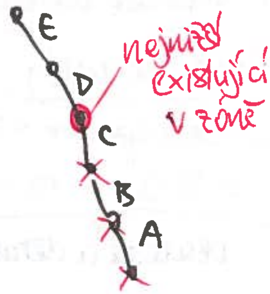
• Neexistence (důkaz) je ale složitější kvůli hranicím zón a wildcardům.

Pro A.B.C.D.E



Vygeneruji:
1. NSEC pro D.E dokazující, že D.E existuje a nemá NS
2. NSEC pro díru pokrývající celé jméno
3. NSEC dokazující neexistenci *.D.E

• Drobná vada na kráse: ~~všechny~~ záznamy v zóně jde pomocí NSEC enumerovat

→ NSEC3: místo jmen používá jejich heše → díry mezi heši.

Ověřování identity aneb kdo to je na druhém konci „drátu"?

- typicky znám veřejný klíč protistrany, ale nevím, komu patří

• PKI – Public Key Infrastructure
  - Certifikační autorita (CA): všichni jí věří a znají její veřejný klíč
    → ke každému klíči vydá certifikát = podepsanou zprávu s hešem veřejného klíče a identitou (a nějakým časovým omezením platnosti)
  - protistrana pak dodá podpis, veřejný klíč a certifikát
    - podpis ↑ ověřím veř. klíčem
    - veřejný klíč ↑ ověřím certifikátem
    - certifikát ↑ ověřím veř. klíčem CA
  - výhody:
    - CA nezná soukromé klíče (oproti Kerberovi)
    - CA je offline, k navázání spojení už není potřeba
    - PKI je univerzální, ověřuje identitu pro všechny aplikace
    - * lze decentralizovat zavedením sub-CA a řetězci certifikátů
  - problémy:
    - nenajdeme nikoho, komu věří všichni (ani většina)
    - co vlastně je identita? (Jan Novák, firma na Bahamách, ...)
    - revokace certifikátů (musí být aspoň částečně online)
      ↳ CRL / online protokol / krátkodobé certifikáty

• Trust On First Use – SSH, ale vlastně tak používáme i většinu webu

• Web Of Trust – PGP – vzájemné podepisování klíčů, důvěra částečně tranzitivní
  - je pro většinu uživatelů příliš složité

Co tedy funguje? - PKI specifická pro aplikaci (třeba klienti banky)
- TOFU + nezávislé ověření při FU

TLS: Transport Layer Security

- původně SSL vyvinuté firmou Netscape pro HTTPS, dnes všudypřítomné
- evoluce: SSL1 → SSL2 → SSL3 → TLS1.0 → 1.1 → 1.2 → 1.3
  - SSL1: nepublikováno
  - SSL2, SSL3, TLS1.0, 1.1: obsolétní a deravé
  - 1.2: dnes běžné (o něm budeme mluvit)
  - 1.3: horká novinka
- v podstatě meta-protokol, skoro vše je volitelné
  - šifrování + MAC — proudová šifra + MAC; blková šifra + MAC } MAC-then-encrypt (!)
    - AEAD (autentizovaná šifra - třeba AES-GCM)
  - PRF pro odvozování klíčů
    - ve starších verzích fixní (založená na HMAC)
    - dnes (1.2) volitelná
  - výměna klíčů: (generuje pre-master secret, z něj master-secret, z ně dalši klíče)
    - RSA: klient generuje klíč, zašifruje veř. klíčem serveru
    - DH+RSA: fixní parametry DH v certifikátu
    - DHE+RSA: Ephemeral DH, podepisuje parametry veř. klíčem
    - ECDHE+ECDSA: podobně s eliptickými křivkami
    - DHE-anon: bez ověření protistrany (MITM!)
    - PSK: pre-shared
    - DHE+PSK: místo certifikátu použije PSK
    - Kerberos
    - (a mnoho dalších)
  - a server a klient se dohodnou na cipher suite, třeba např.:
    TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
    (key xchg, auth, cipher, key size, mode, MAC & PRF)
  - komprese
- základem je Record Protocol:
  - posílá po TCP spojení záznamy, v nich zprávy dalších protokolů
  - záznamy mají: protokol, typ a délku
  - zajišťuje šifrování a MAC dohodnutými alg. (na začátku žádné)

- **Handshake Protocol**

K → S ClientHello
- verze protokolu (max. podporovaná)
- Client random
- podporované suity a kompresní algoritmy
- seznam rozšíření (typ + délka + hodnota)

← ServerHello
- vybraná verze protokolu
- server random
- vybraná suite a mód komprese
- seznam rozšíření

← [Certificate] - certifikát serveru

← [ServerKeyExchange] - závisí na zvoleném algoritmu pro KX

← [CertificateRequest] - chceme i auth. klienta

← ServerHelloDone - deklarace, že server je hotov s KX

→ [Certificate] - cert. klienta

→ ClientKeyExchange - klientova část KX (povinná)

→ [CertVerify] - podpis dosavadních zpráv certifikovaným klíčem

→ ChangeCipherSpec - klient deklaruje přechod na novou šifru
(pozor, tohle je samostatný sub-protokol)

→ Finished - handshake complete
- podpis: PRF(master secret, "client finished" | hash(handshake messages))

↳ spočítá se z premaster secretu a server/client random

← ChangeCipherSpec

← Finished - teprve tímhle se u RSA auth ověří, že server má soukromý klíč ke svému certu

- **zajímavá rozšíření**

- session resume - ServerHello obsahuje ID, pod kterým server ukládá session

další spojení:
- ClientHello požádá o resume s dřívějším ID
- zkrácený handshake
- nové klíče
- ~~nový master secret~~ se odvodí ze starého mast. secretu a nových náhodných dat

jen 2x hello a 2x finished

- session tickets – podobné, ale celý stav si pamatuje klient (zašifrovaný serverovým klíčem)
- server name – pro virtual hosting (viz Hosts v HTTP)
- ALPN (app-level protocol negotiation ... třeba HTTP 1 vs. 2 vs. SPDY)
  - klient nabídne protokoly, server vybere jeden
- **Re-negotiation** – lze iniciovat opětovné spuštění dohadování (od Hello)
  - třeba po přenesení pár GB dat (chceme nové klíče)
  - nebo jsme v průběhu HTTP zjistili, že chceme klientův certifikát
  - TLS ≤ 1.2 má v re-neg zásadní bug: nepodepíše návaznost na předchozí nego. ⇒ elegantní MiTM útok
    - navážu spojení se serverem, pošlu část dat, spustím reneg.
    - pak propojím se skutečným klientem, ten nego dokončí
    - ⇒ umím vnutit prefix relace (třeba HTTP GET, k němuž klient doplní cookies nebo svůj cert)
  - secure reneg. extension ⇒ doplňuje návaznost do podpisů
- **Close Alert** – podepsané ukončení spojení
  - klienti často ignorují ⇒ cookie cutting attack

Útoky na SSL/TLS

- BEAST (Browser Exploit Against SSL/TLS)
  - TLS ≤ 1.0 používá CBC s jednou IV – celé spojení je 1 posloupnost bloků
  - tím pádem víme, jaká IV se použije pro další zprávu
    - ⇒ 1. blok další zprávy je efektivně ECB
  - CPA: umíme zjistit, zda se CP zašifruje stejně jako některý z předchozích bloků
  - navíc můžeme CP paddingem zařídit, aby předch. blok obsahoval hodně známých dat + trochu tajných (třeba 1. znak cookie)
- Compression side-channels
  - ⇒ CRIME (Compression Ratio Info-Leak Made Easy)
  - ⇒ chceme vypnout kompresi

  } útočíme na cookies, XSRF tokens atd
- Lucky 13 – CBC padding oracle (MAC-then-encrypt)
- POODLE (Padding Oracle On Downgraded Legacy Encryption)
  - mnoho implementací lze donutit k downgrade na SSL3

- SSL3 nekontroluje obsah paddingu
- zařídíme, aby posl. blok obsahoval jen padding
  → poslední bajt bloku je B−1, předchozí libovolné
- zašifrovaný blok vyměníme za jiný (o němž chceme něco zjistit)
  → s Pr = 1/256 vyjde po dešifrování a XORu s předch. blokem B−1 na konci; jinak nesedí MAC a spojení se rozpadne
  → tehdy zjistíme posl. bajt vybraného bloku (XOR...)
  → pak posuneme plain-text (CPA) a iterujeme...

- DROWN (Decrypting RSA using Obsolete and Weakened eNcryption)
  - ve starších verzích funguje Bleichenbacherův útok
  - pokud server umí více verzí, použijeme starou jako orákulum na lámání nové
    (se stejným certem)

- ROBOT (Return Of Bleichenbacher Oracle Threat)
  - i TLS 1.2 pořád používá PKCS #1 v1.5, ale s work-aroundy proti Bleich. útokům
  → ještě stále se najdou varianty útoku, které fungují

- Shrnutí:
  - nechceme používat blok. šifry s CBC → buď proudové nebo GCM
  - chceme zakázat obsoletní verze a suity
  - jsou potřeba rozšíření protokolu

Internet PKI

- PKI založená na standardu ITU X.509 ← obskurní, překomplikovaný, ASN.1, původně určený pro jiný svět (ISO/OSI, X.500)

- cert. autority
  - typicky komerční – co je jejich zájmem?
  - pár neziskových – hlavně Let's Encrypt
  - je jich mnoho (Firefox momentálně uznává 181 kořenových certů!)
    → jak pravděpodobné je, že všechny CA jsou
      a) poctivé,
      b) dostatečně důsledné?

- mezilehlé (intermediate) certifikáty
  - podepsané root klíčem, dál podepisují → cert chains
  - některé používá sama CA, jiné deleguje (?) ! složité, ve validaci čask chyby
  - mohou mít omezení na domény / pověření
- jiný distribuovaný model: 1 CA, více registračních autorit

- typy certifikátů: DV = domain-validated (držitel měl pod kontrolou danou doménu)
  OV = organization-validated (legal entity)
  EV = extended validation
- certifikát obsahuje:
  - subject (x.500 DN !)
  - subject alt. names - domény, e-mail. adresy &c
  - heš k veřejného klíče
  - identifikaci vydavatele & podpis
    ↳ vlastně vadřazený cert (root je self-signed)
  - pověření: server / klient / code signing / CA / ...
  - časový interval validity
  - množina rozšíření

- <u>revokace certifikátů</u>:
  - CRL (Cert Revocation List) - velké seznamy, pomalý download → aktualizují se zřídka
  - OCSP (Online Cert Status Protocol) - cert odkazuje na OCSP responder
    - problémy se soukromím (nešifrováno, jen podpisy odpovědí)
    - pomalé a nespolehlivé → klienti dělají soft-fail ↯ (triviální MiTM útok)
    - → TLS extension: OCSP reply stapling
    - → cert extension: must-staple (zatím ji klienti moc neumí...)
  - Google: CRLset
  - Mozilla: OneCRL
    } proprietární seznam, klientem průběžně stahovaný, automaticky se do něj propagují revokace klíčů CA a "high-profile" sítes

... a Chrome dnes ani klasický OCSP nepoužívá 😮

- CA/Browser fórum: stanovuje požadavky na CA
  → pravidla, povinné audity atd.
  → za vážná porušení prohlížeče CA blacklistují (už se párkrát stalo)
- opatření proti podvodně vydaným certům
  - Perspectives – pozorování certů z více míst v síti
  - public key pinning
    - Google v Chrome pinuje klíče svých domén (nečekaně úspěšné)
    - HPKP (HTTP Public-key Pinning): pin v hlavičce odpovědi
      [dosti křehké ...]
    - DANE (DNSSEC Authentication of Named Entities)
      [elegantní, ale odmítané autory prohlížečů kvůli latenci]
  - CAA v DNS – záznam omezující, která CA smí vydávat certy
  - Certificate Transparency (CT)
    - veřejné logy vydaných certů – Merkleovy stromy, lze snadno kontrolovat konsistenci
    - vyhledávač crt.sh
    - CA/B fórum nařizuje pro EV certy a intermediates, občas také za trest 😉
    - HTTP: "Expect-CT" v odpovědi
- problémy s mícháním obsahu na HTTPS a HTTP → warningy
  ... ale co míchání DV a EV certifikace?
- uživatelé často spoléhají na HTTP redirect na HTTPS
  - to by také mohlo řešit DANE, ale...
  - HTTP Strict Transport Security (STS)
    - v hlavičce odpovědi: "zapamatuj si, že tady máš používat jen HTTPS"
    - ale neřeší to first use
  - plugin HTTPS Everywhere → nespolehlivé, občas je na HTTP a HTTPS jiný obsah