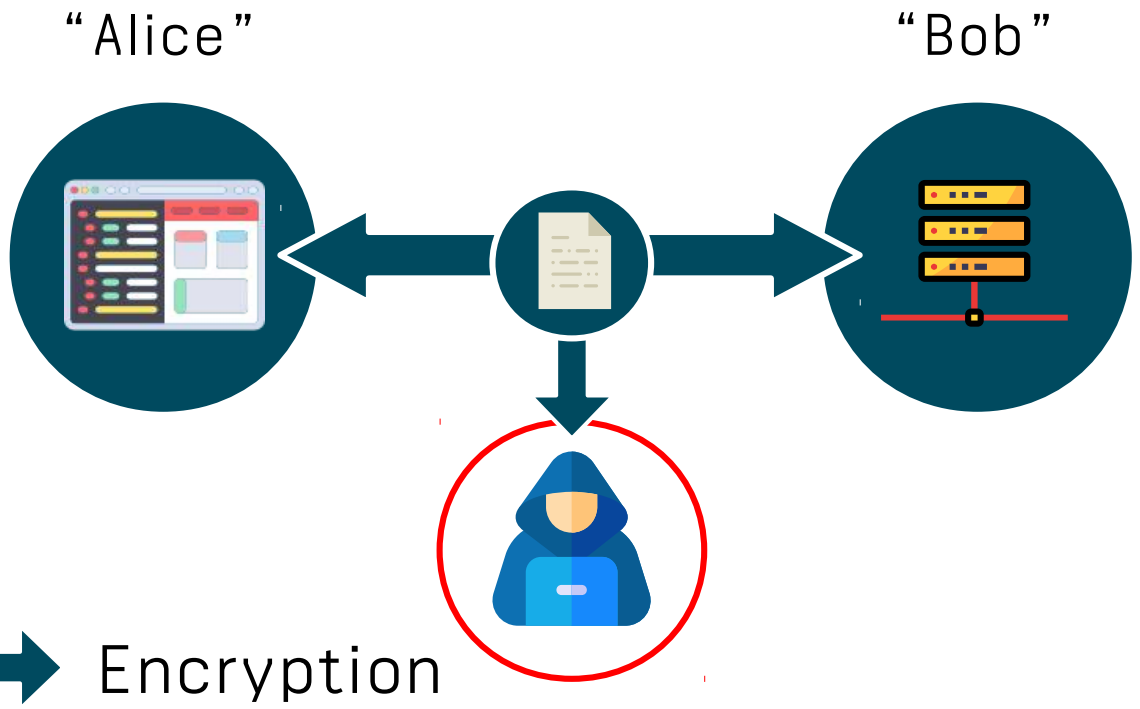
04.05.2018, Andi Umbreit
Check_MK Conference #4

CONFERENCE
MUNICH 2018/5/2-4
#4

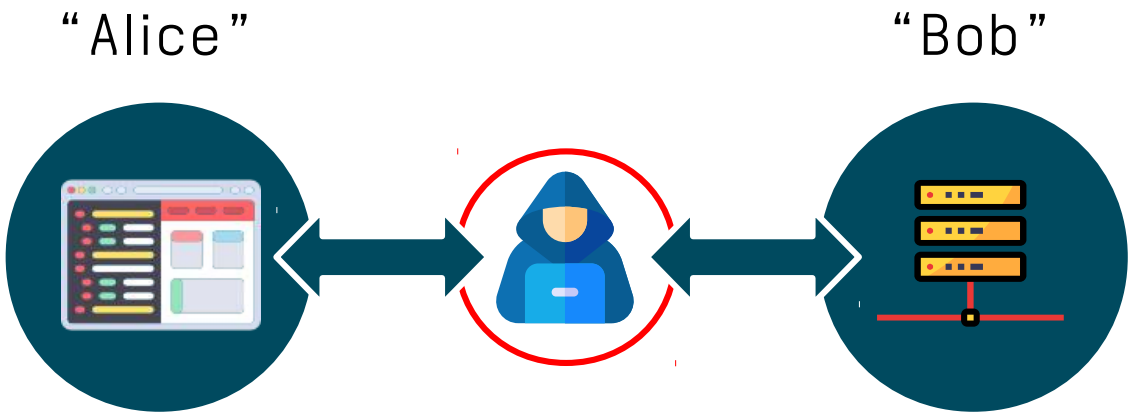
The Scenario



The Scenario



The Scenario



- Encryption
- Authentication

- SSL and Certificates -

SSL (or TLS?)

X.509 and
the certificate chain

Application in Check_MK

Hands-on OpenSSL

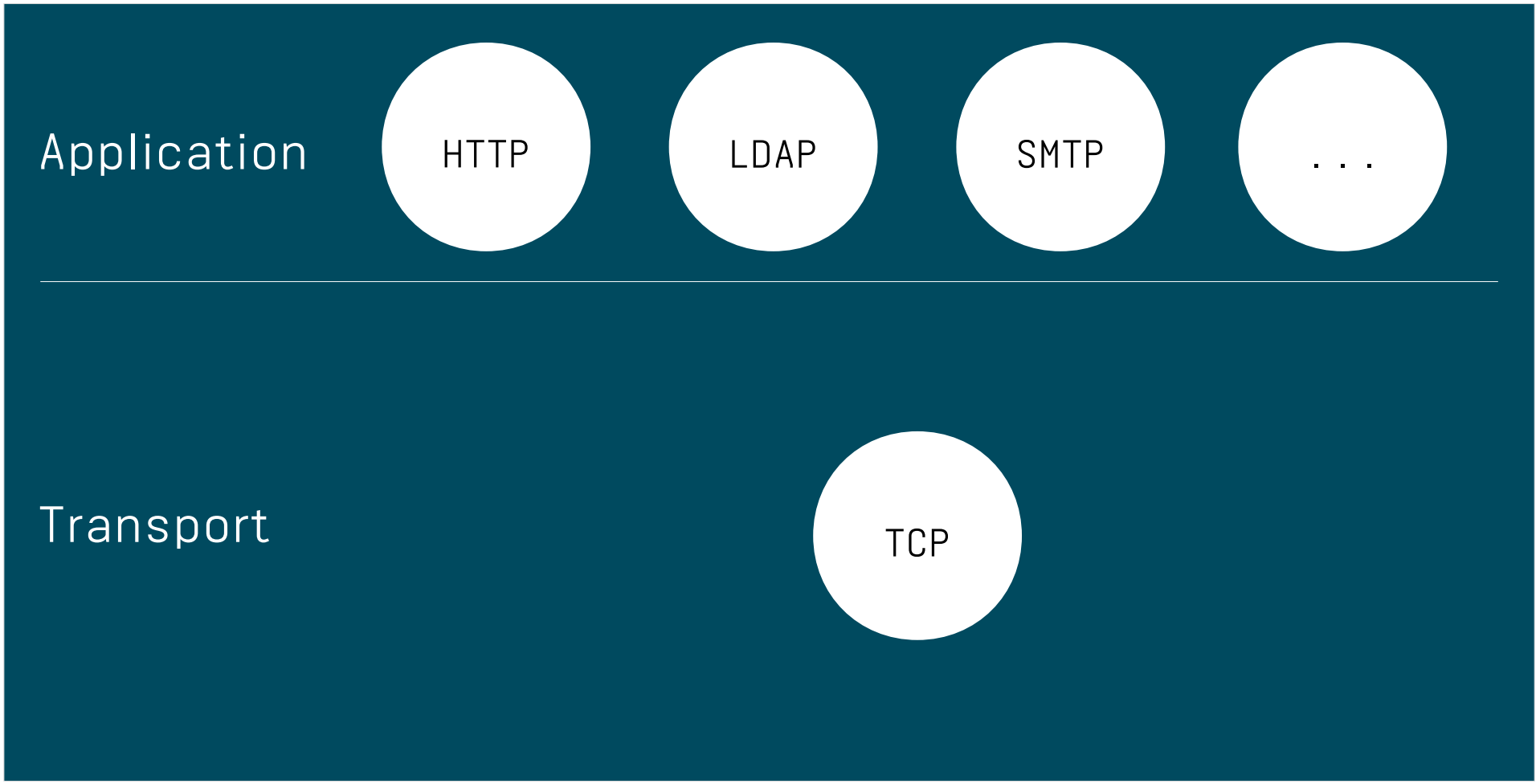
Terminology

SSL — Secure Sockets Layer

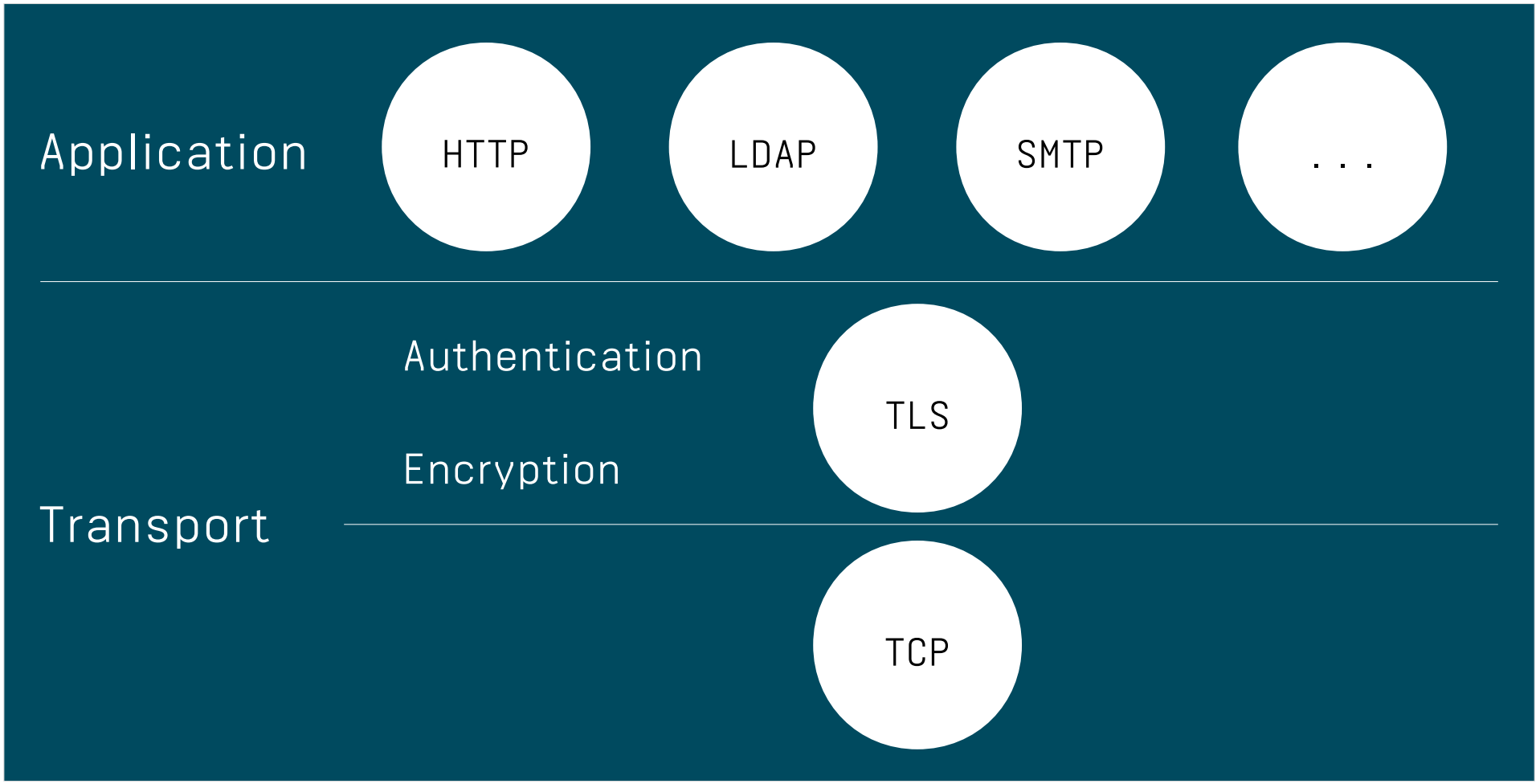
TLS — Transport Layer Security

HTTPS — HTTP with TLS

TLS Protocol

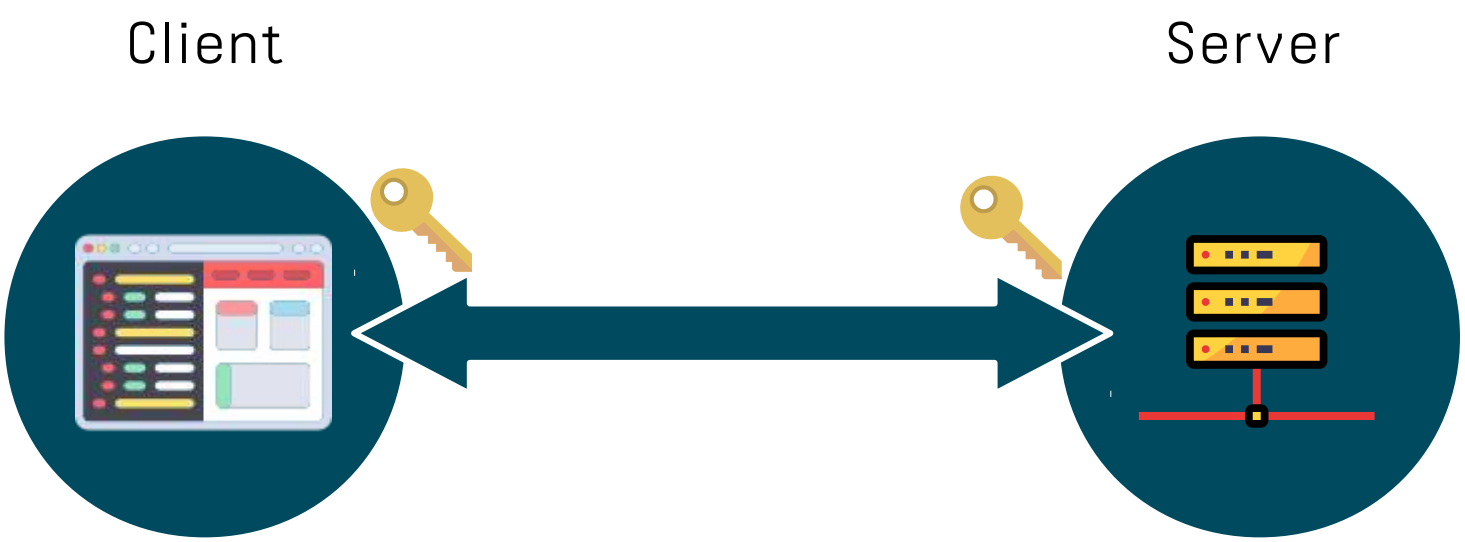


TLS Protocol



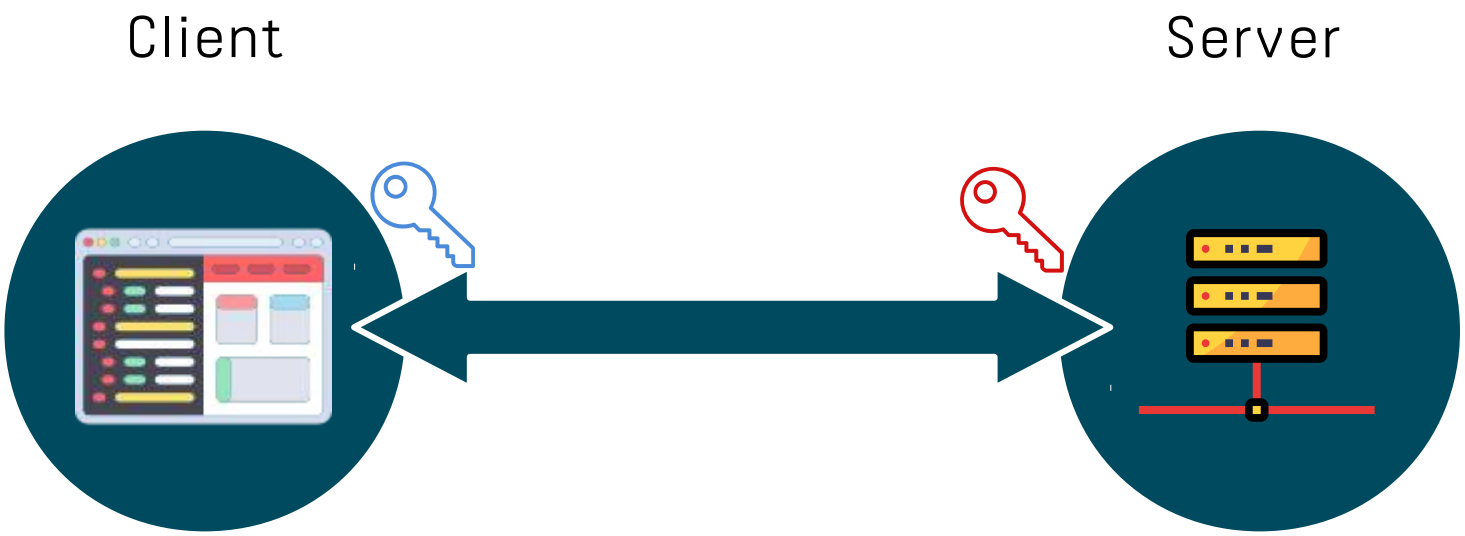
Interlude - Encryption techniques

Symmetric encryption



Interlude - Encryption techniques

Asymmetric encryption



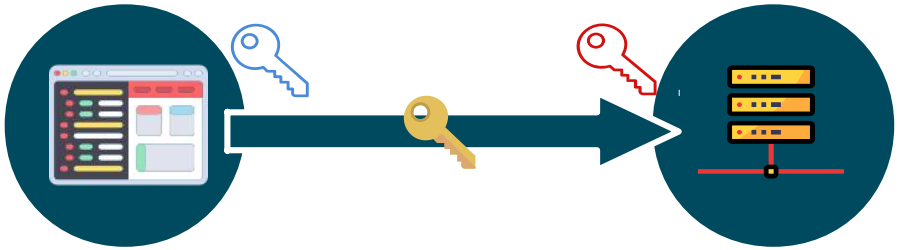
Public Key

Private Key

Interlude – Encryption techniques

Combined

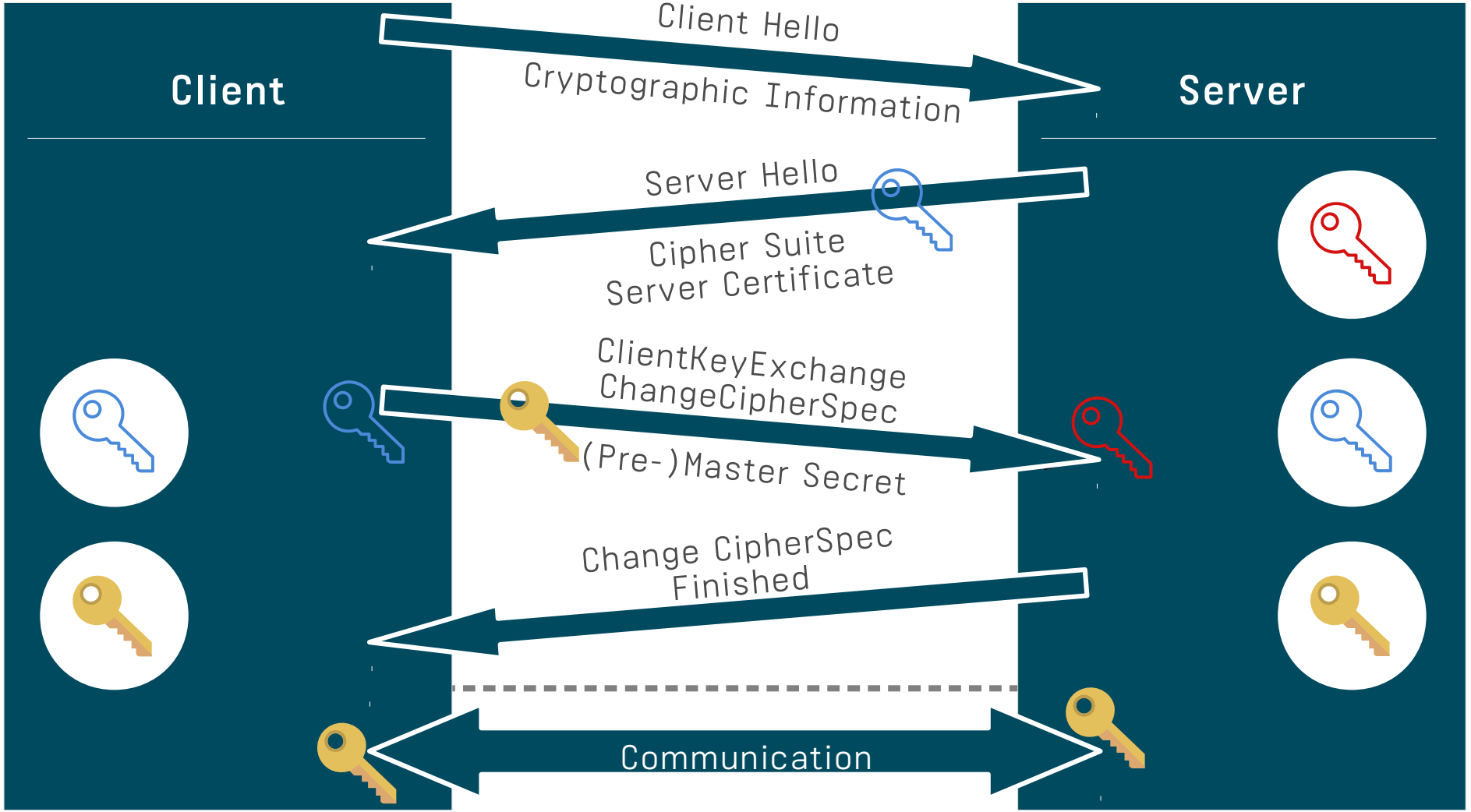
Authentication



Key Exchange

Encrypted
Communication

TLS Handshake



- SSL and Certificates -

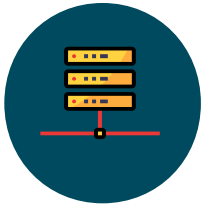
SSL (or TLS?)

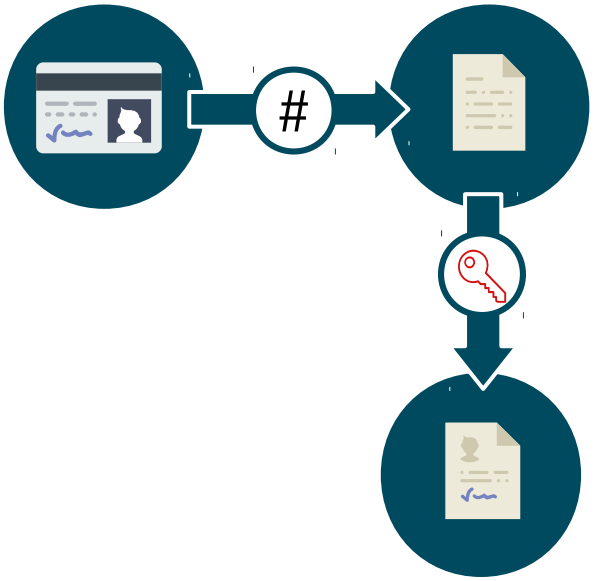
X.509 and
the certificate chain

Application in Check_MK

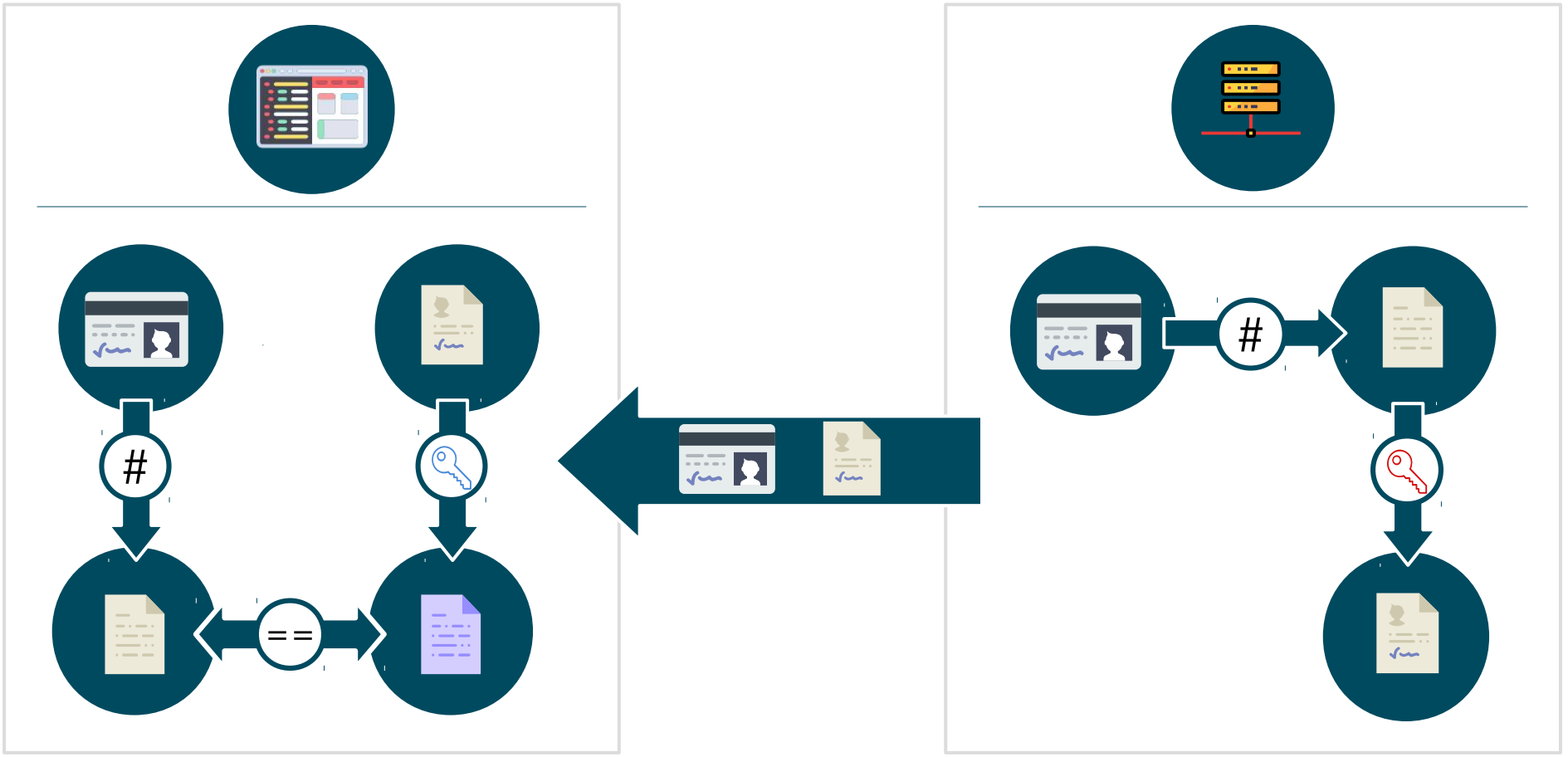
Hands-on OpenSSL

Interlude - Digital Signatures





Interlude - Digital Signatures



Terminology

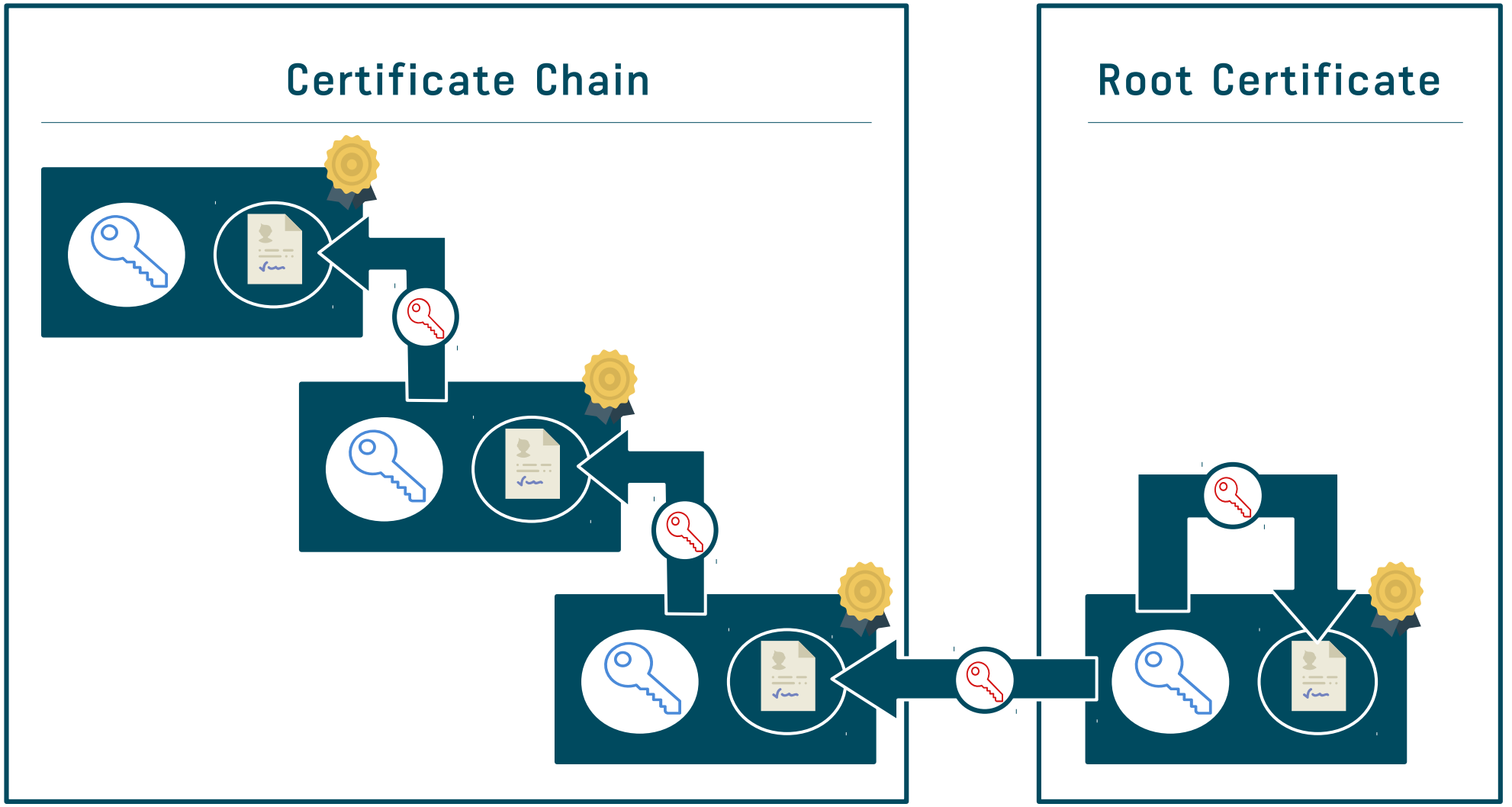
X.509 — ITU-T* standard for a public key intrastructure (pki)

CA — Certification Authority

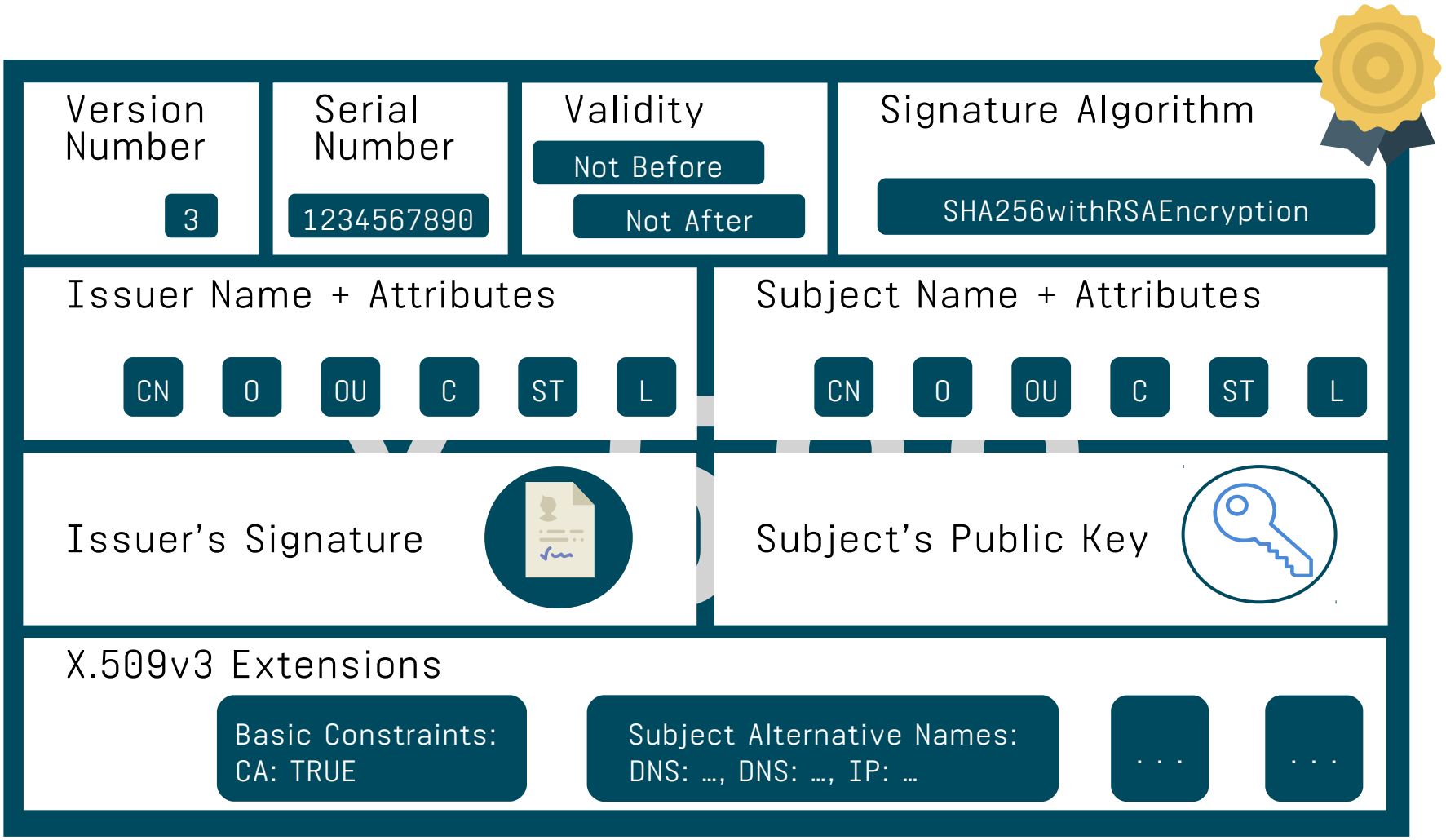
CSR — Certificate Signing Request

* ITU: International Telecommunication Union
ITU-T: ITU Standardization Sector

Gaining Trust



What's in a X.509 certificate?

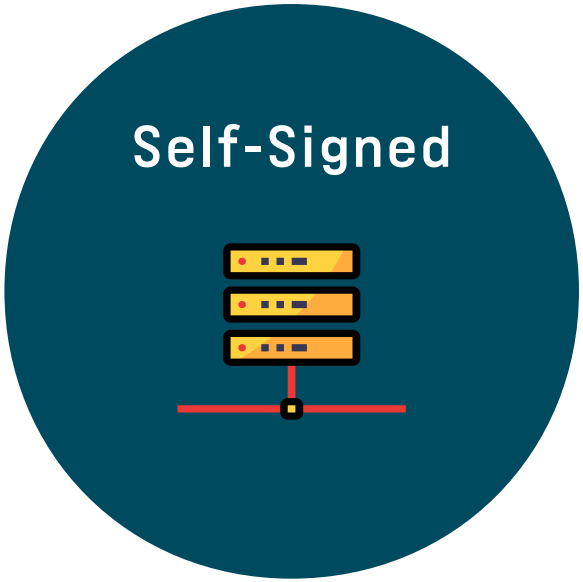


Terminology

CRL — Certificate Revocation List

OSCP — Online Certificate Status Protocol

What should I use?



- SSL and Certificates -

SSL (or TLS?)

X.509 and
the certificate chain

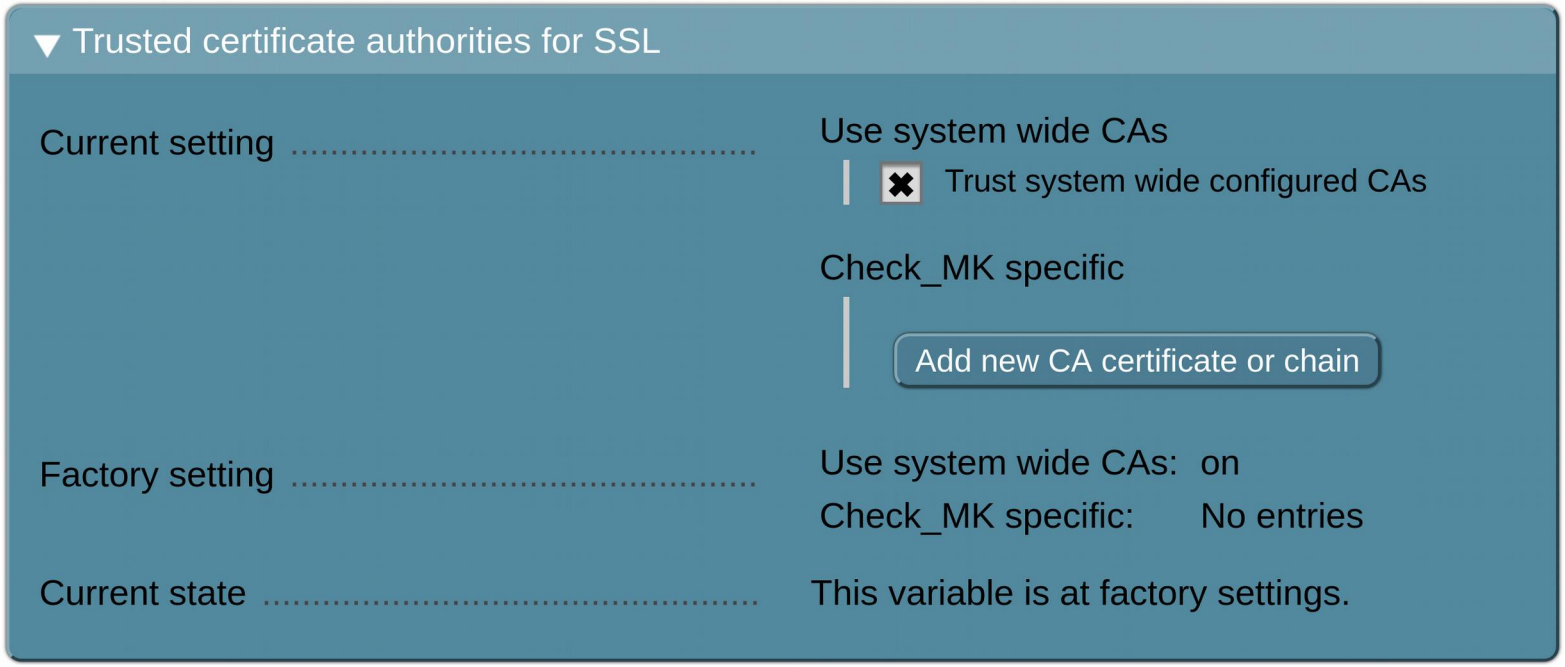
Application in Check_MK

Hands-on OpenSSL

Configuration in Check_MK

Check_MK as a client

Configuration of trusted root certificates in Global Settings – Site Management



Configuration in Check_MK

Check_MK as a client

Global trusted CAs used for

- LDAP user directory
- Distributed Monitoring: master site trusts slave site
- Active Checks (e.g. CheckHTTP)
- Datasource Programs (e.g. Vsphere)
- Everything the Check_MK Site connects to using SSL.

Configuration in Check_MK

Check_MK as a server

- OMD: Configuration of system Apache
  - Provide certificate chain and private key
- Check_MK Appliance: configuration via web-interface
  - Device Settings – Web Access

Web Access

Web Configuration **11:09**

Back | New Certificate | Upload Certificate | Download Certificate | Delete Certificates

▼ Configure Web Access Modes

Web access mode ..... HTTP and HTTPS ▼

- HTTP only (default)
- HTTPS enforced (incl. redirect of HTTP to HTTPS)
- HTTP and HTTPS
- HTTPS only

Save

Current State

HTTP	Active
HTTPS	Active

Current Certificate

Issued To

Country	DE
State or Province Name	BY
Locality Name	MU
Organization Name	MK
Device Host Address	mycma

Issued By

Country	DE
State or Province Name	BY
Locality Name	MU
Organization Name	MK
Device Host Address	mycma

New Certificate

Back

New Certificate

New Certificate .....

Device Host Address (Common Name)

Specify different address...

my-Check-MK-Server.com

Additional Host Addreses (Subject Alternative Names)

my-Check-MK-Server.com

myCheckMKServer.com

Add new element

Email Address

Country Name (2 letter code)

State or Province Name (full name)

Locality Name (e.g. city)

Organization Name (e.g. company)

Organizational Unit Name (e.g. section)

Signing Method

Create a Certificate Signing Request (CSR)

Create a Certificate Signing Request (CSR)

Create a self signed certificate

Save

Upload Certificate

Back

You can use this form to upload certificate files to your device which will then be used to encrypt the HTTPS connection if enabled. If you created a Certificate Signing Request (CSR) and received a signed certificate file now, you need to upload it using this form. Some Certificate Authorities (CAs) provide you with an intermediate or root certficiate (certificate chain) which you can also upload using this form. When you have a private key together with a certificate that has not been created on your device, you need to upload the private key together with the certificate.

Upload Certificate

Upload Certificate .....

- [x] Private Key
  - Paste PEM Content
    - Upload CRT/PEM File
    - Paste PEM Content
  - (empty)
- [x] Certificate
  - Upload CRT/PEM File
  - Choose File No file chosen
- [x] Certificate Chain (Root / Intermediate Certificate)
  - Upload CRT/PEM File
  - Choose File No file chosen

Upload

Configuration in Check_MK

Check_MK as a server

- Visit Check_MK site in browser
- Connect via Agent Updater
- Distributed Monitoring: Provide server certificate for master site

- SSL and Certificates -

SSL (or TLS?)

X.509 and
the certificate chain

Application in Check_MK

Hands-on OpenSSL

Useful commands

```
$ openssl x509 -in COMODO_Certification_Authority.pem -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            4e:81:2d:8a:82:65:e0:0b:02:ee:3e:35:02:46:e5:3d
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=GB, ST=Greater Manchester, L=Salford, O=COMODO CA Limited, CN=COMODO Certification Authority
        Validity
            Not Before: Dec  1 00:00:00 2006 GMT
            Not After : Dec 31 23:59:59 2029 GMT
        Subject: C=GB, ST=Greater Manchester, L=Salford, O=COMODO CA Limited, CN=COMODO Certification Authority
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:d0:40:8b:8b:72:e3:91:1b:f7:51:c1:1b:54:04:
                    98:d3:a9:bf:c1:e6:8a:5d:3b:87:fb:bb:88:ce:0d:
                    e3:2f:3f:06:96:f0:a2:29:50:99:ae:db:3b:a1:57:
                    b0:74:51:71:cd:ed:42:91:4d:41:fe:a9:c8:d8:6a:
                    86:77:44:bb:59:66:97:50:5e:b4:d4:2c:70:44:cf:
                    da:37:95:42:69:3c:30:c4:71:b3:52:f0:21:4d:a1:
                    d8:ba:39:7c:1c:9e:a3:24:9d:f2:83:16:98:aa:16:
                    7c:43:9b:15:5b:b7:ae:34:91:fe:d4:62:26:18:46:
                    9a:3f:eb:c1:f9:f1:90:57:eb:ac:7a:0d:8b:db:72:
                    30:6a:66:d5:e0:46:a3:70:dc:68:d9:ff:04:48:89:
                    77:de:b5:e9:fb:67:6d:41:e9:bc:39:bd:32:d9:62:
                    02:f1:b1:a8:3d:6e:37:9c:e2:2f:e2:d3:a2:26:8b:
                    c6:b8:55:43:88:e1:23:3e:a5:d2:24:39:6a:47:ab:
                    00:d4:a1:b3:a9:25:fe:0d:3f:a7:1d:ba:d3:51:c1:
                    0b:a4:da:ac:38:ef:55:50:24:05:65:46:93:34:4f:
                    2d:8d:ad:c6:d4:21:19:d2:8e:ca:05:61:71:07:73:
                    47:e5:8a:19:12:bd:04:4d:ce:4e:9c:a5:48:ac:bb:
                    26:f7
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                0B:58:E5:8B:C6:4C:15:37:A4:40:A9:30:A9:21:BE:47:36:5A:56:FF
            X509v3 Key Usage: critical
                Certificate Sign, CRL Sign
            X509v3 Basic Constraints: critical
                CA:TRUE
            X509v3 CRL Distribution Points:

                Full Name:
                  URI:http://crl.comodoca.com/COMODOCertificationAuthority.crl

    Signature Algorithm: sha1WithRSAEncryption
```

Analyze connection to server

```
$ openssl s_client -connect mathias-kettner.de:443

CONNECTED(00000003)
depth=2 O = Digital Signature Trust Co., CN = DST Root CA X3
verify return:1
depth=1 C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
verify return:1
depth=0 CN = mathias-kettner.de
verify return:1
---
Certificate chain
 0 s:/CN=mathias-kettner.de
   i:/C=US/O=Let's Encrypt/CN=Let's Encrypt Authority X3
 1 s:/C=US/O=Let's Encrypt/CN=Let's Encrypt Authority X3
   i:/O=Digital Signature Trust Co./CN=DST Root CA X3
---
```

Analyze connection to server

```
Server certificate
-----BEGIN CERTIFICATE-----
MIIGKDCCBRCgAwIBAgISBG3nWe0MKO7NhxCO6Xx9L1b/MA0GCSqGSIb3DQEBCwUA
MEoxCzAJBgNVBAYTAlVTMRYwFAYDVQQKEw1MZXQncyBFbmNyeXB0MSMwIQYDVQQD
ExpMZXQncyBFbmNyeXB0IEF1dGhvcml0eSBYMzAeFw0xODAzMjkxOTI0MTFaFw0x
[...]
n9AmCuoNnLFSv5+crzG6ae3CVNtkIZHQjAGYP/adXa7kfdcO9NXzG2jbO0XucU52
Q22WgS9AhLsKjmDN16oZW2n4nNK9Q4PIVHaqMmLD7kAtuG5+6B2malzX1bQjW6nr
nv6p+LkAyisOh6GNhBmPlOcuSqQuhVPTiuEFkdW1aRnCuMNiFcFmU3q4h34=
-----END CERTIFICATE-----

subject=/CN=mathias-kettner.de
issuer=/C=US/O=Let's Encrypt/CN=Let's Encrypt Authority X3
---
```

Analyze connection to server

```
No client certificate CA names sent
Peer signing digest: SHA512
Server Temp Key: ECDH, P-256, 256 bits
---
SSL handshake has read 3448 bytes and written 431 bytes
---
New, TLSv1/SSLv3, Cipher is ECDHE-RSA-AES256-GCM-SHA384
Server public key is 2048 bit
Secure Renegotiation IS supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
```

Analyze connection to server

```
SSL-Session:
    Protocol  : TLSv1.2
    Cipher    : ECDHE-RSA-AES256-GCM-SHA384
    Session-ID: F5B38C476B0FC4A050A6EDD61C52059100F66B1[...]
    Session-ID-ctx:
    Master-Key: 3281A124802A2965A8827A3D59FEA279D9D2BF4[...]
    Key-Arg   : None
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    TLS session ticket lifetime hint: 300 (seconds)
    TLS session ticket:
    0000 - b4 81 01 [...]

    Start Time: 1525191939
    Timeout   : 300 (sec)
    Verify return code: 0 (ok)
---
closed
```

Analyze X.509 certificate

```
$ openssl x509 -in /etc/ssl/certs/DST_Root_CA_X3.pem -text

Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            44:af:b0:80:d6:a3:27:ba:89:30:39:86:2e:f8:40:6b
    Signature Algorithm: sha1WithRSAEncryption
        Issuer: O=Digital Signature Trust Co., CN=DST Root CA X3
        Validity
            Not Before: Sep 30 21:12:19 2000 GMT
            Not After : Sep 30 14:01:15 2021 GMT
        Subject: O=Digital Signature Trust Co., CN=DST Root CA X3
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:df:af:[...]
                Exponent: 65537 (0x10001)
```

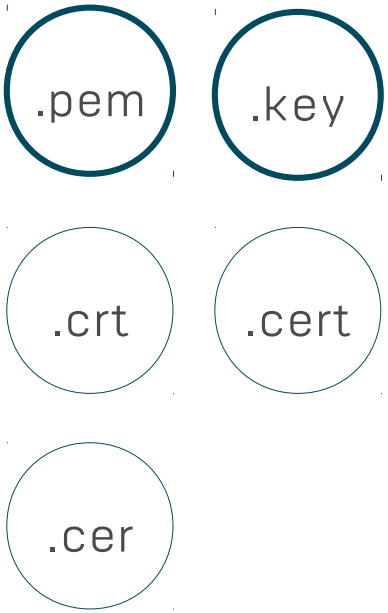
Analyze X.509 certificate

```
    X509v3 extensions:
        X509v3 Basic Constraints: critical
            CA:TRUE
        X509v3 Key Usage: critical
            Certificate Sign, CRL Sign
        X509v3 Subject Key Identifier:
            C4:A7:B1:[...]
Signature Algorithm: sha1WithRSAEncryption
     a3:1a:2c:9b:17:00:5c:a9:1e:ee:28:66:37:3a:bf:83:c7:3f:
     4b:c3:09:a0:95:20:5d:e3:d9:59:44:d2:3e:0d:3e:bd:8a:4b:
     [...]
```

File Types

```
-----BEGIN CERTIFICATE----
. . .
----END CERTIFICATE----
-----BEGIN RSA PRIVATE KEY----
. . .
-----END RSA PRIVATE KEY-----
```

PEM

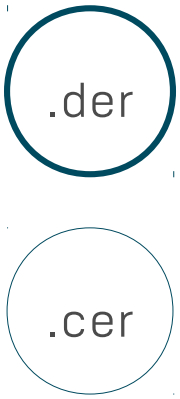


- Text format, Base64-encoded
- Most common format,
  e.g. Apache Server
- Can hold multiple certificates
  and private key

File Types

```
01100111001110011011000110
10110000110101011100010101
11110001110000111101101111
11100000000011110011011111
11000000000000111111101110
```

DER



- Binary format, “PEM without Base64”
- Holds single certificate or private key
- Common for Windows, Java

```
$ openssl x509 -inform der
  -in certificate.cer
  -out certificate.pem
```

File Types

```
-----BEGIN PKCS7----
...
----END PKCS7----
```

PKCS#7

.p7b

.p7c

- Text format, Base64-encoded
- Holds multiple certificates
- Common for Windows, Tomcat server

```
$ openssl pkcs7 -print_certs
  -in certificate.p7b
  -out certificate.pem
```

File Types

```
01100111001110011011000110
10110000110101011100010101
11110001110000111101101111
11100000000011110011011111
11000000000000111111101110
```

PKCS#12

- Binary format
- Can hold multiple certs and private key
- Password-protected
- Common for private key and certificate chain storage on Windows

```
$ openssl pkcs12
  -in certificate.pfx
  -out certificate.pem -nodes
```

Further reading

RFC #5280 (X.509 specification):

- https://tools.ietf.org/html/rfc5280

Setup your own CA using OpenSSL:

- https://gist.github.com/Soarez/9688998

Wikipedia articles:

- https://en.wikipedia.org/wiki/Transport_Layer_Security
- https://en.wikipedia.org/wiki/X.509

Browse Google

Browse YouToube

Credits

Comic “Protocol” by Randall Munroe taken from https://xkcd.com/1323

CHECK_MK

CONFERENCE

MUNICH 2018/5/2-4

#4