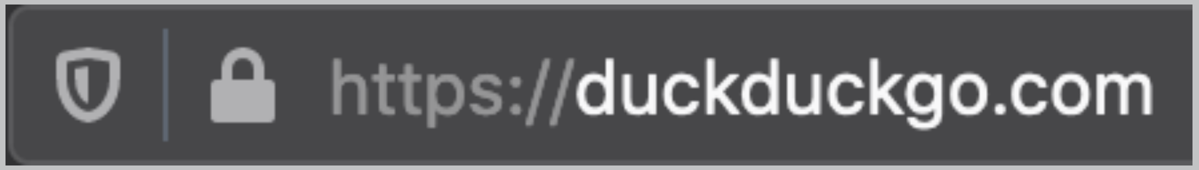
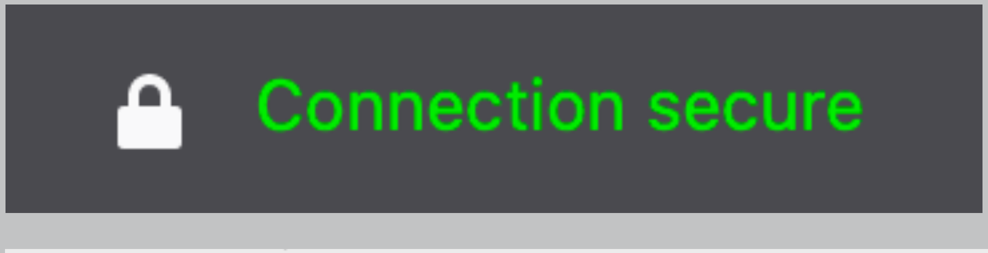
SECURITY (COMP0141): DIGITAL CERTIFICATES

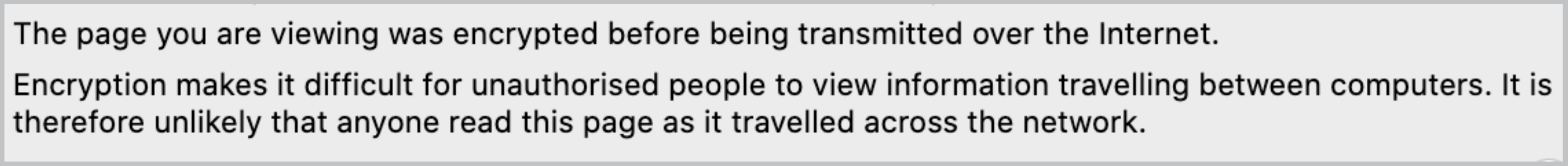
HTTPS INDICATORS TODAY: DEMO



https://duckduckgo.com



Connection secure

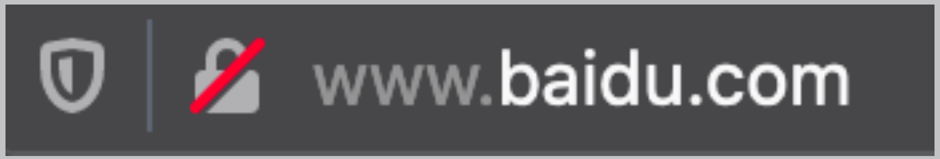


The page you are viewing was encrypted before being transmitted over the Internet.

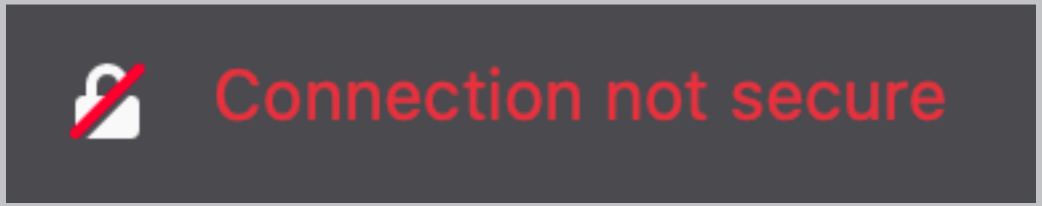
Encryption makes it difficult for unauthorised people to view information travelling between computers. It is therefore unlikely that anyone read this page as it travelled across the network.

secure
(encrypted)

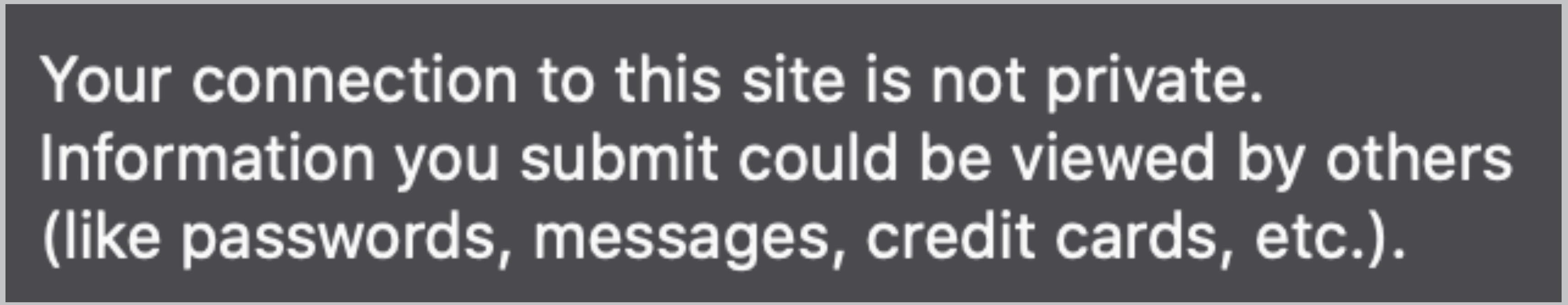
insecure
(unencrypted)



www.baidu.com

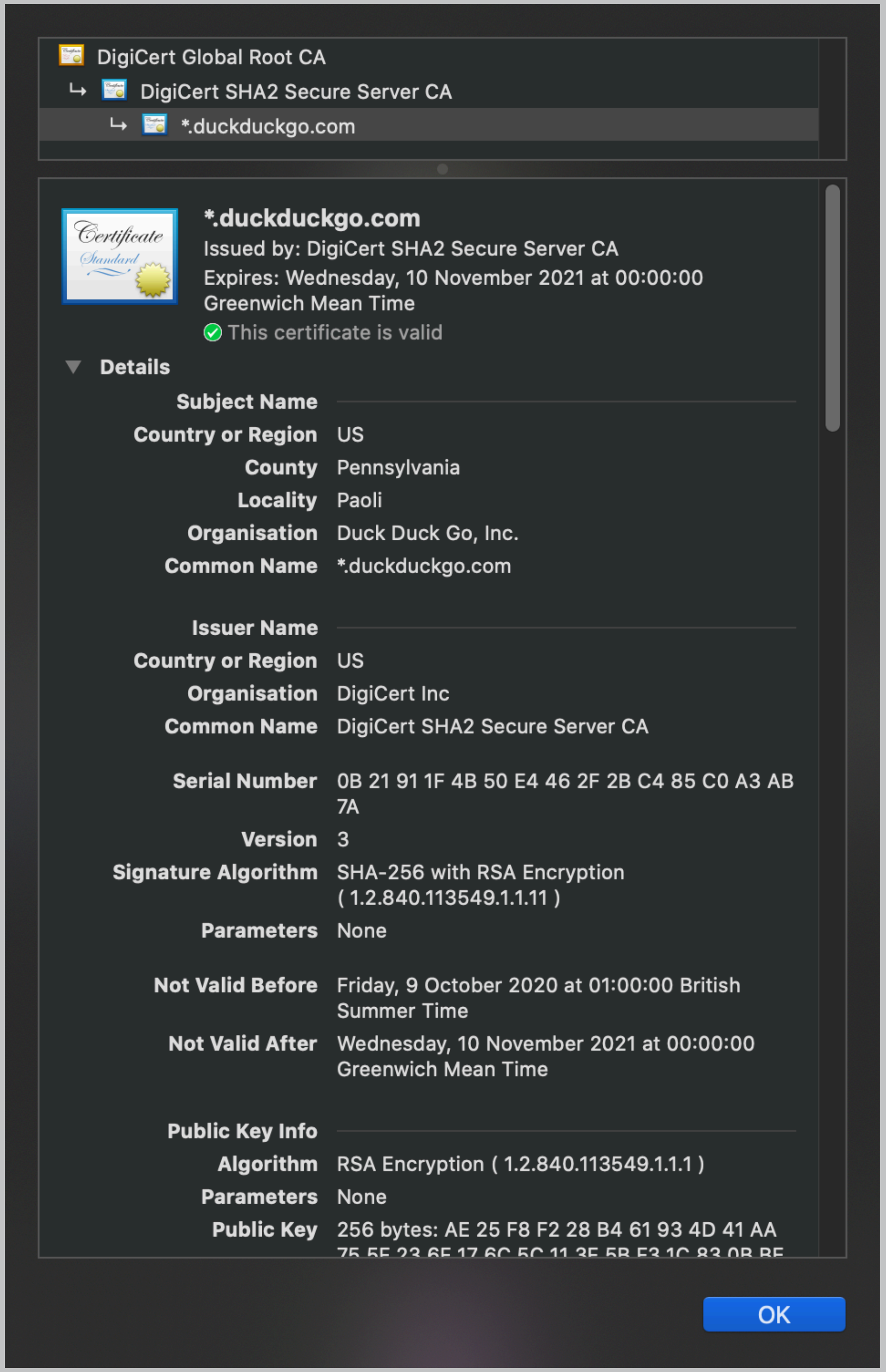


Connection not secure



Your connection to this site is not private. Information you submit could be viewed by others (like passwords, messages, credit cards, etc.).

DIGITAL CERTIFICATES DEMO



***.duckduckgo.com**
Issued by: DigiCert SHA2 Secure Server CA
Expires: Wednesday, 10 November 2021 at 00:00:00 Greenwich Mean Time
✓ This certificate is valid

▼ **Details**

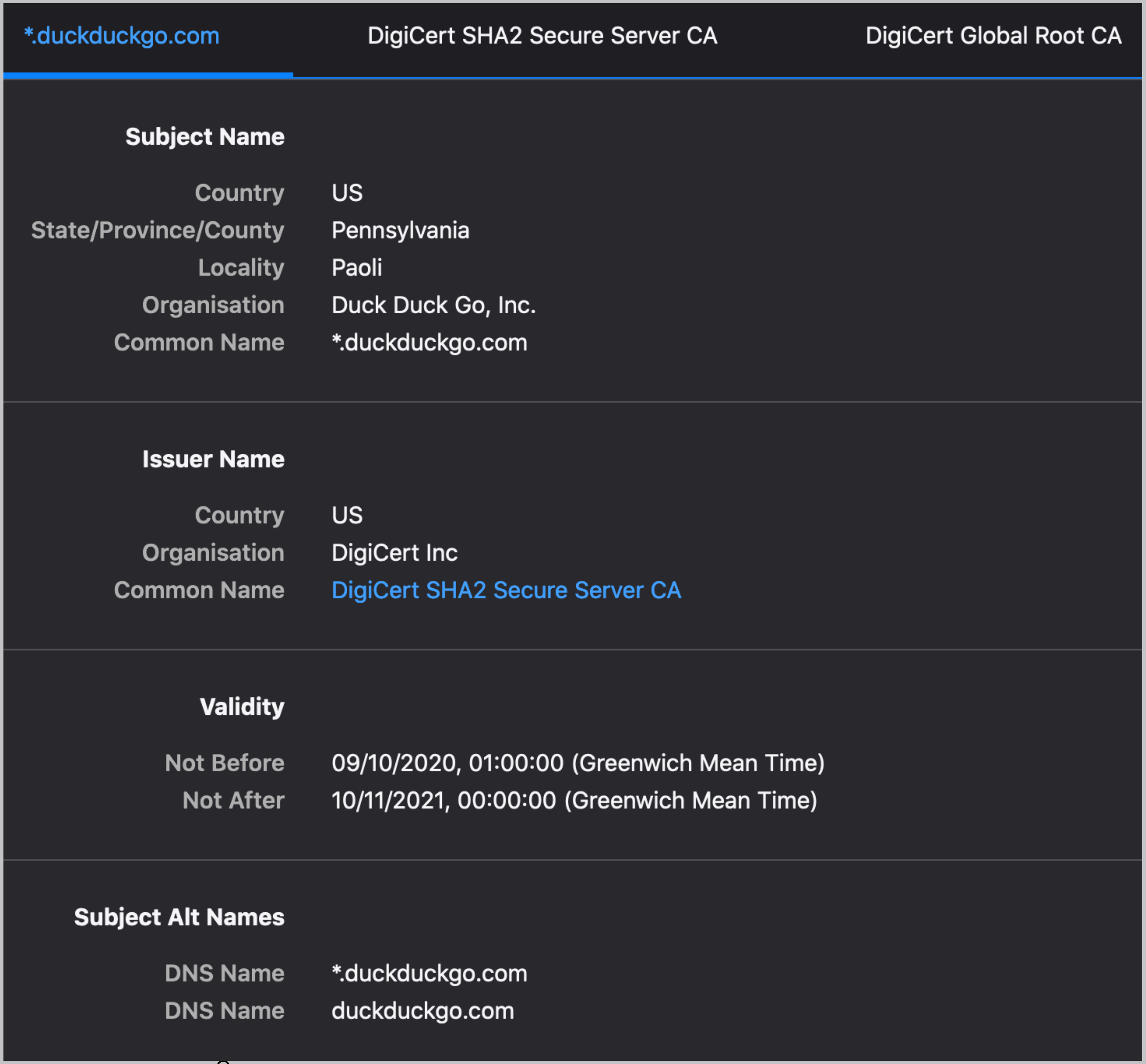
Subject Name
Country or Region US
County Pennsylvania
Locality Paoli
Organisation Duck Duck Go, Inc.
Common Name *.duckduckgo.com

Issuer Name
Country or Region US
Organisation DigiCert Inc
Common Name DigiCert SHA2 Secure Server CA

Serial Number 0B 21 91 1F 4B 50 E4 46 2F 2B C4 85 C0 A3 AB 7A
Version 3
Signature Algorithm SHA-256 with RSA Encryption (1.2.840.113549.1.1.11)
Parameters None

Not Valid Before Friday, 9 October 2020 at 01:00:00 British Summer Time
Not Valid After Wednesday, 10 November 2021 at 00:00:00 Greenwich Mean Time

Public Key Info
Algorithm RSA Encryption (1.2.840.113549.1.1.1)
Parameters None
Public Key 256 bytes: AE 25 F8 F2 28 B4 61 93 4D 41 AA 75 5F 23 6F 17 6C 5C 11 3F 5B F3 1C 83 0B BF



Subject Name
Country US
State/Province/County Pennsylvania
Locality Paoli
Organisation Duck Duck Go, Inc.
Common Name *.duckduckgo.com

Issuer Name
Country US
Organisation DigiCert Inc
Common Name DigiCert SHA2 Secure Server CA

Validity
Not Before 09/10/2020, 01:00:00 (Greenwich Mean Time)
Not After 10/11/2021, 00:00:00 (Greenwich Mean Time)

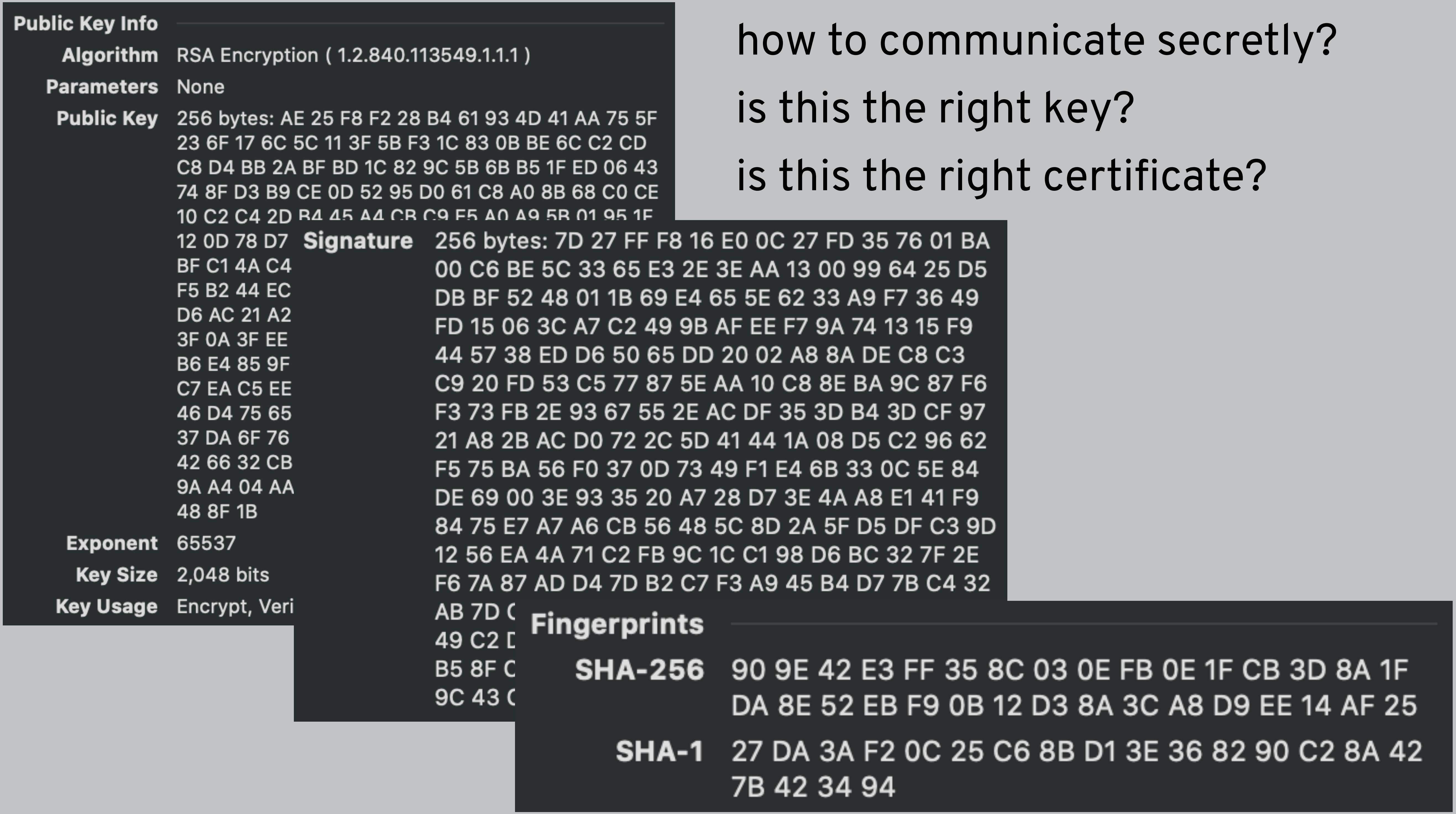
Subject Alt Names
DNS Name *.duckduckgo.com
DNS Name duckduckgo.com

DIGITAL CERTIFICATES

how to communicate secretly?

is this the right key?

is this the right certificate?

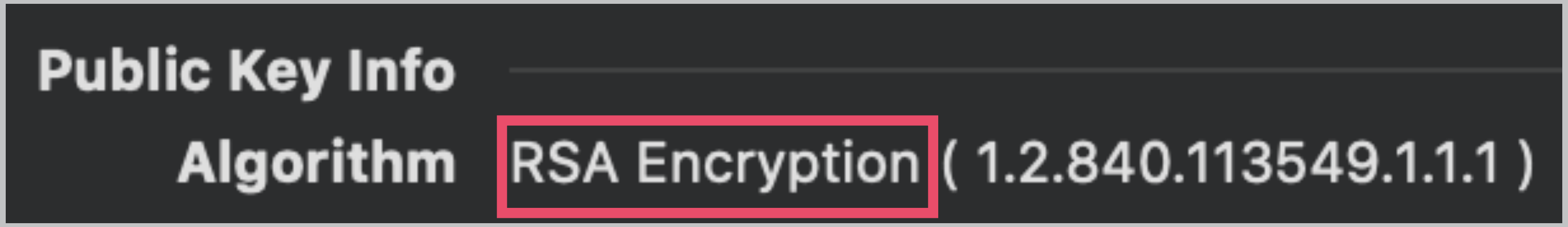


```
Public Key Info
     Algorithm   RSA Encryption ( 1.2.840.113549.1.1.1 )
    Parameters   None
    Public Key   256 bytes: AE 25 F8 F2 28 B4 61 93 4D 41 AA 75 5F
                 23 6F 17 6C 5C 11 3F 5B F3 1C 83 0B BE 6C C2 CD
                 C8 D4 BB 2A BF BD 1C 82 9C 5B 6B B5 1F ED 06 43
                 74 8F D3 B9 CE 0D 52 95 D0 61 C8 A0 8B 68 C0 CE
                 10 C2 C4 2D B4 45 A4 CB C9 F5 A0 A9 5B 01 95 1F
                 12 0D 78 D7
                 BF C1 4A C4
                 F5 B2 44 EC
                 D6 AC 21 A2
                 3F 0A 3F EE
                 B6 E4 85 9F
                 C7 EA C5 EE
                 46 D4 75 65
                 37 DA 6F 76
                 42 66 32 CB
                 9A A4 04 AA
                 48 8F 1B
      Exponent   65537
      Key Size   2,048 bits
     Key Usage   Encrypt, Veri
```

```
Signature   256 bytes: 7D 27 FF F8 16 E0 0C 27 FD 35 76 01 BA
            00 C6 BE 5C 33 65 E3 2E 3E AA 13 00 99 64 25 D5
            DB BF 52 48 01 1B 69 E4 65 5E 62 33 A9 F7 36 49
            FD 15 06 3C A7 C2 49 9B AF EE F7 9A 74 13 15 F9
            44 57 38 ED D6 50 65 DD 20 02 A8 8A DE C8 C3
            C9 20 FD 53 C5 77 87 5E AA 10 C8 8E BA 9C 87 F6
            F3 73 FB 2E 93 67 55 2E AC DF 35 3D B4 3D CF 97
            21 A8 2B AC D0 72 2C 5D 41 44 1A 08 D5 C2 96 62
            F5 75 BA 56 F0 37 0D 73 49 F1 E4 6B 33 0C 5E 84
            DE 69 00 3E 93 35 20 A7 28 D7 3E 4A A8 E1 41 F9
            84 75 E7 A7 A6 CB 56 48 5C 8D 2A 5F D5 DF C3 9D
            12 56 EA 4A 71 C2 FB 9C 1C C1 98 D6 BC 32 7F 2E
            F6 7A 87 AD D4 7D B2 C7 F3 A9 45 B4 D7 7B C4 32
            AB 7D C
            49 C2 D
            B5 8F C
            9C 43 C
```

```
Fingerprints
     SHA-256   90 9E 42 E3 FF 35 8C 03 0E FB 0E 1F CB 3D 8A 1F
               DA 8E 52 EB F9 0B 12 D3 8A 3C A8 D9 EE 14 AF 25
       SHA-1   27 DA 3A F2 0C 25 C6 8B D1 3E 36 82 90 C2 8A 42
               7B 42 34 94
```

STANDARDS

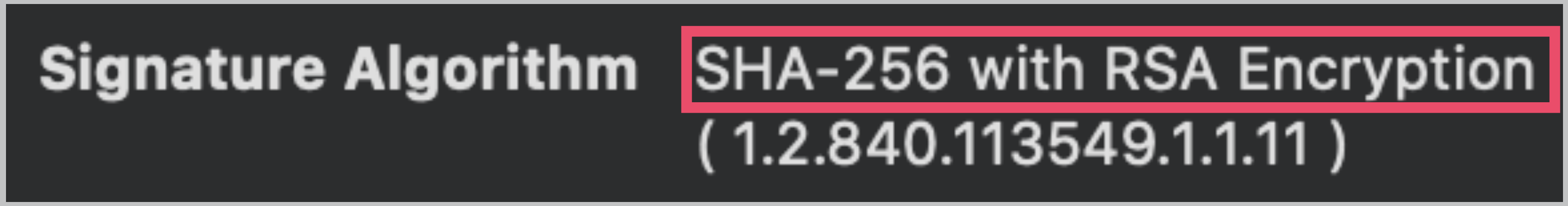


Public Key Info

Algorithm RSA Encryption (1.2.840.113549.1.1.1)

public-key encryption

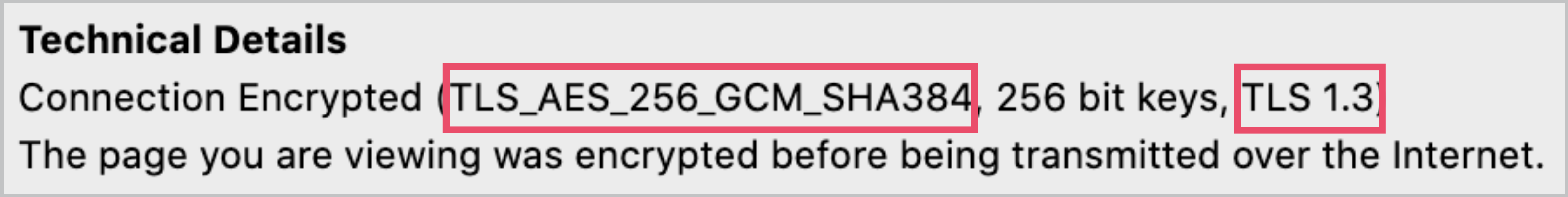
FDH digital signature
(also DSA,ECDSA)



Signature Algorithm SHA-256 with RSA Encryption
(1.2.840.113549.1.1.11)

AEAD

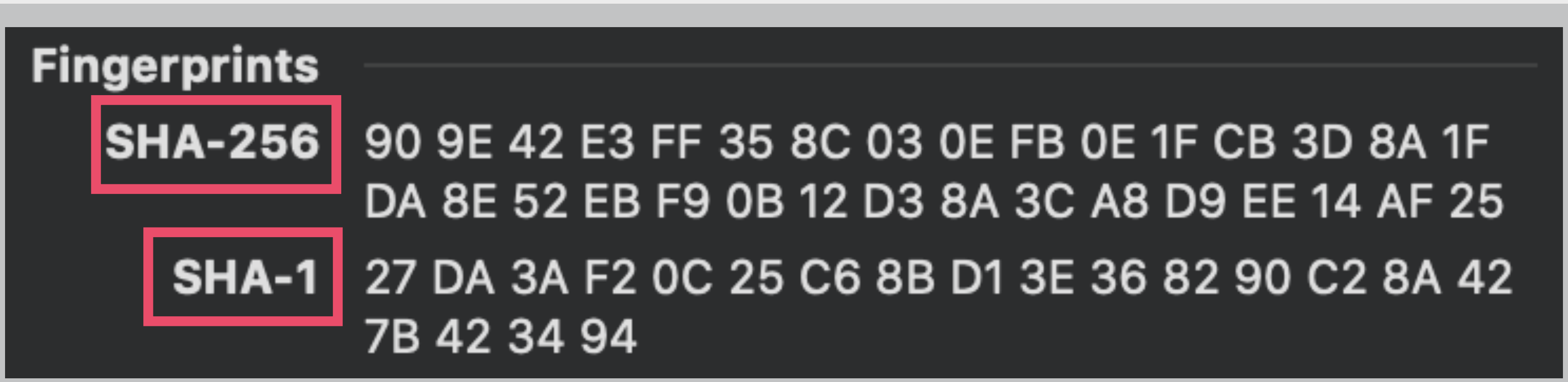
collection of protocols



Technical Details

Connection Encrypted (TLS_AES_256_GCM_SHA384, 256 bit keys, TLS 1.3)
The page you are viewing was encrypted before being transmitted over the Internet.

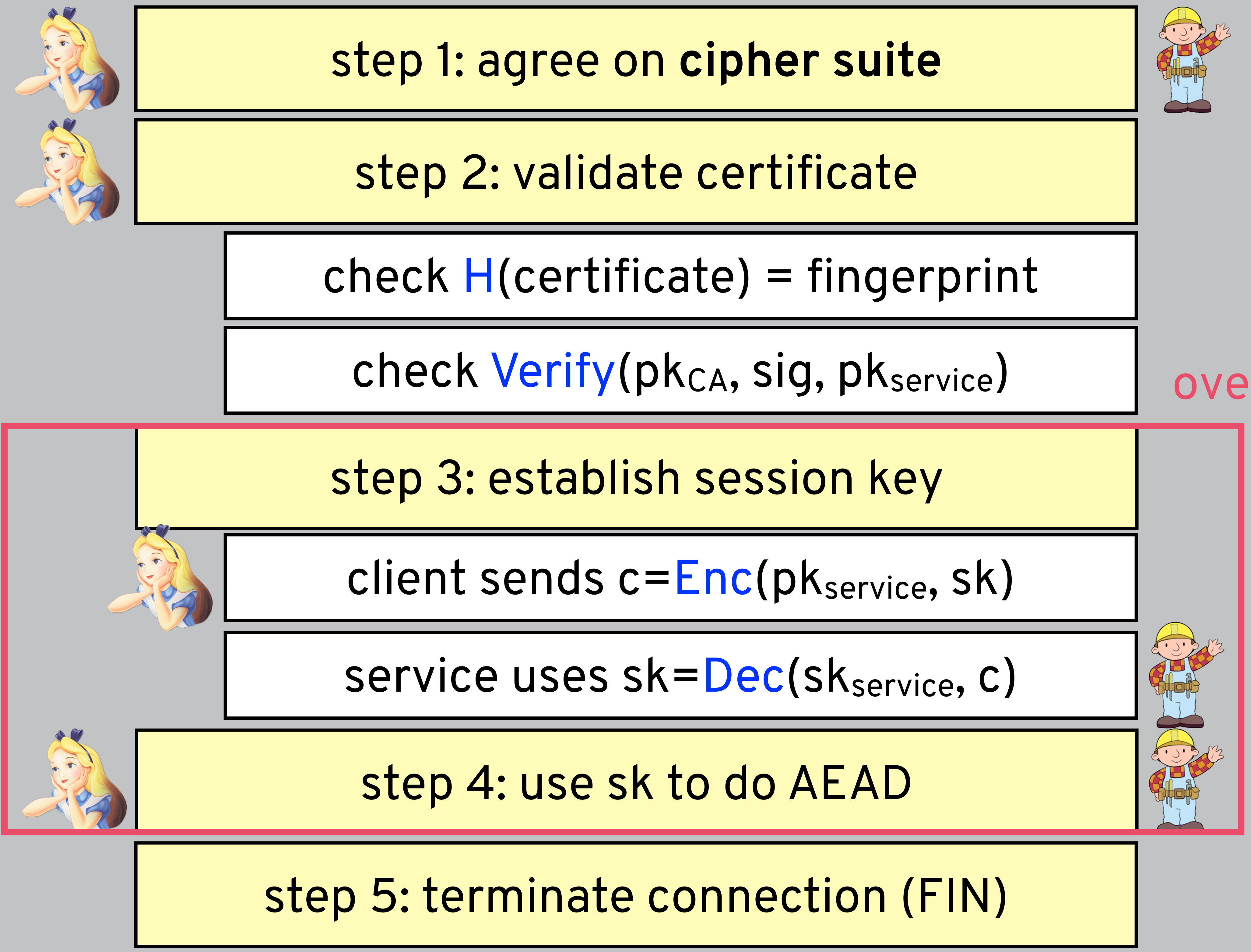
hash functions
(also SHA-3)



Fingerprints

```
SHA-256  90 9E 42 E3 FF 35 8C 03 0E FB 0E 1F CB 3D 8A 1F
         DA 8E 52 EB F9 0B 12 D3 8A 3C A8 D9 EE 14 AF 25
  SHA-1  27 DA 3A F2 0C 25 C6 8B D1 3E 36 82 90 C2 8A 42
         7B 42 34 94
```

SSL/TLS HANDSHAKE



step 1: agree on **cipher suite**

step 2: validate certificate

check H(certificate) = fingerprint

check Verify(pk_{CA}, sig, $pk_{service}$)

step 3: establish session key

client sends c=Enc($pk_{service}$, sk)

service uses sk=Dec($sk_{service}$, c)

step 4: use sk to do AEAD

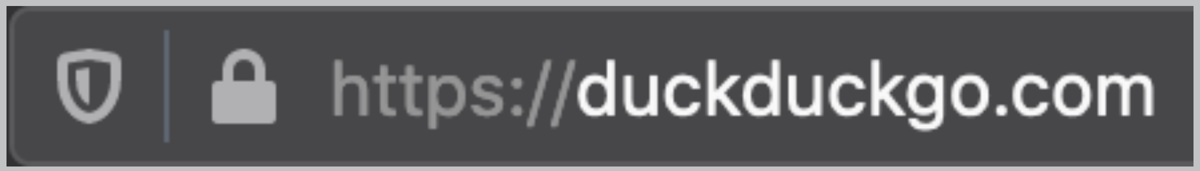
step 5: terminate connection (FIN)

oversimplified!

TLS

TLS (Transport Layer Security) is the standard for secure communication on the Internet today, **SSL** (Secure Socket Layer) is its predecessor

HTTPS (Secure HTTP) means you are running HTTP over TLS

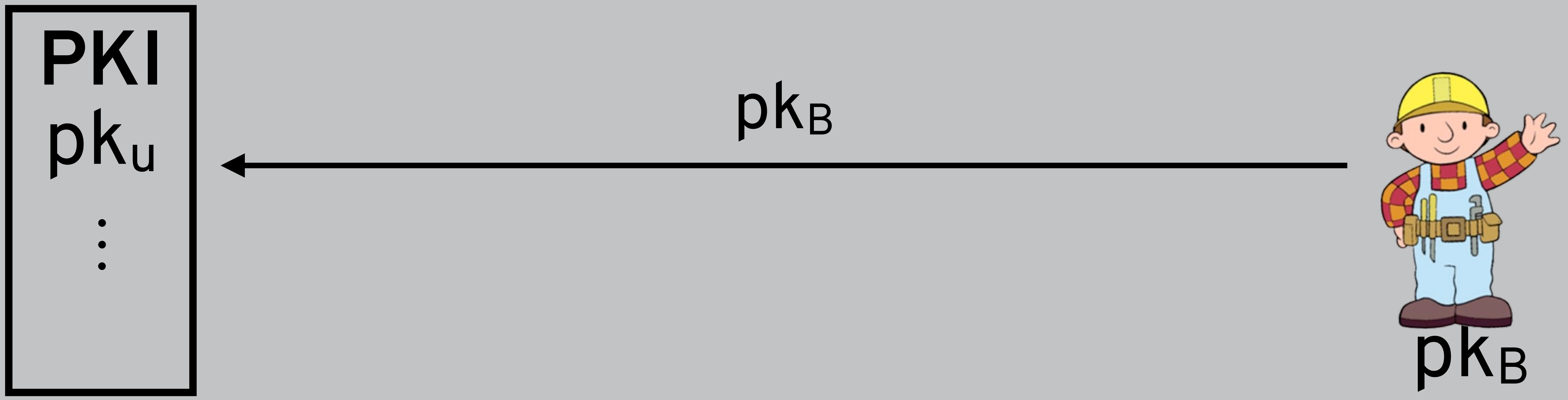


https://duckduckgo.com

AS ALWAYS, SOME QUESTIONS...

q: did we really provide a public-key infrastructure (PKI)?

PUBLIC-KEY INFRASTRUCTURE



PKI

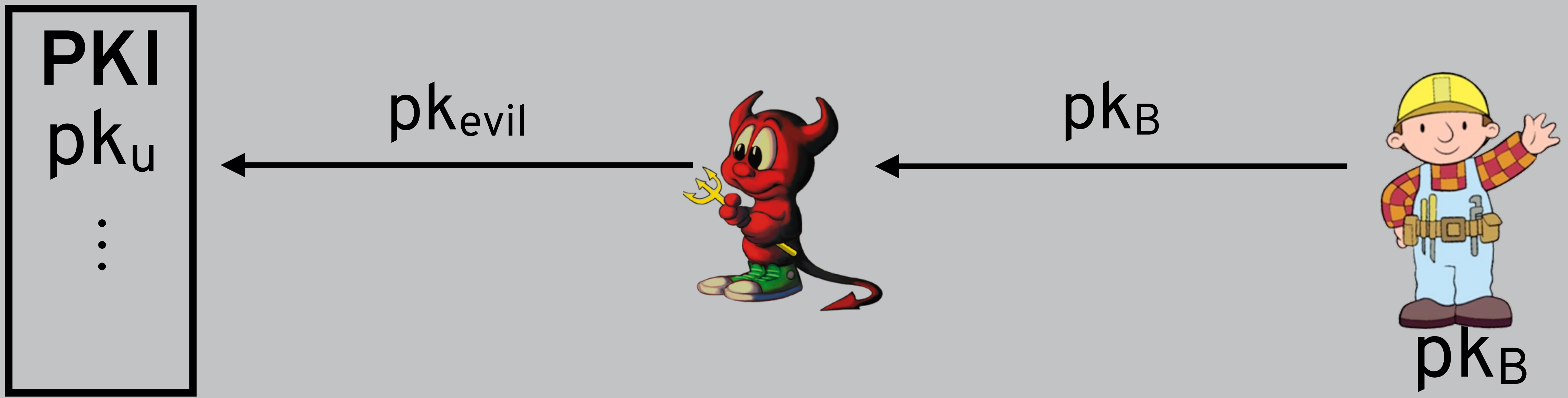
pk_u

⋮

pk_B

pk_B

PUBLIC-KEY INFRASTRUCTURE



PKI

pk_u

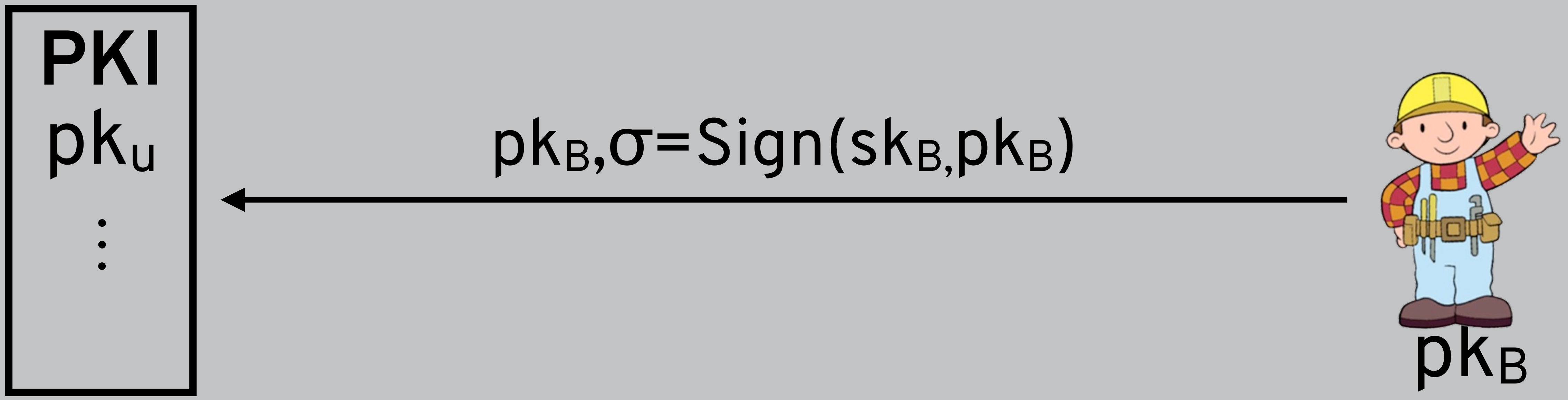
⋮

pk_{evil}

pk_B

pk_B

PUBLIC-KEY INFRASTRUCTURE



PKI

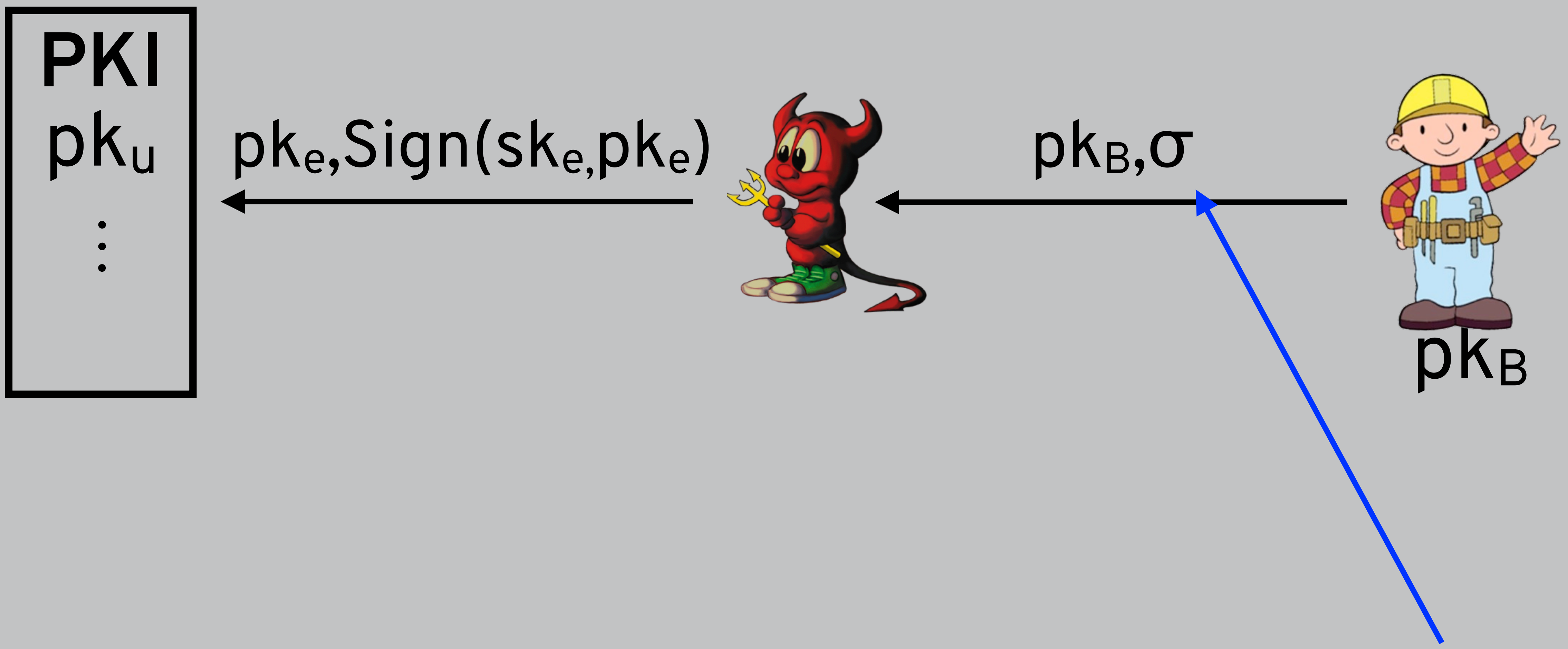
pk_u

⋮

$pk_B, \sigma = \mathrm{Sign}(sk_B, pk_B)$

pk_B

PUBLIC-KEY INFRASTRUCTURE



PKI

pk_u

⋮

pk_e,Sign(sk_e,pk_e)

pk_B,σ

pk_B

Bob could try to sign with a different key, but then how would we verify signature?

PUBLIC-KEY INFRASTRUCTURE

PKI

pk_u

⋮

alice.com	pk_A	Sign(sk_{CA},pk_A)
eve.com	pk_E	Sign(sk_{CA},pk_E)
…	…	…
bob.com	pk_B	Sign(sk_{CA},pk_B)

pk_B

check Verify(pk_{CA}, sig, $pk_{service}$)

(pk_{CA}, sk_{CA})

certificate authority

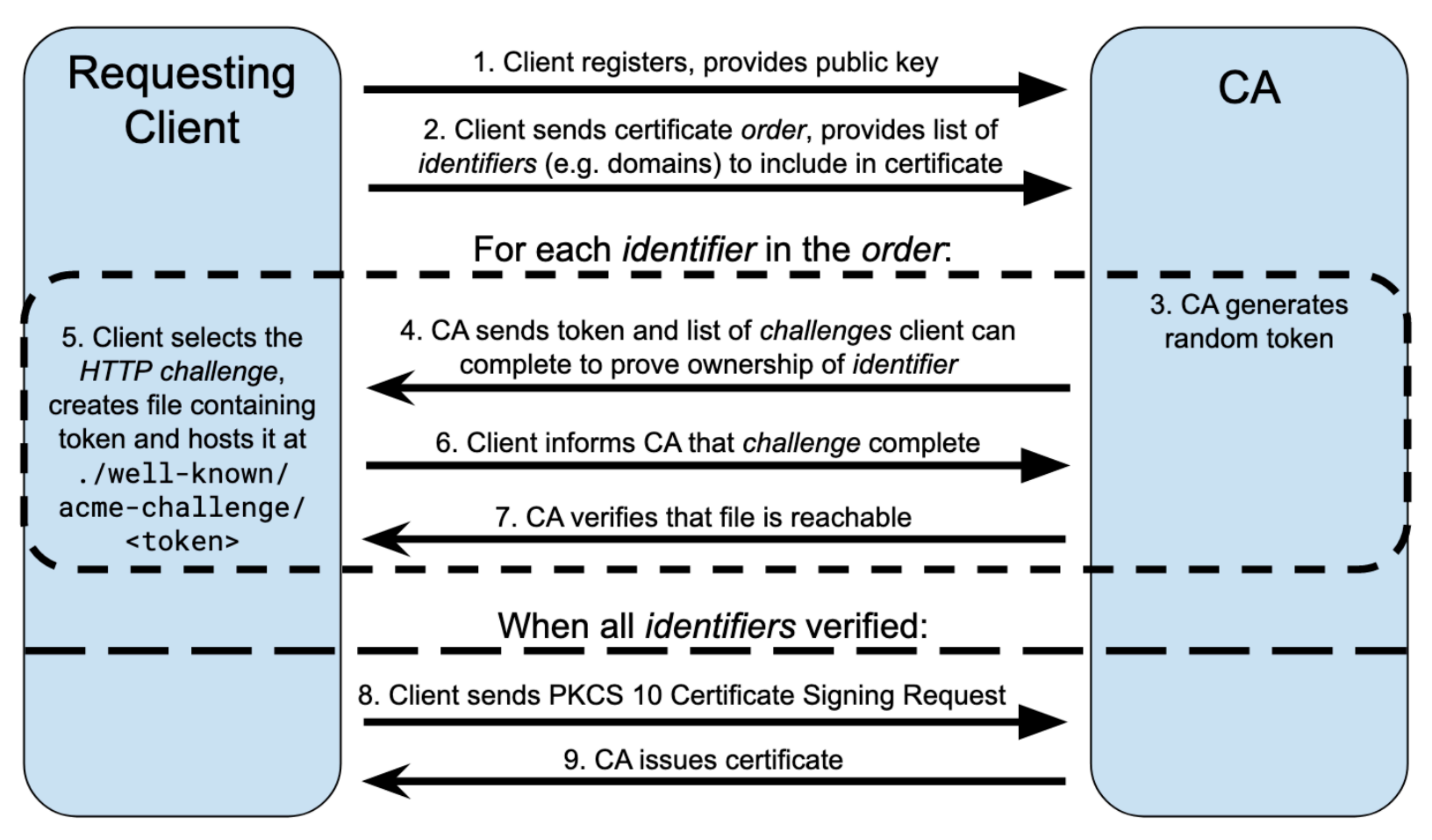
certificate signing request

so we've reduced key distribution problem to CA keys

LET'S ENCRYPT

Let's Encrypt is a fully automated (and free!) CA

- Performs only **domain validation**, stronger validations (organisational and extended) require human intervention
- Automated validation via ACME protocol



Requesting Client
CA
1. Client registers, provides public key
2. Client sends certificate *order*, provides list of *identifiers* (e.g. domains) to include in certificate
For each *identifier* in the *order*:
3. CA generates random token
4. CA sends token and list of *challenges* client can complete to prove ownership of *identifier*
5. Client selects the *HTTP challenge*, creates file containing token and hosts it at ./well-known/acme-challenge/<token>
6. Client informs CA that *challenge* complete
7. CA verifies that file is reachable
When all *identifiers* verified:
8. Client sends PKCS 10 Certificate Signing Request
9. CA issues certificate

HTTPS ADOPTION

62,142 in September 2015

20,362 in January 2019

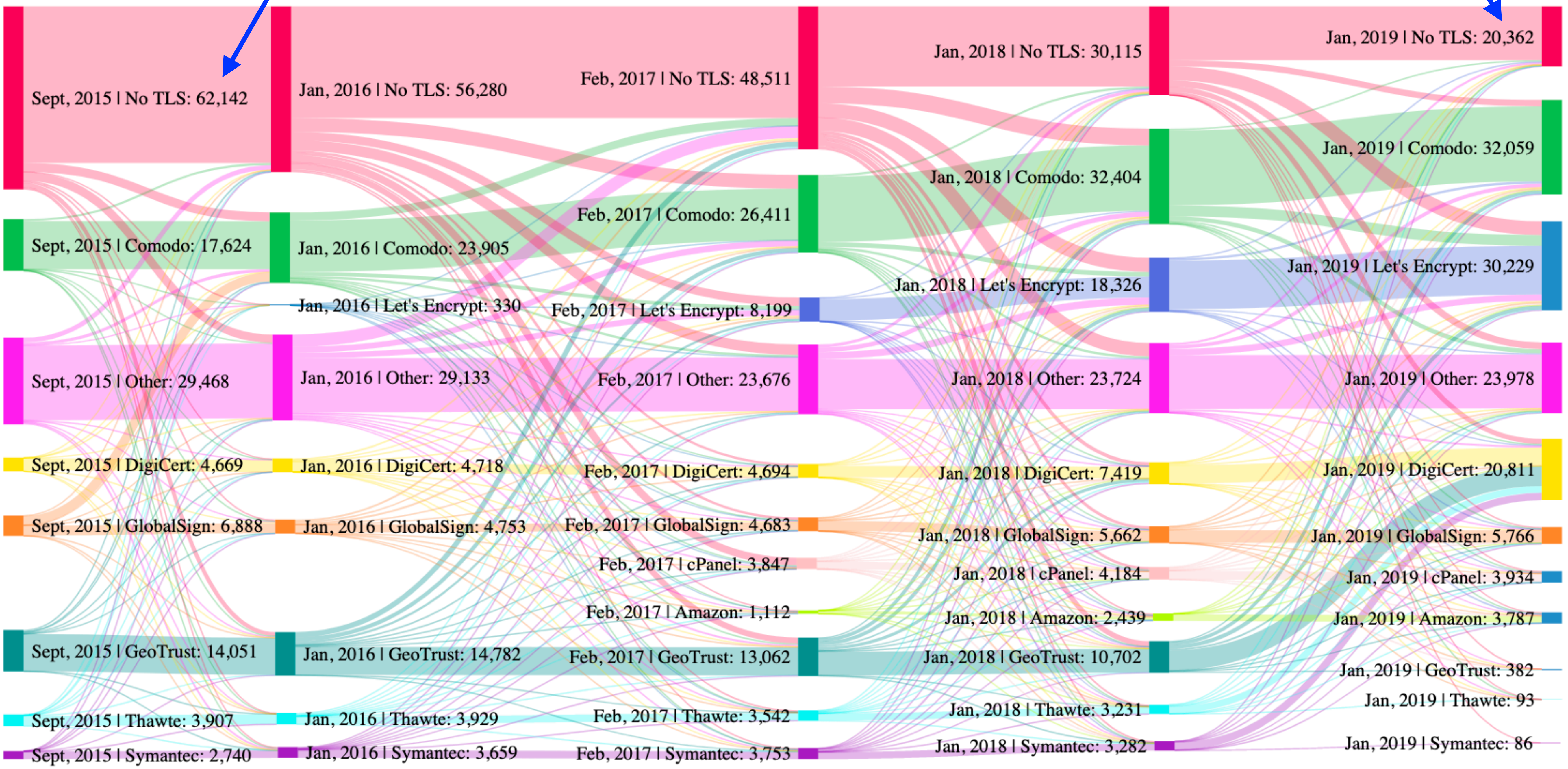


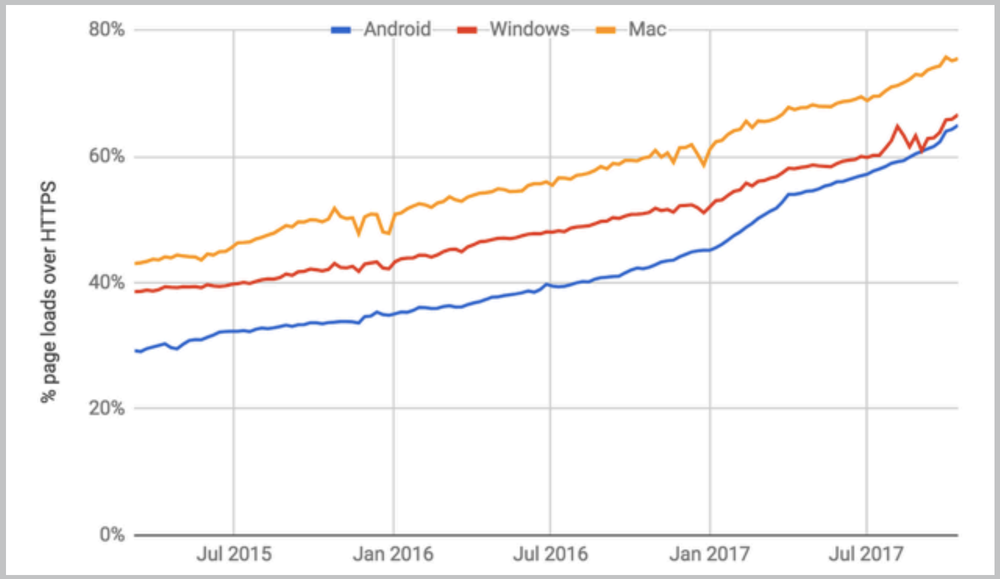
Figure 8: Certificate authority flow among stable, popular sites. We track CA choice for 141K domains over five snapshots, from 7/2015 to 1/2019. The included sites are those that were ranked in the Alexa Top Million at every snapshot, and so are likely more popular and long-lived than the top million overall.

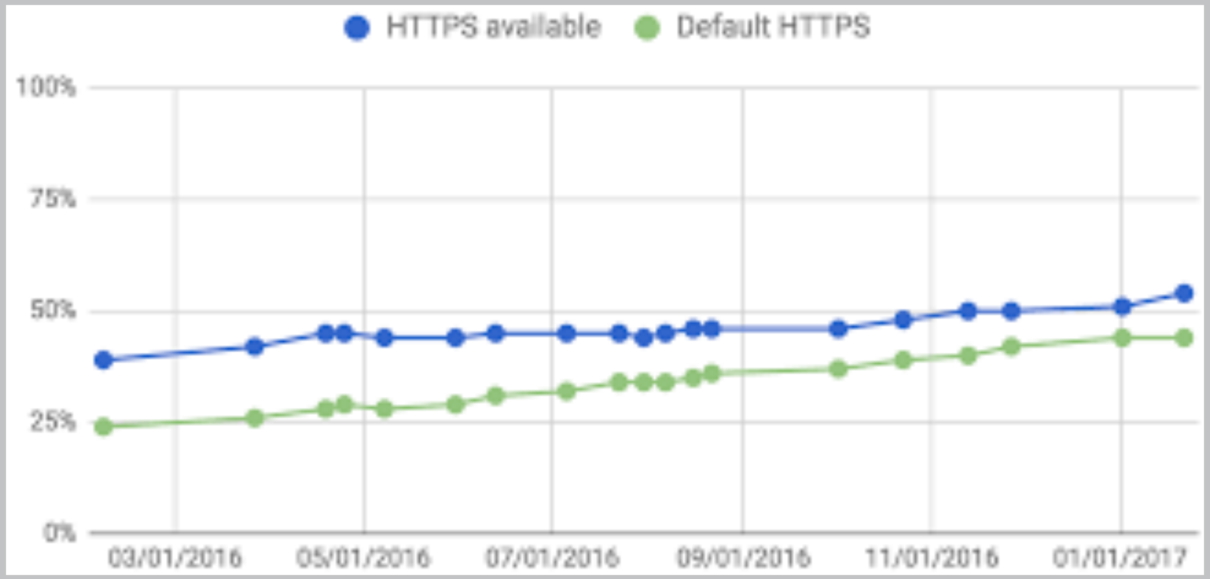
HTTPS ADOPTION

Success story for usable security!

- Studies showed that old (positive) indicators were not usable and thus did not protect users
- As a result, browsers moved or are moving towards negative indicators instead

This was enabled by **technological advances (which we'll see next week) that also made HTTPS much more widespread**

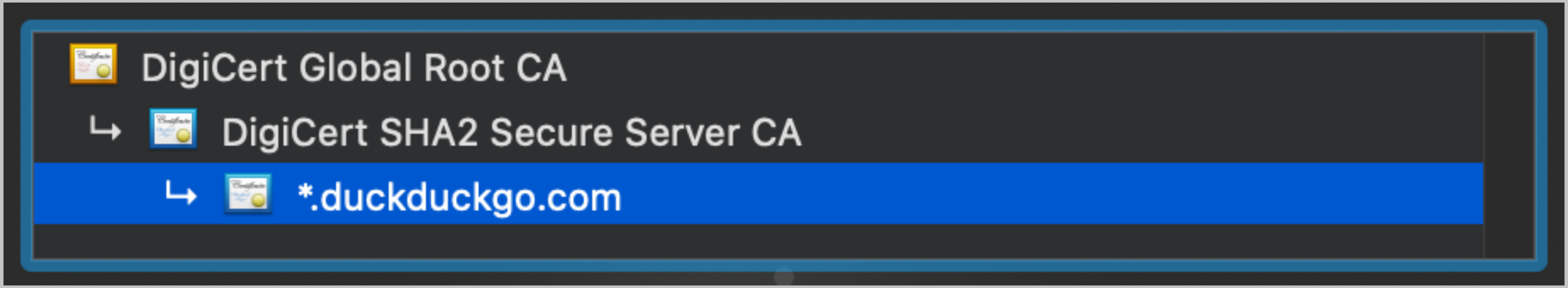




X.509 CERTIFICATES

The process we've just seen is typical of the X.509 standard

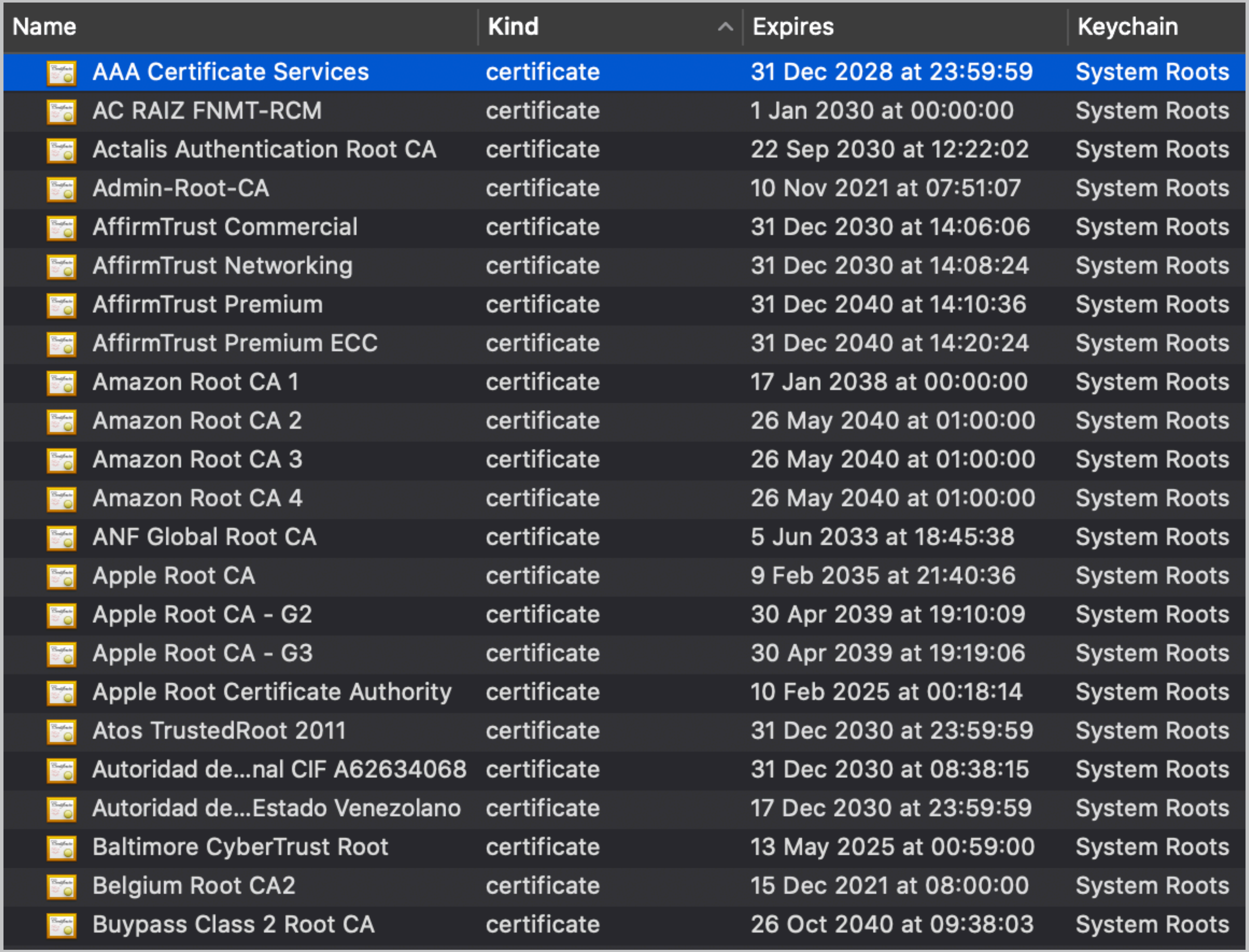
This also defines the structure of certificates and the concept of a **certificate chain**



DigiCert Global Root CA
↳ DigiCert SHA2 Secure Server CA
↳ *.duckduckgo.com

Root certificate in the chain is treated as a **trust anchor**

ROOT CERTIFICATES



Name	Kind	Expires	Keychain
AAA Certificate Services	certificate	31 Dec 2028 at 23:59:59	System Roots
AC RAIZ FNMT-RCM	certificate	1 Jan 2030 at 00:00:00	System Roots
Actalis Authentication Root CA	certificate	22 Sep 2030 at 12:22:02	System Roots
Admin-Root-CA	certificate	10 Nov 2021 at 07:51:07	System Roots
AffirmTrust Commercial	certificate	31 Dec 2030 at 14:06:06	System Roots
AffirmTrust Networking	certificate	31 Dec 2030 at 14:08:24	System Roots
AffirmTrust Premium	certificate	31 Dec 2040 at 14:10:36	System Roots
AffirmTrust Premium ECC	certificate	31 Dec 2040 at 14:20:24	System Roots
Amazon Root CA 1	certificate	17 Jan 2038 at 00:00:00	System Roots
Amazon Root CA 2	certificate	26 May 2040 at 01:00:00	System Roots
Amazon Root CA 3	certificate	26 May 2040 at 01:00:00	System Roots
Amazon Root CA 4	certificate	26 May 2040 at 01:00:00	System Roots
ANF Global Root CA	certificate	5 Jun 2033 at 18:45:38	System Roots
Apple Root CA	certificate	9 Feb 2035 at 21:40:36	System Roots
Apple Root CA - G2	certificate	30 Apr 2039 at 19:10:09	System Roots
Apple Root CA - G3	certificate	30 Apr 2039 at 19:19:06	System Roots
Apple Root Certificate Authority	certificate	10 Feb 2025 at 00:18:14	System Roots
Atos TrustedRoot 2011	certificate	31 Dec 2030 at 23:59:59	System Roots
Autoridad de...nal CIF A62634068	certificate	31 Dec 2030 at 08:38:15	System Roots
Autoridad de...Estado Venezolano	certificate	17 Dec 2030 at 23:59:59	System Roots
Baltimore CyberTrust Root	certificate	13 May 2025 at 00:59:00	System Roots
Belgium Root CA2	certificate	15 Dec 2021 at 08:00:00	System Roots
Buypass Class 2 Root CA	certificate	26 Oct 2040 at 09:38:03	System Roots

AS ALWAYS, SOME QUESTIONS…

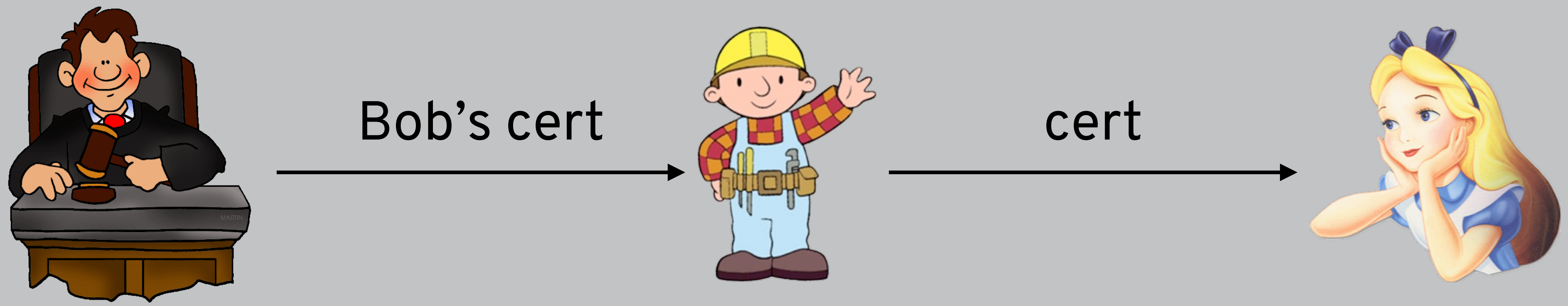
q: did we really provide a public-key infrastructure (PKI)?

a: yes, but we still need to distribute keys for CAs.

q: so we're really trusting those CAs, huh?

a: yes! but Certificate Transparency (CT) tries to reduce this trust.

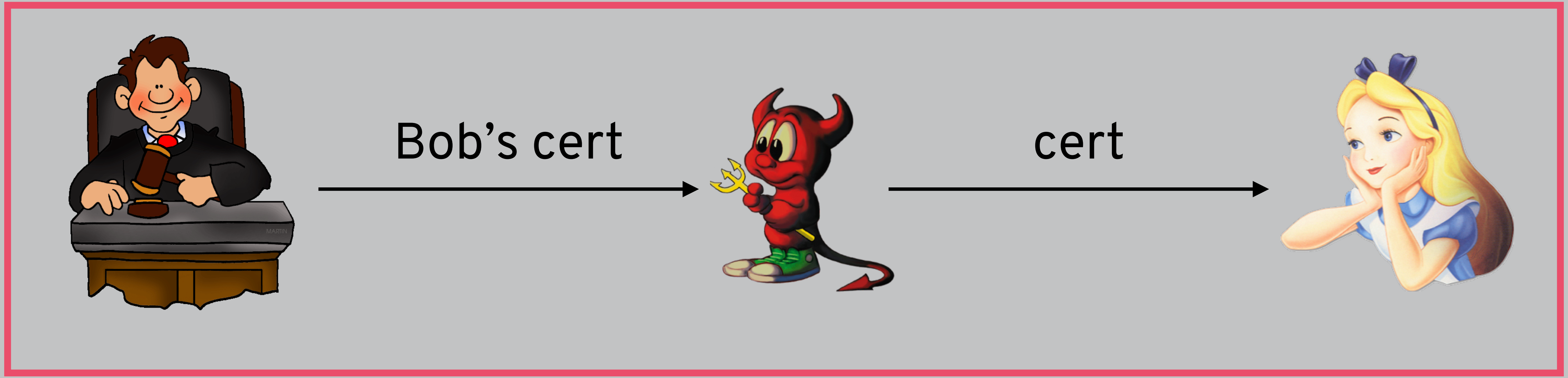
CERTIFICATE MISISSUANCE



Bob’s cert

cert

vs.

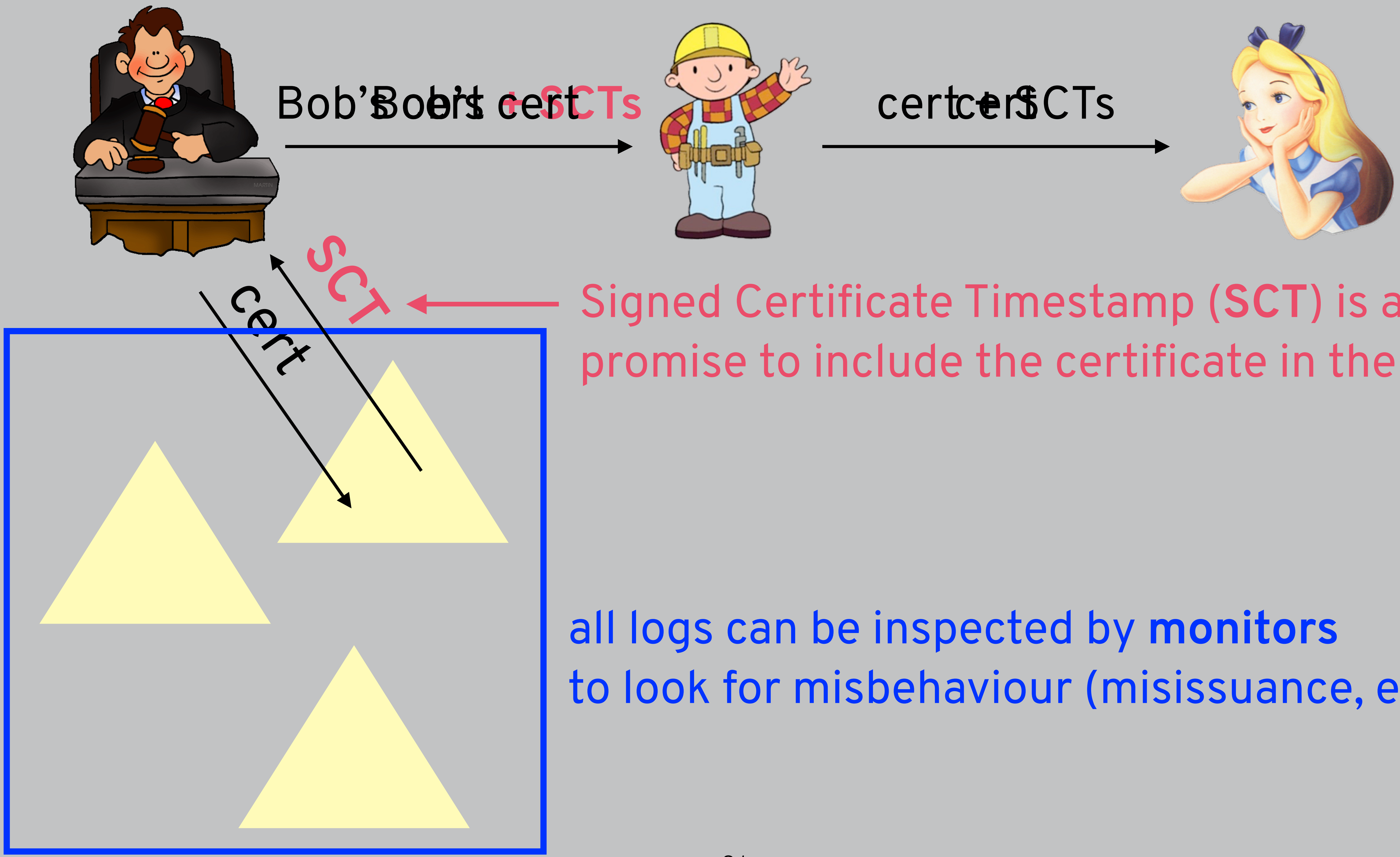


certificate misissuance

Bob’s cert

cert

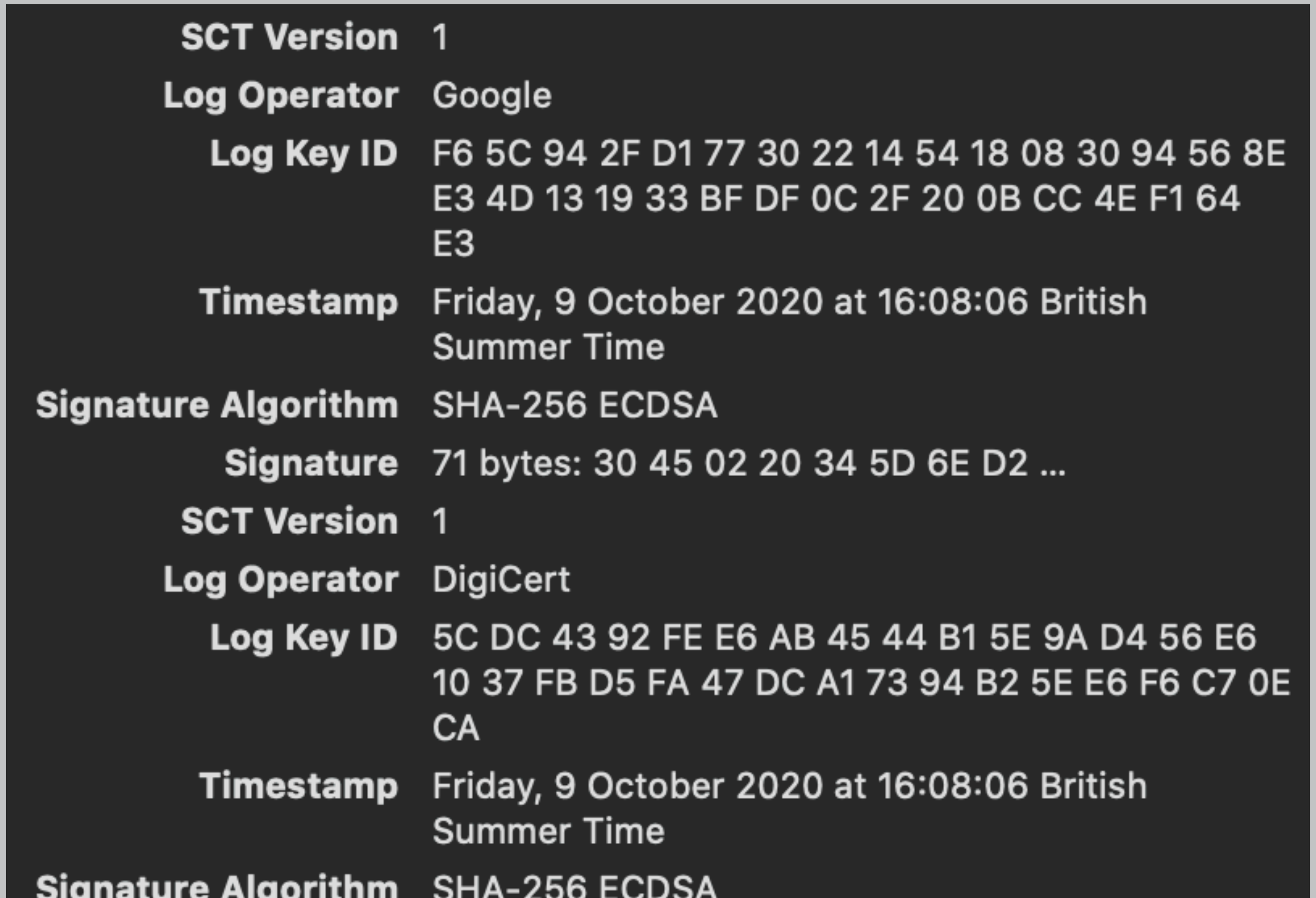
CERTIFICATE TRANSPARENCY



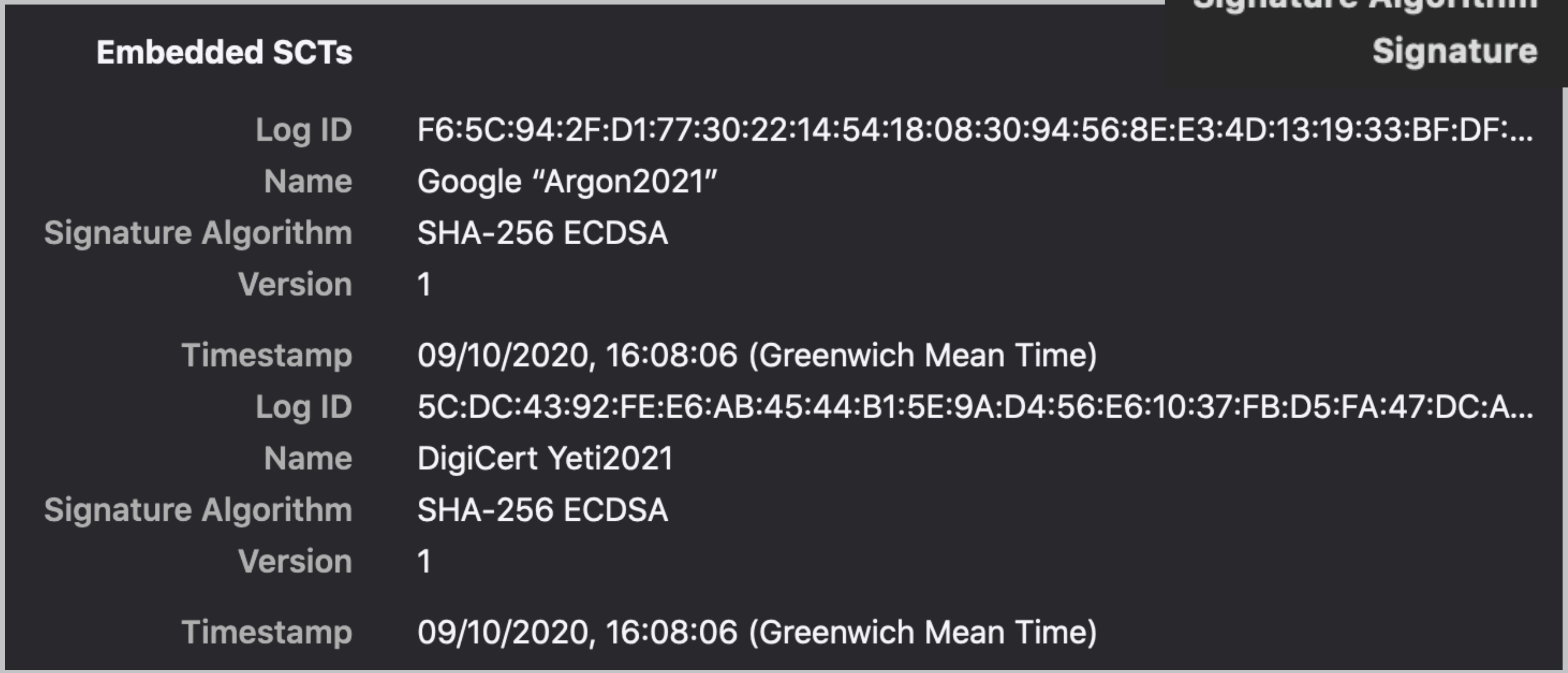
Signed Certificate Timestamp (**SCT**) is a promise to include the certificate in the log

all logs can be inspected by **monitors**
to look for misbehaviour (misissuance, etc.)

CERTIFICATE TRANSPARENCY DEMO



```
       SCT Version  1
      Log Operator  Google
        Log Key ID  F6 5C 94 2F D1 77 30 22 14 54 18 08 30 94 56 8E
                    E3 4D 13 19 33 BF DF 0C 2F 20 0B CC 4E F1 64
                    E3
         Timestamp  Friday, 9 October 2020 at 16:08:06 British
                    Summer Time
Signature Algorithm SHA-256 ECDSA
         Signature  71 bytes: 30 45 02 20 34 5D 6E D2 ...
       SCT Version  1
      Log Operator  DigiCert
        Log Key ID  5C DC 43 92 FE E6 AB 45 44 B1 5E 9A D4 56 E6
                    10 37 FB D5 FA 47 DC A1 73 94 B2 5E E6 F6 C7 0E
                    CA
         Timestamp  Friday, 9 October 2020 at 16:08:06 British
                    Summer Time
Signature Algorithm SHA-256 ECDSA
         Signature  71 bytes: 30 45 02 20 29 B7 04 F4 ...
```



```
Embedded SCTs

             Log ID  F6:5C:94:2F:D1:77:30:22:14:54:18:08:30:94:56:8E:E3:4D:13:19:33:BF:DF:...
               Name  Google “Argon2021”
Signature Algorithm  SHA-256 ECDSA
            Version  1

          Timestamp  09/10/2020, 16:08:06 (Greenwich Mean Time)
             Log ID  5C:DC:43:92:FE:E6:AB:45:44:B1:5E:9A:D4:56:E6:10:37:FB:D5:FA:47:DC:A...
               Name  DigiCert Yeti2021
Signature Algorithm  SHA-256 ECDSA
            Version  1

          Timestamp  09/10/2020, 16:08:06 (Greenwich Mean Time)
```

AS ALWAYS, SOME QUESTIONS…

q: did we really provide a public-key infrastructure (PKI)?

a: yes, but we still need to distribute keys for CAs.

q: so we're really trusting those CAs, huh?

a: yes! but Certificate Transparency (CT) tries to reduce this trust.

q: does the client authenticate itself to the server?

a: no! we'll see client authentication later on.

PUBLIC-KEY CRYPTOGRAPHY

secrecy without shared secrets

anyone can encrypt to Bob (or many other websites)

important in huge open environment like the Internet

integrity without key exchange

use digital signatures

small number of distributed keys

small key distribution

restricted to certificate authorities

(disadvantages? slow, uses strong assumptions)

QUIZ!

Please go to

`https://moodle.ucl.ac.uk/mod/quiz/view.php?id=2754465`

to take this week's quiz!