Kriptográfia és Információbiztonság
8. előadás

MÁRTON Gyöngyvér

Sapientia Egyetem, Matematika-Informatika Tanszék
Marosvásárhely, Románia
mgyongyi@ms.sapientia.ro

2024

Miről volt szó az elmúlt előadáson?

- a diszkrét logaritmus (DL) probléma, a DL feltételezés
- hatványok és generátor elemek
- a Diffie-Hellman kulcscsere
- DL problémán alapuló digitális aláírások:
  - az ElGamal digitális aláírás
  - a DSA (Digital Signature Algorithm) vagy DSS (Digital Signature Standard)
- elliptikus görbéken alapuló kriptográfia

Miről lesz szó?

- Webbiztonság
- SSL/TLS
- HTTPS
- end-to-end encryption: Signal

Webbiztonság (web security)

a WWW lényegében egy kliens/szerver alkalmazás:

- a webböngészők használata egyszerű,
- a webszerverek konfigurálása és kezelése egyszerű,
- a webtartalom fejlesztése egyre egyszerűbb,
- a háttérben levő szoftver komplex:
  - biztonsági hibák
  - sebezhető a különféle támadásokkal szemben
- webszerver: sikeres támadás esetén a támadó hozzáférhet a lokális szerveren tárolt adatokhoz

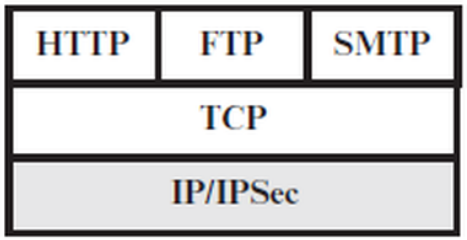
Webbiztonság

- hálózati rétegben (network level)
  - átlátható a végfelhasználók és az alkalmazások számára
  - általános célú biztonsági megoldások
  - szűrési lehetőségek: csak a kiválasztott adatforgalom engedélyezett
  - az operációs rendszer része, módosításához az operációs rendszert is módosítani kell
- szállítási rétegben (transport level): a webes tranzakciók biztonságának a megvalósítására használják
- alkalmazási rétegben (application level):
  - a szolgáltatások egy adott alkalmazásba vannak beágyazva
  - a szolgáltatás az adott alkalmazás speciális igényeihez szabható
- a tartalmak jogosulatlan másolása:
  - vízjelek (speciális minta) beágyazása
  - igazolható a tulajdonjog, azonosítható a jogsértő személy
  - nyomon követhető az üzenet/dokumentum terjesztése
  - tájékoztathatók a felhasználók az adatok jogosultságáról

Webbiztonság, hálózati rétegben

a hálózati rétegben megvalósuló webbiztonságot az IP security (IPsec) biztosítja

- előnye: átlátható a végfelhasználók és az alkalmazások számára
- általános célú megoldást kínál, szűrési lehetőséggel: csak a kiválasztott forgalom kell átmenjen az IPsec-feldolgozáson
- az operációs rendszer része, ezért módosításához az OP-t is módosítani kell
- gyakran használják virtuális magánhálózatok - Virtual Private Network VPN létrehozására

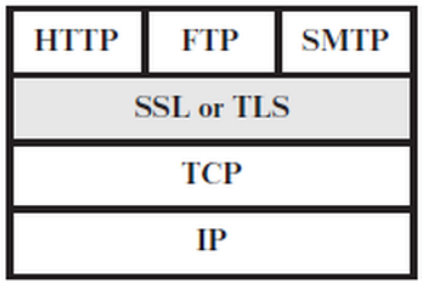


(a) Network level

Webbiztonság, szállítási rétegben

a szállítási rétegben megvalósuló biztonságot a TCP felett implementálják

- a Secure Sockets Layer (SSL) és az azt követő Transport Layer Security (TLS) biztosítja a biztonságot
- hasonlóan az IPsec-hez titkosítást, integritásvédelmet és hitelesítést biztosít, csak egyszerűbb, és elsősorban a webes tranzakciók biztonságának a megvalósítására használják,
- két implementációs lehetőség ismert:
  - az SSL/TLS az alapul szolgáló protokollcsomag része, ezért átlátható az alkalmazások számára
  - az SSL/TLS meghatározott csomagokba van beágyazva, például minden böngészőben benne van a TLS

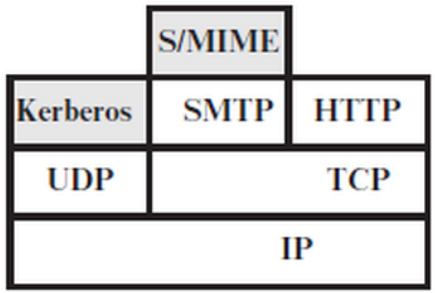


(b) Transport level

Webbiztonság, alkalmazási rétegben

az alkalmazási rétegben megvalósuló biztonság

- a biztonsági szolgáltatások egy adott alkalmazásba vannak beágyazva
- előnye, hogy a szolgáltatás az adott alkalmazás speciális igényeihez szabható
- a Secure/Multipurpose Internet Mail Extension (S/MIME) az RSA Data Security technológiáján alapuló MIME internetes e-mail formátumszabvány továbbfejlesztése, több szabvány is leírja, például RFC 5322
- Kerberos: elnevezése a görög mitológiából ered, hitelesítést biztosít szimmetrikus kriptot használva, általában lokális hálózatokban használják



(c) Application level

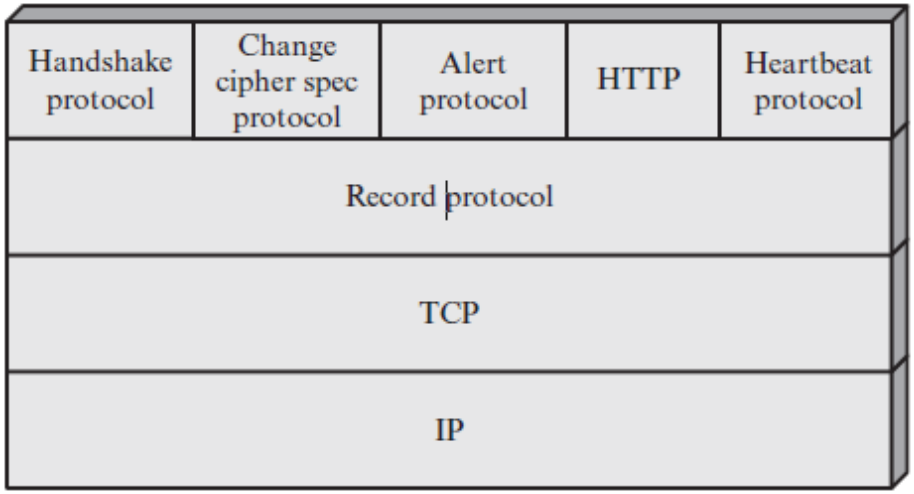
Az SSL és a TLS

- kliens/szerver alkalmazásokban a kommunikáció lehallgatása és manipulálása ellen nyújt védelmet
- a TLS az első SSL specifikációknak is része volt (1994, 1995, 1996), amelyeket a Netscape Communications fejlesztett a saját webböngészői számára
- az közzétett verziója alapvetően az SSLv3.1-nek tekinthető
- az IETF által ajánlott internetes szabvány
- először 1999-ben írták le, a jelenlegi standard a TLS 1.3, ezt 2018-ban definiálták,
- legszélesebb körben használt standard leírása az rfc8446 dokumentumban található: *link*
- úgy volt tervezve, hogy TCP-t használva megbízható, teljes körű biztonságos szolgáltatást nyújtson
- a TCP-n alapuló protokollok részeként valósítják meg
- TLS elleni több támadást is leíró dokumentum: *link*

A TLS

A TLS nem egyetlen protokoll, hanem két protokollrétegből áll:

- Record Protocol alapvető biztonsági szolgáltatásokat nyújt a különböző magasabb szintű protokollokhoz, például a Hypertext Transfer Protocol (HTTP) adatátviteli szolgáltatást biztosít a webes kliens/szerver interakcióhoz, ez a TLS-n felül is működhet
- a Handshake, a Change Cipher Spec, az Alert, és a Heartbeat protokollok további magasabb szintű protokollok



A TLS

Két fontos TLS-fogalom:

- TLS-kapcsolat (TLS-connection):
  - peer-to-peer típusú kapcsolat (a hálózat végpontjain levő eszközök közvetlenül egymással kommmunikálnak),
  - átmeneti
  - minden kapcsolat egy munkamenethez van társítva
- TLS-munkamenet (TLS-session):
  - a Handshake Protocol hozza létre,
  - a biztonsági paraméterek készletét határozza meg,
  - a paraméterek több kapcsolat között megoszthatók

A TLS

TLS-kapcsolat (connection):

- a kliens/szerver alkalmazásokban az alkalmazásoknak kérnie kell a szervertől a TLS-kapcsolat létrehozását (anélkül is tudnak kommunikálni)
- a TLS kapcsolat létrehozásához:
  - különböző portszámok, pl.:
    - a 80-as portot a titkosítatlan HTTP-forgalomhoz,
    - a 443-as portot a titkosított HTTPS-forgalomhoz
  - a kliens protokoll-specifikus kérést küld a szervernek, hogy kapcsoljon át TLS-re; például STARTTLS kérést kezdeményez a levelezési, a hír-protokollok esetében
- a felek között több biztonságos TLS-kapcsolat is létezhet

A TLS

TLS-munkamenet (session):

- a munkameneteket a Handshake protokoll hozza létre
- egy munkameneten belül a felek több kriptográfiai paraméterben is megegyeznek, amelyeket több kapcsolaton belül is lehet használni
- elméletileg egyidejűleg több munkamenet is előfordulhat a felek között, de ezt a funkciót a gyakorlatban nem használják
- az egyes munkamenetekhez több állapot tartozik: aktuális működési állapot íráshoz/olvasáshoz, függőben lévő működési állapot íráshoz/olvasáshoz
- költséges, mert minden session alkalmával publikus kulcsok cseréjére kerül sor
- ha már létezik egy TLS-session, akkor egy TLS-connectiont hatékonyan ki lehet építeni: a felek már osztoznak egy szimmetrikus kulcson, amelyet fel lehet használni új connectionok kiépítésére

A TLS

Record Protokoll: hitelesített titkosítást végez (authenticated encryption), a bizalmas kommunikációt és az üzenetek sértetlenségét teszi lehetővé

- a Handshake Protokollban létrehozott szimmetrikus/titkos kulcsot használja a titkosításhoz
- a használt titkosítási algoritmusok:
  - AES-GCM: 128 vagy 256 bites kulcsméret,
  - ChaCha20-Poly1305.

Change Cipher Spec Protokoll:

- egyetlen egy bájtos üzenetből áll; jelzi, ha a munkamenet titkosítási módját egy másik rekord módosítja

Alert Protokoll:

- két bájtból áll, az első warning, vagy fatal értéket jelent, a második a riasztás kódját tartalmazza
- normal handshake vagy alkalmazáscsere esetén nem kerül elküldésre, de bármikor amikor riasztás szükséges elküldhető
- a munkamenet lezárását is maga után vonhatja

A TLS

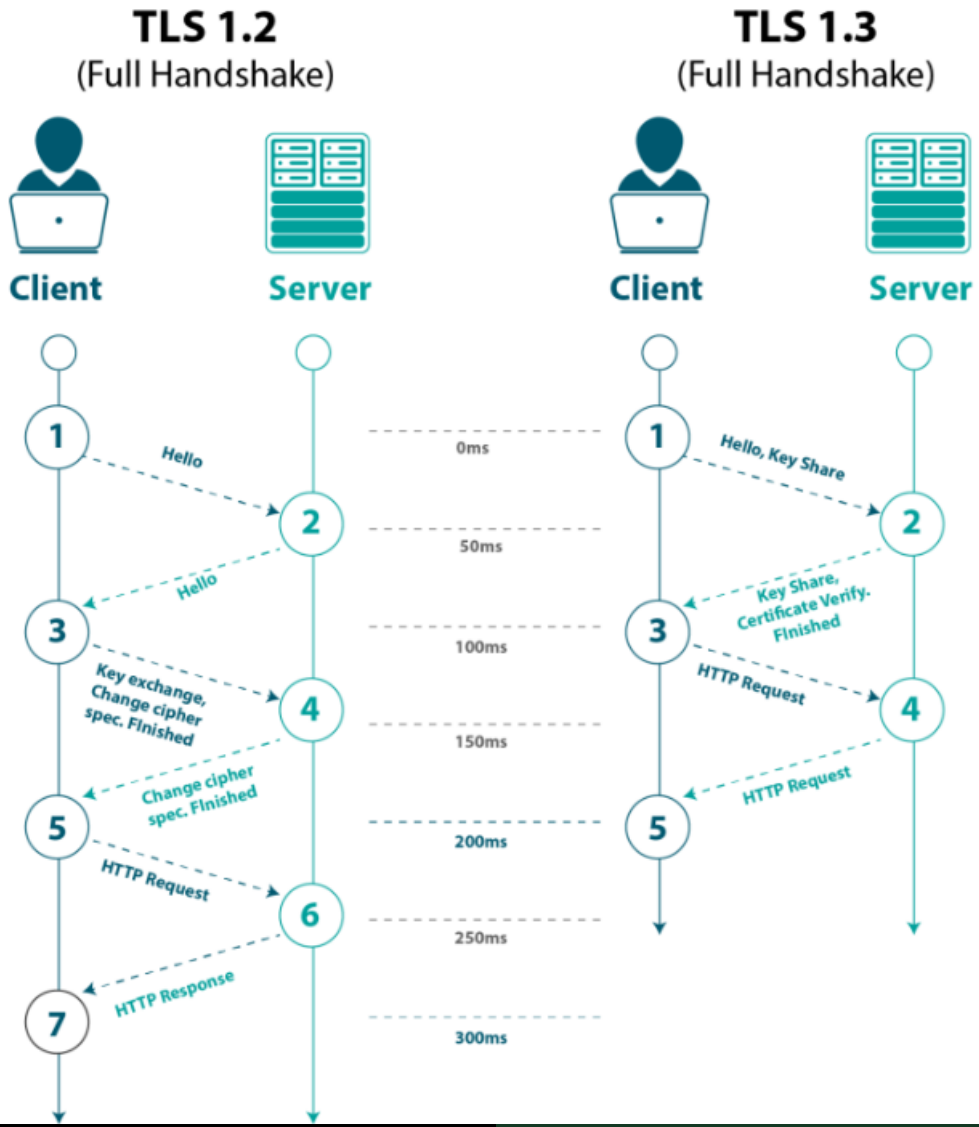
Heartbeat Protokoll: 2012-ben határozták meg az RFC 6250-ben

- két célt szolgál:
  - biztosítja a feladót, hogy a címzett még mindig *életben* van, még akkor is, ha egy ideig nem volt semmilyen tevékenység
  - aktivitást generál a kapcsolaton keresztül tétlenségi a időszakokban, így elkerülhető a tétlen kapcsolatokat nem toleráló tűzfal azonnali bezárása
- egy hardver vagy szoftver által generált periodikus jel, amely jelzi a normál működést, vagy a rendszer más részeinek szinkronizálását végzi

Handshake Protokoll: a TLS legösszetettebb része

- lehetővé teszi a szerver és a kliens kölcsönös hitelesítését
- lehetővé teszi a szerver és kliens közti kulcscserét
- lehetővé teszi, hogy a felek megállapodjanak, hogy milyen hitelesített titkosítót használjanak
- ugyanahhoz a szerverhez történü ismételt csatlakozás esetében: fast-track secure session

A TLS Handshake Protokoll



A TLS Handshake Protokoll

- a kliens kezdeményezésére elkezdődik az egyezkedés (negotiation), a kliens és a szerver is konfigurálható:
  - kinek mi a verzió száma?
  - ki milyen titkosító algoritmust fogad el?
  - cél megegyezni, hogy a továbbiakban biztonságosan történhessen a kommunikáció
- kulcscsere (key exchange):
  - a Handshake lényege, a leggyengébb pontja minden TLS-nek
  - FREAK - *Faktoring RSA Export Keys*:
    - az USA korlátozta az RSA kulcsméretet max 512 bitre
    - 1990-től kezdődött, 2015-ig tartott
  - LOGJAM - *Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice*:
    - 2015-ben egy protokoll tervezési hiba, és az 512 bites DH kulcsméret miatt volt lehetséges volt a ManInTheMiddle támadás
    - *Critical Review of Imperfect Forward Secrecy*
  - DROWN - *Decrypting RSA with Obsolete and Weakened eNcryption*: a Bleichenbacher támadás újragondolása

A TLS Handshake Protokoll

Kulcscsere módszerek a TLS 1.3-ban:

- RSA kulcscsere: a korábbi TLS verziók tartalmazták, de a TLS 1.3-ban már nem szerepel
- FFDH: Diffie–Hellman típusú kulcscsere 2048 bites prímszám használatával, az rfc7919-ben található leírás alapján
- ECDH: elliptikus görbén alapuló kulcscsere: `P-256`, `P-384`, `P-521`, `X25519`, `X448` típusú görbék használatával

Digitális aláírások a TLS 1.3-ban:

- RSA PKCS#1 1.5 verzió
- RSA-PSS, ahol a hash függvény SHA-256 vagy SHA-512 lehet
- ECDSA és EdDSA: elliptikus görbéken alapuló digitális aláírás

Hash függvények a TLS 1.3-ban:

- az SHA-2-nek két változtatát lehet használni: SHA-256, SHA-384
- a digitális aláírásokhoz mindig meg van adva, hogy milyen hash-t kell használni, ezeket a hash-ek többi kripto primitív esetén is használják

A TLS Handshake Protokoll

TLS 1.2 kulcscsere:

- csak azután kezdődött el miután megállapodtak a kulcscsere módszerben

TLS 1.3 kulcscsere:

- az egyezkedés és kulcscsere egyszerre történik:
  - a kliens az első lépésben már kiválaszt egy általa támogatott kulcscsere módszert
  - ha ez nem jár sikerrel, akkor újabb `Hello, Key Share` üzenetet küld
- forward secrecy:
  - ephemeral kulcs: a különböző munkamenetekhez generált különböző kulcsok, amelyeket a session-ok végén megsemmisít
  - egy munkamenet kulcsának a kompromitálása nem vonja maga után a korábbi munkamenetek kompromitálását
- a kulcscsere után minden további üzenetváltás már titkosítva történik (a szerver paraméterek megosztása, a hitelesítés), ez nem így volt a TLS 1.2-ben

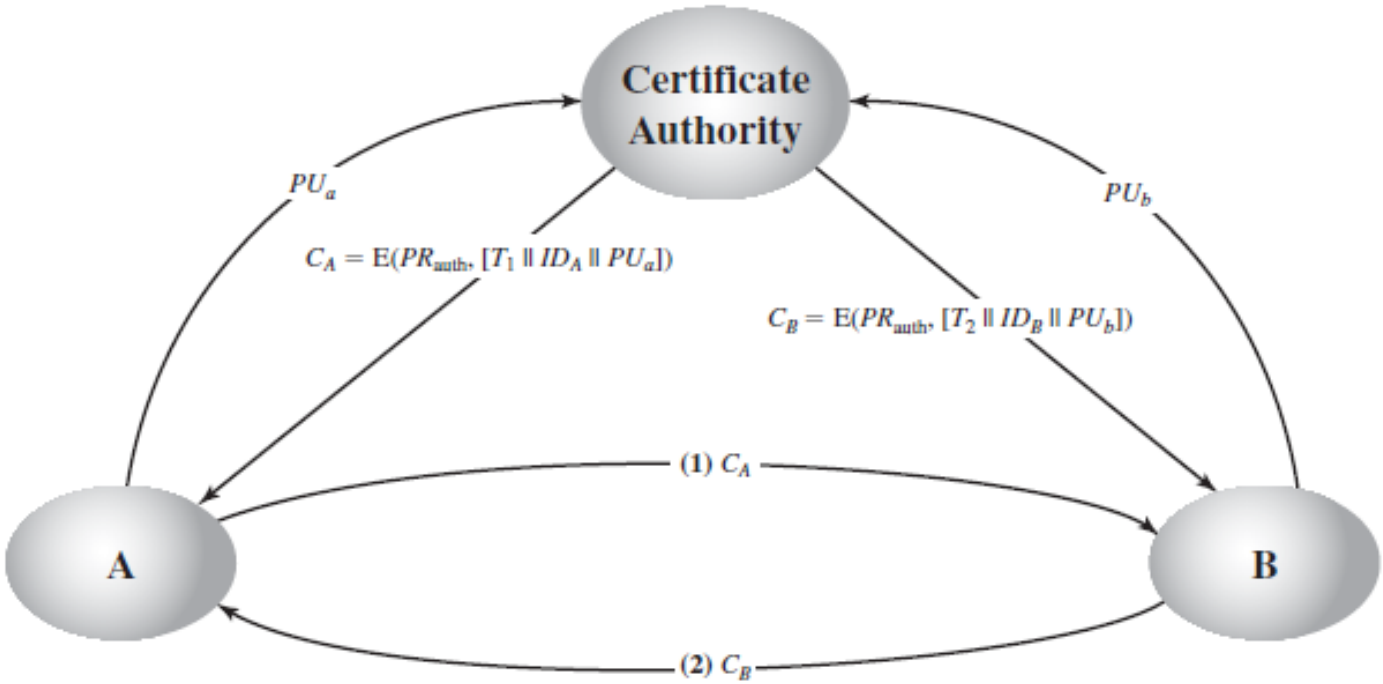
A TLS Handshake Protokoll

TLS 1.3 kulcscsere:

- minden munkamenethez a szerver/kliens más-más random értéket használ
- HKDF-t (Hash Key Derivation Function) használ, hogy a megosztott kulcsból és a random értékekből további, különböző kulcsokat generáljon: HMAC alapú, SHA-2 hash-t használ (4. előadás)
- lehetőség van PSK handshake-re (pre-shared keys):
  - a publikus kulcs infratruktrúrát kerüli ki
  - a kliens erre külön figyelmezteti a szervert
  - akkor lehetséges, ha a szerver és a kliens mindegyike ismer egy közös PSK értéket, amiből származtatni tudják a szimmetrikus kulcsot
  - ilyenkor elmarad a kulcscsere, a hitelesítés

A TLS Handshake Protokoll

A publikus kulcsok hitelesítése tanúsítványok használatával történik (public-key certificates):



Publikus kulcsok megosztása

Publikus tanúsítványok használata:

- egy tanúsítvány a következőket tartalmazza: a publikus kulcsot, a kulcstulajdonos azonosítóját és ezen adatok digitálisan aláírt értékét
- a tanúsítványt egy megbízható intézmény/hatóság kell kiállítsa
- a tanúsítvány kiállítása azt jelenti, hogy a felhasználó (pl. egy webszerver) bemutatja publikus kulcsát a hatóságnak, amit az *aláír* alkalmazva a saját privát kulcsát
- az aláírt tanúsítványt, az aláírt értékkel a felhasználó ezután közzé teszi
- ha egy felhasználó publikus kulcsát használni szeretnénk (pl. a böngészőnk akarja használni), akkor le kell ellenőrizni, ami azt jelenti, hogy a hatóság publikus kulcsát használva el kell végezni az aláírás-ellenőrzési folyamatot
- ha az ellenőrzés sikerrel jár az azt jelenti, hogy a publikus kulcs tulajdonosa az akinek mondja magát, ezután tehát biztonságosan használható kulcseréhez a publikus kulcs
- ne tévesszük össze a hatóság publikus/privát kulcspárját a felhasználó publikus privát kulcspárjával

HTTPS

- a HTTPS (HTTP over SSL) a HTTP és az SSL kombinációjára utal, nem egy önálló protokoll, a HTTP felett fut és a biztonsági megoldásokra a TLS/SSL szabványt használja
- a felhasználó számára biztosítja a kommunikáció titkosságát, integritását és hogy a felkeresett weboldal hiteles, pl. védett a kommunikáció a *man-in-the-middle* típusú támadásokkal szemben
- a webszerver hitelesítését harmadik fél által kibocsátott tanúsítványok használatával oldja meg
- a biztonságos kommunikáció azonban csak akkor valósul meg, ha megfelelő titkosítási csomagok kerültek kiválasztásra, illetve megtörtént a webszerver hitelességének ellenőrzése
- minden modern webböngészőbe be van építve, használata webszervertől függ: egyes keresőmotorok például nem támogatják a HTTPS-t
- 2016 óta használják egyre szélesebb körben, dokumentációja az RFC 2818-ben található: *link*

A HTTP és HTTPS közötti különbségek

HTTP:

- az URL-cím (uniform resource locator - egységes erőforrás-kereső): `http://`-vel kezdődik
- a 80-as portot használja
- az OSI modell szerint az alkalmazási rétegben helyezkedik el
- nem használ titkosítást, hitelesítést
- sebezhető a *man-in-the-middle* típusú támadással
- a biztonsági beállítások hiánya miatt gyorsabb
- információ megosztásra használt weboldalak esetében használják
- fenntartása nem jár költségekkel

HTTPS:

- az URL-cím: `https://`-vel kezdődik
- a 443-as portot használja
- az OSI modell szerint a szállítási rétegben helyezkedik el
- bizalmas információ cserét, integritást, hitelesítést biztosít
- nem támadható *man-in-the-middle* típusú támadással
- az alkalmazott biztonsági beállítások miatt lassúbb
- *érzékeny* adatokat (jelszavak, kártya adatok, stb) kezelő weboldalak esetében kell használni,
- fenntartása költségekkel jár: az tanúsítványokért fizetni kell

HTTPS

HTTPS esetén a következő elemek vannak titkosítva:

- a kért dokumentum URL címe
- a dokumentum tartalma
- a böngésző űrlapok tartalma (a böngésző felhasználója tölti ki)
- a böngészőről a szerverre és a szerverről a böngészőre küldött cookie-k
- a HTTP-fejléc tartalma

Akkor lehet megbízni egy weboldalban, amelyet HTTPS kapcsolaton keresztül értünk el, ha a következőkben is megbízunk:

- a böngésző helyesen implementálta a HTTPS-t, illetve az előre telepített tanúsító hatóságokat
- a tanúsító hatóság csak legális webhelyeket garantál
- a weboldal érvényes tanúsítvánnyal rendelkezik
- a tanúsítvány helyesen azonosítja a weboldalt
- a titkosító algoritmusok elég *erősek*


HOW
HTTPS.
WORKS

Végpontok közötti titkosítás (End-to-end encryption)

- miért nem elég a TLS?
  - a rendszerek/eszközök/felek közötti biztonságos kommunikációt biztosítja,
  - meg kell bízni egy központi hatóságban, a CA-ban,
  - a szerver/szerverek előtt nincs rejtve a kommunikáció,
  - a visszafejtés után a kommunikáció tartalma ismert a szerverek számára
- Signal, *link*:
  - a kommunikáló felek (Alice, Bob) egy szervert használva egyeznek meg a közös kulcsban
  - aszinkron (az egyik fél úgy is kezdeményezhet kommunikációt, ha a másik fél offline van) üzenetváltást tesz lehetővé, ezért a kulcscsere nem interaktív
  - különbség van aközött, hogy Alice kezdeményezi a kommunikációt, illetve amikor Bob
  - forward secrecy-t biztosít

Signal

- trust on first use (TOFU):
  - minden crypto alkalmazásban szükség van egy kiinduló pontra, root of trust-ra, amely alapján fel lehet építeni a további funkcionalitásokat
  - az első adatcsere nem baj ha nem biztonságos, mert a későbbiekben le lehet ellenőrizni, hogy állt-e fenn MITM,
  - az első nem biztos, hogy biztonságos adatcserét követően azonban garantálja, a kommunikáció további adatcseréjének biztonságát, mert ezt később bármikor le lehet ellenőrizni,
- extended Triple Diffie-Hellman (X3DH) kulcscsere: a forward secrecy megvalósítása minden beszélgetés megkezdésekor,
- double ratchet (dupla racsni):
  - symmetric ratchet: a forward secrecy megvalósításához minden egyes üzenetváltáskor,
  - DH ratchet: post-compromise security (PCS) megvalósításához, megoldja azt a helyzetet amikor kompromitálódik két felhasználó szimmetrikus kulcsa,

Signal - TOFU (trust on first use)

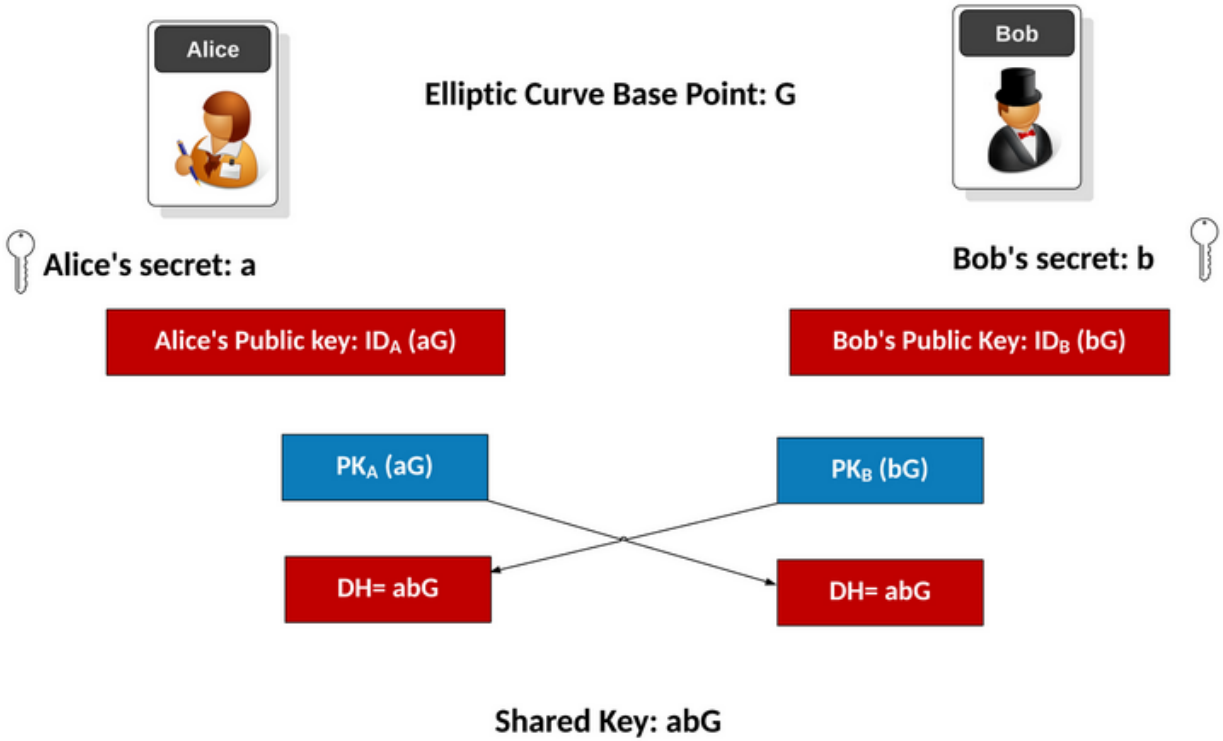
- ha az első kapcsolatlétesítéskor kicsi az esélye annak hogy MITM támadás lépjen fel,
- az első kulcscserekor megállapodott kulcsban bíznak meg, és minden további változást elutasít
- a gyakorlatban megadott lenyomatok (fingerprint) vagy QR kódok értékeit hasonlítják össze:
  - az egyik fél felhasználóneve után hozzáfűzik a telefonszámát és ennek meghatározzák a hash értékét:

    $$\text{Hash}(\text{telefonszám} \,||\, \text{azonosítókulcs})$$

  - majd a másik fél esetében is ugyanezt meghatározzák,
  - a kapott két érték konkatenációjából kapott értéket hasonlítják össze, ez kell megegyezzen

Signal - egy DH

A Signal három vagy több DH-t kombinál *egy* kulcscsere során, ahol egy DH:



Kép forrása: https://asecuritysite.com/encryption/go_x3dh

Signal - X3DH

többféle kulcs típust használ, ezek a következők, Alice estében A lesz az index (Bob esetében B lesz az index):

- identity key (azonosító kulcs): hosszútávú kulcspár, ahol a privát kulcs (a), a publikus kulcs ($IK_A = aG$)
- one-time signed prekeys (egyszer használatos aláírt előkulcsok):
  - privát előkulcs: a',
  - publikus előkulcs: $SPK_A = a'G$
  - ezeket periodikusan (pl hetente) szükséges lecserélni, cserekor Alice törli a használtat,
  - a publikus előkulcsot aláírja a privát kulcsával
- one-time prekey (egyszer használatos előkulcsok):
  - privát kulcs: a_1, publikus kulcs $OPK_A = a_1 G$,
  - ezeket egyszeri használat után töröli a szerver
- ephemeral key (egyszer használatos kulcs):
  - short term secret, egyszeri privát kulcs: a'
  - ephemeral key: $EK_A = a'G$
  - minden protokoll véletlenszerűen, és újat generál
  - használat után Alice törli

Signal - X3DH

Regisztrációkor a felhasználó, legyen ez most Alice elküldi a szervernek:

- az azonosítókulcsát: IK_A
- egy aláírt előkulcsot: SPK_A, és ennek az aláírását: $Sig(a, SPK_A)$
- több egyszer használatos előkulcsot: $OPK_A^1, OPK_A^2, OPK_A^3, \ldots$
- az IK_A-t egyszer küldi el a szervernek
- hetente/havonta újraküldi az előkulcsot és az aláírt értékét, melyekre a szerver lecseréli a régit
- ha a szerver kéri (ha elfogytak az egyszer használatos előkulcsok), akkor további egyszerhasználatos előkulcsokat küld a szervernek

Signal - X3DH

Kulcscserekor, ha Alice kezdeményezi a kulcscserét:

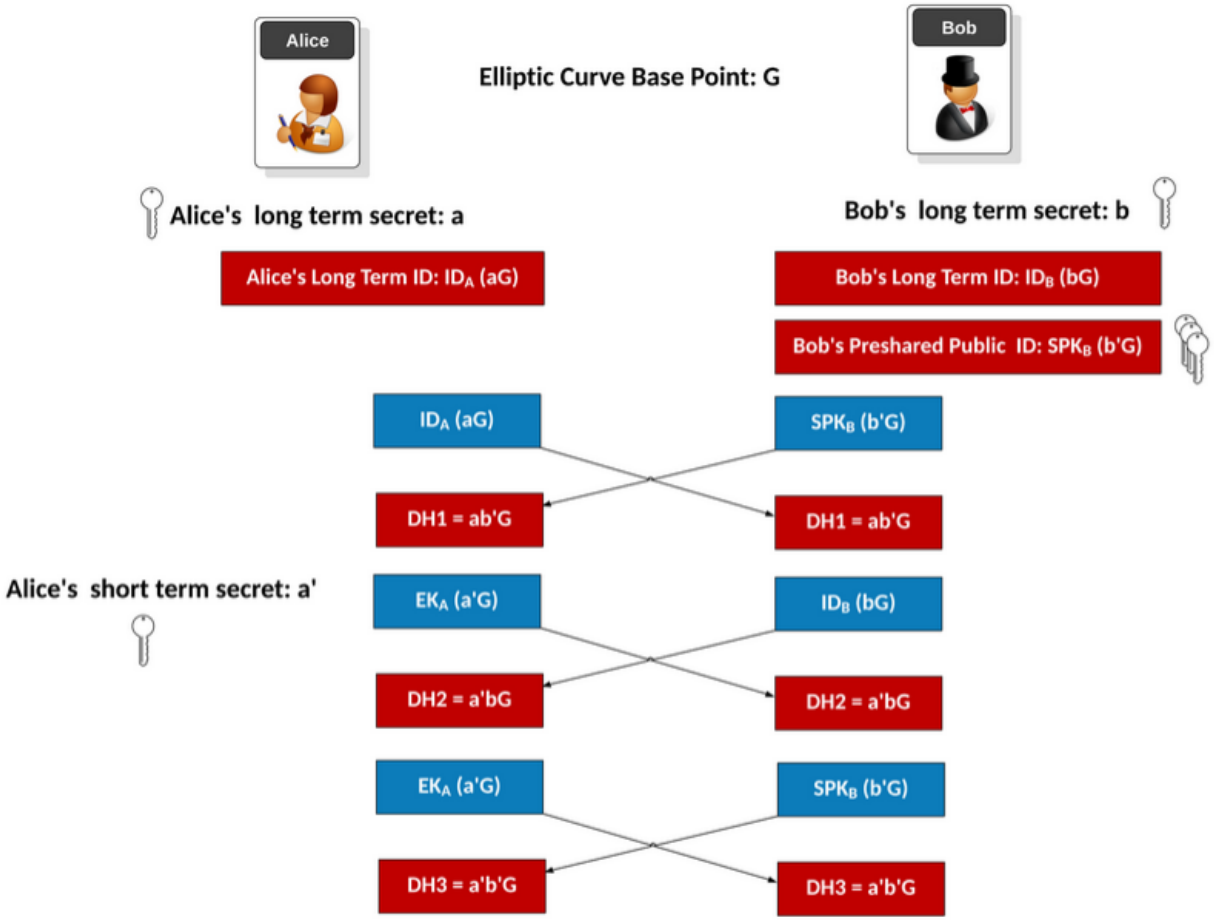
1. lekéri a szervertől:
   - Bob IK_B-t
   - SPK_B-t, $Sig(b, SPK_B)$-t
   - opcionálisan egy OPK_B-t
2. ellenőrzi az aláírást
3. generál egy ephemeral kulcsot: a'
4. ha a szervertől lekért értekek között NEM szerepel egy OPK_B:

$$DH1 = DH(IK_A, SPK_B)$$
$$DH2 = DH(EK_A, IK_B)$$
$$DH3 = DH(EK_A, SPK_B)$$
$$SK = KDF(DH1||DH2||DH3)$$

5. ha a szervertől lekért értekek között szerepel egy OPK_B, akkor még egy DH-t elvégez:

$$DH4 = DH(EK_A, OPK_B)$$
$$SK = KDF(DH1||DH2||DH3||DH4)$$

Signal - X3DH



Kép forrása: https://asecuritysite.com/encryption/go_x3dh

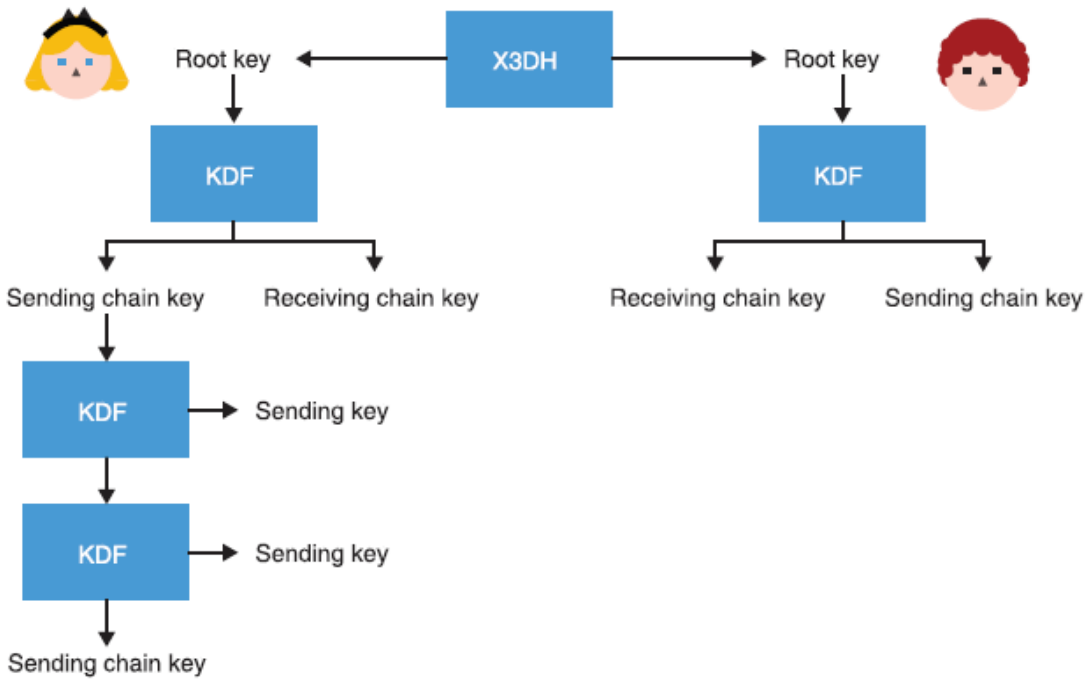
Signal - X3DH

Megjegyzések:

- az $DH1, DH2$ a kölcsönös hitelesítést mutual authentication-t biztosítja
- a $DH3, DH4$ a forward secrecy-t biztosítja
- Bob is az ő paramétereivel hasonlóan jár el, mint Alice
- a mindkét fél által meghatározott SK-t, a Root key-t használják a kommunikáció titkosítására
- az aláírások meghatározásához mindig az identity key privát kulcsát használja, ez gyenge pontja lehet a protokollnak

Signal - symmetric ratchet

- KDF-el az *SK*-ból két kulcsot hoz létre:
  - egyet használ Alice a titkosításhoz, mint seed,
  - a másikat Bob használja a titkosításhoz, mint seed,
  - a két kulcs nem lehet egyforma
- a titkosításhoz használt seed-ből egy láncot készít → sending chain key: hash függvényt alkalmaz a következő sending key előállításához



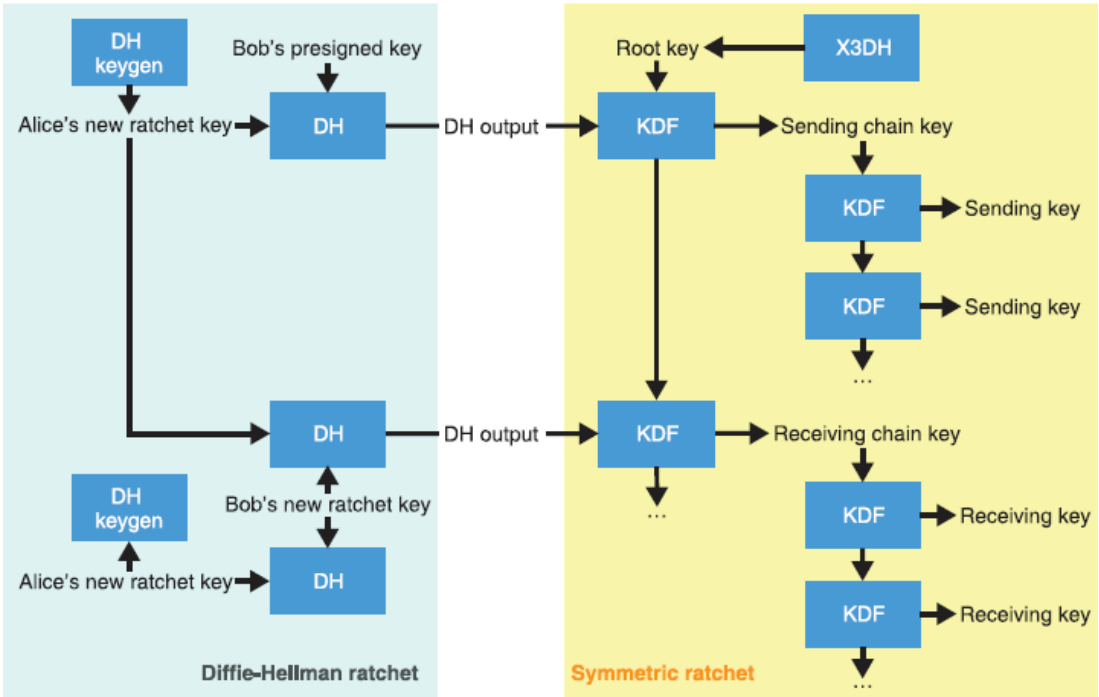
Kép forrása: D. Wong, Real-World Cryptography. Manning Publications Co., 2021.

Signal - DH Ratchet

- a post compromise security-hez:
- ha egy adott ponton kompromiálódnak a kulcsok a protkoll megújítja magát
- ha egy támadó hozzáfér az eszközhöz, akkor nincs hatása
- új entrópiát alkalmaz: egy új ephemeral key-el ismételt DH kulcscserét végez

Signal - Double Ratchet

symmetric ratchet + DH ratchet



Kép forrása: D. Wong, Real-World Cryptography. Manning Publications Co., 2021.