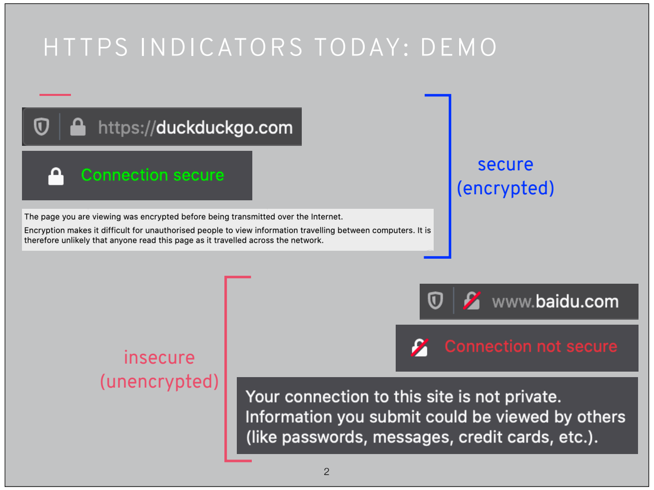
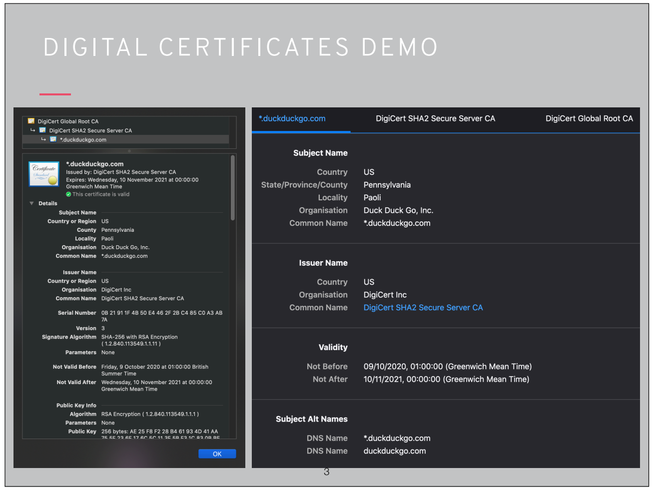
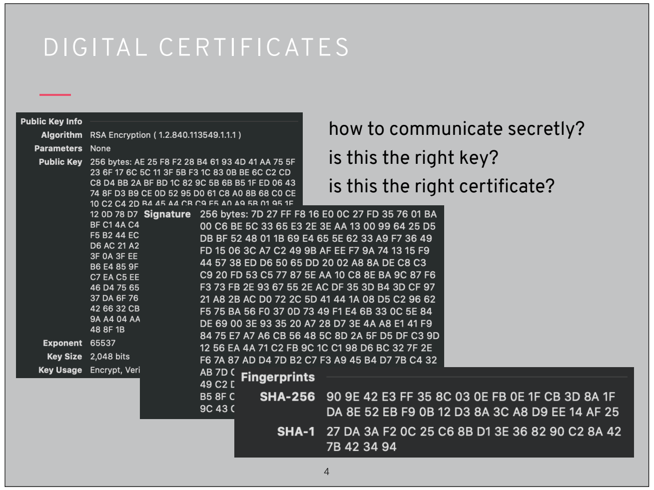


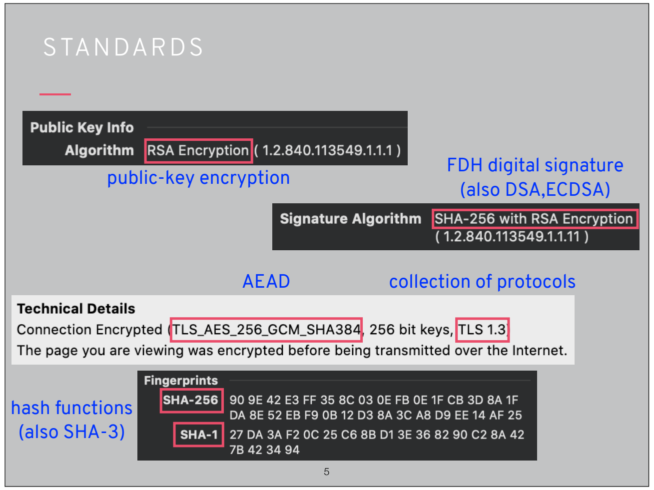
Let’s go back to the demo we did last week exploring HTTPS and go a bit deeper this time



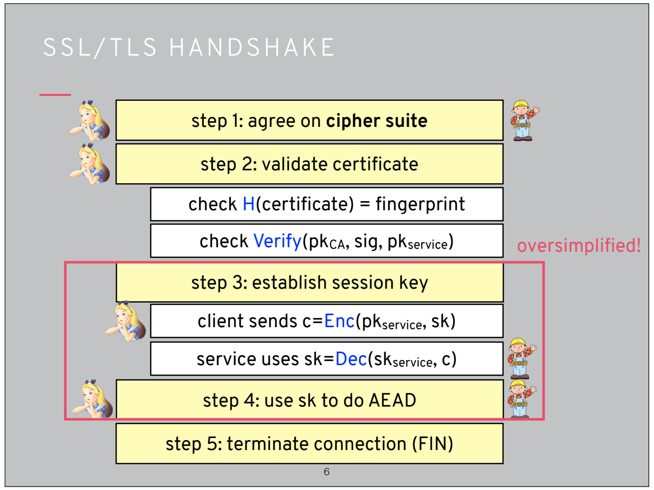
Example on the left is using Chrome, on the right using Firefox



Three key components of the certificate address three separate issues. Encryption provides private communication with website, signature provides guarantee that the encryption key is the right one, and fingerprints provide guarantee that the information in the certificate is correct



All of these things are associated with different cryptographic standards, these are the names to look out for and that we've already seen. TLS 1.3 was introduced in 2018 and replaces TLS 1.2



These are the steps you take to communicate securely with a website

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

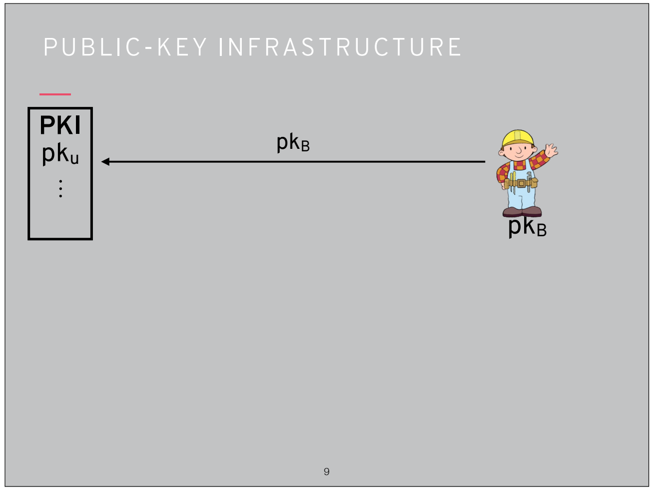
So this is what the padlock means, that we're running all of these steps in the background

---

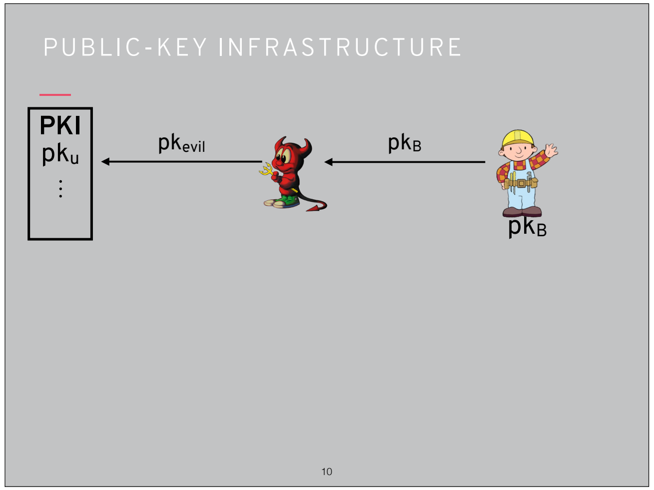
AS ALWAYS, SOME QUESTIONS…

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

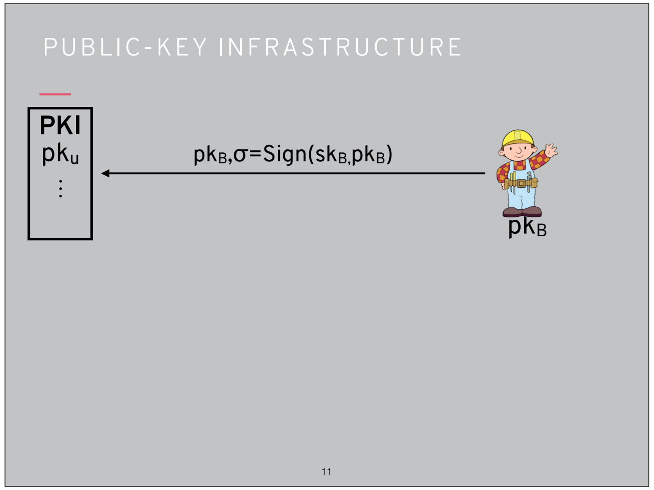
Let's revisit that magic box in the corner of our slides



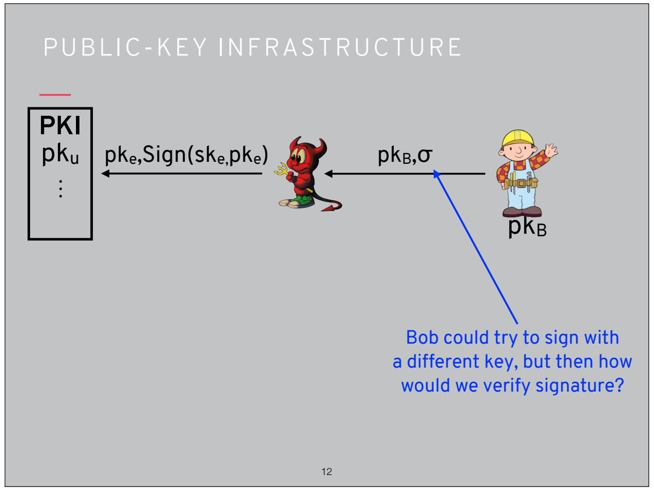
Most naive way to try to register a key is just to send it



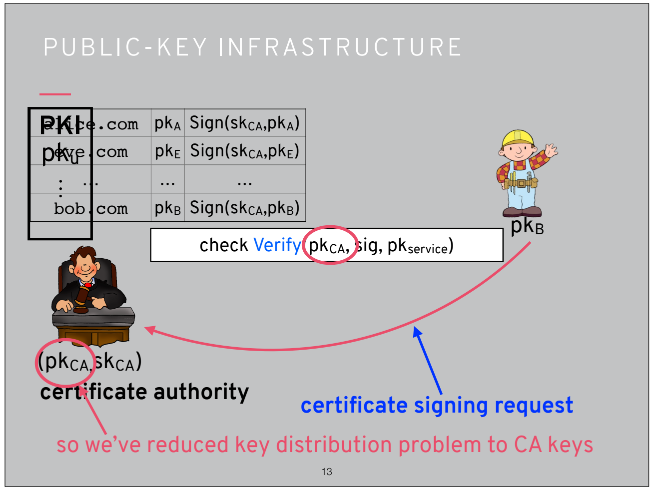
Obviously this can be subject to a MitM attack



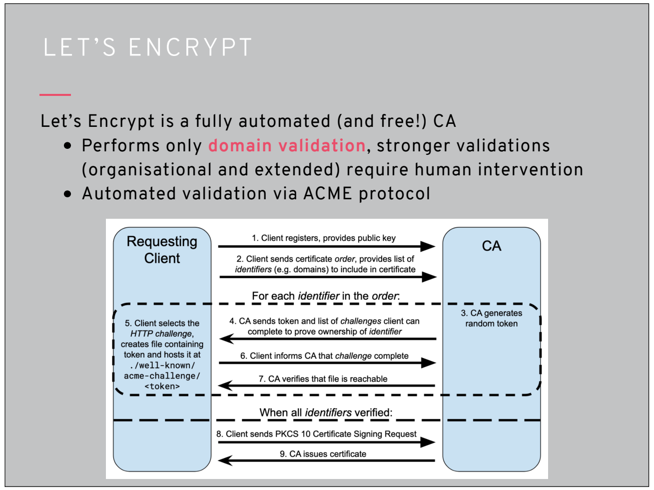
We saw a way to deal with MitM attacks, right? Just sign the data



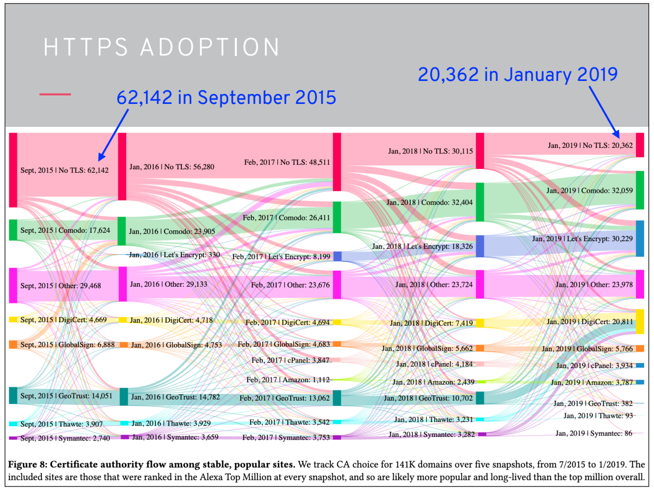
But of course then the adversary can just forge the signature since it's under the key that Bob is sending. We could try for a different key but then how would we know what key to verify under? This is a bit of a chicken-and-egg situation, and results in a self-signed certificate (which most browsers consider insecure)



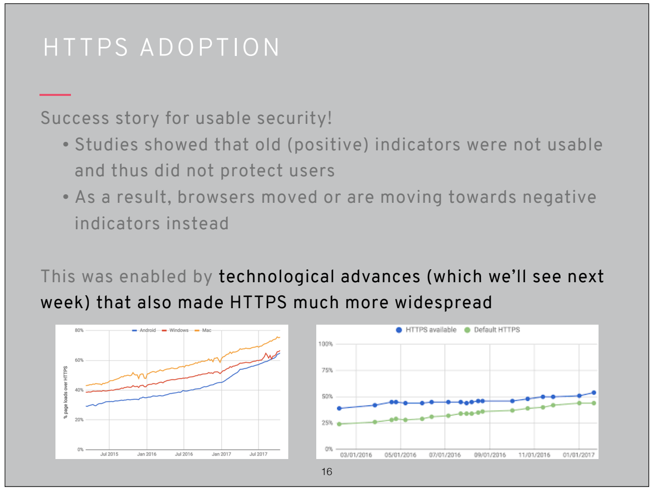
The solution is that we send the key out-of-band, in a formal application to a certificate authority (CA). They then check the request and add the key if they feel it's okay. We verify the signatures under their keys though, so still need to distribute their keys, but we've at least reduced the number of keys we need to distribute



These days, it is possible to get a certificate quickly and for free thanks to the Let's Encrypt (LE) CA (https://letsencrypt.org/), which operates in a completely automated way



See growth in LE certificates but also threefold drop in sites that don't use HTTPS / TLS at all



This is very much related to the increased adoption of HTTPS in recent years that we saw last week

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**

Certificate hierarchy is not actually flat (meaning CAs sign individual domains), instead root CAs sign certificates of intermediate CAs, and so on down to certificates for individual domains

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

Like we saw with DNS and other parts of the Internet, these root CAs are essentially hardcoded into our computers

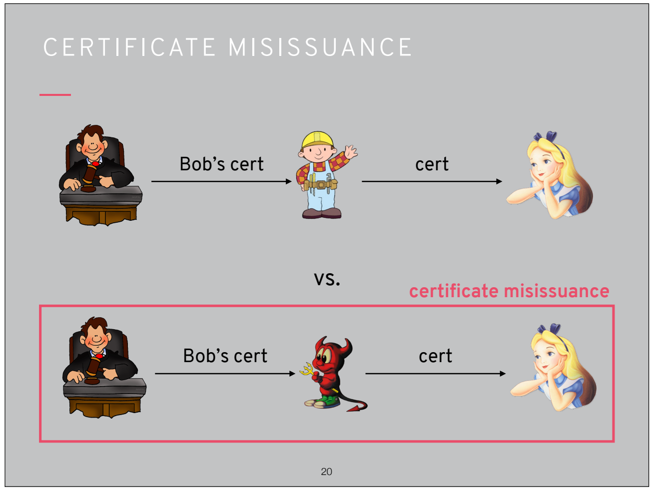
AS ALWAYS, SOME QUESTIONS…

q: did we really provide a public-key infrastructure (PKI)?
a: yes, but we still need to distribute keys for CAs.

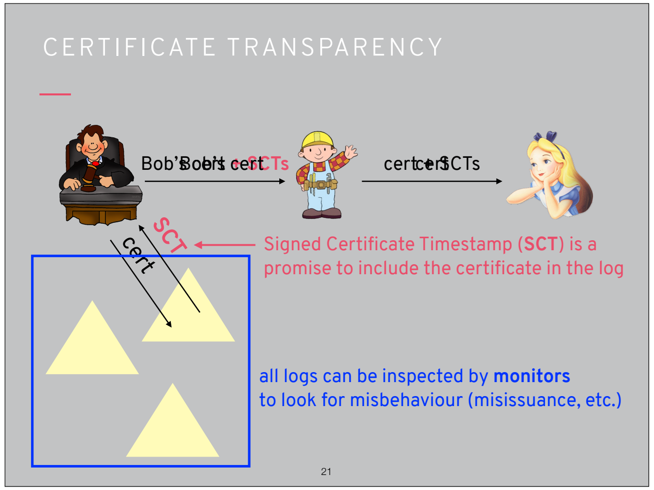
q: so we're really trusting those CAs, huh?
a: yes! but Certificate Transparency (CT) tries to reduce this trust.

19

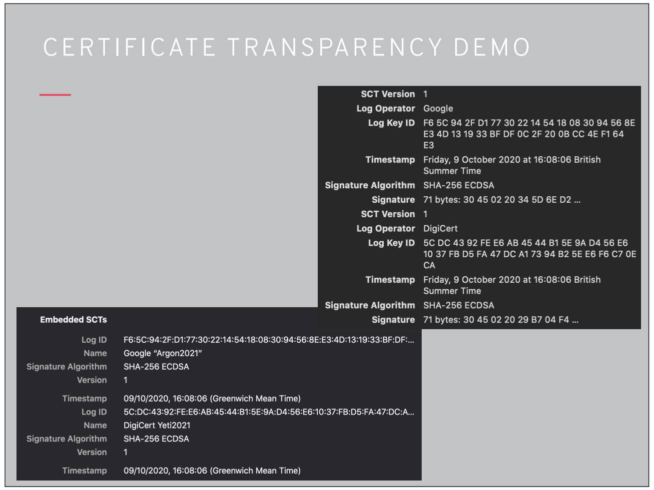
We do really trust CAs though to do those checks, and there are many examples of failures in this system (for example with DigiNotar and Comodo)



It can be very hard to detect certificate misissuance, which can have a serious impact, since the certificate will verify and end users cannot be relied upon to notice the difference



In Certificate Transparency (CT, https://certificate.transparency.dev/), all issued certificates are stored in globally visible logs, which can be continuously monitored to identify misbehaviour more quickly. This doesn't eliminate trust but it spreads it beyond the CA



The screenshot in the bottom left is from Firefox, and the one in the upper right is from Chrome

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.

Client authentication looks more like entering a username and password

---

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)

To summarise, public-key cryptography is a nice fit for the open nature of the Internet

QUIZ!

Please go to

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

to take this week's quiz!