SSL/TLS for Mortals

Maarten Mulders (@mthmulders)

Transport Layer Security

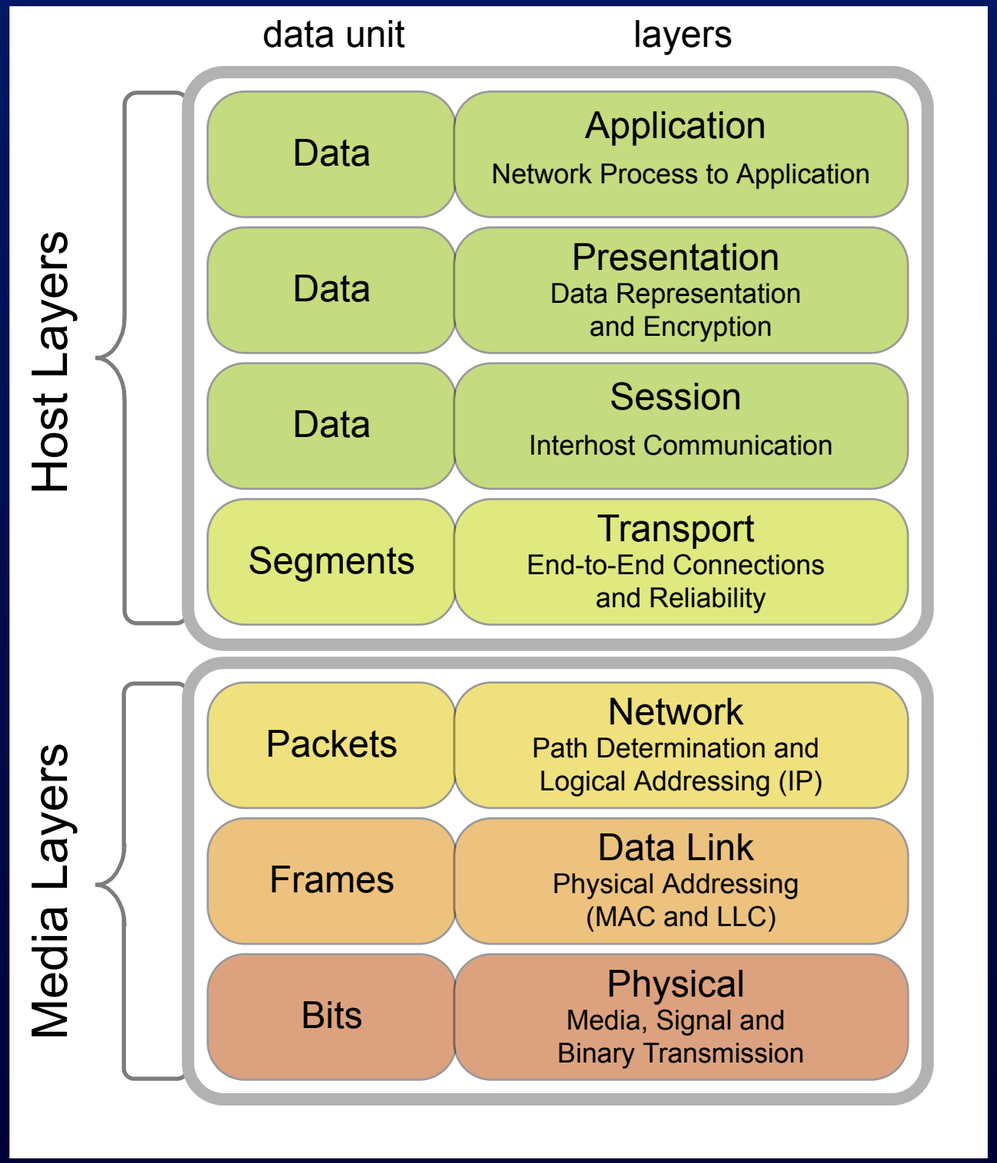
Awesome Sauce, or...

Headaches 🤯

```
Exception in thread "main" javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path building failed:
    at sun.security.ssl.Alerts.getSSLException(Alerts.java:192)
    at sun.security.ssl.SSLSocketImpl.fatal(SSLSocketImpl.java:1949)
    at sun.security.ssl.Handshaker.fatalSE(Handshaker.java:302)
    at sun.security.ssl.Handshaker.fatalSE(Handshaker.java:296)
    at sun.security.ssl.ClientHandshaker.serverCertificate(ClientHandshaker.java:1506)
    at sun.security.ssl.ClientHandshaker.processMessage(ClientHandshaker.java:216)
    at sun.security.ssl.Handshaker.processLoop(Handshaker.java:979)
    at sun.security.ssl.Handshaker.process_record(Handshaker.java:914)
    at sun.security.ssl.SSLSocketImpl.readRecord(SSLSocketImpl.java:1062)
    at sun.security.ssl.SSLSocketImpl.performInitialHandshake(SSLSocketImpl.java:1375)
    at sun.security.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.java:1403)
    at sun.security.ssl.SSLSocketImpl.startHandshake(SSLSocketImpl.java:1387)
    at sun.net.www.protocol.https.HttpsClient.afterConnect(HttpsClient.java:559)
    at sun.net.www.protocol.https.AbstractDelegateHttpsURLConnection.connect(AbstractDelegateHttpsURLConnection.java:185)
    at sun.net.www.protocol.http.HttpURLConnection.getInputStream0(HttpURLConnection.java:1512)
    at sun.net.www.protocol.http.HttpURLConnection.getInputStream(HttpURLConnection.java:1440)
    at sun.net.www.protocol.https.HttpsURLConnectionImpl.getInputStream(HttpsURLConnectionImpl.java:254)
    at it.mulders.maarten.Demo.main(Demo.java:13)
Caused by: sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderExc
    at sun.security.validator.PKIXValidator.doBuild(PKIXValidator.java:387)
    at sun.security.validator.PKIXValidator.engineValidate(PKIXValidator.java:292)
    at sun.security.validator.Validator.validate(Validator.java:260)
    at sun.security.ssl.X509TrustManagerImpl.validate(X509TrustManagerImpl.java:324)
    at sun.security.ssl.X509TrustManagerImpl.checkTrusted(X509TrustManagerImpl.java:229)
    at sun.security.ssl.X509TrustManagerImpl.checkServerTrusted(X509TrustManagerImpl.java:124)
    at sun.security.ssl.ClientHandshaker.serverCertificate(ClientHandshaker.java:1488)
    ... 13 more
Caused by: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target
    at sun.security.provider.certpath.SunCertPathBuilder.build(SunCertPathBuilder.java:146)
    at sun.security.provider.certpath.SunCertPathBuilder.engineBuild(SunCertPathBuilder.java:131)
    at java.security.cert.CertPathBuilder.build(CertPathBuilder.java:280)
    at sun.security.validator.PKIXValidator.doBuild(PKIXValidator.java:382)
```


SRW2024
24-port 10/100/1000 Gigabit Switch with WebView
Link/Act
Gigabit
SYSTEM
LINKSYS
SRW2016
16-port 10/100/1000 Gigabit Switch with WebView
Link/Act
Gigabit
SYSTEM
LINKSYS
t4sb
Electronics

7 Layers of OSI Model



History of SSL & TLS

SSL 1.0	*never released*
SSL 2.0	1995 - 2011 (*POODLE*)
SSL 3.0	1996 - 2014 (*POODLE*)
TLS 1.0	1999 - 2011 (*BEAST*)
TLS 1.1	2006
TLS 1.2	2008
TLS 1.3	2018

Demo time 🤞

What's the issue?!

How to prevent this?

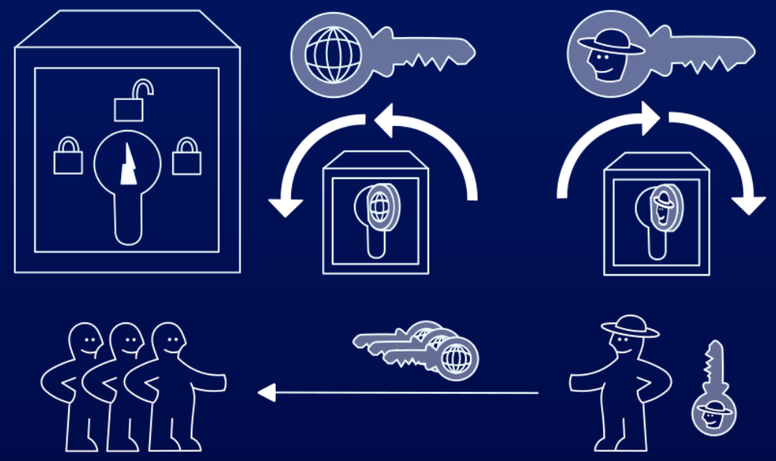
1. public/private key encryption
2. signed certificates
3. certificate authorities

1. Public & Private Key Encryption

PUBLIC KEY KRÜPTO

idea-instructions.com/public-key/
v1.0, CC by-nc-sa 4.0

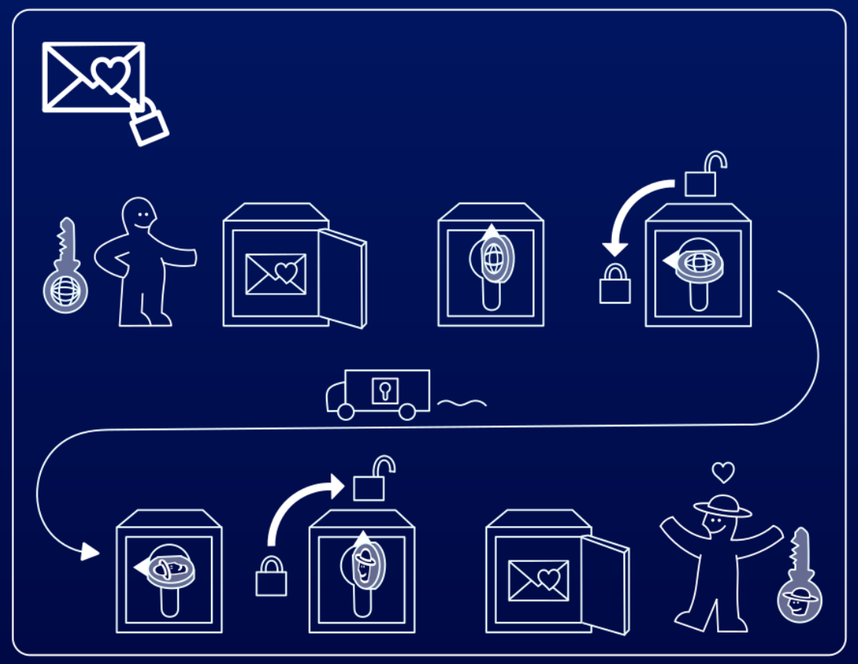
IDEA



PUBLIC KEY KRÜPTO

idea-instructions.com/public-key/
v1.0, CC by-nc-sa 4.0

IDEA



Math time!

1. Select two prime numbers: $p = 11, q = 17$
2. Calculate *product*: $p * q = 187$
3. Select random number < product: $e = 3$
4. Find *d*, so that $(d * e) - 1 \mod (p - 1) * (q - 1) = 0$
   a. $(d * 3) - 1 \mod (10 * 16) = 0$
   b. $320 \mod 160 = 0$
   c. $(321 - 1) \mod 160 = 0$
   d. $(107 * 3) = 321 \Rightarrow d = 107$

Note that d varies with e: when $e = 75, d = 183$.

Now, what if P and Q are unknown?

1. $p * q = 299, e = 5$
2. Find *d*, so that $(d * e) - 1 \mod (p - 1) * (q - 1) = 0$

Pretty hard without knowing p and q!

As soon as we know $p = 13, q = 23$, calculating $d = 317$ is trivial (again).

For big enough p and q, finding those factors will cost an eternity!

So we can distribute $p * q$ and even e!

Let's encrypt "G"

$p * q = 187, e = 3, G \Rightarrow 7$

$7^e = 7^3 = 343$

$343 \mod 187 = 156$

Let's decrypt "156"

Since *we* know p and q, we can calculate $d = 107$

$$156^{d} = 156^{107} \approx 4.6 * 10^{234}$$

$$156^{107} \mod 187 = 7$$

$$7 \Rightarrow G$$

Negotiating a secure connection

	Client		Server
1	ClientHello	→	
2		←	ServerHello
3		←	Certificate
4		←	ServerKeyExchange
5		←	ServerHelloDone
6	ClientKeyExchange	→	
7	ChangeCipherSpec	→	
8	Finished	→	
9		←	ChangeCipherSpec
10		←	Finished

Demo time 🤞

No-one is eavesdropping!

2. Signed Certificates

A certificate contains:

- Serial Number
- Subject
- Validity
- Usage
- Public Key
- Fingerprint Algorithm
- Fingerprint

But wait... anyone could create a certificate!

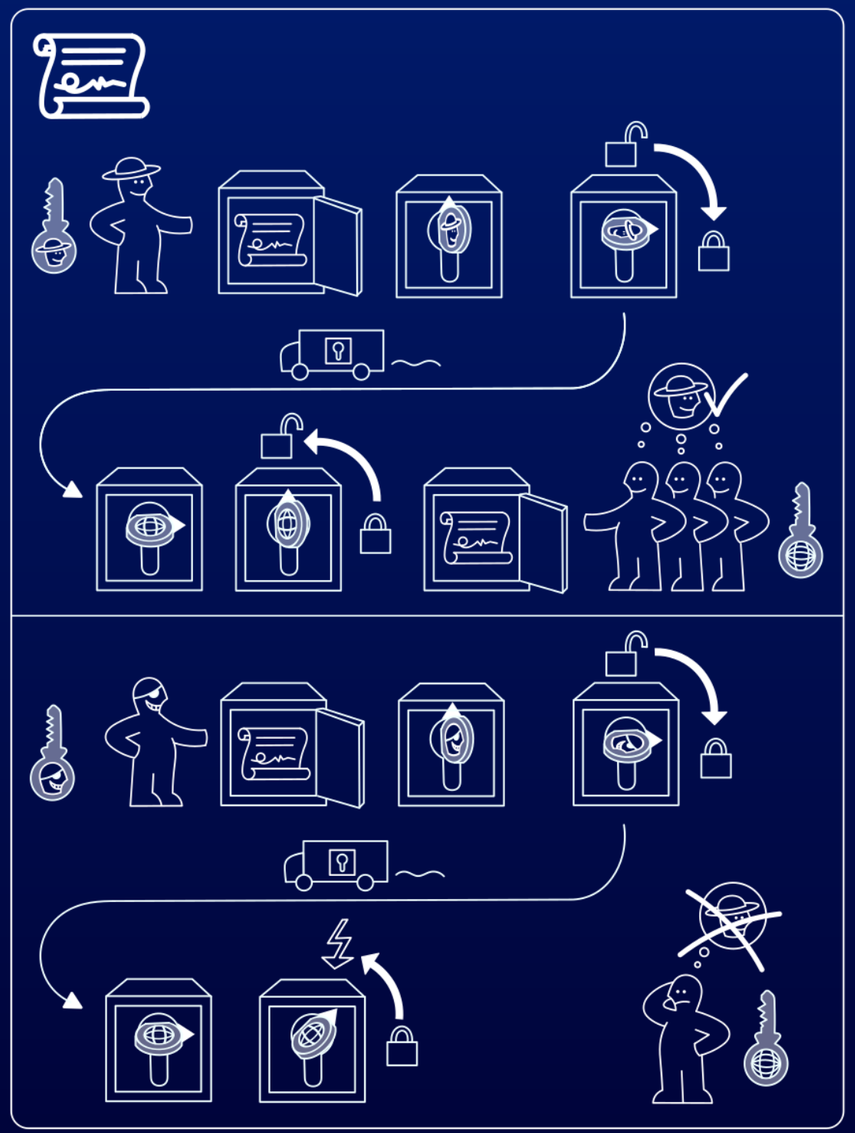
So we also need

- Signature Algorithm
- Signature
- Issuer

... and a way to sign certificates.

PUBLIC KEY KRÜPTO

IDEA



A *signature* is a mathematical relationship between a message x, a private key sk and a public key pk.

It consists of two functions:

1. *signing function* $t = f(sk, x)$
2. *verifying function* $[accept, reject] = g(pk, t, x)$

So, given x and t and knowing pk, we can tell if x is indeed signed by sk.

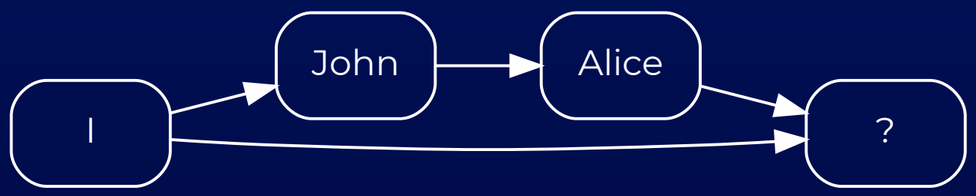
3. Certificate Authorities

An entity that issues *digital certificates*,

certifying the *ownership* of a *public key*

by the *subject* of the certificate.

> *I can trust you, because I trust John, and John trusts Alice, and Alice trusts you.*



So, who is John, anyway?

Many John's in todays browsers and operating systems!

Top-notch security procedures, including "key ceremonies"

```
/** intentionally left blank */
```

What happened next

- Google blacklists 247 certificates in Chromium
- Microsoft removes the DigiNotar root certificate from all supported Windows-releases *
- Mozilla revokes trust in the DigiNotar root certificate in all supported versions
- Apple issued Security Update 2011-005

- Update *Certificate Revocation Lists* (although these are self-signed)

Demo time 🤞🏻

Trust (for what it's worth)

Tools, Tips & Tricks

- Simple HTTP client with TLS support:
  `curl -v -k <address>`
- Troubleshoot trust issues and see certificates being used:
  `openssl s_client -showcerts -servername <address> -connect <address>:443`
- Troubleshoot supported protocols, ciphers, ...:
  `nmap --script ssl-enum-ciphers -p 443 <address>`
- Trace (or even decrypt) TLS traffic:
  `ssldump -i eth0 port 443 and host <address>` (add `-Ad -k <keyfile> -p <password>` for decryption)

JVM Settings

`-Djavax.net.ssl.trustStore=<file>`

Denotes where a *truststore* can be found: a file that contains *trusted certs*.

`-Djavax.net.ssl.trustStorePassword=changeit`

is the password to that file.

JVM Settings

`-Djavax.net.ssl.keyStore=<file>`

Denotes where a *keystore* can be found: a file that contains *public* and/or *private keys*.

`-Djavax.net.ssl.keyStorePassword=changeit`

is the password to that file.

JVM Settings

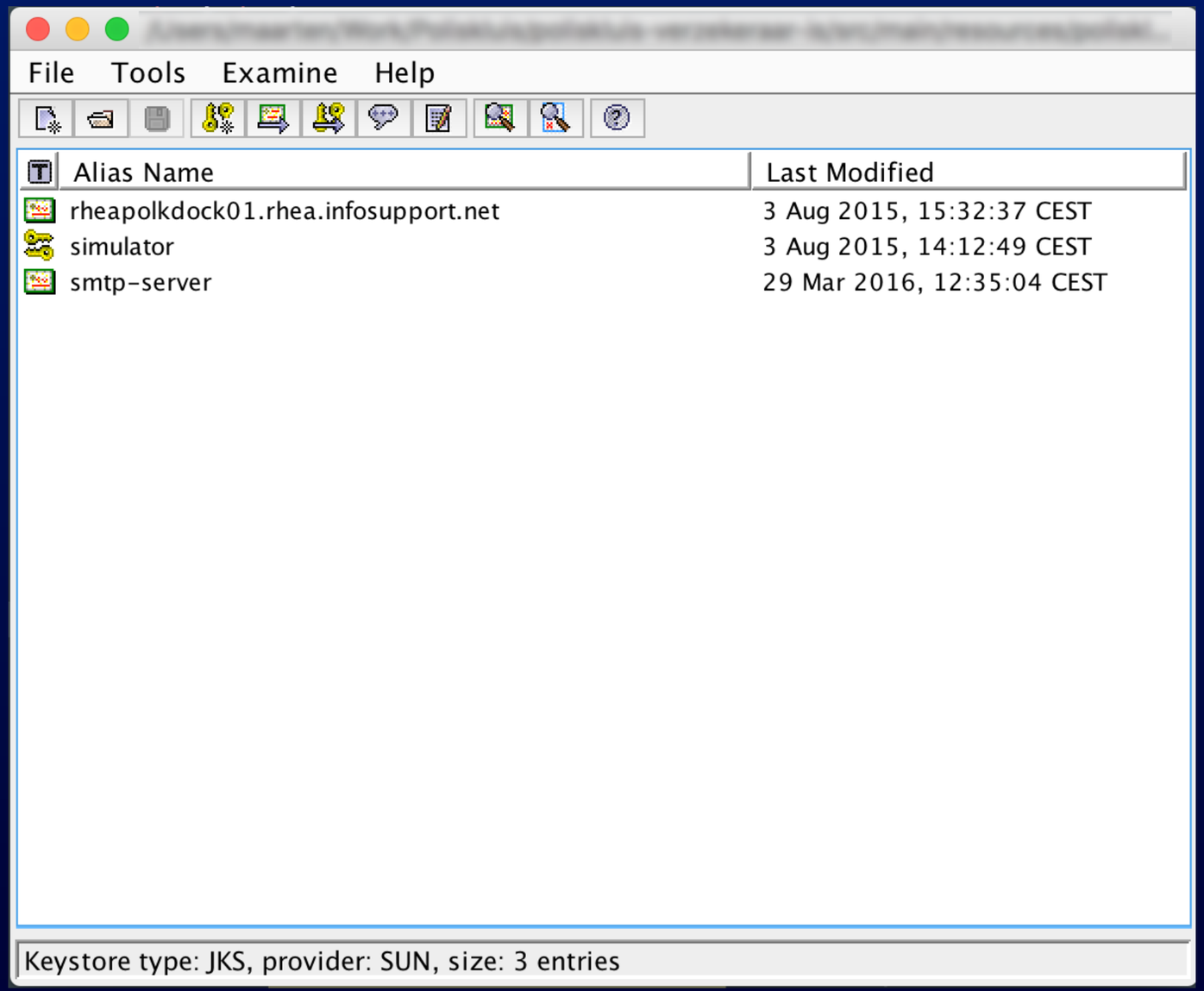
`-Djavax.net.debug=ssl[:flag]`

Include debug logging for TLS handshake and connections.

Additional flags:

record	session	sessioncache	pluggability	plaintext
handshake	defaultctx	keymanager	data	packet
keygen	sslctx	trustmanager	verbose	

Portecle


File
Tools
Examine
Help
Alias Name
Last Modified
rheapolkdock01.rhea.infosupport.net
3 Aug 2015, 15:32:37 CEST
simulator
3 Aug 2015, 14:12:49 CEST
smtp-server
29 Mar 2016, 12:35:04 CEST
Keystore type: JKS, provider: SUN, size: 3 entries

Public 🔑 take-aways

1. Don't use SSL!
   Use TLS v1.2 or v1.3.
2. Be careful whom you trust!
3. When in doubt, open your toolbox:
   openssl, curl, nmap, ssldump and Portecle

Questions?

Image attributions

Router by unknown author
Public Key Krüpto by Sándor P. Fekete, Sebastian Morr, and Sebastian Stiller (@ideainstruction)
Puss In Boots by kisspng
IANA Root KSK Ceremony #36 @ https://www.iana.org/dnssec/ceremonies/36
Beverwijk by Gerard Hogervorst @ Wikimedia Commons