Standard Security Protocols

Going up the stack:

PKI, DNS, TLS, IPsec and all that

Next hour(s)...

Read source RFC's for these Exam Material.

- PKI model/protocols and DNS
- SMIME formats and (a bit on) email
- **TLS protocol (TLS1.2 or TLS1.3)**
- IPsec
- Kerberos
- SSH
- Wireguard
- MLS

Knowing **one** of these in detail (and PKI and DNS) is enough for exam purposes – you **will** need to read the source materials

First bit...

- Yawning :-)
- Why standards?
- Standards Development Organisations (aka alphabet soup)
- Couple of examples
- When to standardise?

Why standardise?

- Generic reasons:
  - Multi-Vendor Interoperability
  - “Open”-ness
- Security Reasons
  - Remember: Crypto => Interop is hard!
  - Review: Many (non-security) standards decrease security
- NB: Not everything needs to be standardised!

Different provider products work together

↦ Standards make Crypto interop easier.

A Possibly New Why?

- Human Rights Protocol Considerations Research Group
  - https://irtf.org/hrpc
- “Rethinking Privacy Online and Human Rights: The Internet’s Standardisation Bodies as the Guardians of Privacy Online in the Face of Mass Surveillance”
  - Adamantia Rachovitsa, Conference Paper No.5/2016 2016 ESIL Research Forum, Istanbul, 21-22 April 2016 (see materials page for copy and link to original)

Standardization helps protect human rights (Apple data case)

Standards Devlopment Organisations (SDOs) and related

- International organisations
  - UN/ITU, ISO
- "Open" Internet Standards Development Organisations
  - IEEE, IETF, W3C
- Commercial Enterprise Driven SDOs
  - OASIS, FIDO Alliance
- Company-specific pet projects
  - FB Free basics/internet.org
- Open-source projects
  - Apache, WHATWG, OpenSSL, …
- Open-source commercial alliances
  - OpenStack
- Operational entities:
  - ICANN, RIPE, ARIN,…
- Topic specific alliances
  - M3AAWG, Lora alliance, …

Types of standards

SDOs produce different categories of output, not all standards, e.g. RFC1149

- Draft versions may be published or not, long-lived or not, rubbish or not

Almost all interesting standards in this space are openly available - some for a fee!

But note: All SDOs have some business model, even the best/most-open ones

ISO/ITU-T

- National bodies are members (NIST, Enterprise-Ireland)
  - https://www.iso.org
  - https://www.itu.int
- Occasional ITU-T attempts to take over the Internet;-)
- Security stuff:
  - Cryptographic mechanisms (ISO)
  - Even more X.509 (PKI stuff ITU-T, mostly useless/irritating)
  - Various ITU telephony specs
    - Some inheriting from/profiling IETF

Internet Engineering Task Force (IETF)

- No members: (in theory) a group of individuals developing the Internet

https://www.ietf.org https://www.irtf.org

- Security:
  - About 1-2 dozen of about 100 working groups usually doing interesting security stuff
- Best of a bad lot really!
  - But I would say that having been involved with this lot for >25 years;-)

World-Wide-Web Consortium (W3C)

- Membership organisation ($5-50k per annum) plus “team” plus invited experts

https://www.w3.org

- WebRTC
- Focus more on browser APIs and not protocols
- WGs: WebRTC, Federated Identity, Privacy, Web authentication…
- See also: WHATWG https://whatwg.org/

IEEE Standards Association

- Individual memberships but meeting attendance counts

https://www.ieee.org/

- Security:
  - IEEE 802 – various security things, WPA etc.
  - MAC address randomisation,
  - MACsec (layer 2 crypto)

Examples of “standards”

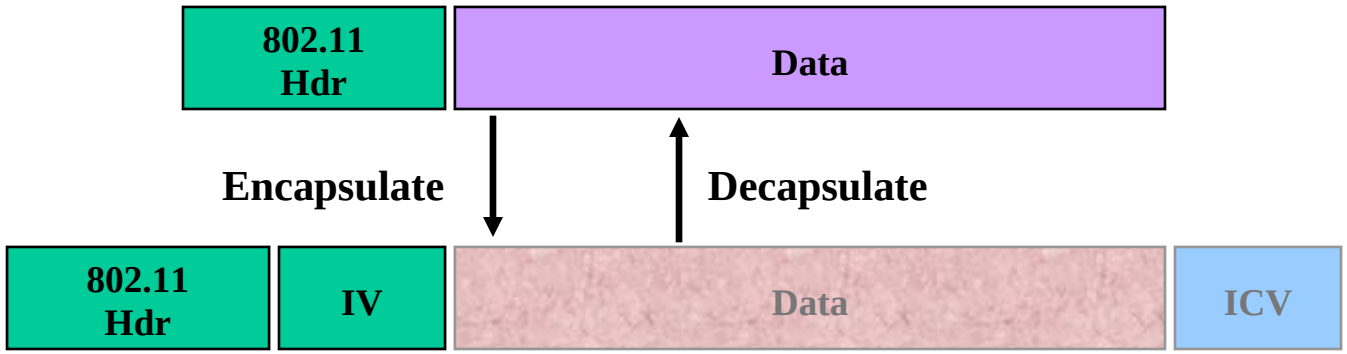
- Really good: RFC822/2822/5322 – Mail message format
- Middling: RFC3185 “Re-use of CMS Content Encryption Keys”
  - (Ahem!) Co-author present :-)
  - Length: 10pp (-crud=3pp)
  - Duration: ~18 months
  - Purpose: Fix a problem for RADIUS/Diameter
  - BUT: Zero implementations

“Bad” examples

- Many to choose from
- IETF: IKEv1 (see later) → good to know why its bad.
- IEEE: WEP (or was it?)
  - https://www.ieee802.org/11/Documents/DocumentArchives/1994_docs/1194249_scan.pdf version 1.0 November 1994

WEP Encapsulation

Wired Equivalent Privacy



Goal: make wifi as good as ethernet packets across a wire

WEP Encapsulation Summary:

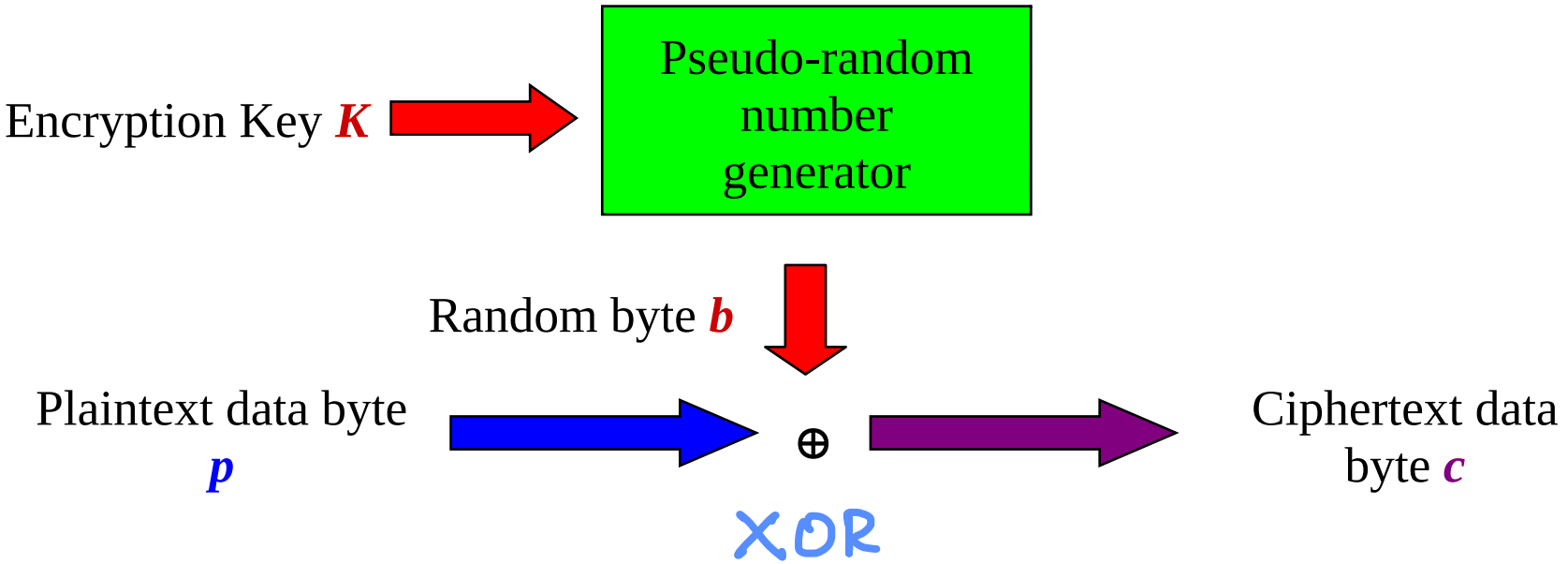
- Encryption Algorithm = RC4
- Per-packet encryption key = 24-bit IV concatenated to a pre-shared key
- WEP allows IV to be reused with any frame
- Data integrity provided by CRC-32 of the plaintext data (the "ICV")
- Data and ICV are encrypted under the per-packet encryption key

too short

It's broken

Properties of Vernam Ciphers (1)

The WEP encryption algorithm RC4 is a Vernam Cipher:



Decryption works the same way: $p = c \oplus b$

Properties of Vernam Ciphers (2)

Thought experiment 1: what happens when p'_1 and p'_2 are encrypted under the same "random" byte b?

$$c'_1 = p'_1 \oplus b \qquad\qquad c_2 = p_2 \oplus b$$

Then:

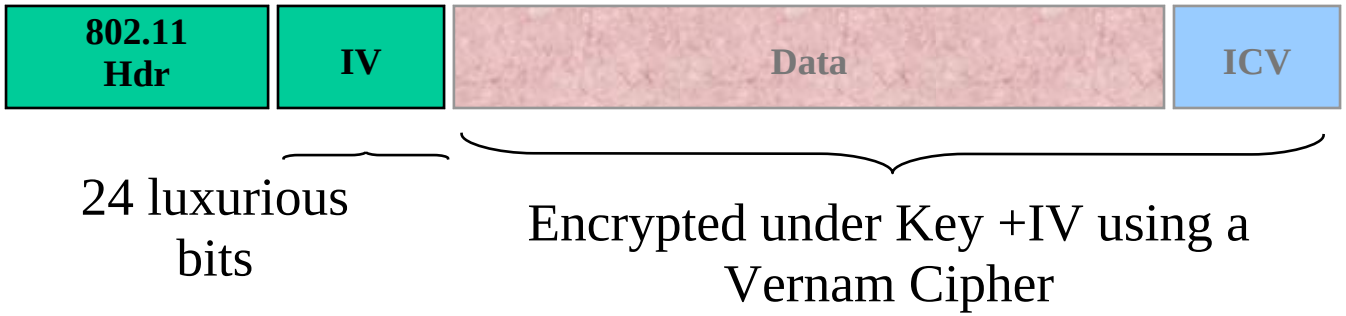
$$c'_1 \oplus c_2 = (p'_1 \oplus b) \oplus (p_2 \oplus b) = p'_1 \oplus p_2$$

Conclusion: it is a very bad idea to encrypt any two bytes of data using the same byte output by a Vernam Cipher PRNG.

Ever.

Immediately broken

How to Read WEP Encrypted Traffic (1)



- By the Birthday Paradox, probability P_n two packets will share same IV after n packets is $P_2 = 1/2^{24}$ after two frames and

$$P_n = P_{n-1} + (n-1)(1-P_{n-1})/\ 2^{24} \text{ for } n > 2.$$

- 50% chance of a collision exists already after only 4823 packets!!!
- Pattern recognition can disentangle the XOR'd recovered plaintext.
- Recovered ICV can tell you when you've disentangled plaintext correctly.
- After only a few hours of observation, you can recover all 2^{24} key streams.

How to Read WEP Encrypted Traffic (2)

- Accelerate the process!
- Send spam into the network: no pattern recognition required!
- Get the victim to send e-mail to you
  - The AP creates the plaintext for you!
- Decrypt packets from one Station to another via an Access Point
  - If you know the plaintext on one leg of the journey, you can recover the key stream immediately on the other
- Etc., etc., etc.

You know the plaintext

Question

- WEP was broken after being deployed, and then fixed. Same is true of SSL which became TLS.
- Was that better or worse than IPsec/IKE that took 10 years to develop?

Make a mistake and expedite development to fix it?

Or

Develop it correctly but it takes too long?

When to standardise

- When many implementer's codebases have to talk a protocol
  - HTTP, SMTP,... (many, many examples)
- When one implementer has to use another vendor's API
  - WebRTC, PKCS#11
- When serious review is required
  - Routing (BGP) or TCP changes like RED, AQM
  - Crypto algs, e.g. AES, PQ algs

When not to…

- When you just want your name in “lights”
- When your clever algorithm is the tenth way to do the job
- When your scheme is patented
  - Or secretly about to be patented!
- When no-one cares
- See RFC6417 for guidance for researchers

More on when not to…

- If you're an open-source team and don't have the cycles to engage with all the axe-grinders who'll get involved when you engage in a really open process (and they will) – e.g. Tor
- If you claim that implementation agility and speed is more important than multi-implementer interop - e.g. Signal, maybe wireguard

Tor probably shouldn't be standard

Standard Security Structures and Protocols

PKI, S/MIME, SSL, Kerberos, IPsec, SSH, Wireguard

Basically, things that do automated key management and secure application data transport

Materials

- Lots of RFCs
- https://datatracker.ietf.org/wg/
- Bleichenbacher/"Avoiding the million-message attack" RFC 3218
  - Good one for those who want to read ahead…

Next hour(s)…

- PKI model/protocols and DNS
- SMIME formats and (a bit on) email
- **TLS protocol (TLS1.2 or TLS1.3)**
- IPsec
- Kerberos
- SSH
- Wireguard
- MLS

Knowing **one** of these in detail (and PKI and DNS) is enough for exam purposes – you **will** need to read the source materials

Public Key Infrastructure (PKI)

Public Key Infrastructure (PKI)

- Key management is a scaling problem
- Possible to push “trust” (bad term!) up to a certification authority (CA)
- “Trust” here is: CA is responsible for binding information about an entity (esp. names) with a public key
  - Anyone who trusts that CA for that purpose can then check that entity's signature or encrypt to it

public key certification

"Trust": who is trusting who for what.

Bad to use word "trust" without this. ↑

PKI History

- Original concept: publishing public keys in newspapers
- Electronic equivalent suggested in 1978 undergrad work:
  - Kohnfelder, ”Toward a practical public key cryptosystem,” Bachelors Thesis, MIT Dept. of Electrical Engineering, May 1978 https://dspace.mit.edu/handle/1721.1/15993
- First standard was X.509 in 1988
- IETF X.509 profile is in **RFC5280** from 2008, used by TLS, S/MIME and lots of other protocols and libraries

Everyone hates it →

X.509-based PKI Problems

- Everyone sensible has disliked X.509 since about the late 1990's
  - It's old, gnarly & horrible
- Every now and then someone suggests replacing it with <foo>
  - Sadly, so far, no <foo> has been sufficiently better to ***displace*** X.509 based PKI

Nothing good enough to swap it out

- Maybe in 5-10 years it'll be less important, but for now we have to suffer
  - I said the above 5-10 years ago too;-(

Certificates (1)

```
Certificate  ::=  SEQUENCE  {
 tbsCertificate       TBSCertificate,
 signatureAlgorithm   AlgorithmIdentifier,
 signatureValue       BIT STRING  }
```

RSA SHA256

- Who knows what ASN.1 is?
  - An **A**bstract **S**yntax **N**otation
  - With tag, length value encoding schemes (BER, DER, PER)
    - SEQUENCE -> 0x30, INTEGER -> 0x02
  - PITA, as are all marshalling schemes in the end

Certificates (2)

```
TBSCertificate  ::=  SEQUENCE  {
version         [0]  EXPLICIT Version DEFAULT v1,
serialNumber         CertificateSerialNumber,
signature            AlgorithmIdentifier,
issuer               Name,
validity             Validity,
subject              Name,
subjectPublicKeyInfo SubjectPublicKeyInfo,
issuerUniqueID [1]IMPLICIT UniqueIdentifier OPTIONAL,
subjectUniqueID[2]IMPLICIT UniqueIdentifier OPTIONAL,
          -- If present, version MUST be v2 or v3
extensions      [3]  EXPLICIT Extensions OPTIONAL
               -- If present, version MUST be v3
}
```

3 →

Unique combo

Amazon.com

isA CA == No (to tell CAs from end points with certificates)

Certificate Revocation Lists

- A list of "bad" certificate serial numbers (plus list- and entry-extensions)
- Periodically issued by a CA
- Revocation = putting on block-list
- Revocation information can also be fetched via on-line certificate status protocol (OCSP)
- OCSP-responses can be stapled
- Reasons to revoke: Key compromise, Key loss, Change of function, Uninstall web server...

List of certificates to ignore / not trust

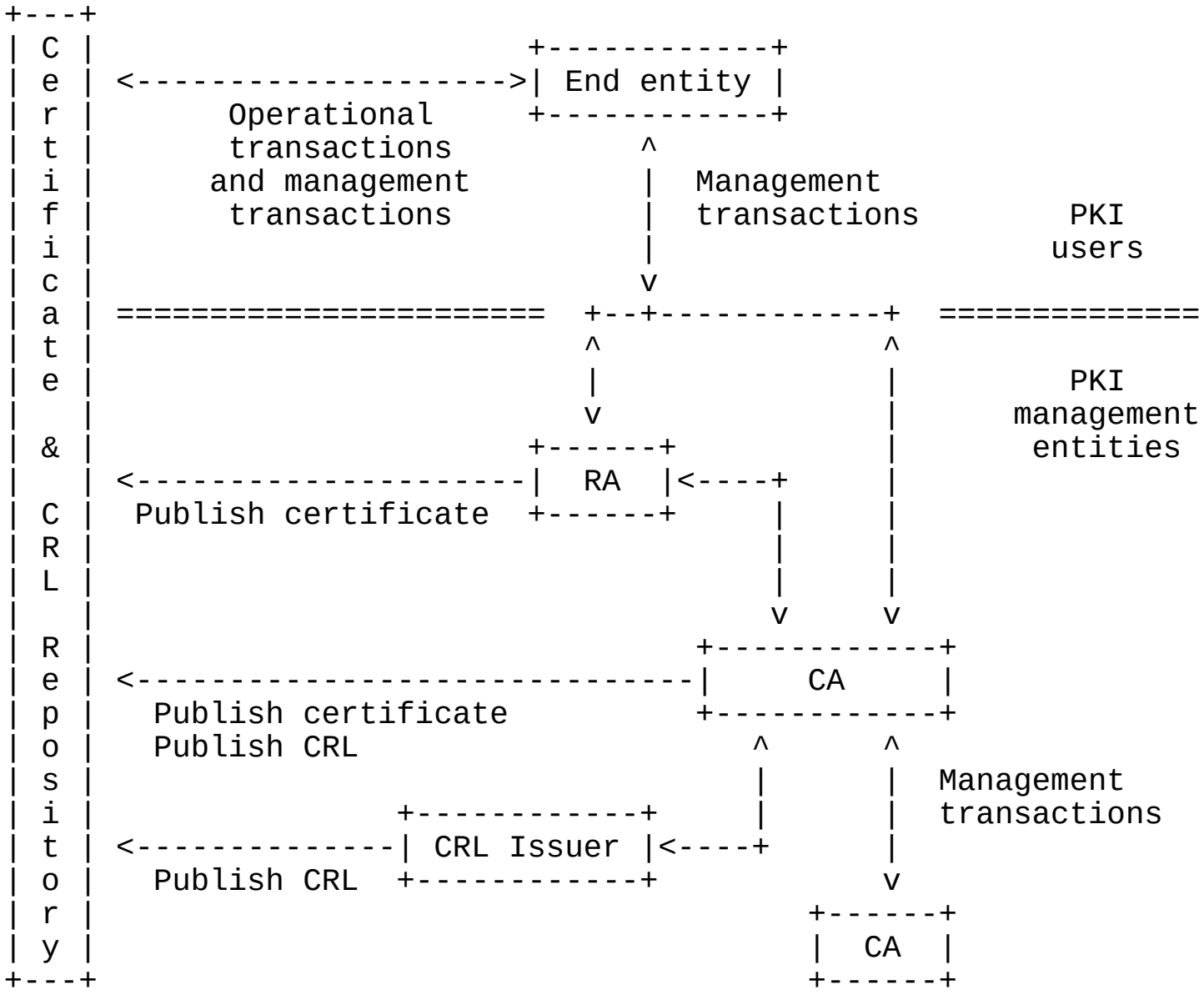
OSCP can cause privacy leaks (CA knows you're going to OCSP after visiting website)

To check a certificate

- Start with a (set of) locally trusted CA public keys
- Progress down certification path:
  - Is next-signature ok with previous public?
  - Is certifcate revoked?
  - Continue
- Missing lots (see rfc5280)
  - E.g. Policy mappings (yuk), but do ignore those!

don't need heavy detail but need details

PKI Entities



PKI Protocols

Certificate Management Protocol

- Registration & renewal: PKCS#10, proprietary, CMP, CMC, EST and now **ACME**
  - CMP (RFC4210) still used by 3gpp, and for trains!
- Certificate retrieval: **in-band,** LDAP, DAP, HTTP, FTP, DPD
- Certificate status checking: **OCSP**, DPV, CRL processing, and now (maybe) application specifics like CRLite
  - https://blog.mozilla.org/security/2020/01/21/crlite-part-3-speeding-up-secure-browsing/
  - Browsers perhaps only caring about CA and popular-domain revocations these days, not sure
- Certificate Transparency (**CT**) for logging certificate issuance has been a success

CMP is poorly designed

Roots/Trust Points

- Applications using PKI need to have a local set of (root) CA public keys they “trust”
- Browsers and OSes have those – each with hundreds of CA organisations in the list
- In/ex-clusion is highly political CA browser forum
  - https://cabforum.org/ is a venue for some of that
  - Mozilla operate a public discussion list (other tend to be less public) https://groups.google.com/a/mozilla.org/g/dev-security-policy

• The WebPKI is a special case today – biggest and most important PKI but quite a few others do exist

  - Other applications and OSes handle things similarly, often with some overlap with WebPKI

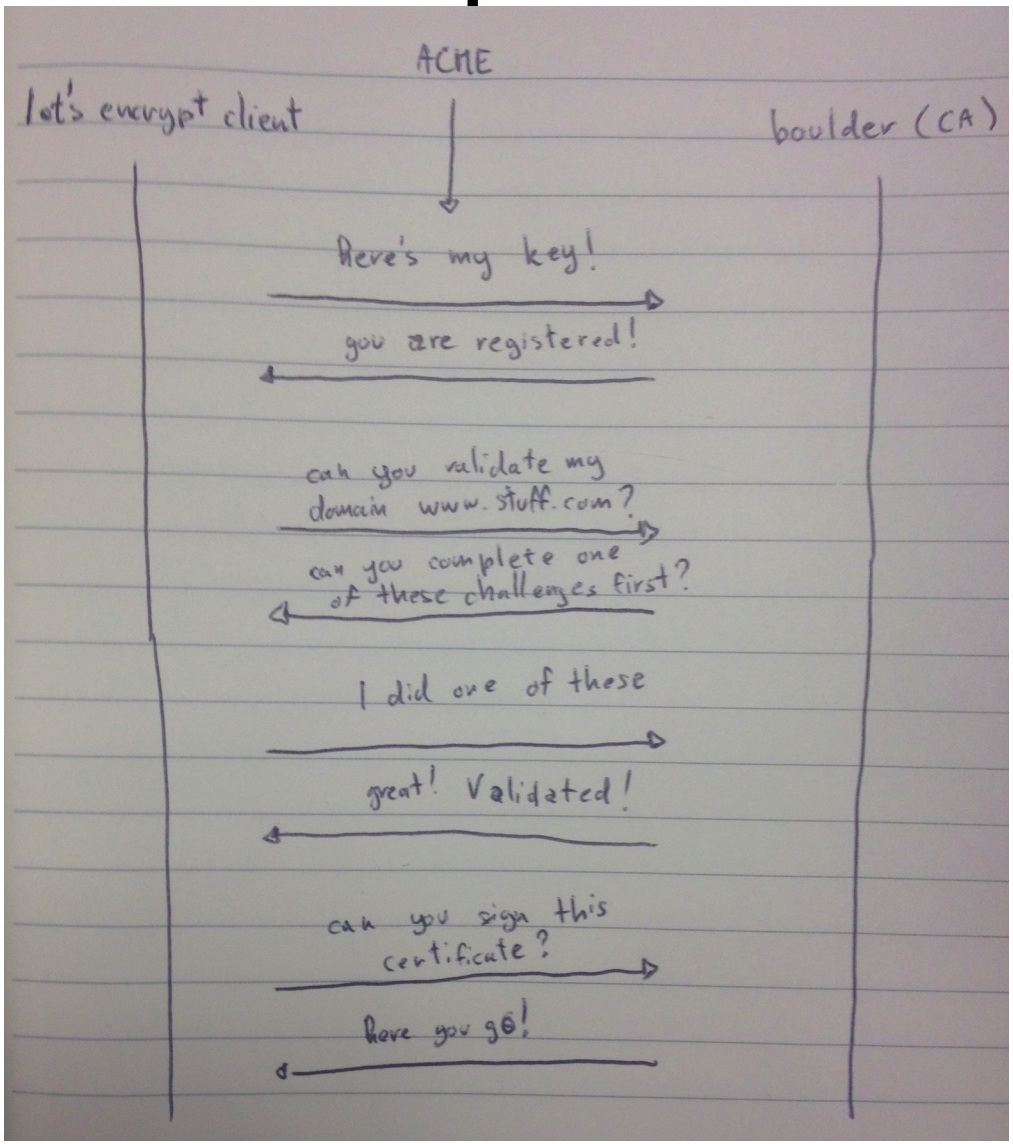
Some good news though...

- https://letsencrypt.org/ operating a free CA since Dec 2015
- ACME protocol for automated certificate management
  - JSON based
  - RFC8555

ACME protocol

- Automated Certificate Management Environment (ACME) is an HTTP-based protocol for managing certificates
  - Works with LE and other CAs for acquiring Domain Validated (DV) certificates
- Quite a few client implementations:
  - certbot, ACME.sh, …
- Typically: command line registration then (~weekly) cron job for renewal

ACME protocol



https://cryptologie.net/article/274/lets-encrypt-overview/

ACME protocol

- ACME challenges:
  - Evidence for domain validation
  - There's a history of these turning out tricky in some deployment scenarios
- Challenge types:
  - http-01: random content at e.g. "/.well-known/acme-challenges/abcdef0123456"
  - dns-01: As above but in DNS
  - tls-sni-01/02: deprecated! Demonstrate ability to use random TLS SNI value

Certificate Transparency (CT)

- There have been cases where CA's have been hacked or misbehave (more later)
- CT improves the WebPKI
  - CT specified in (non-standard) RFC6962 in 2013
  - Standardisation process hugely slooooow.... V2.0 is defined in RFC9162 (2021)
  - I suspect everyone may stick with RFC6962
- Idea: append-only public logs of all certificates issued to allow detection of mis-issuance (not prevention, detection!)
- Nice (if not quite reliable) search UI: https://crt.sh/
  - Reminder to self: pop up this URL and show stuff:
    - https://crt.sh/?q=tcd.ie
- Also of interest: https://ct.cloudflare.com/

Attribute certificates

- RFC5755 - Certificate-like thing with generic attributes (e.g. group membership) and no subjectPublicKey
- Treat as experimental if you ever hear about them
- Can be used for role-based access control etc.
- BUT… when these seem useful there's almost always a better way!

PQ certificates

- New PQ algs will require public keys in certificates, which is fine (e.g. Dilithium, Kyber public keys)
- Some people are proposing ways to embed two public keys in one certificate to support use of hybrid protocols
  - That could end up a disaster, but we'll see
- Even if nothing silly happens there are transition challenges
- A bunch of documents are visible at: https://datatracker.ietf.org/wg/lamps/documents/

X.509-based PKI Summary

- Mix of mature and new(ish) technology
  - Plenty of open-source, products and services and significant deployment experience
- Deployment and application integration problems will always exist
  - Can be overcome but expensive
- Still technology-of-choice for scalable public key management
- Improvement is possible: e.g. ACME/CT
- Might be affected by post-quantum crypto
  - Or, hopefully PQC will tip us into finally moving to something better (but that hope is fading…)

The Domain Name System (DNS)

Domain Name System (DNS)

- Internet Assigned Numbers Authority (IANA, https://iana.org/) keeps lists of "top level domain names" (TLDS, e.g. .com, .ie), IP address allocations and protocol numbering registrations
  - That's a fantastic bit of bookkeeping but no more than that, policies are set elsewhere despite what some "Internet Governance" folks might say
  - IANA is homed in ICANN which is one of the policy setting/operations entities
  - The Regional Internet Regisgtries (RIRs) are another, and the IETF controls the protocols that create the space in which the policies operate
- The DNS is structured as a tree with a single root, below which we have .com, .ie, etc. and below those we have example.com, tcd.ie etc. and within tcd.ie we have whatever e.g. TCD might want e.g. down.dsg.cs.tcd.ie
- There are a set of (13 logical, physically: ~1000) DNS root servers on the Internet that serve as the DNS "root zone" and who can tell you set of IP addresses where you can find more authoritative information about .com or .ie. or any of the Top Level Domains (TLDs)
  - Entities that do the bookkeeping for a TLD are called registries

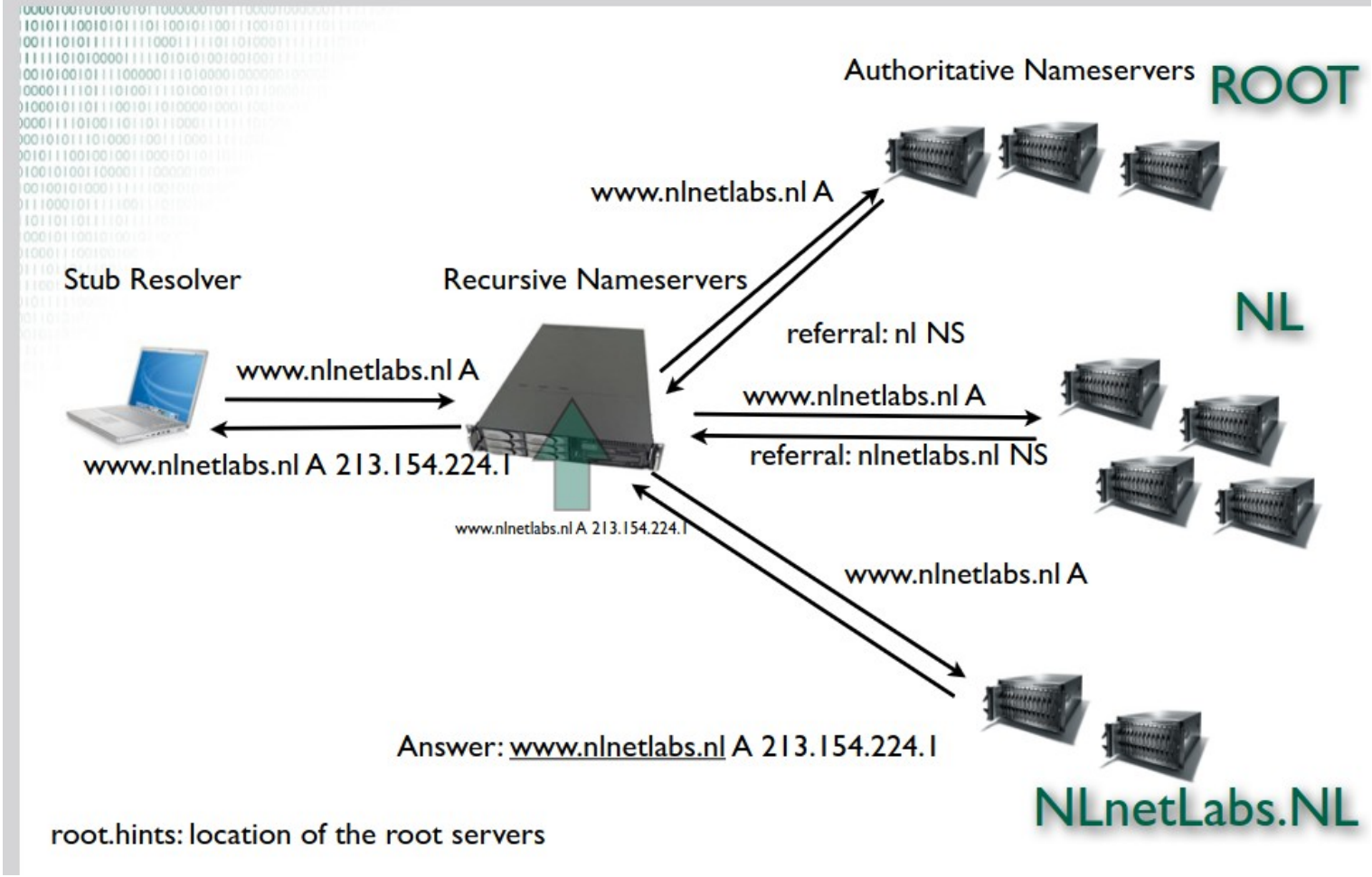
Domain Name System (DNS)

- There are 1591 TLDs, all recent ones being outrageously expensive, about 316 of those are country-code TLDs (ccTLDs), e.g. for “.ie” or “.de”
  - https://www.iana.org/domains/root/db
- DNS supports internationalised domain names (IDNs) because the world has many languages and writing systems (the Internet started out in ASCII)
  - ουτοπία.δπθ.gr is a (no longer functioning) Greek IDN https://en.wikipedia.org/wiki/Internationalized_domain_name
    - another representation for that is (in Punycode) is xn—kxae4bafwg.xn--pxaix.gr)
  - You may see IDNs listed in punycode as XN--<<stuff>>
  - IDNs create lots of confusion possibilities! Similar looking characters in names
- DNS authoritative servers store the canonical information for some “zone” (e.g. all the hosts in/below tcd.ie) but can also delegate to another server as happens in the case of cs.tcd.ie
- When you want a new name (e.g. jell.ie) you have to go to a registrar who works with the relevant registry (e.g. Tolerant Networks Limited is me-as-a-registrar for .ie) and then pay to rent that for a few years (IEDR, rebranded as https://weare.ie/ is the name of the Irish ccTLD registry)
  - Not all names are available – could be taken or a trade mark – some lawyers love this stuff!

Domain Name System (DNS)

- You need a server machine with an IP address for a DNS name to be useful
  - You often get from a hosting company or cloudy service provider like AWS or Azure or whomever
- And then that machine's name and IP address need to be published in the DNS of the parent zone
- Then you can e.g. install nginx (a web server) and make your web site and interact with e.g. LetsEncrypt.org to get a public key certificate for your DNS name so TLS to your server will work
  - Then browsers can nicely visit your web site
- Web crawlers and attackers of all sorts will also constantly ping your server, all the time
- At that point you may decide to be an advertiser or not, if you do, you'll probably start to record things about people who visit you and maybe you'll sign up to some advertising platform to make money for you and them
- But you might also decide not to track anyone (what I do)
  - Modulo normal web logs!

A DNS Query



Recursively searches for the name by starting with the end of the name and working up to the full name.

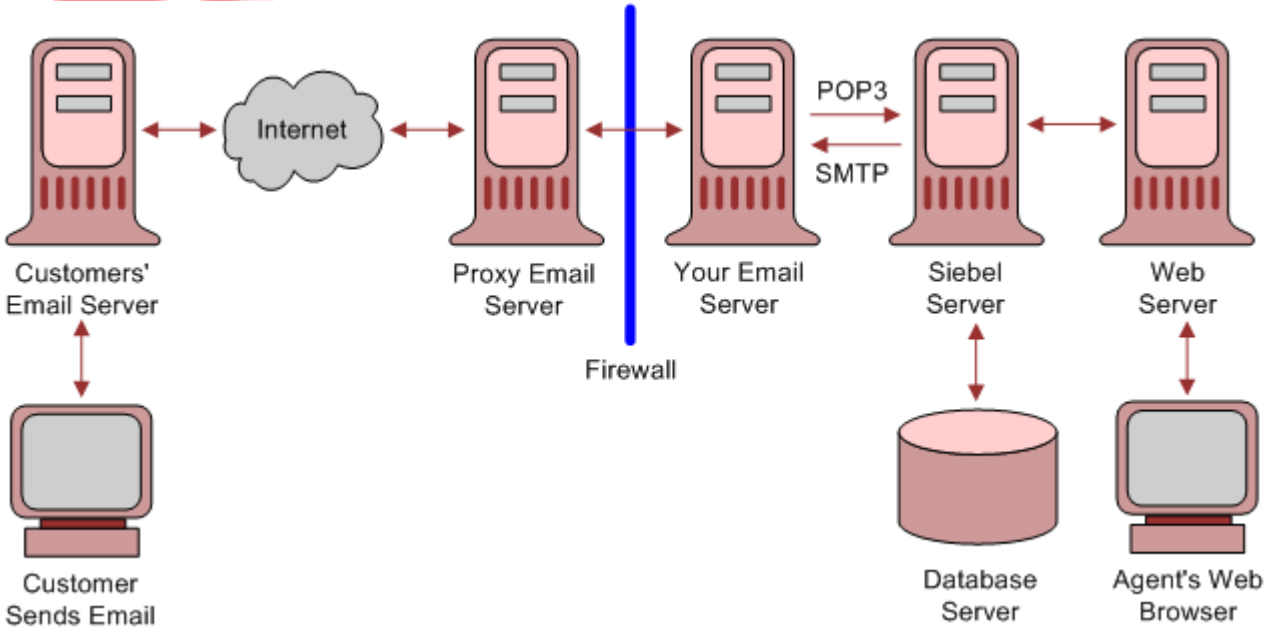
Domain Name System (DNS)

- We'll return to the DNS later, esp for DNSSEC, and to discuss HTTPS, SVCB and (maybe) DELEG RRs
- For now though, note that the webPKI depends on the DNS
  - Most certificates provide "domain validation" and no more
  - If I can bork the DNS, I can get a webPKI certificate for names I don't really control

e-Mail, spam, S/MIME and PGP

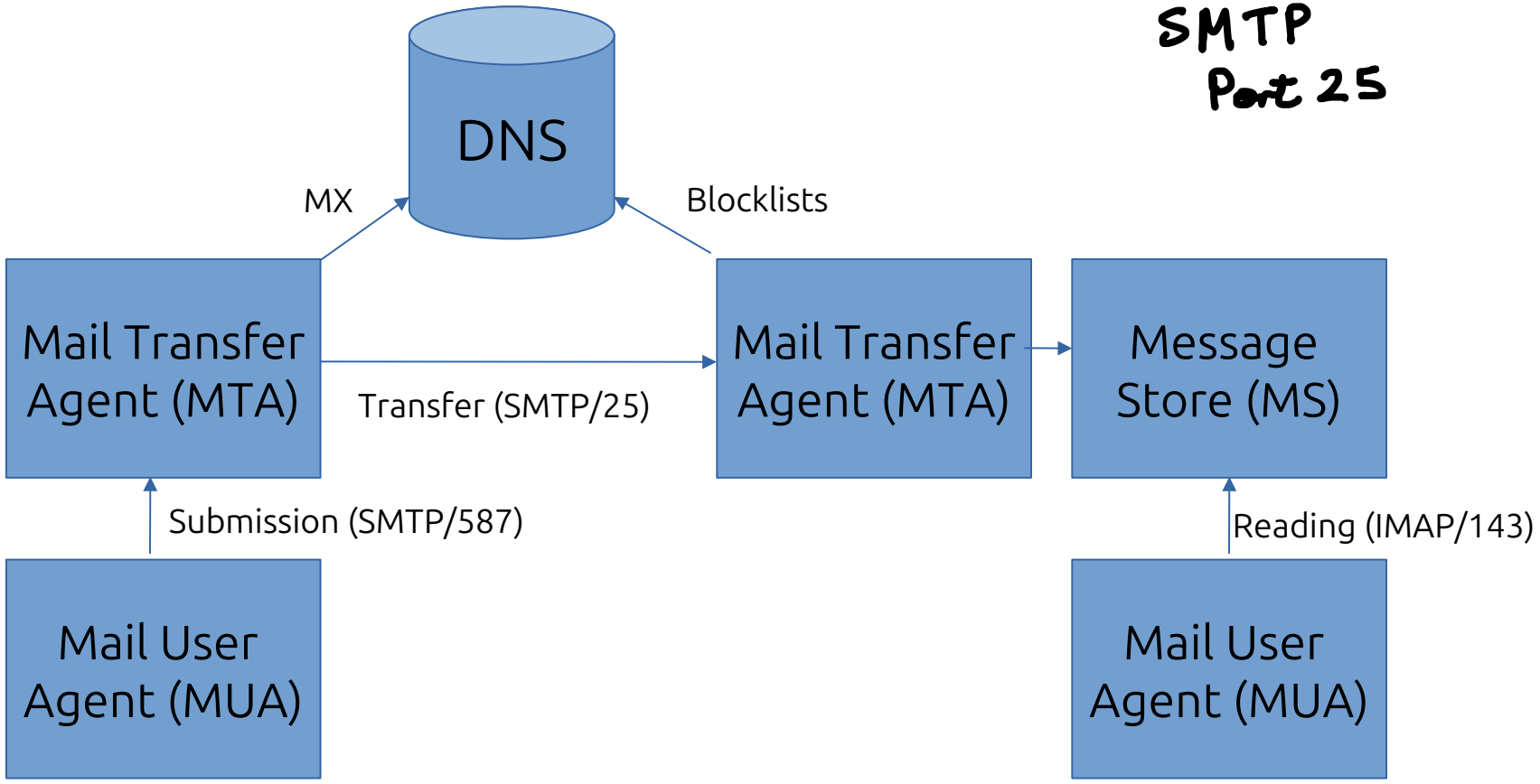
Email Architecture

- See RFC 5598



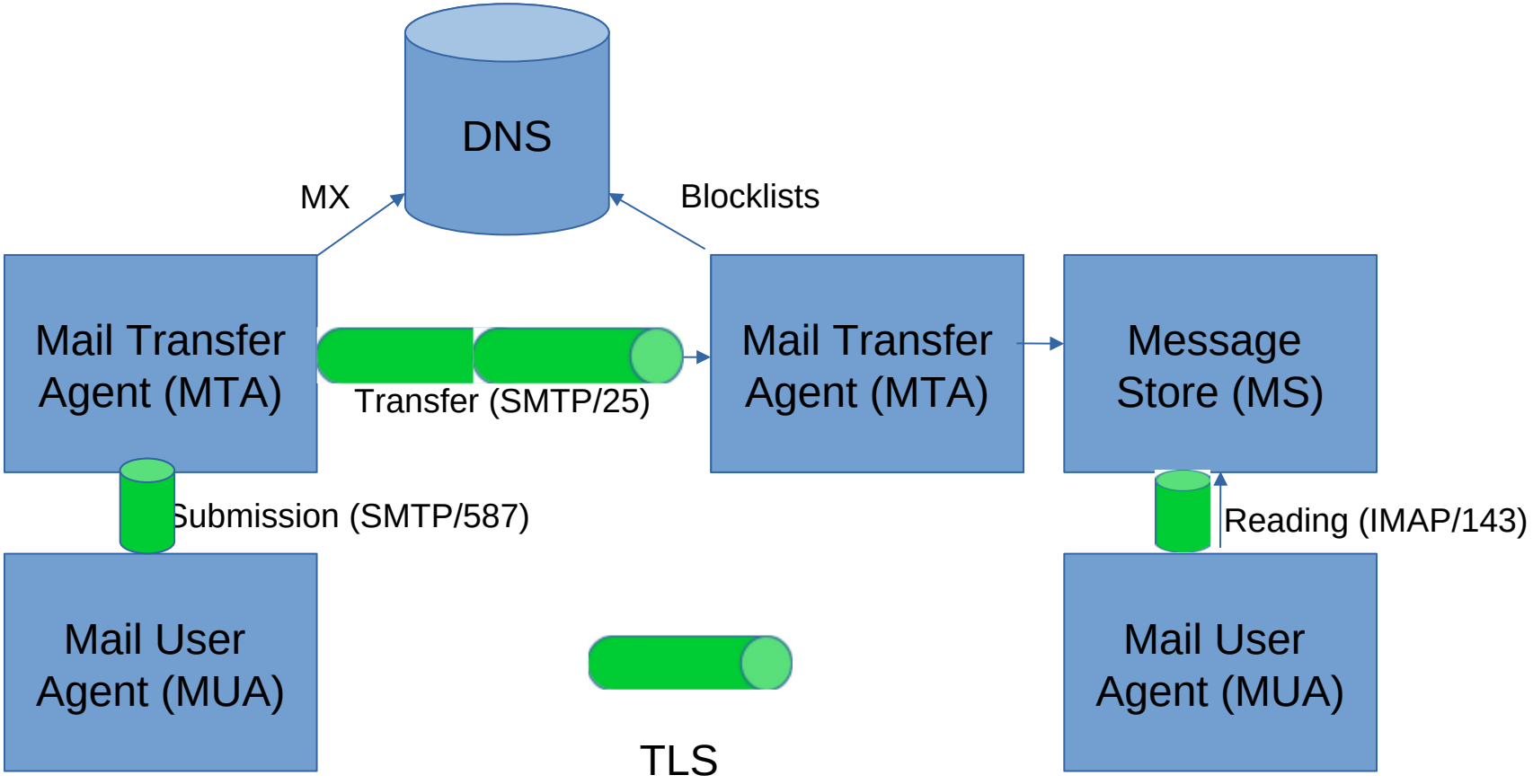
https://docs.oracle.com/cd/E63029_01/books/SecurHarden/img/architecture_email_v.gif

Another architectural view



low score for HAM, high score for spam => inbox spam ranking

Mail Transport Security

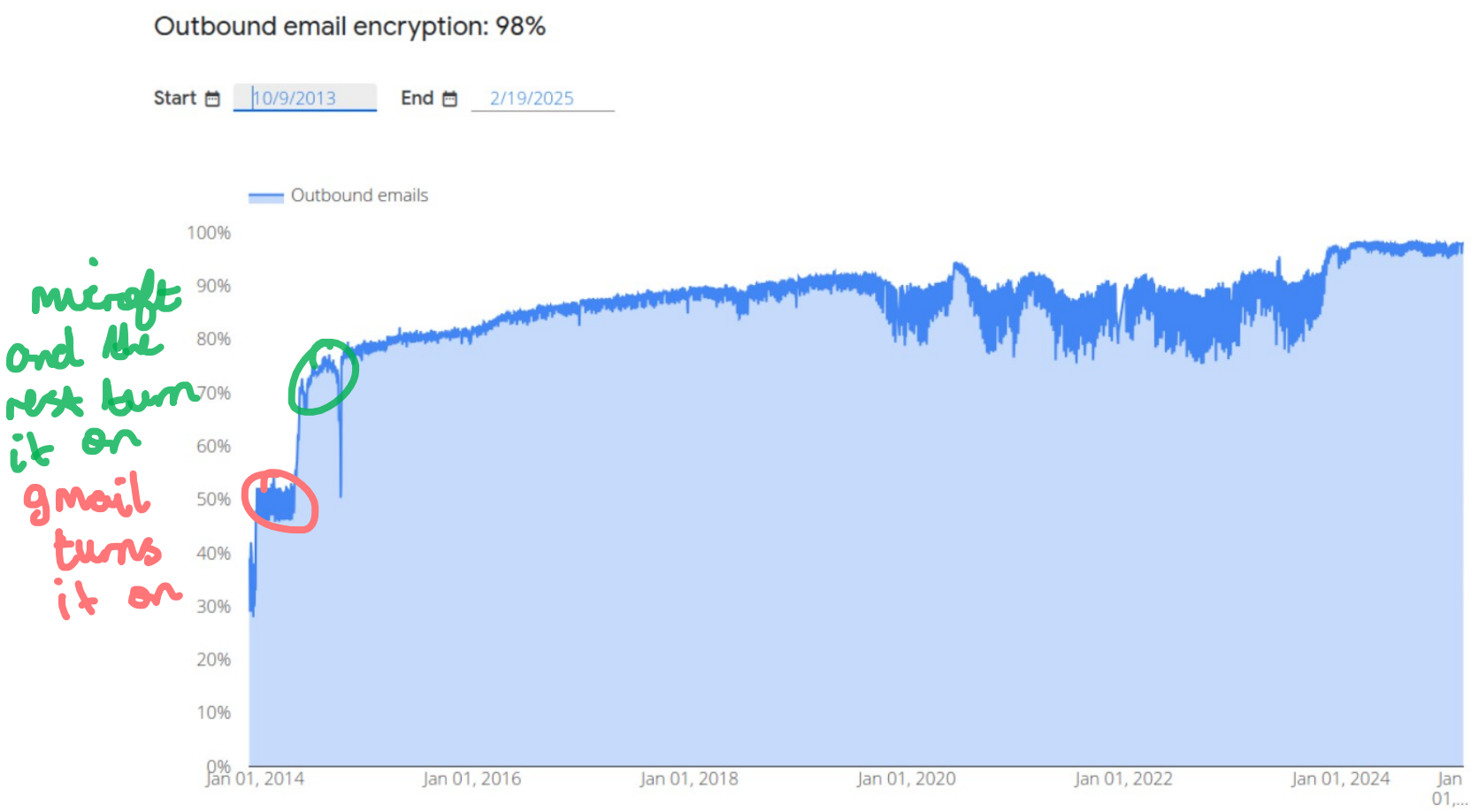


Mail Transport Security sometimes described as “hop-by-hop security”

Mail Transport Security Stats

- Gmail publish “transparency report” stats
  - https://transparencyreport.google.com/safer-email/overview
- Other service providers see commensurate numbers
- Recently: 98% of outbound and 100% of inbound are protected via some form of TLS
  - Note: “Some form” == possibly opportunistic
  - First time I’ve seen 100%!
- Variations remain

HBH over time (still @gmail)



Opportunistic SMTP/TLS (1)

- Sending MTA looks up MX RR of recipient's DNS domain
- MX name might not be same as recipient's domain, e.g., tcd.ie's MX is msft/outlook
- Sending MTA is TLS client, receiving MTA is TLS server
- Question: which name to require in TLS server cert? MX name or recipient domain?
- Answer: many receiving domains outsource MX to someone big and good at anti-spam meaning sending MTA has to deal with MX names and can't insist MX be able to serve TLS certificate of recipient domain

Opportunistic SMTP/TLS (2)

- Today many MX's still serve self-signed certificates or expired certificates or certificates signed by some CA not trusted by sending MTA (e.g. a corporate CA)
- For all the above reasons, SMTP/TLS is often still "opportunistic" in that MTAs enable it, but don't impose the same level of checks as are done on the web
- Note: on the web, there's often a warm body who can see/react to errors or warnings – there is no human user present when a sending MTA tries to establish SMTP/TLS with a receiving MTA

MTA-STS

- To improve on opportunistic SMTP/TLS we need receiving MTAs to publish a signal that it's ok for sending MTAs to be more strict
- MTA Strict Transport Security (RFC 8461) defines a way to publish that information in the DNS (so sending MTA can look for it before starting to connect)
  - Modelled on HSTS and DKIM
- Important: has a "testing" phase and defines how sending MTAs can report stats occasionally

testing and reporting helps convince people to use SMTP/TLS

Mail and SPAM

…just can't seem to get enough…

https://www.theguardian.com/environment/2017/feb/13/extraordinary-levels-of-toxic-pollution-found-in-10km-deep-mariana-trench#img-1

What is spam?

- Various acronyms:
  - Unsolicited bulk email (UBE)
  - Unsolicited commercial email (UCE)
- Spam is bad:
  - Resource consumption
    - Filters, scanners etc. cost time & money
  - Malware
  - Phishing attempts

Junk Mail

Sometimes hard to know…

HILARY TERM GREETINGS FROM THE COLLEGE CHAPLAINS The College Chaplains send best wishes to all, and would like to bring the following upcoming events to your attention. They are open to any students or staff members who wish to join us. …

Original spam tricks

- Just send email!
  - Ahh…the naivety of it all!
- Email to list
  - Listservers got better, e.g. subscriber only with controlled subscription
- Forge headers
- Send via open relay
  - Used to be a lot of these, very few now
    - toad.com is (or was) an exception!

More spam tricks…

- Confusion:
  - accounts@paypa1.com
  - support@eboy.com
  - postmaster@boi-support.com
  - About to get worse thanks to I18N
  - security@bigbank.com
    ^ Unicode 0430 is cryllic small ‘a’
- Throwaway domains/addresses
- Zombie’d hosts
- BGP spoofing
- DKIM replay

Yet more

- HTML messing
  - Colour-related
  - Relay sites
  - Encoded URIs
  - Font size 0: break words with zero width spaces

How much spam is there?

- Lots
  - Hard to get good figures, these are ones I've overheard
- ISP backbones:
  - 70% + of email traffic
- Delivered mail:
  - 40% + delivered
- Increasing or not?
  - Harder to tell if MTAs silently filter

Recent Anti-spam techniques

- Beware of FUSSP - https://www.dmuth.org/fussp/
- Content filtering (Bayesian, etc.)
  - Misc. AI magic
- DNS Block Lists (SORBS, DNSBL)
- Register of known spam operators (ROKSO)
- Greylisting
  - All in widespread use today

if the sender seems suspicious, fake an error and ask to resend in few minutes. Spam most likely won't be resent.

- Sender Policy Framework (SPF)
- Domain Keys Identified Mail (DKIM)
- DMARC

SPF, DKIM, DMARC (v. briefly)

- Sender Policy Framework (SPF): publish IP addresses of sending MTA in DNS; receiving MTA checks source IP of inbound connections – RFC 7208
- Domain Keys Identified Mail (DKIM): sending MTA digitally signs outbound emails with private key associated to sending domain; receiving MTA looks up public key in DNS and checks signature; missing/bad signature treated as unsigned message - RFC 6376
- Domain-based Message Authentication, Reporting, and Conformance (DMARC): sending MTA publishes “policy” in DNS stating how it would like receiving MTAs to treat messages, e.g. requiring SPF and DKIM to “pass” before accepting email, up to and including asking receivers to “reject” inbound mail that fails SPF or DKIM – RFC 7489

DMARC Downsides (1)

- DMARC is great for transactional mail (e.g. receipts), but, there are valid and (to some) important uses of 3rd party sending: Mailing lists, alumni addresses
- Mailing lists are the main tool for discussing how Internet standards (incl. Mail, incl. DKIM, incl. DMARC) should work, so breaking those seems dim
- But some mail service providers will prefer to reduce their costs even so
  - https://www.pcworld.com/article/2141120/yahoo-email-antispoofing-policy-breaks-mailing-lists.html
  - https://wiki.asrg.sp.am/wiki/Mitigating_DMARC_damage_to_third_party_mail

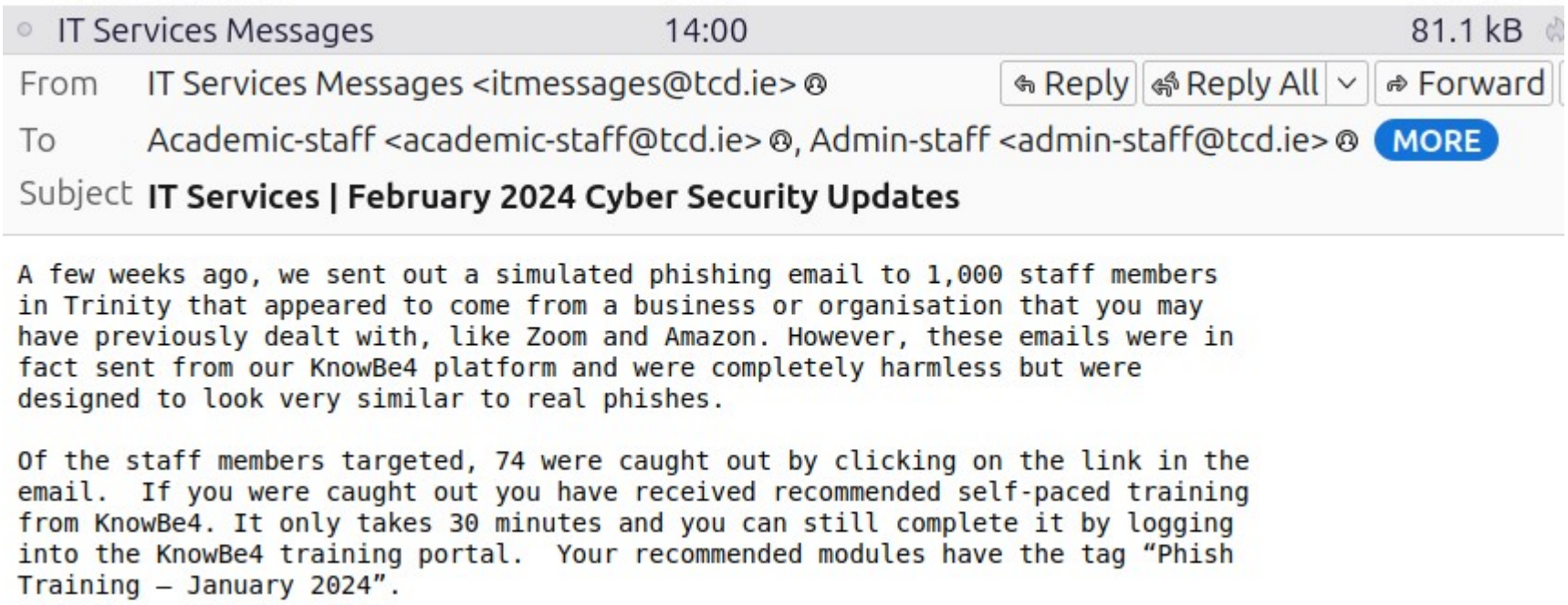
DMARC Downsides (2)

- When sender to mailing list is from a domain that publishes “p=reject” policy, other list-recipients may bounce the mail, leading to non-delivery, but also, after a few such bounces, leading the mailing list to unsubscribe the sender
  - Result: <user>@yahoo.com can no longer participate in e.g. IETF
- Risk: If gmail.com published “p=reject” that would likely cause havoc for organisations that depend on mailing lists
  - Google said they won’t until the impact of that would be reduced sufficiently
- Workarounds
  - Use gmail or a vanity sending domain you control
  - IETF mailing lists re-write sender’s address, e.g. to first.last=40example.com@dmarc.ietf.org
- “Solution” for DMARC downsides (ARC) involves DKIM signing by sender’s MTA and a 2nd DKIM(-like) signature from list agent, but is complex, and interpreting ARC protections on a message is even more complex so not clear if ARC deployment will help
  - Authenticated Received Chain (ARC) is RFC 8617 but mostly ignored – some people are starting work on a DKIM2 that could improve this situation

DKIM vs DNSSEC

- We’re jumping miles ahead here, but will return to this: DKIM and DNSSEC involve publishing public keys in DNS
- DNSSEC DS requires talking to parent domain, chains up to signed root zone
    - ~4% of .com domains are DNSSEC signed
- DKIM publishes public key in DNS in TXT record with no additional DNS security
    - Today DKIM is almost mandatory if you want your email delivered
    - Partly, that’s due to the power of the major mail service providers
- Think about the deployment pros and cons?

Simulated Phishing



IT Services Messages 14:00 81.1 kB

From IT Services Messages <itmessages@tcd.ie> Reply Reply All Forward

To Academic-staff <academic-staff@tcd.ie>, Admin-staff <admin-staff@tcd.ie> MORE

Subject **IT Services | February 2024 Cyber Security Updates**

A few weeks ago, we sent out a simulated phishing email to 1,000 staff members in Trinity that appeared to come from a business or organisation that you may have previously dealt with, like Zoom and Amazon. However, these emails were in fact sent from our KnowBe4 platform and were completely harmless but were designed to look very similar to real phishes.

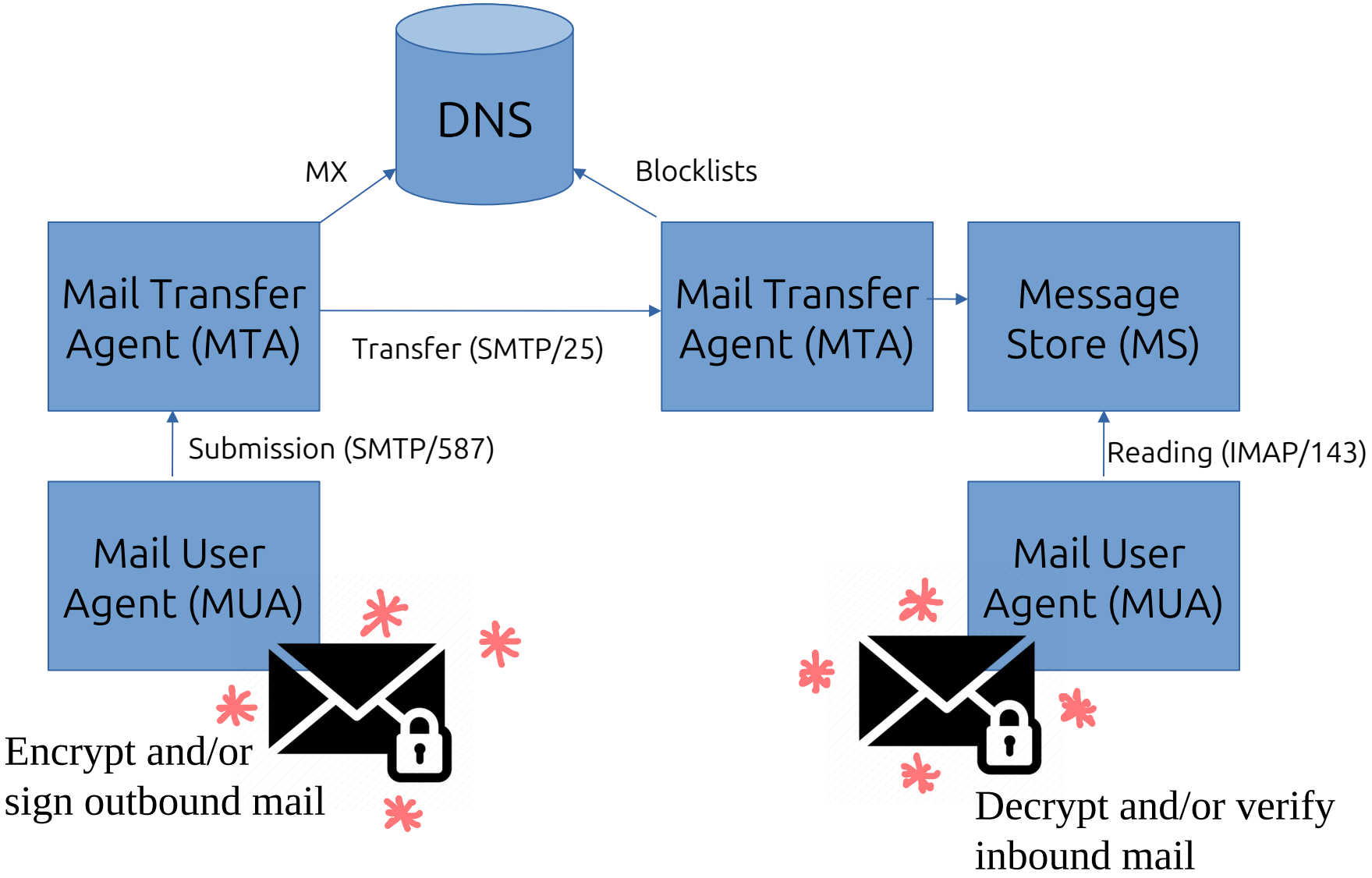
Of the staff members targeted, 74 were caught out by clicking on the link in the email. If you were caught out you have received recommended self-paced training from KnowBe4. It only takes 30 minutes and you can still complete it by logging into the KnowBe4 training portal. Your recommended modules have the tag “Phish Training – January 2024”.

SPAM summary

- Email is historically and still currently a very important Internet application
- Hop-by-hop/transport security for mail has significantly improved since 2013
- SPF, DKIM and DMARC are deployable and very widely used
- DMARC is good for some but not all
- None of the above “solves” spam
- Spam/Anti-spam is an arms-race, beware the FUSSP!

  http://www.circleid.com/posts/20131226_the_naive_arrogance_of_fussps/

End-to-end (E2E) Email Security



Focus on TLS for the exam

E2E mail security: S/MIME

- There's a history here too!
  - PGP, PEM and MOSS
  - RSADSI's PKCS#7 based proposal
- Cryptographic Message Syntax (CMS) is the basis for S/MIME and various other crypto applications using ASN.1
  - So S/MIME = CMS
    + Message-Specification
    + Certificate-specification

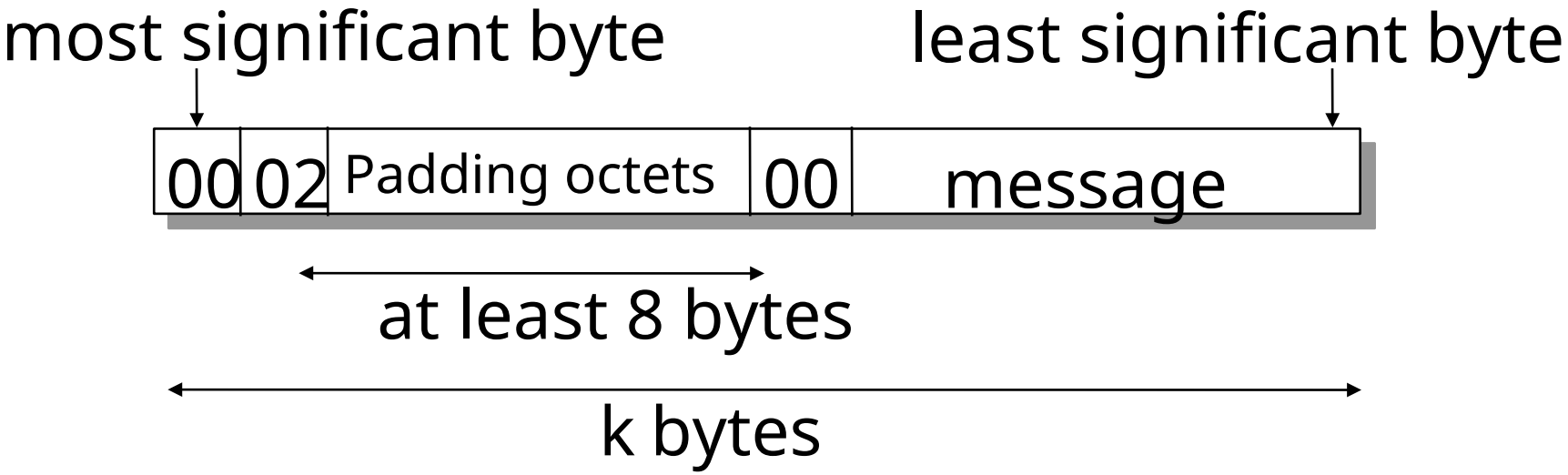
CMS

- How to MAC, sign and/or encrypt application data in an ASN.1 oriented way
  - E.g. CMS defines: SignedData and EnvelopedData
  - XML, JSON and CBOR equivalents started from here
- Algorithms and options same as X.509
- Latest CMS specification: RFC 5652
- But first, to encrypt, we need a bit of glue between arithmetic (for RSA) and ASN.1 BIT STRINGs – often uses PKCS#1 v1.5 format…

PKCS#1 v1.5 Padding (encryption) RFC 8017

RSA modulus: n=pq of length k bytes;
i.e. 256 ** (k-1) < n < 256 ** k

Padding octets are random but non-zero



CMS SignedData (1)

```
SignedData ::= SEQUENCE {
 version CMSVersion,
 digestAlgorithms DigestAlgorithmIdentifiers,
 encapContentInfo EncapsulatedContentInfo,
 certificates [0] IMPLICIT CertificateSet
                          OPTIONAL,
 crls [1] IMPLICIT CertificateRevocationLists
                          OPTIONAL,
 signerInfos SignerInfos }
```

CMS SignedData (2)

```
SignerInfo ::= SEQUENCE {
 version CMSVersion,
 sid SignerIdentifier,
 digestAlgorithm DigestAlgorithmIdentifier,
 signedAttrs [0] IMPLICIT SignedAttributes OPTIONAL,
 signatureAlgorithm SignatureAlgorithmIdentifier,
 signature SignatureValue,
 unsignedAttrs [1] IMPLICIT UnsignedAttributes OPTIONAL }

SignerIdentifier ::= CHOICE {
 issuerAndSerialNumber IssuerAndSerialNumber,
 subjectKeyIdentifier [0] SubjectKeyIdentifier }

SignedAttributes ::= SET SIZE (1..MAX) OF Attribute
UnsignedAttributes ::= SET SIZE (1..MAX) OF Attribute
Attribute ::= SEQUENCE {
 attrType OBJECT IDENTIFIER,
 attrValues SET OF AttributeValue }
AttributeValue ::= ANY
SignatureValue ::= OCTET STRING
```

CMS Message Specification

- Latest specification is RFC 8551
- Tells you how to:
  - Start with a MIME email message
  - Treat that like plaintext the CMS way
  - Then take the resulting bytes and make them into a MIME message
- Note: LARGE messages exist
  - Have to handle BER as well as DER

CMS Certificate Specification

- Latest specification is RFC 8550
- Tells you how to interface an s/mime mail user agent with a PKI
- Tells you how to interpret RFC 5280 for s/mime purposes
  - E.g. How to include email addresses in certificates

Wanna Try S/MIME?

- It’s been a few years since I had a working s/mime setup (I use PGP mostly), but for your benefit…
- In 2022, I tried this “free” service: https://www.actalis.com/s-mime-certificates.aspx
- Verification email was Italian-first:-)
- Then they sent me a pkcs#12 file!
- That means they know the private key!!
- I didn’t install that in my mail user agent
- I got Italian language emails a year later me that a certificate was going to expire (I think:-)
- Doesn’t look good for ad-hoc S/MIME use: https://kb.mozillazine.org/Thunderbird_:_FAQs:_Get_an_SMIME_certificate

Pretty Good Privacy (PGP)

- PGP can do all that S/MIME does
- PGPMime is RFC 3156
- PGP's basic formats in RFC 9580
  - Not ASN.1 based (home-grown TLVs)
  - Recently updated (I help co-chair that), sadly accompanied by a schism in the PGP community
- Web-of-trust model != X.509 PKI
  - But you don't have to
- Most important use-case? Maybe package signing
- Now natively supported in Thunderbird

PGP Key Management

- X.509-based PKIs are hierarchies
- PGP web of trust (WoT) based on user's signing one another's keys, with possibly many signatures per public key
- PGP Key IDs are (truncated) hashes
- PGP key servers exist
- Usability: sucks:-(
- Details: lots of HOWTOs on the web

S/MIME and PGP Deployment

- Most MUAs suport s/mime or PGP either built-in or as an option
  - There are also “plug-in” products
- And mostly then *can* work together
  - I’ve used both, PGP currently seems more usable (with Thunderbird)
- But e2e secure mail is not ubiquitous
  - Why?

e2e email security barriers

- Designs pre-date web user agent which changes trust model (where's the private key kept? Needs new infrastructure)
- Needs all major mailbox providers (yahoo, hotmail, gmail) to deploy the same thing which also needs to be implemented by all major user agent developers (microsoft, mozilla, apple, google)
- Public key retrieval needs to be fixed (doable if the above done, but a killer if not done), likely with some new PKI (doable but who's gonna pay?)
- Mail headers need to be protected as users don't get that S/MIME and PGP only protect body and not e.g. Subject, From (new enveloping protocol needed, work-in-progress but kludgy)
- We need to unify S/MIME and PGP or pick one or we'll lose interop (it's ok if the other soldiers on for some niches)
- Users don't care much, so it has to be entirely transparent for them (needs significant UI work, co-ordinated across MUAs and significant web-UAs)

e2e email current attempts

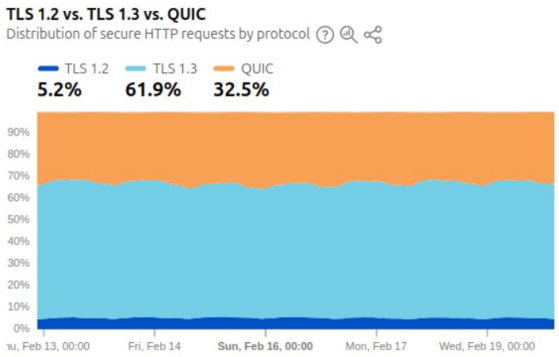
- (At least) two current projects have are trying to address general e2e mail security:
  - Autocrypt: https://autocrypt.org/ → auto use PGP whenever possible
  - ~~Sadly defunct: p≡p :~~ https://pep.foundation/
- Some service providers try make e.g. PGP easier:
  - ProtonMail: https://protonmail.com/
- All worthy, none (yet) with real MUA/mail-mega-provider traction
- Most email security today depends on TLS for mail transport security which is hop-by-hop and not end-to-end
- e2e email security doesn't play so well with server-side anti-spam/malware/phishing techniques
- BUT, without e2e email security, it's all postcards!
  - And there are people reading those: http://www.spiegel.de/international/europe/gchq-monitors-hotel-reservations-to-track-diplomats-a-933914.html

Transport Layer Security (TLS)

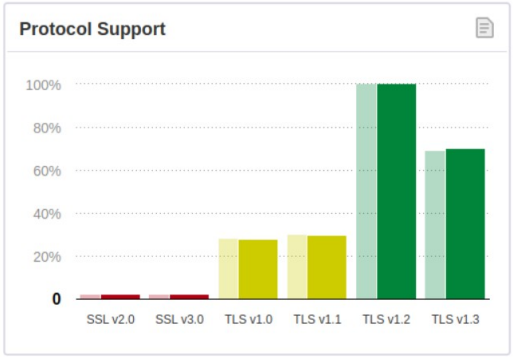
Transport Layer Security (TLS)

Secure Sockets Layer (SSL) proposed by Netscape in 1994

- Works with any application that uses sockets to communicate e.g., ftp, http, nntp, telnet...
- Standardised as Transport Layer Security (TLS) in 1999
  - Name was compromise with Microsoft (their version was called PCT)
  - https://tim.dierks.org/2014/05/security-standards-and-name-changes-in.html
- Latest: TLS1.3, RFC 8446; Oddly: RFC6101 is SSL3.0!



https://radar.cloudflare.com/adoption-and-usage



https://www.ssllabs.com/ssl-pulse/

SSL original services

Amazon don't care who you are so long as your credit card works.

- Server Authentication - “buyer” can believe they are dealing with a bona fide “merchant”
  - (Optional) Client authentication
  - Digital certificates (X.509)
- Message Encryption - Buyer can send credit card details across the network without fear of interception
  - Also message integrity and replay-detection
- Relatively transparent to the user and application developer
  - Hence the name “secure sockets layer”

Deployment of TLS

- Mid '90's: Used in Netscape Commerce Server
  - International “Step-up” encryption: Strong crypto for international banks, with special Verisign certificate
  - Built into Communicator 4.0 and later
- Now: everywhere, standard part of Web browsers, servers and all development platofrms (PHP, python, golang etc.)
  - online trading, banking, commerce…
- “foo”/TLS/TCP on port 443 is now de-facto baseline for Internet-scale interop

TLSv1.2 and TLSv1.3

- We'll look first at TLSv1.2, then lots more on TLSv1.3
- TLSv1.3 is a **major update** despite the minor version number change
- For exam purposes, I'd recommend you spend time on TLSv1.3, but it's (just about) still OK to choose TLSv1.2
- About 90%+ of you will choose to study TLS for exam purposes

- WRT older versions (SSL3, TLSv1.0, TLSv1.1): see next slide and RFC8996, but those are not valid study subjects for exam purposes

From: draft-ietf-tls-oldversions-deprecate-12 Best Current Practice
Errata exist
Internet Engineering Task Force (IETF) K. Moriarty
Request for Comments: 8996 CIS
BCP: 195 S. Farrell
Obsoletes: 5469, 7507 Trinity College Dublin
Updates: 3261, 3329, 3436, 3470, 3501, 3552, March 2021
3568, 3656, 3749, 3767, 3856, 3871,
3887, 3903, 3943, 3983, 4097, 4111,
4162, 4168, 4217, 4235, 4261, 4279,
4497, 4513, 4531, 4540, 4582, 4616,
4642, 4680, 4681, 4712, 4732, 4743,
4744, 4785, 4791, 4823, 4851, 4964,
4975, 4976, 4992, 5018, 5019, 5023,
5024, 5049, 5054, 5091, 5158, 5216,
5238, 5263, 5281, 5364, 5415, 5422,
5456, 5734, 5878, 5953, 6012, 6042,
6083, 6084, 6176, 6347, 6353, 6367,
6460, 6614, 6739, 6749, 6750, 7030,
7465, 7525, 7562, 7568, 8261, 8422
Category: Best Current Practice
ISSN: 2070-1721

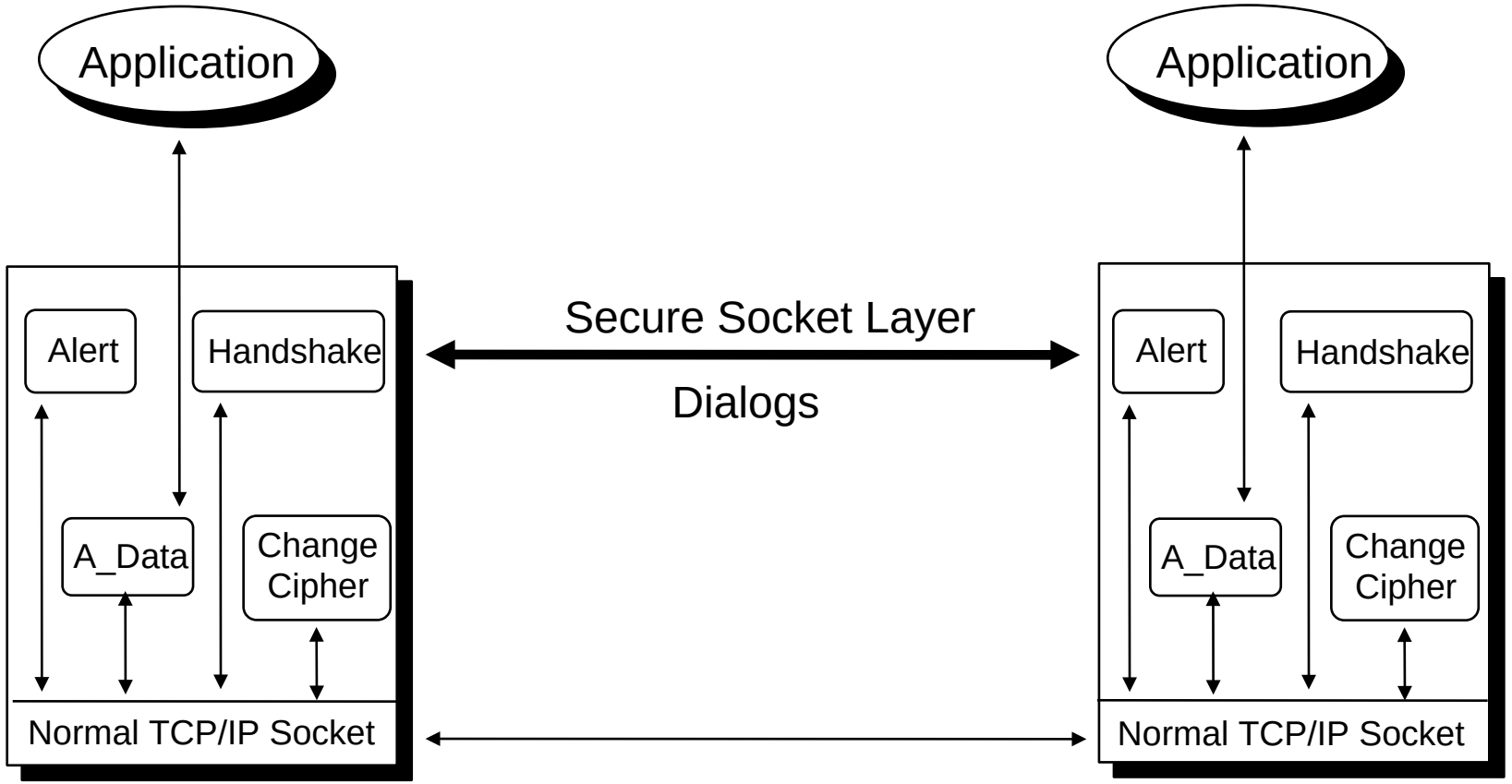
Deprecating TLS 1.0 and TLS 1.1

Abstract

This document formally deprecates Transport Layer Security (TLS) versions 1.0 (RFC 2246) and 1.1 (RFC 4346). Accordingly, those documents have been moved to Historic status. These versions lack

Components of the TLS Protocol

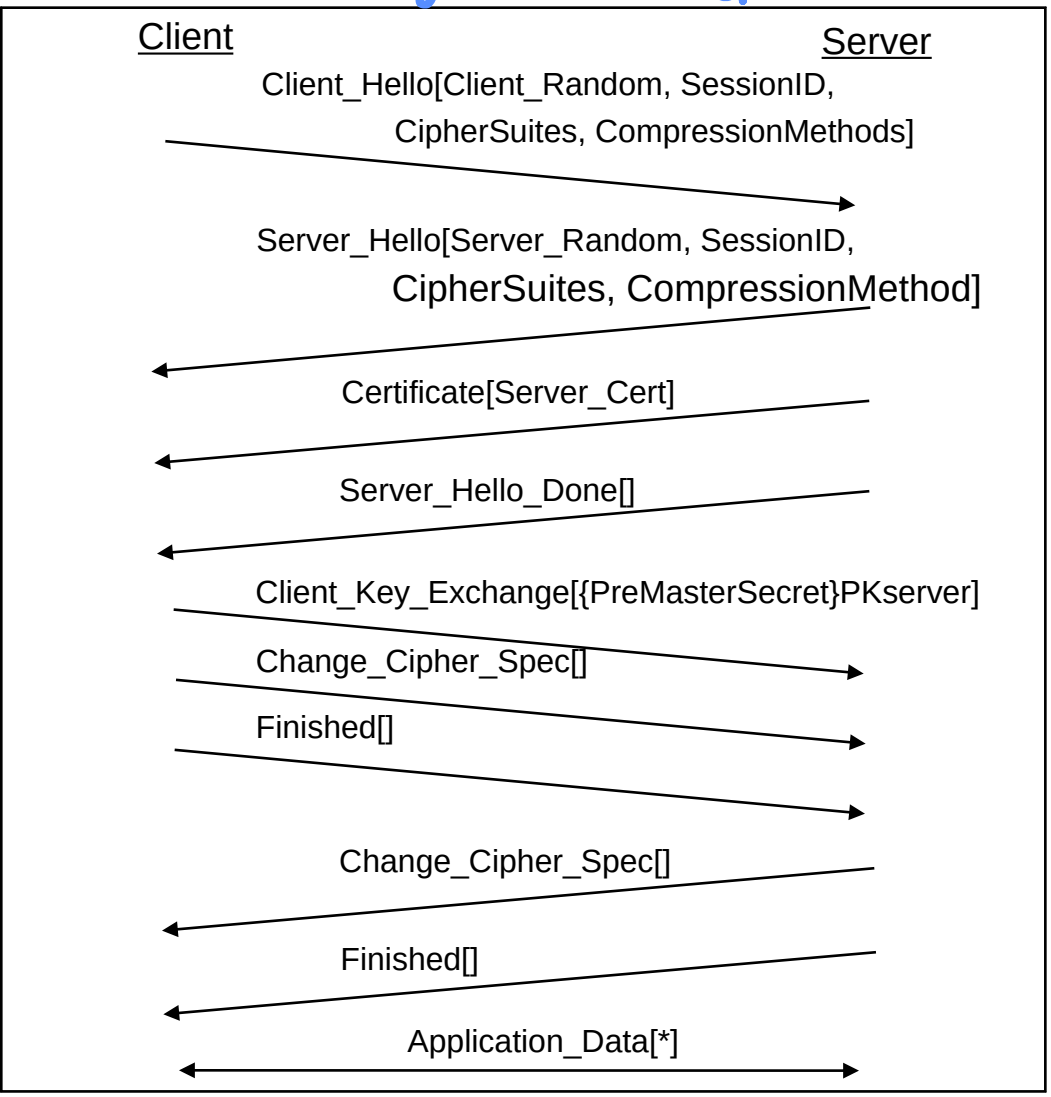
- TLS broken into 4 interrelated sub-protocols



A_Data = Application Data

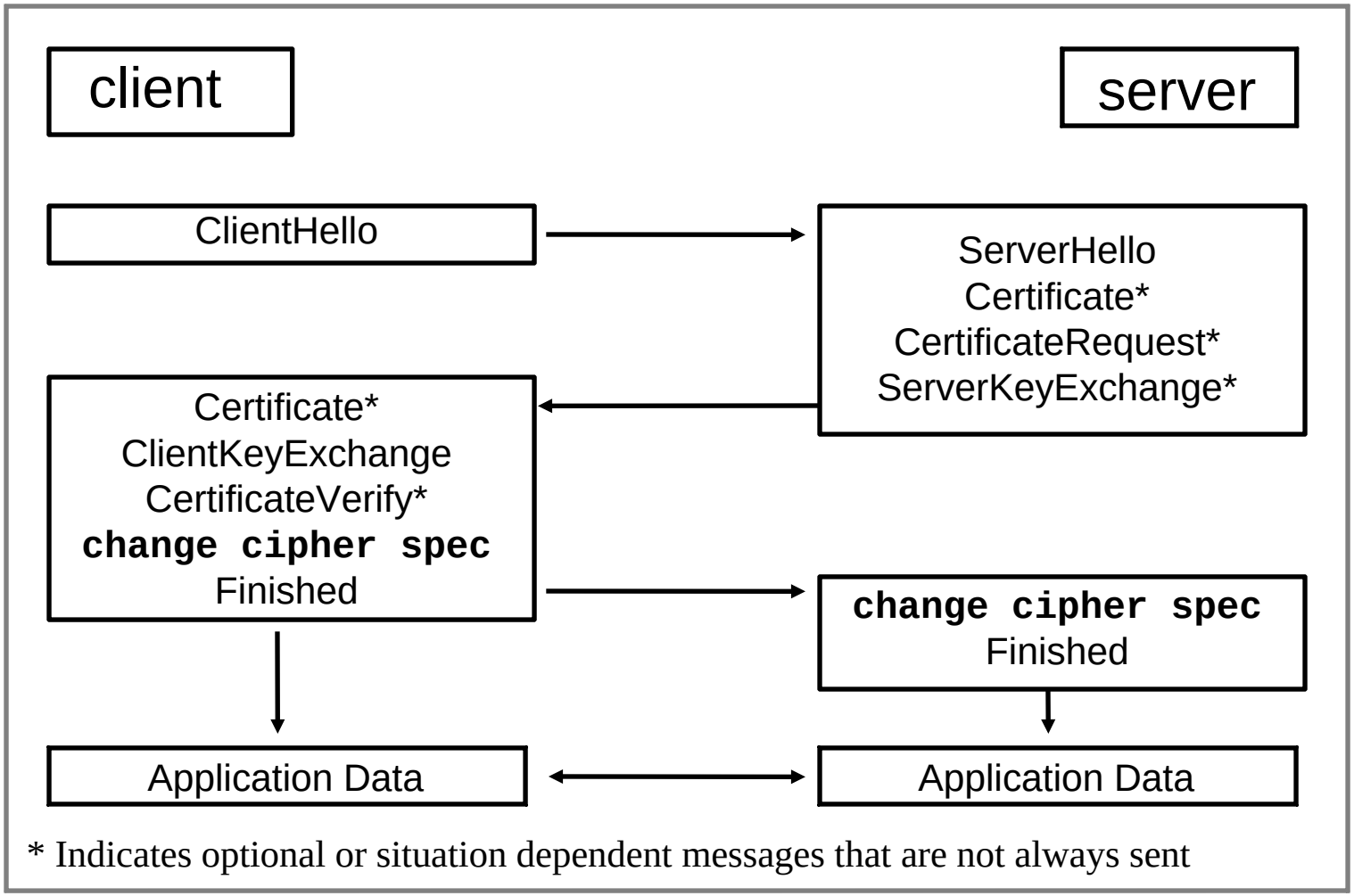
TLSv1.2 Handshake Protocol

Compression before encryption



- Negotiate Compression Method and Cipher Suite
- Swap random quantities
- Client obtains server certificate path
- Client invents PreMasterSecret (48 bytes) and securely sends it to the server
- Keys Calculated by both
- Finished message using new algorithms
- Data send in A_Data Units

TLSv1.2 Handshake Protocol
summary



Computing Keys from the Pre-MasterSecret

- First compute MasterSecret
- MasterSecret = f(PreMasterSecret, Client.Random, Server.Random)

- MasterSecret Used to prime key-generator
- KeyBlock = f(MasterSecret, Client.Random, Server.Random)

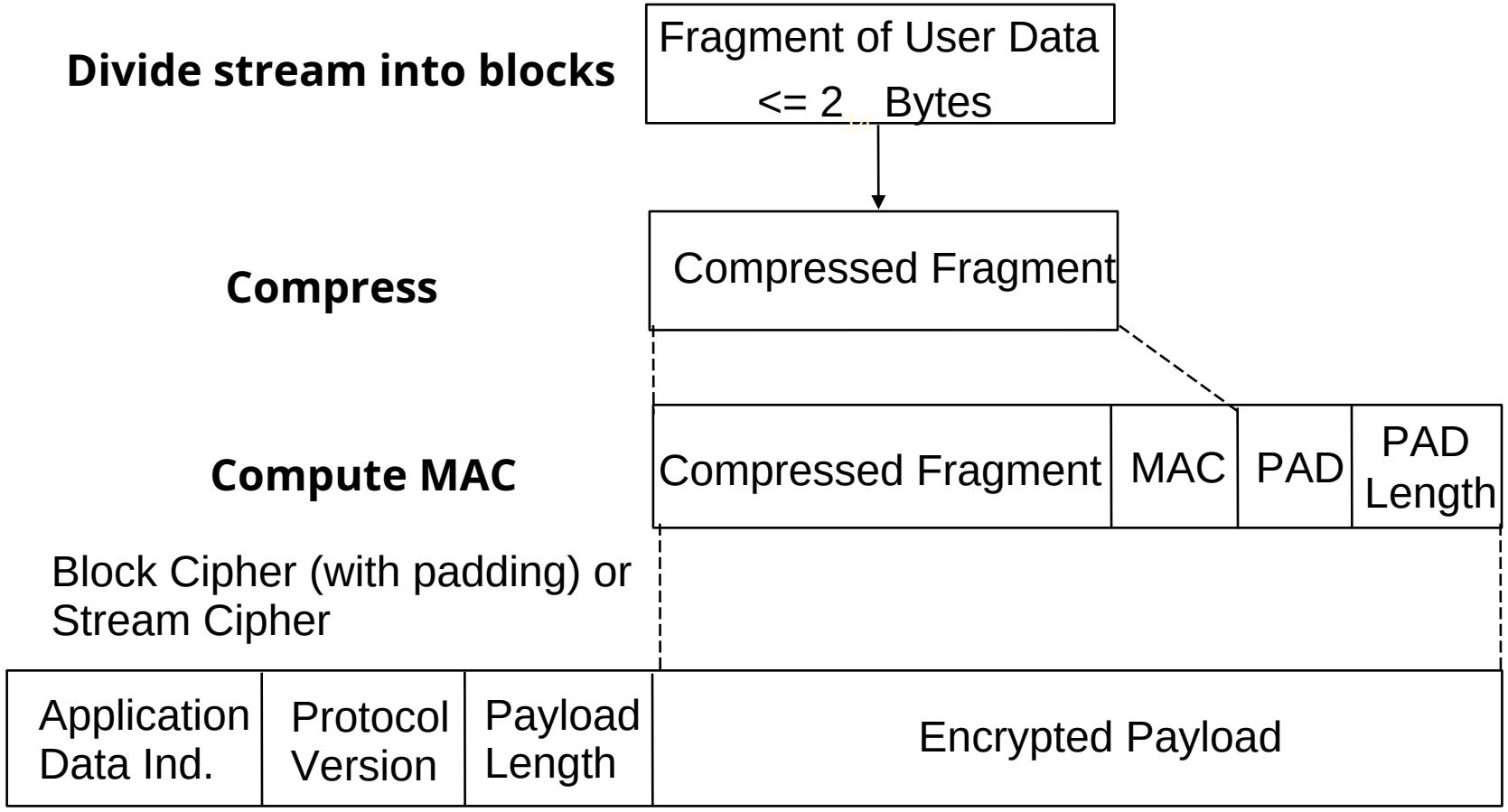
$$\begin{aligned}\text{KeyBlock} = {} & \text{MD5}(\text{MasterSecret} + \text{SHA}(\text{'A'} + \text{MasterSecret} + \text{ServerHello.Random} + \text{ClientHello.Random})) + \\ & \text{MD5}(\text{MasterSecret} + \text{SHA}(\text{'BB'} + \text{MasterSecret} + \text{ServerHello.Random} + \text{ClientHello.Random})) + \\ & \text{MD5}(\text{MasterSecret} + \text{SHA}(\text{'CCC'} + \text{MasterSecret} + \text{Server.Hello.Random} + \text{ClientHello.Random})) + [\ldots]\end{aligned}$$

KeyBlock

Client_MAC_Secret
Server_MAC_Secret
Client_Key
Server_Key
Client_Stream_Key
Server_Stream_Key

Partition key-stream into individual quantities

Applying the Keys to Application Data (TLSv1.2 Record Layer)



TLSv1.2 Ciphersuites

- SSL/TLS supports various cryptographic options
  - Digest algorithms, key transport, …
- Design decision was to represent all choices made in a single value
  - Ciphersuite – a 16 bit number
  - TLS_RSA_WITH_3DES_EDE_CBC_SHA
- Interesting consequences…

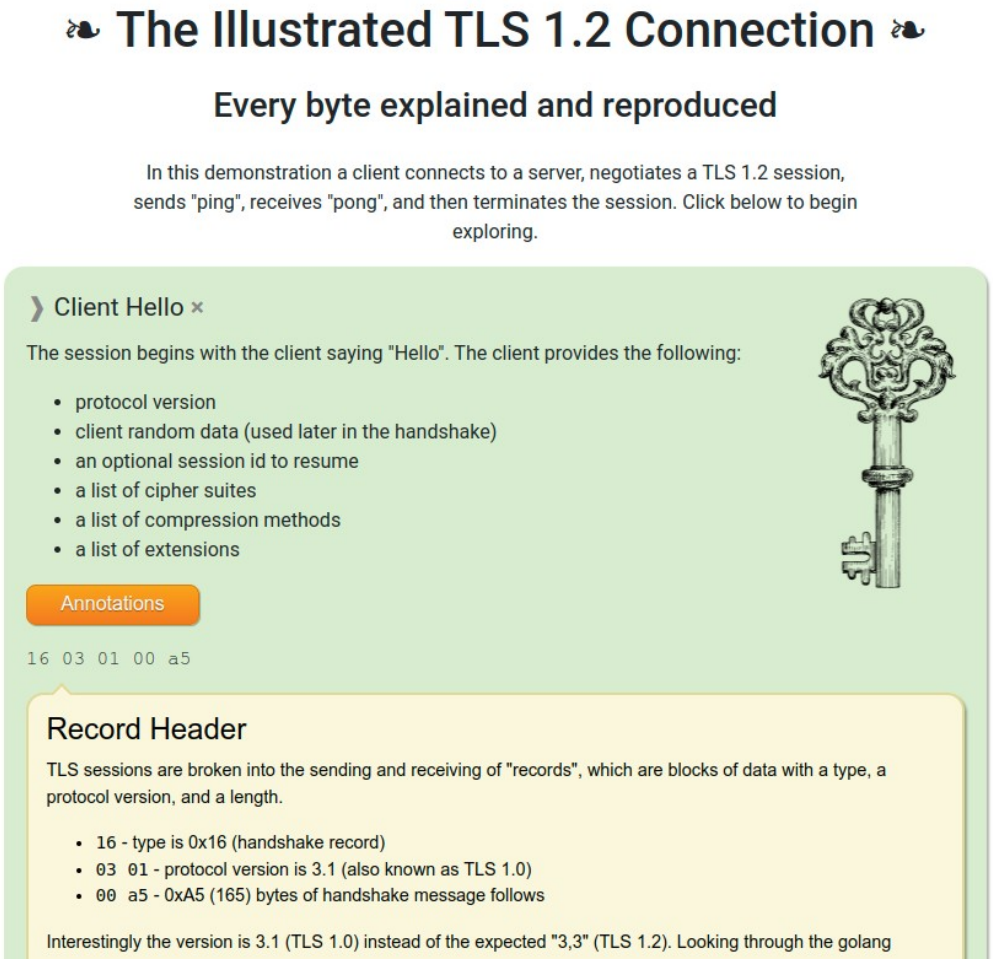
A TLSv1.2 Handshake Visualisation

https://tls12.xargs.org/

Shows a sample handshake with annotations

https://tls13.xargs.org/

similar for TLSv1.3



The Illustrated TLS 1.2 Connection

Every byte explained and reproduced

In this demonstration a client connects to a server, negotiates a TLS 1.2 session, sends "ping", receives "pong", and then terminates the session. Click below to begin exploring.

❯ Client Hello ×

The session begins with the client saying "Hello". The client provides the following:

- protocol version
- client random data (used later in the handshake)
- an optional session id to resume
- a list of cipher suites
- a list of compression methods
- a list of extensions

Annotations

```
16 03 01 00 a5
```

Record Header

TLS sessions are broken into the sending and receiving of "records", which are blocks of data with a type, a protocol version, and a length.

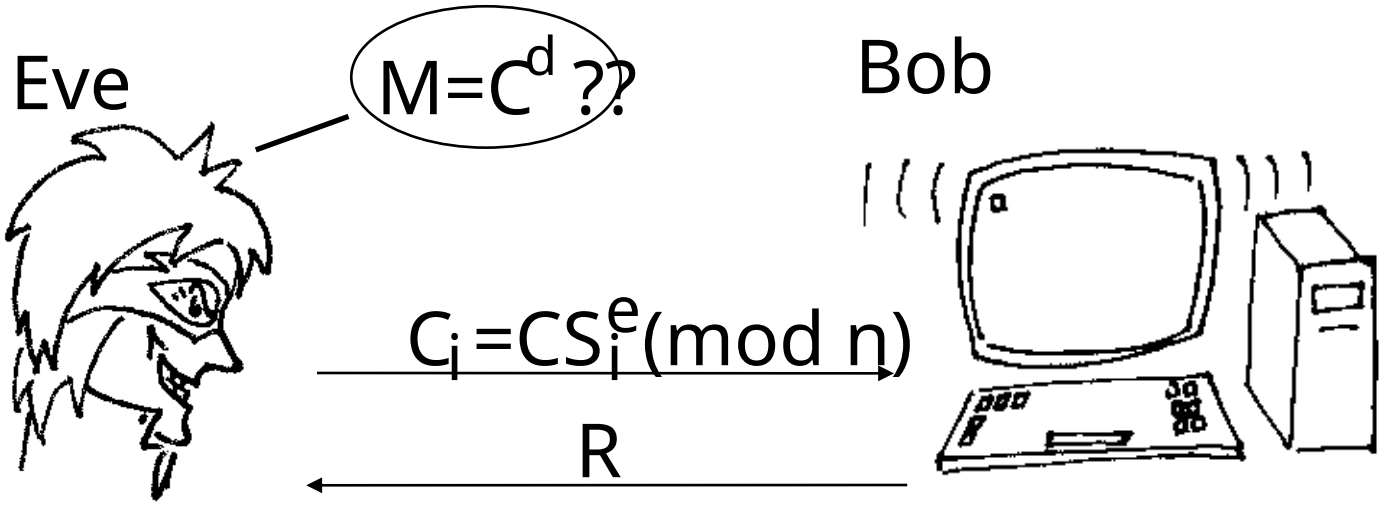
- `16` - type is 0x16 (handshake record)
- `03 01` - protocol version is 3.1 (also known as TLS 1.0)
- `00 a5` - 0xA5 (165) bytes of handshake message follows

Interestingly the version is 3.1 (TLS 1.0) instead of the expected "3,3" (TLS 1.2). Looking through the golang

Bleichenbacher

- “Materials” section has a full paper & ppt
  - If you **completely** understand the attack, you're doing very well!
- Basis
  - PKCS#1 v1.5 padding adds formatting to the data
  - If I have an “oracle” that'll attempt decryption and tell me when the recovered plaintext has the PKCS#1 v1.5 padding then I gain knowledge
- This happened with TLS!
  - “Million message” attack

How the attack works: Overview



If a message C_I is PKCS conforming then

$2*256**(k-2) \le MS < 3*256**(k-2)$

- This is an adaptive chosen-ciphertext attack
- RFC3218 describes the attack and ways to avoid it.

Other TLS-relevant Stuff

- ~~Key-pinning in HTTP (HPKP): RFC 7469~~
- HTTP Strict Transport Security (HSTS): RFC 6797
- OSCP Stapling (RFC 6066)
- MTA Strict Transport Security (RFC 8461)
- Datagram TLS (DTLS) for connectionless applications (e.g. RTP): RFC 9147 for DTLS1.3, RFC 6437 for DTLS1.2

- We'll look at TLSv1.3 (RFC8446) in detail in a bit and Encrypted Client Hello (for TLSv1.3 only)
  - https://tools.ietf.org/html/draft-ietf-tls-esni

IP Security (IPsec)

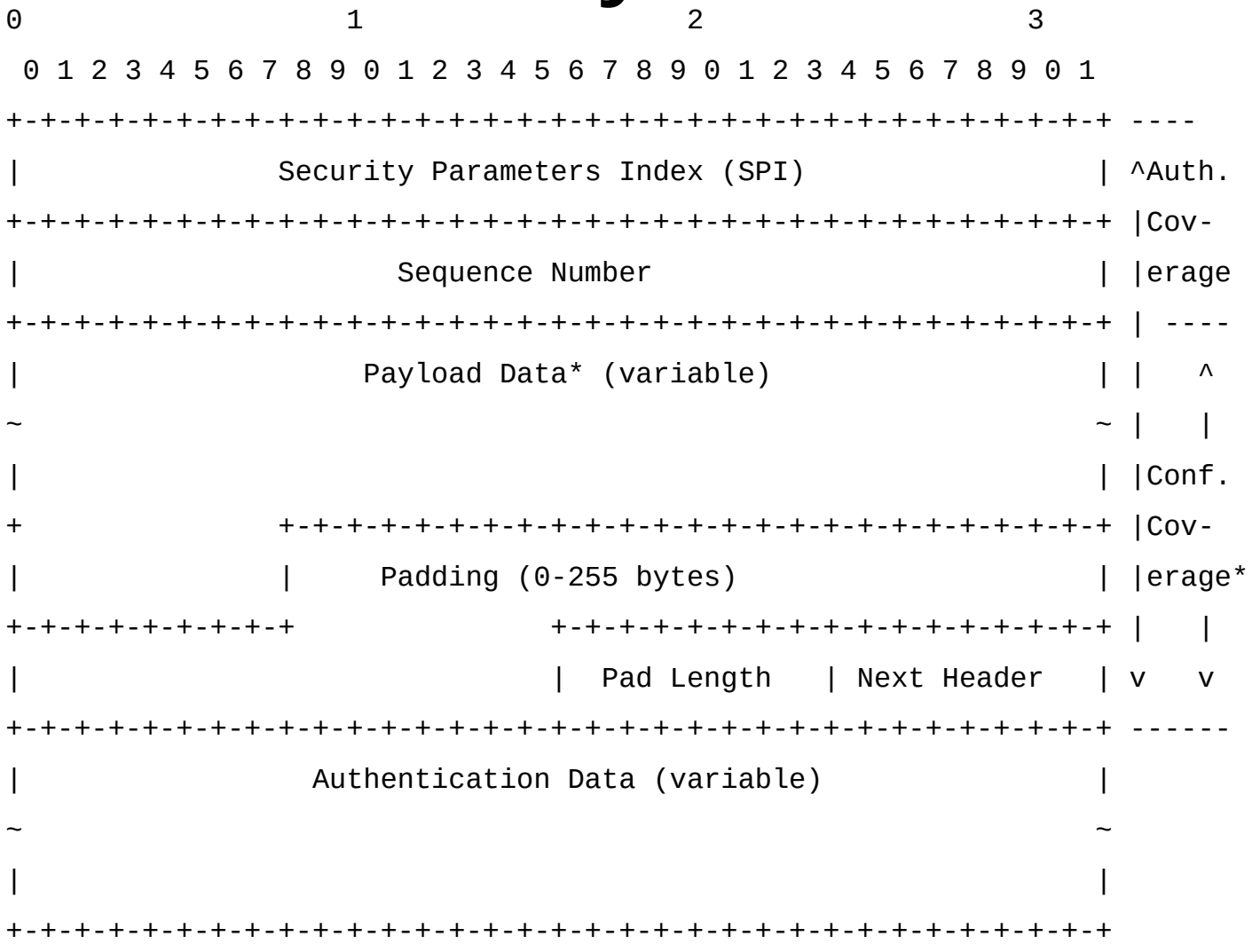
Ipsec overview

- “Security Architecture” RFC 4301
  - Key exchange, data encryption and data integrity mechanisms at the IP layer
- Optional as part of an IPv4 stack
  - (Used to be) Mandatory to implement as part of IPv6 stacks
  - Note: Mandatory to implement (MTI) != “MUST use”
- Tunnel/Transport modes
  - VPNs using tunnel mode common
- Security policy DB
  - How to handle (un)protected packets

IP Security (IPsec)

- Took sooooo looooong and produced such a complex outcome that they have a 2011 document roadmap (RFC 6071).
- Start with RFC 4301, the architecture
- Main moving parts: AH (RFC 4302), ESP (RFC 4303), and IKEv2 (RFC 7296)
  - AH mostly deprecated
  - IKE -> IKEv2 due to IKE/ISAKMP/... complexity, do NOT bother with IKEv1
- So, we'll only look at ESP & IKEv2

ESP – Encapsulated Security Payload



ESP terms

- Specification: RFC 4303
- Security Parameters Index (SPI) selects amongst different
  - Algorithms; Keys; etc.
  - SPI (+ dest. IP) = Security Association (SA)
- Seq# for anti-replay
  - Never cycles (new SA needed)
- Next header specifies payload protocol
- SA's are directional
  - So we use different keys between Alice and Bob as compared to between Bob and Alice

ESP – some features

- ESP allows some basic level of traffic hiding
  - Padding (up to 255 bytes) to disguise data length
  - Tunnel mode between gateways
- Data expansion
  - About +19 bytes per packet, possibly worse
  - Can be serious (telnet)

IKE - Internet Key Exchange

- IKEv1 (RFC 2409) is a mess of:
  - ISAKMP (rfc2408)
  - OAKLEY (rfc2412) and SKEME (not an rfc)
  - DOI (rfc2407)
- IKEv2 (RFC 7296) is a single document that's much better
  - But is 142 pages long;-(
- Wasn't that simple!

IKEv2 (1)

- Provides protocol to…
  - Mutually authenticate
  - Exchange keys
- Based on:
  - D-H and
  - Pre-shared secrets or RSA

IKEv2 (2)

- Establishes an IKE-SA and one (or more) CHILD-SA
- Generally requires 4 or 6 messages
  - IKE_SA_INIT (req/rep)
  - IKE_AUTH (req/rep)
  - CREATE_CHILD_SA (req/rep)

IKEv2 Phase1
IKE_SA_INIT & IKE_AUTH

```
Initiator                                       Responder
-----------                                     ---------

HDR, SAi1, KEi, Ni   -->

                     <--    HDR, SAr1, KEr, Nr, [CERTREQ]

HDR, SK {IDi, [CERT,] [CERTREQ,] [IDr,]
 AUTH, SAi2, TSi, TSr}     -->

                     <--    HDR, SK {IDr, [CERT,] AUTH,
                                     SAr2, TSi, TSr}
```

IKEv2 Phase2 CREATE_CHILD_SA

```
Initiator                                   Responder
-----------                                 ---------

HDR, SK {SA, Ni,[Kei],[TSi,Tsr]} -->

          <-- HDR, SK {SA, Nr,[Ker],[Tsi, TSr]}
```

Some IKEv2 Features

- Not IKEv1!
- Uses ESP to protect things
- Misc:
  - PFS, NAT-capable, Traffic Selectors, Liveness checks, Cookies
- Still a bit clunky though!
  - E.g. Compression (rfc2393), ESP and then AH nested SAs

IPsec summary

- Tunnel mode widely used for VPNs and works just fine
  - Transport mode hardly used at all
- IKE interop today isn't perfect (with certs), but is ok
  - Some vendor-specific stuff, e.g. For RADIUS/legacy auth
- Deployment issues:
  - Windows, NAT, Firewall, ECN, Opportunistic Keying, APIs
  - IKEv2 work aims to address a bunch (but not all!) of these

Another IPsec overview

- Good March 2022 presentation on IPsec by an expert (Paul Wouters)
  - https://datatracker.ietf.org/meeting/113/materials/slides-113-saag-introduction-to-ipsec-00
- If his content contradicts mine, believe him:-)

Other protocols: kerberos, SSH, wireguard

Kerberos

- Originally developed as part of MIT project athena
  - RFC4120 (but RFC1510 may be easier)
- Designed for many users working with few(-ish) servers
  - Largely symmetric key (shared secret) based
- Based around clients interacting with a Key Distribution Centre (KDC) aka:
  - Authentication server (AS)
  - Ticket Granting Server (TGS)

Kerberos

- Uses ASN.1 again! (sort-of)
- Latest spec: RFC 4120
- Typical use:
  - Client sends AS_REQ to KDC gets AS_REP containing TGT
  - Client sends TGS_REQ to KDC (includes TGT) and gets TGS_REP (including Ticket)
  - Client sends KRB_SAFE to server including Ticket

A Ticket

```
Ticket ::= [APPLICATION 1] SEQUENCE {
tkt-vno [0] INTEGER (5),
realm [1] Realm,
sname [2] PrincipalName,
enc-part [3] EncryptedData -- EncTicketPart }

-- Encrypted part of ticket

EncTicketPart ::= [APPLICATION 3] SEQUENCE {
flags [0] TicketFlags,
key [1] EncryptionKey,
crealm [2] Realm,
cname [3] PrincipalName,
transited [4] TransitedEncoding,
authtime [5] KerberosTime,
starttime [6] KerberosTime OPTIONAL,
endtime [7] KerberosTime,
renew-till [8] KerberosTime OPTIONAL,
caddr [9] HostAddresses OPTIONAL,
authorization-data [10]
         AuthorizationData OPTIONAL }
```

Kerberos Flows

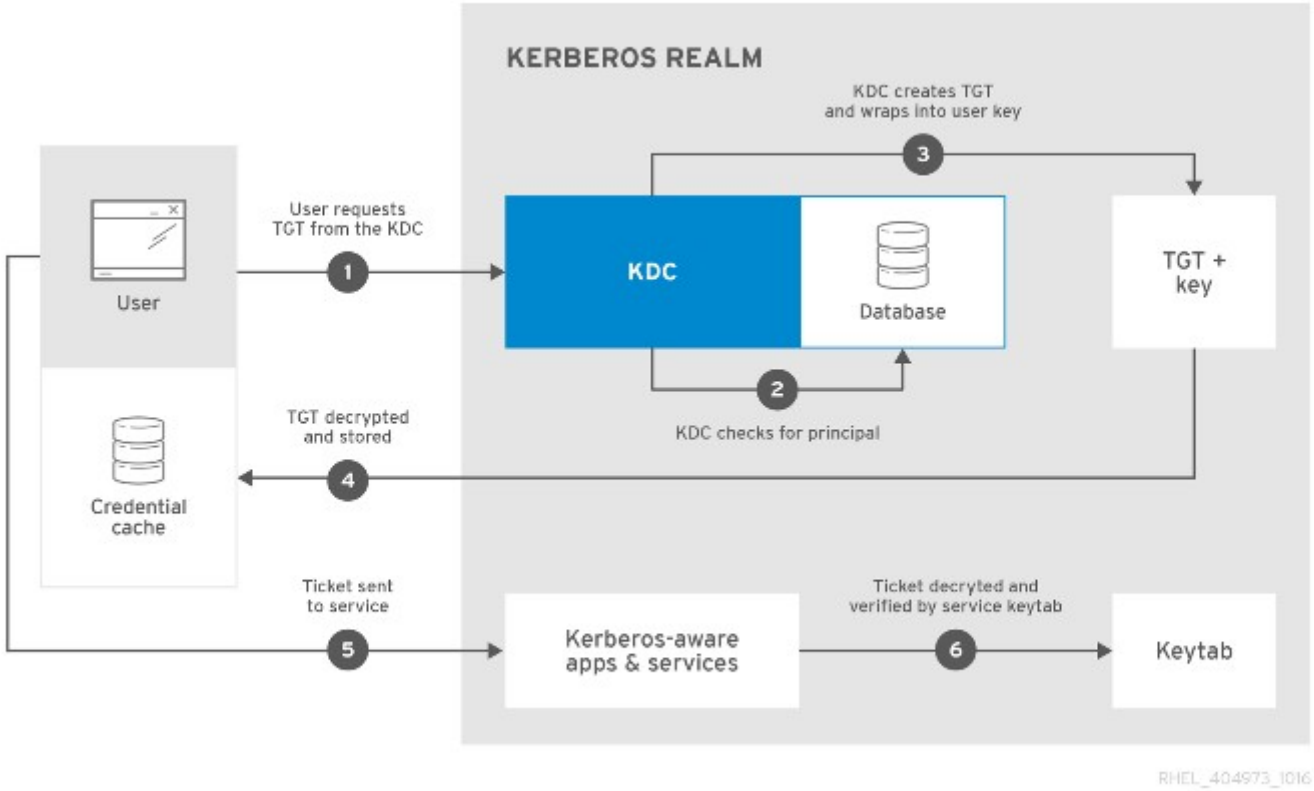


Figure 11.1. Kerberos Authentication

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system-level_authentication_guide/using_kerberos

Kerberos Things to Note

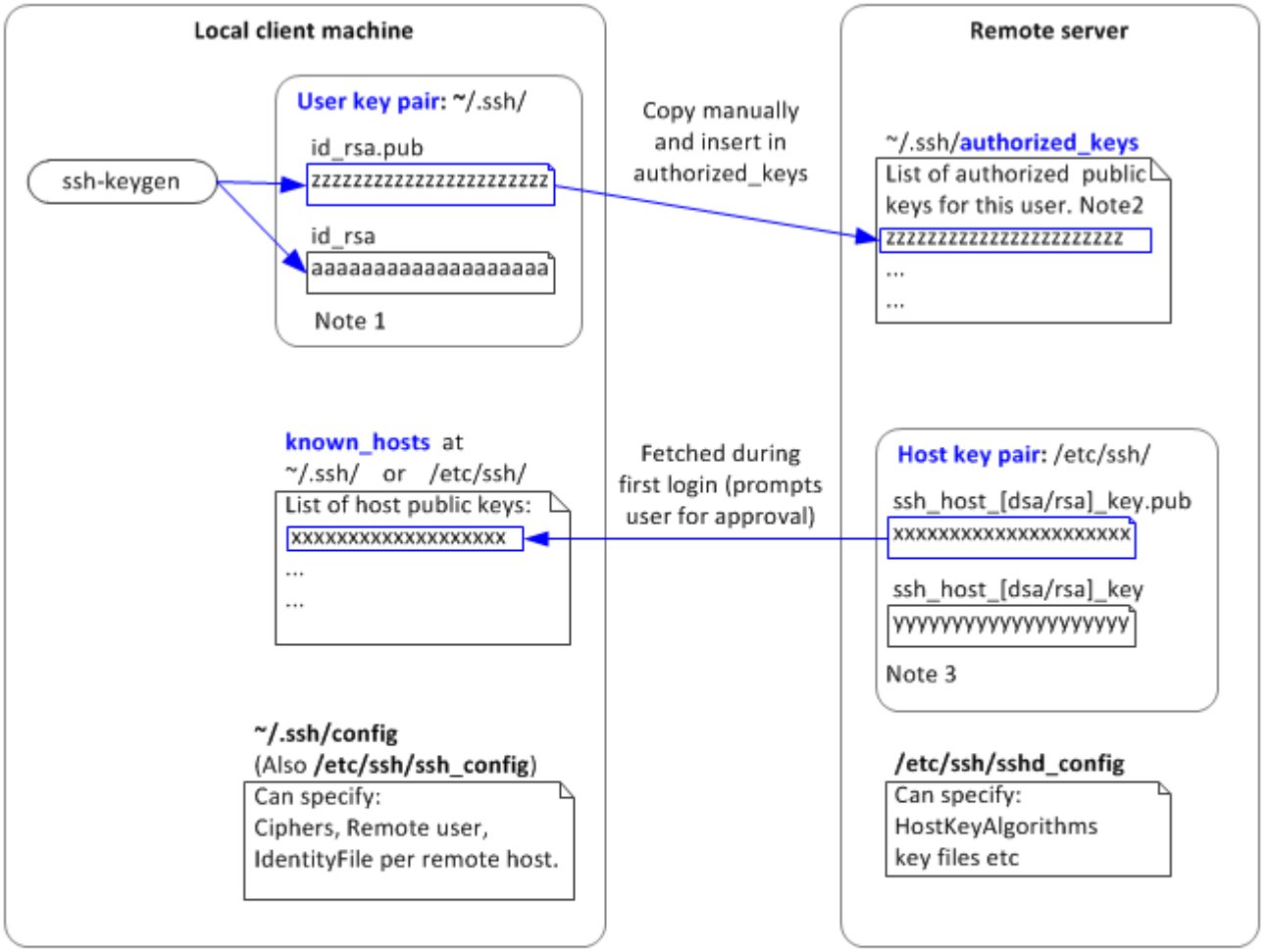
- Loose clock sync. Required
- Kerberos realm is like but not the same as a DNS domain
- Inter-realm and public key based operation defined but not used that often
- Used in Windows security: KDC is part of ActiveDirectory server
- Various authorization data extension have been done over the years
- AS_REQ can contain dictionary attackable password or something better (usually the latter in modern implementations)

Secure Shell (SSH)

- Architecture: RFC 4251
- Details in RFCs 4252 (transport protocol), 4253 (user auth), 4254 (multiplexing)
- Entirely typical flows – do D-H, authenticate in encrypted tunnel, application data in channel in encrypted tunnel
- Tends to be more driven by widely-used implementations rather than the RFCs, so generally implementations are ahead of the RFCs, sometimes significantly so esp. for extensions and algorithms, e.g.: https://www.openssh.com/specs.html
- Next 2 slides from: https://serverfault.com/questions/935666/ssh-authentication-sequence-and-key-files-explain

Files involved in SSH connection: preparation

gwideman 2018-10-14



Notes:

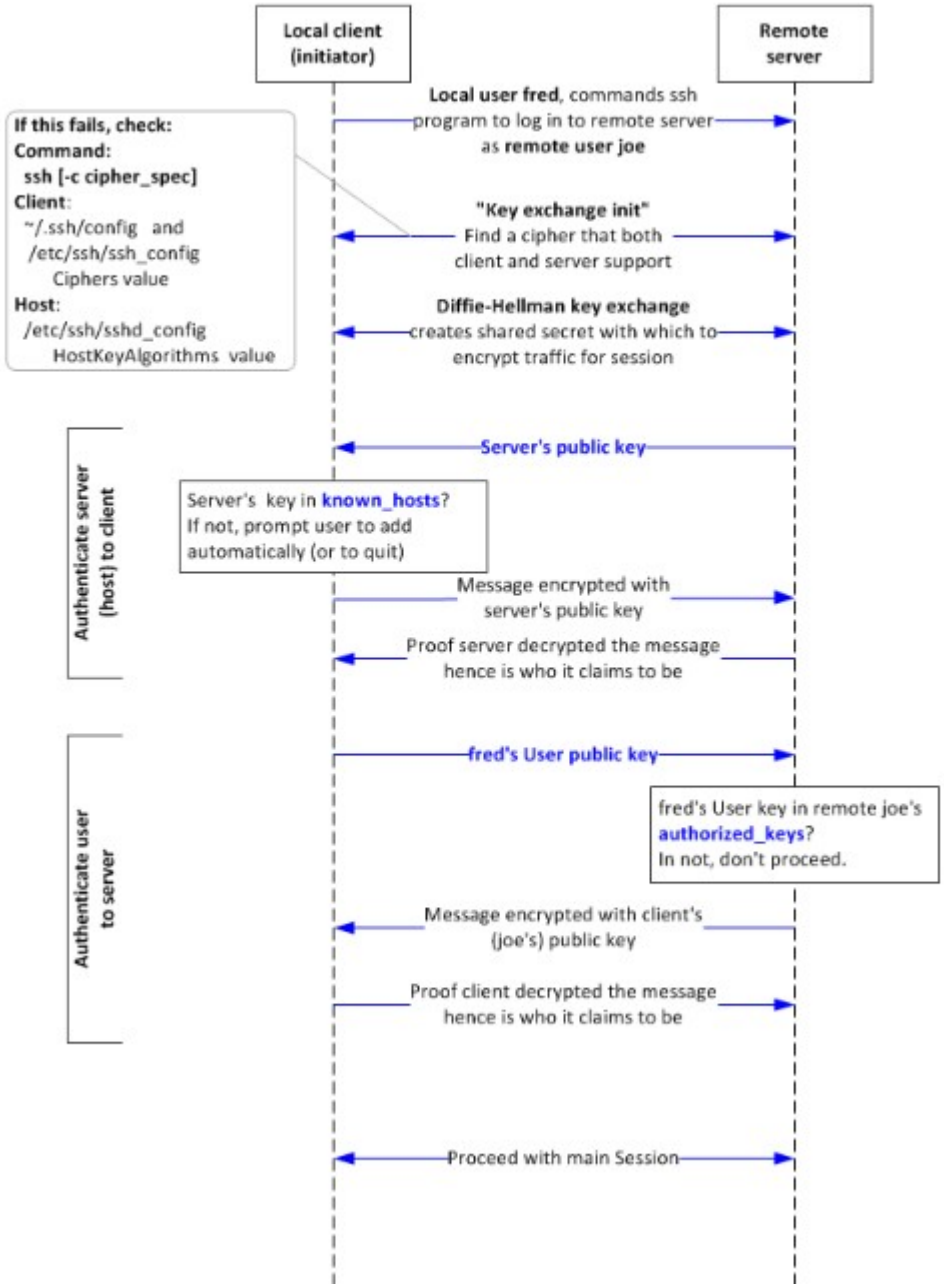
1. A user can, and often will, have more than one user key pair, in files with distinctive names. Commands that use key files allow specifying which files to use.

2. User account on the remote host need not have the same username as that on the local initiating machine. Ie: In local ssh or rsync command, specify the remote user account by which to log in.

3. Host key pair is created at previous installation time, such as when installing openssh.

Files involved in SSH connection: in use

gwideman 2018-10-14



SSH Messages

Old style message descriptions, e.g. RFC 4252 describes the value of 'signature' as a signature by the corresponding private key over the following data, in the following order:

```
    string    session identifier
    byte      SSH_MSG_USERAUTH_REQUEST
    string    user name
    string    service name
    string    "publickey"
    boolean   TRUE
    string    public key algorithm name
    string    public key to be used for
authentication
```

SSH things to note

- SSH is the epitome of the Trust On First Use (TOFU) model in how it handles known host keys on the client
- Be careful of VMs – easy to clone host key accidentally, e.g. if using puppet or similar
- In almost all cases: turn off password authentication and only allow public key based auth for systems you manage
- Be careful of user key pairs – it's very easy to end up with lots of old keys (e.g. for ex-employees!) in authorized_keys files
- SSH tunnelling is a fine thing – use a jump-off host to which you have a login to connect to others inside the LAN of the jump-off host – but as usual be careful in allowing this if you're a sysadmin

Wireguard-1

- Newish (2016), non-standard, work-in-progress VPN tunnelling protocol, but now (Jan 2020) accepted into upstream Linux kernel
- Protocol description: https://www.wireguard.com/papers/wireguard.pdf and/or https://www.wireguard.com/protocol/
- Design goals: simpler, quicker than IPsec or OpenVPN using only modern crypto, ~4K LOC in kernel
- `wg` tool acts like `iwconfig` for managing interfaces, e.g. `wg0`, use normal OS tools (e.g. `ip`) to create i/f and handle addresses and routes
- Local interfaces are associated with a static curve25519 key pair, remote peers with a public key, ephemeral curve25519 keys are generated in 1RTT handshake
- Has a cookie mechanism for DoS mitigation, triggered if responder is “under load”
- Optional PSK can be bound with key exchange for post-quantum future-proofing
- Traffic is protected using Chacha20poly1305 and encapsulated in UDP

Wireguard-2

- Public key distribution is out of scope – requires some other tooling, same as SSH
- Wireguard doesn't allocate IP addresses for clients – that's also considered out of scope and needs other tooling (apparently, haven't tried) – claimed to be problematic for some VPN operators who don't want to configure/log anything per-client
    - https://git.zx2c4.com/wg-dynamic was being developed in 2019, not sure of status
- Various timers and optional keepalives built-in (for NAT)
- If you want firewalling – just use a firewall (e.g. `ufw`) as if the traffic were in clear (no IPsec policy DB here:-)
- Protocol (designer) is v. opinionated – deliberately no crypto agility at all within protocol versions (that's also "modern":-)
- Performance and attack surface look good, will be interesting to see how this evolves

Mix'n'match

- PKI vs. Shared-secret vs. Trusted public keys
- TLS vs IPsec vs S/MIME vs CMS vs PGP vs Kerberos vs SSH vs Wireguard
- When should you pick which?

Next hour(s)…

- PKI model/protocols and DNS
- SMIME formats and (a bit on) email
- **TLS protocol (TLS1.2 or TLS1.3)**
- IPsec
- Kerberos
- SSH
- Wireguard
- MLS

Knowing **one** of these in detail (and PKI and DNS) is enough for exam purposes – you **will** need to read the source materials