České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra telekomunikační techniky

SSL/TLS, DTLS

Ing. Tomáš Vaněk, Ph.D.
tomas.vanek@fel.cvut.cz

Obsah

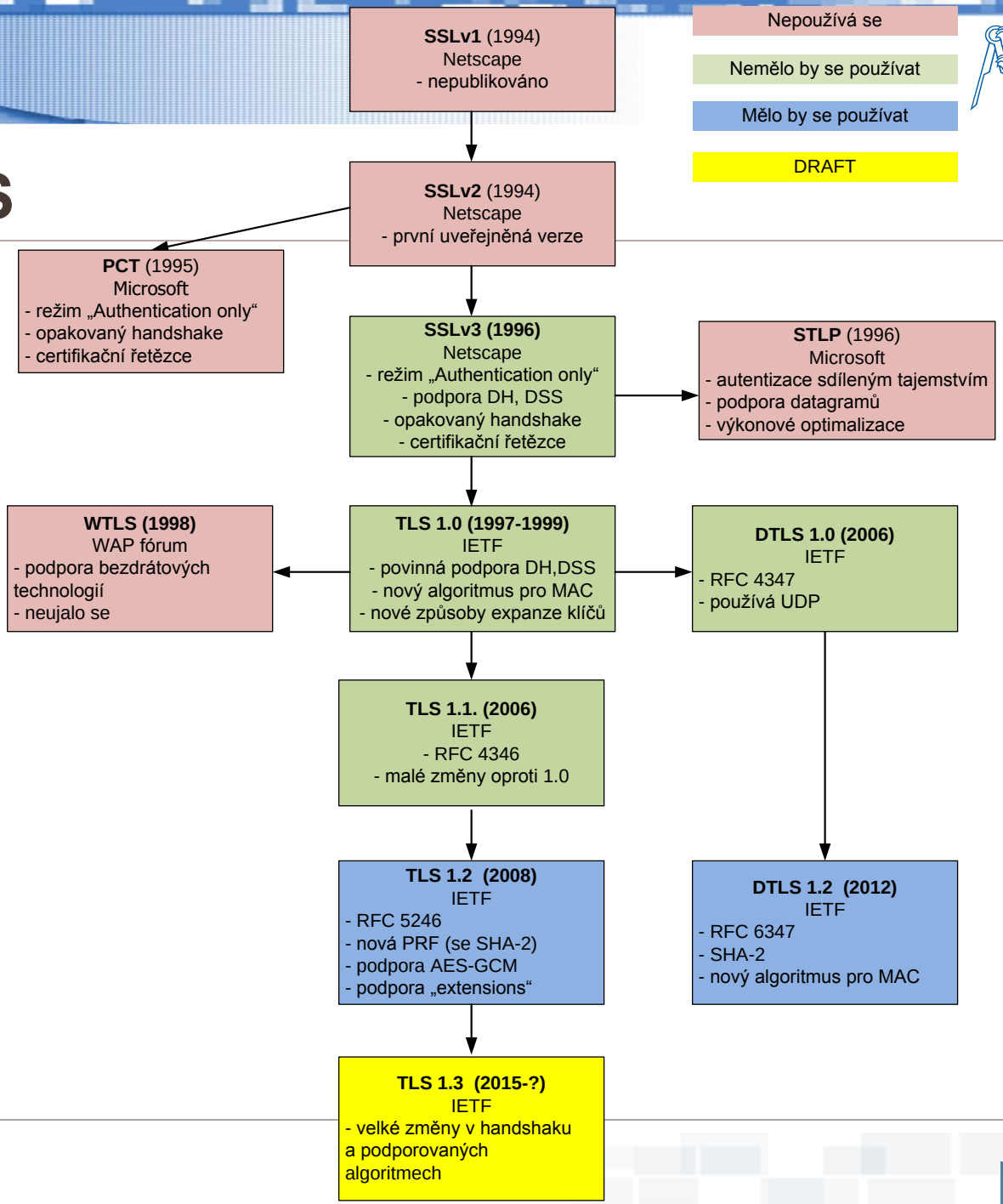
- SSL
  - RLP
  - HP
  - CCSP
  - AP
- TLS
- DTLS

SSL - Secure Socket Layer

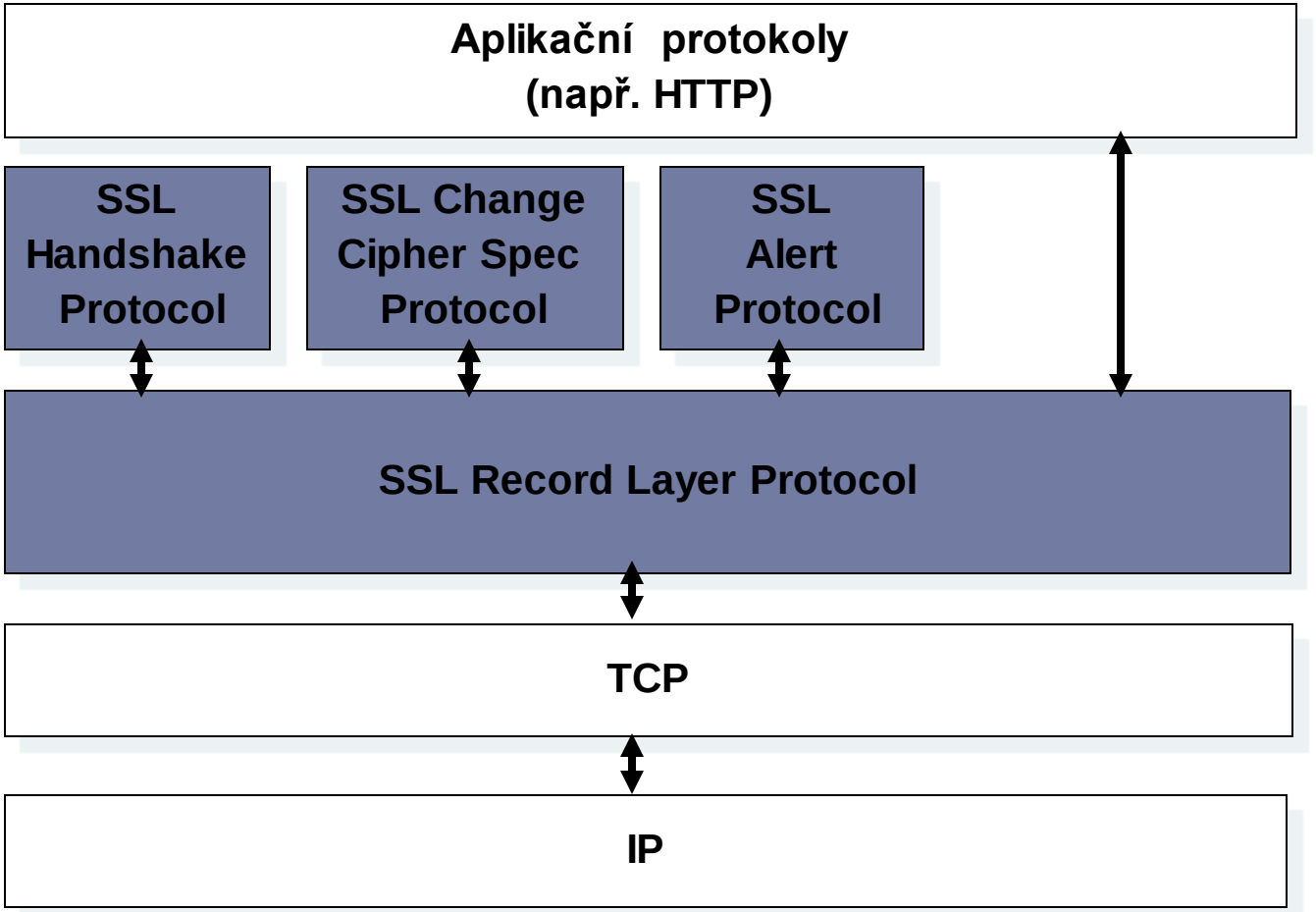
- vyvinut firmou Netscape
- nadstavba TCP/IP
- umístěna mezi aplikační a transportní vrstvou
- zajišťuje bezpečnou komunikaci mezi dvěma uzly
  - utajení přenášených informací
  - integrita přenášených zpráv
  - autentizace komunikujících stran
- dobrá interoperabilita
- snadná rozšiřitelnost o nové algoritmy
- používá spolehlivý transportní protokol (TCP)
- vytvořen s cílem zabezpečit prostředí WWW

- architektura klient/server
  - klient …ten, kdo zahajuje spojení
  - server…ke komu se klient připojuje
- duplexní komunikace
- existuje mnoho implementací protokolu SSL/TLS
- OpenSSL – www.openssl.org
  - aktuální stabilní verze – 1.0.2n (7.12.2017)
  - do 09/2000 se nesmělo používat v USA v RSA režimu (algoritmus RSA byl patentově chráněn)
  - zdarma pro nekomerční i komerční použití
- CyaSSL, yaSSL - http://www.yassl.com
  - TLS1.2, DTLS1.2 – 20x menší než OpenSSL
  - zaměřeno na embedded a RTOS prostředí

Vývoj SSL/TLS



Spolupráce jednotlivých protokolů z rodiny SSL/TLS



RLP – Record Layer Protocol

1) přebírá data od aplikačních protokolů
2) rozděluje data na bloky dané délky (< 2^{14}B)
   - výsledek operace je nazýván *SSLPlaintext*
3) komprimuje datové bloky (pokud bylo dohodnuto pomocí SSL-HP)
   - bezeztrátová komprese
   - nárůst délky bloku max. o 1024B
   - délka *SSLCompressed* musí být menší než < 2^{14}B, jinak dojde ke kritické chybě „ decompression_failure alert“
   - výsledek operace je nazýván *SSLCompressed*
4) doplní fragmenty na délku rovnou celistvému násobku délky šifrovacího bloku

RLP – Record Layer Protocol

5) zašifruje datové bloky

- 2 možné typy algoritmů

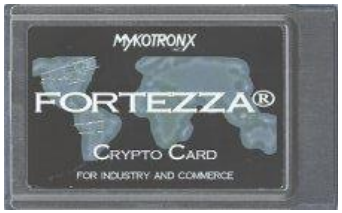
proudové (**stream**) - nepotřebuje doplňování velikosti a záznam může být použit přímo ve formátu v jakém jej předal předchozí proces

blokové (**block**) - potřebují doplnění délky záznamu na požadovanou délku násobku bloku. Celková délka výstupu ze šifrování nesmí překročit 2^{14}B.

Blokové šifry	Proudové šifry
IDEA - 128 bit	RC4 - 40 bit
RC2 - 40 bit	**RC4 - 128 bit**
DES - 40 bit	
DES - 56 bit	
3DES - 168 bit	
Fortezza - 80 bitů (HW karta)	

Šifrovací algoritmy SSL 3.0

http://en.wikipedia.org/wiki/Fortezza



RLP – Record Layer Protocol

6) vypočte a k *SSLCompressed* připojí SSL-MAC podle předem dohodnutého algoritmu MAC je vypočítána jako hash funkce s následujícími parametry:

MAC-DATA = H(MAC-WRITE-SECRET || OPAD || H(MAC-WRITE-SECRET || IPAD, SEQUENCE-NUMBER || type || length ||fragment))

SSLCompressed*

- konkrétní algoritmus (MD-5, nebo SHA-1) je domluven pomocí HP
- význam ostatních hodnot je následující:

  IPAD - 0x36 byte opakovaný 48x pro MD5 nebo 40x pro SHA-1

  OPAD - 0x5C byte opakovaný 48x pro MD5 nebo 40x pro SHA-1

  SEQUENCE-NUMBER - „počítadlo“;

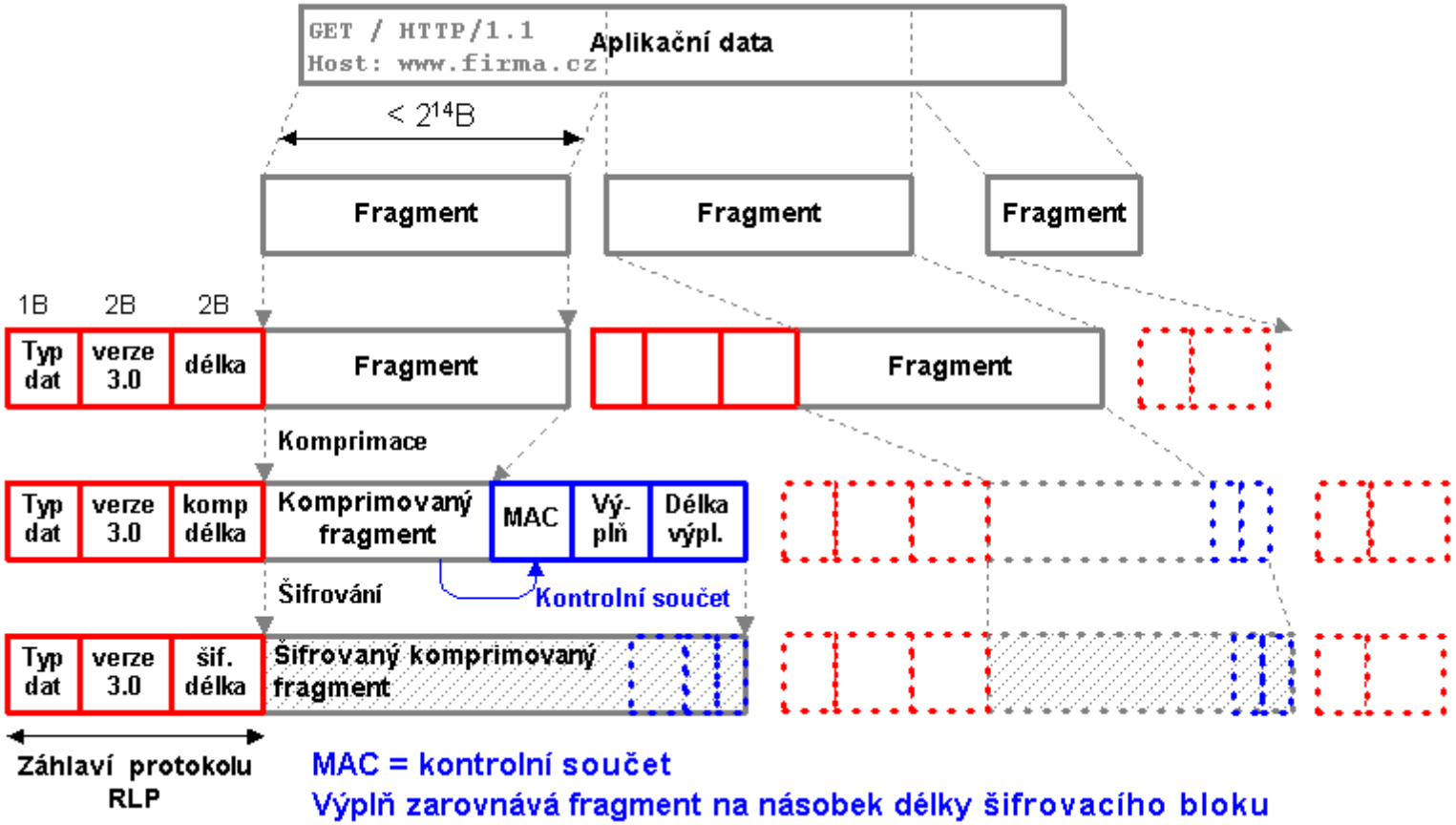
  každý účastník má dvě počitadla – odeslané a přijaté zprávy;

  v případě změny parametrů šifrování – reset počítadla

7) přidá hlavičku RLP před datový fragment

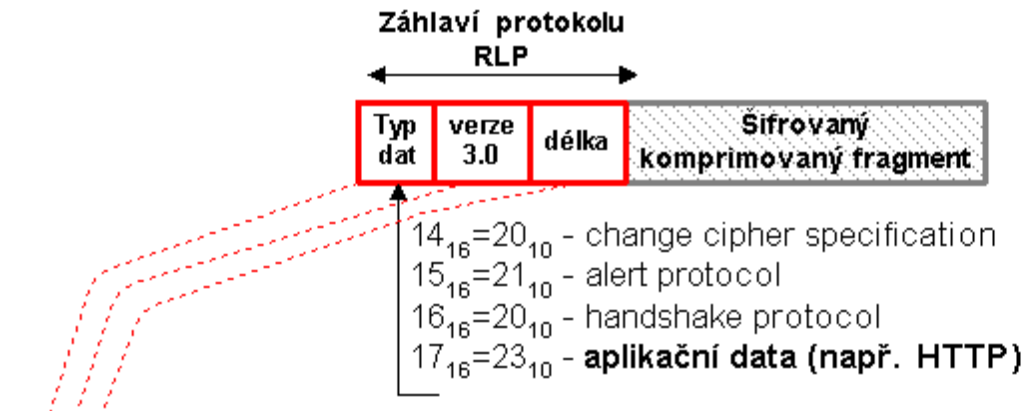
8) odevzdá data transportnímu protokolu TCP

RLP - úprava aplikačních dat



http://www.cpress.cz/knihy/tcp-ip-bezp/Cd-II/CD-ssl/ssl.htm

RLP - záhlaví



IP a TCP záhlaví je odříznuto

```
16 03 00 00 4F 01 00 00 4B 03 00 34 F3 C2 D1 1E AD 65 2B 7A 2F
95 A5 F0 68 AA 8E 55 F8 53 B3 B3 AC C7 9E E9 0C B6 5C 62 A8 BE
9F 20 03 99 14 66 7F DE 5A 47 10 19 F6 47 C7 4D 15 6A 4A 43 3A
BA B1 65 82 AC 98 52 68 7D 06 EE 2D E1 00 04 00 03 00 06 01 00
```

$4F_{16}=79_{10}$ takže se jedná o 79 bajtů handshake protokolu.

http://www.cpress.cz/knihy/tcp-ip-bezp/Cd-II/CD-ssl/ssl.htm

RLP - záhlaví

Typ dat - délka 1B

- specifikace přenášených dat:

0x14h – CCSP
0x15h – AP
0x16h – HP
0x15h – aplikační data (HTTP)

Verze - délka 2B

- Major a Minor verze
- prakticky

2.0…SSL v2.0
3.0…SSL v3.0
3.1…TLS v1.0
3.2…TLS v1.1
3.3…TLS v1.2
254.255 DTLS v1.0
254.253 DTLS v1.2

Délka - délka 2B - určení délky fragmentu

SSL Handshake Protokol

SSL-RLP neřeší:

- dojednání algoritmů, které se budou používat
- dojednání a bezpečnou výměnu klíčů
- tyto parametry jsou mu předány v tzv. protokolové sadě během vyjednávací fáze (handshake)

- z pohledu RLP je HP aplikační protokol
- HP se zapouzdřuje do RLP
  - slouží k:
    - autentizaci serveru
    - autentizaci klienta (nepovinná)
    - přípravě protokolové suity (typ šifrovacího algoritmu, typ protokolu pro výpočet kontrolního součtu)
    - dohodě na komprimačním algoritmu
    - výměně dat pro ustanovení hlavního tajemství (Master Secret)
    - ustavení zabezpečeného SSL spojení (connection)

SSL Handshake Protokol

Klient **Server**

client_hello →

← server_hello

Fáze 1: Vyjednání: ID relace, algoritmu pro výměnu klíčů, algoritmů pro MAC , šifrovacích algoritmů a výměna počátečních náhodných čísel

← certificate

← server_key_exchange

← certificate_request

← server_hello_done

Fáze 2: Server odešle svůj certifikát a zprávu pro výměnu klíčů, může si také vyžádat odeslání certifikátu klienta. Server oznamuje konec „uvítací (hello)“ fáze.

certificate →

client_key_exchange →

certificate_verify →

Fáze 3: Klient odešle svůj certifikát (pokud byl požadován) a pošle zprávu ověřující certifikát serveru. Klient vždy odesílá svou zprávu „key exchange message“

change_cipher_spec →

finished →

← change_cipher_spec

← finished

Fáze 4: výměna zprávy „Change Cipher Specification“ a ukončení vyjednácání

SSL Handshake Protokol

Autentizace serveru – vždy povinná

- umožňuje klientovi ověřit identitu serveru.
- klient využívá asymetrických algoritmů k tomu, aby ověřil
  - pravost certifikátu serveru a jeho identifikátoru
  - fakt že certifikát serveru byl vydán důvěryhodnou CA

Tento krok je důležitý pokud budou vytvořeným kanálem přenášena citlivá data, například čísla platebních karet.

Autentizace klienta - volitelná

- umožňuje serveru ověřit identitu klienta
- technicky se realizuje obdobně jako autentizace serveru
- autentizace klienta vůči serveru není příliš obvyklá

Relace vs. Spojení

Dva způsoby komunikace pomocí HP:

Vytvoření nové relace (session)

- autentizace účastníků (server vždy, klient pouze v případě neanonymního serveru)
- výměna klíčů
- Fáze 1-4

Obnova existující relace

- pokud již existuje relace a je potřeba nové spojení
- není nutné provádět úplný handshake
- stačí zkrácená verze využívající existující session ID
- Fáze1 + Fáze 4

Relace vs. Spojení

Relace (Session)

- ustavení spojení mezi dvěma komunikujícími uzly
- mezi těmito uzly je vytvořen zabezpečený „tunel“
- relace v sobě může obsahovat jedno nebo více spojení
- struktura relace je definována šifrovacími parametry (algoritmy + master_secret)

V rámci relace se udržuje:

Session ID – sekvence libovolných oktetů zvolená serverem sloužící k jednoznačné identifikaci konkrétní relace

Peer certificate – certifikát klienta podle standardu X.509v3

Cipher Specification - protokolová sada (množina) specifikující použitý šifrovací algoritmus a algoritmus pro výpočet kontrolního součtu (MAC)

Komprimační algoritmus – použitý kompresní algoritmus

Master secret – 48B, sdílené tajemství mezi C-S

Příznak - informace, indikující možnost využití ustaveného sezení pro navazování dalších případně nových spojení

Relace vs. Spojení

Spojení (Connection)

- jeden specifický komunikační kanál (obvykle mapovaný na TCP spojení)
- každé spojení má vlastní šifrovací klíče, klíče pro MAC a IV
- každé spojení je svázáno pouze s jednou relací

V rámci Spojení se udržuje:

Client_Radom - náhodné číslo generované klientem

Server_Radom - náhodné číslo generované serverem

Server_write_MAC_secret - tajná informace použitá pro výpočet MAC připojený k datům, které odesílá server

Client_write_MAC_secret - tajná informace použitá pro výpočet MAC připojený k datům, které odesílá klient

Initialization vectors – obsahuje inicializační vektory, pokud je použita bloková šifra v režimu CBC.

Sequence number – sekvenční čísla pro odeslané a obdržené zprávy

Server_write_key – symetrický klíč, kterým šifruje data server a dešifruje klient

Client_write_key – symetrický klíč, kterým šifruje data klient a dešifruje server

Příklad kryptografických sad pro TLS v1.0

```
CipherSuite TLS_RSA_WITH_NULL_MD5                          = 00 01
...
CipherSuite TLS_RSA_WITH_RC4_128_SHA                       = 00 05
...
CipherSuite TLS_RSA_WITH_3DES_EDE_CBC_SHA                  = 00 0A
CipherSuite TLS_ECDH_ECDSA_WITH_NULL_SHA                   = C0 00
CipherSuite TLS_ECDH_ECDSA_WITH_RC4_128_SHA                = C0 01
CipherSuite TLS_ECDH_ECDSA_WITH_DES_CBC_SHA                = C0 02
CipherSuite TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA           = C0 03
CipherSuite TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA            = C0 04
CipherSuite TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA            = C0 05
CipherSuite TLS_ECDHE_ECDSA_WITH_NULL_SHA                  = C0 06
...
CipherSuite TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA           = C0 09
CipherSuite TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA           = C0 0A
CipherSuite TLS_ECDH_RSA_WITH_NULL_SHA                     = C0 0B
CipherSuite TLS_ECDH_RSA_WITH_RC4_128_SHA                  = C0 0C
CipherSuite TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA             = C0 0D
CipherSuite TLS_ECDH_RSA_WITH_AES_128_CBC_SHA              = C0 0E
...
```

Výměna klíčů

Cíl výměny: sestavení šifrovaného kanálu pro přenos hodnoty PreMasterSecret ve směru C→S

Lze realizovat více způsoby:

- RSA se šifrovacími klíči
- RSA s podepisovacími klíči
- DSA
- DH výměna s jednorázovými parametry
- DH výměna s permanentními parametry
- anonymní DH výměna

Výměna klíčů

RSA bez autentizace klienta – verze 1

- server ve zprávě `server_certificate` odešle svůj veřejný RSA klíč
- zpráva `server_key_exchange` se neposílá
- klient ve zprávě `client_key_exchange` odešle zašifrovanou hodnotu `pre-master secret`
- zprávy `client_certificate` a `certificate_verify` nejsou odeslány

- pre-master secret je zašifrován pomocí veřejného RSA klíče serveru
- kryptografické sada začíná na SSL_RSA_with…

Výměna klíčů

RSA bez autentizace klienta – verze 2

- server ve zprávě `server_certificate` odešle svůj veřejný podepisovací klíč RSA nebo DSS klíč
- server ve zprávě `server_key_exchange` odešle jednorázový/dočasný RSA veřejný klíč
- klient ve zprávě `client_key_exchange` odešle zašifrovanou hodnotu `pre-master secret`
- zprávy `client_certificate` a `certificate_verify` nejsou odeslány

- pre-master secret je zašifrován pomocí veřejného RSA klíče serveru
- kryptografické sada začíná na SSL_RSA_with...

Výměna klíčů - alternativní

RSA s autentizací klienta – verze 1

- server ve zprávě `server_certificate` odešle svůj veřejný RSA klíč
- zpráva `server_key_exchange` se neposílá
- klient ve zprávě `client_certificate` odešle svůj veřejný podepisovací klíč (algoritmus RSA nebo DSS)
- klient ve zprávě `client_key_exchange` odešle zašifrovanou hodnotu `pre-master secret`
- klient ve zprávě `certificate_verify` odešle podpis všech odeslaných předchozích zpráv
- pre-master secret je zašifrován pomocí veřejného RSA klíče serveru
- kryptografické sada začíná na SSL_RSA_with...

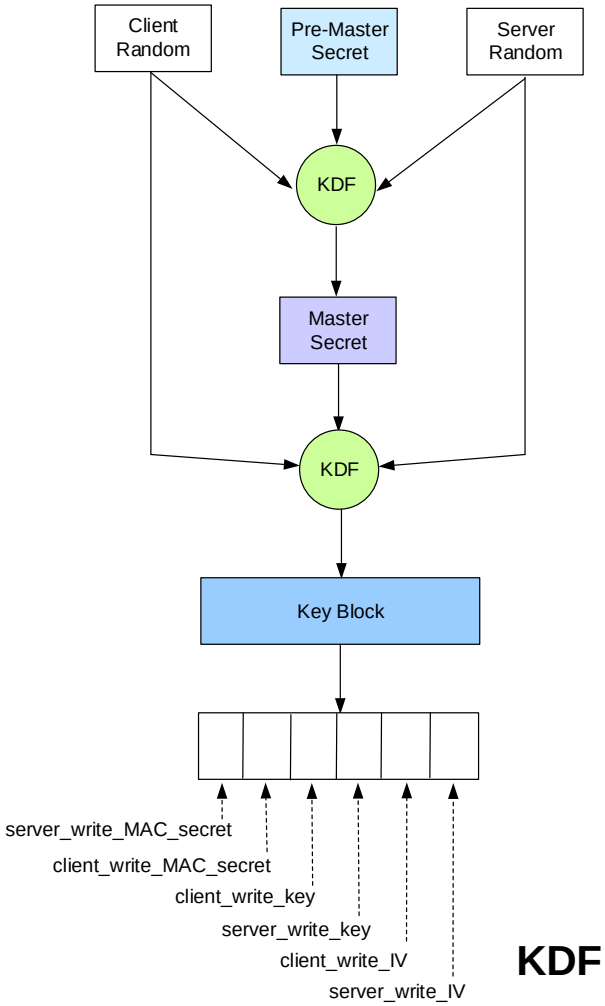
Výměna klíčů - alternativní

Jednorázová DH bez autentizace klienta

- server odešle ve zprávě `server_certificate` svůj podepsaný veřejný klíč (algoritmus RSA nebo DSA)
- server ve zprávě `server_key_exchange` odešle veřejné hodnoty pro jednorázovou DH výměnu podepsané soukromým RSA/DSA klíčem
- klient ve zprávě `client_key_exchange` odešle veřejné hodnoty pro jednorázovou DH výměnu
- zprávy `client_certificate` a `certificate_verify` nejsou odeslány

- kryptografické sada začíná na SSL_DHE_…

SSL – odvození MasterSecret a následně klíčů



KDF – Key Derivation Function

Odvození klíčů z Master Secret

- Hodnota **master secret** slouží pro výpočet šifrovacích klíčů.
- Pro její výpočet je použito dvou různých hashovacích funkcí, což by mělo zabezpečit komunikaci i pro případ, kdyby se v jednom hashovacím algoritmu našel fatální bezpečnostní problém.

1. Klient ve zprávě *Client_Hello* odešle náhodné číslo *Client_random*
2. Server ve zprávě *Server_Hello* odešle náhodné číslo „*Server_random*“
3. Klient vygeneruje *Pre-Master Secret* , zašifruje veřejným RSA klíčem Serveru a odešle jej ve zprávě *ClientKeyExchangeMessage*

Odvození klíčů z Master Secret

4. Obě strany si spočtou „*Master_Secret*“ podle vzorce:

```
master_secret =
    MD5(PreMasterSecret + SHA('A' + PreMasterSecret +
ClientRandom + ServerRandom)) +
    MD5(PreMasterSecret + SHA('BB' + PreMasterSecret +
ClientRandom + ServerRandom)) +
    MD5(PreMasterSecret + SHA('CCC'+ PreMasterSecret +
ClientRandom + ServerRandom)
```

kde A=0x41, B=0x4242, C=0x434343

5. Po předání zprávy *ClientKeyExchange* pak jak klient, tak i server znají
   - *ClientRandom* (ze zprávy ClientHello)
   - *ServerRandom* (ze zprávy ServerHello)
   - *Pre_Master_secret* (ze zprávy *ClientKeyExchange*)
   - *Master_Secret* (spočítali si)

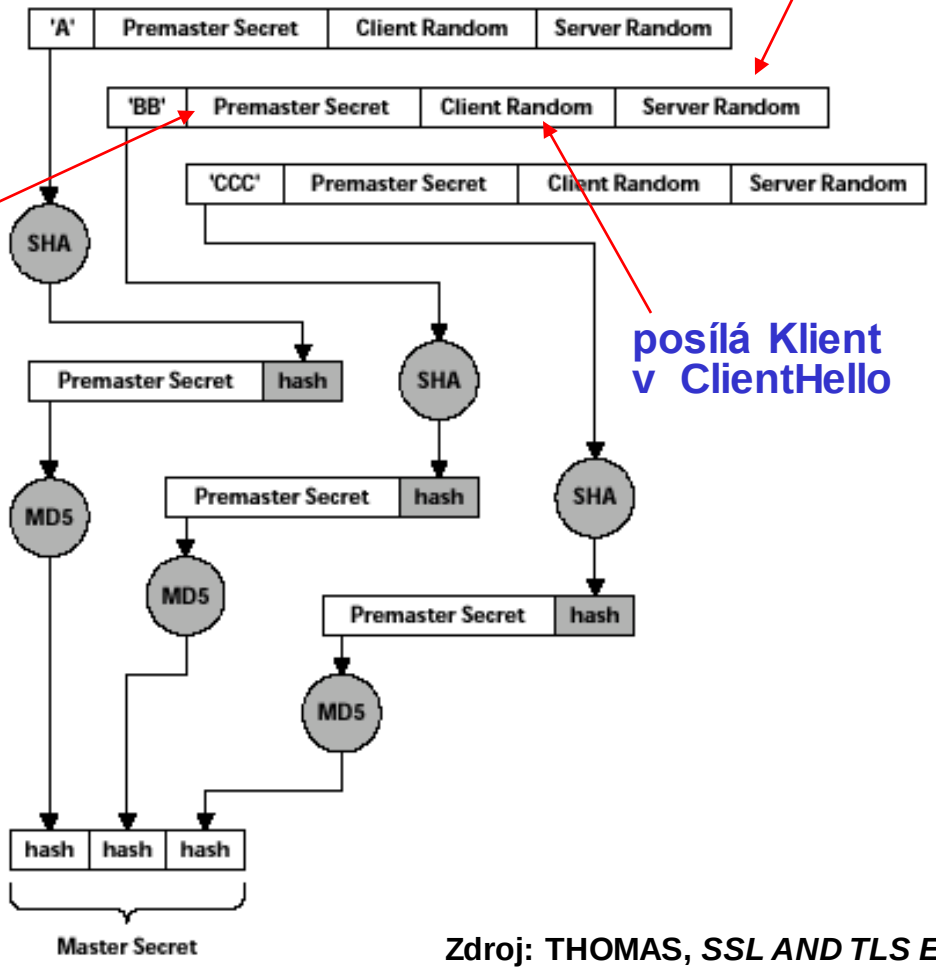
posílá Server
v ServerHello

Veřejný klíč Serveru je odeslán ve zprávě ServerKeyExchange

Klient vygeneruje PREMASTER SECRET

Klient zašifruje PREMASTER SECRET veřejným klíčem Serveru

Klient odešle PREMASTER SECRET ve zprávě ClientKeyExchange



posílá Klient
v ClientHello

MASTER SECRET jsou tři MD5 HASHE spojené dohromady = 384 bitů

Zdroj: THOMAS, *SSL AND TLS ESSENTIALS*

Odvození klíčů z Master Secret

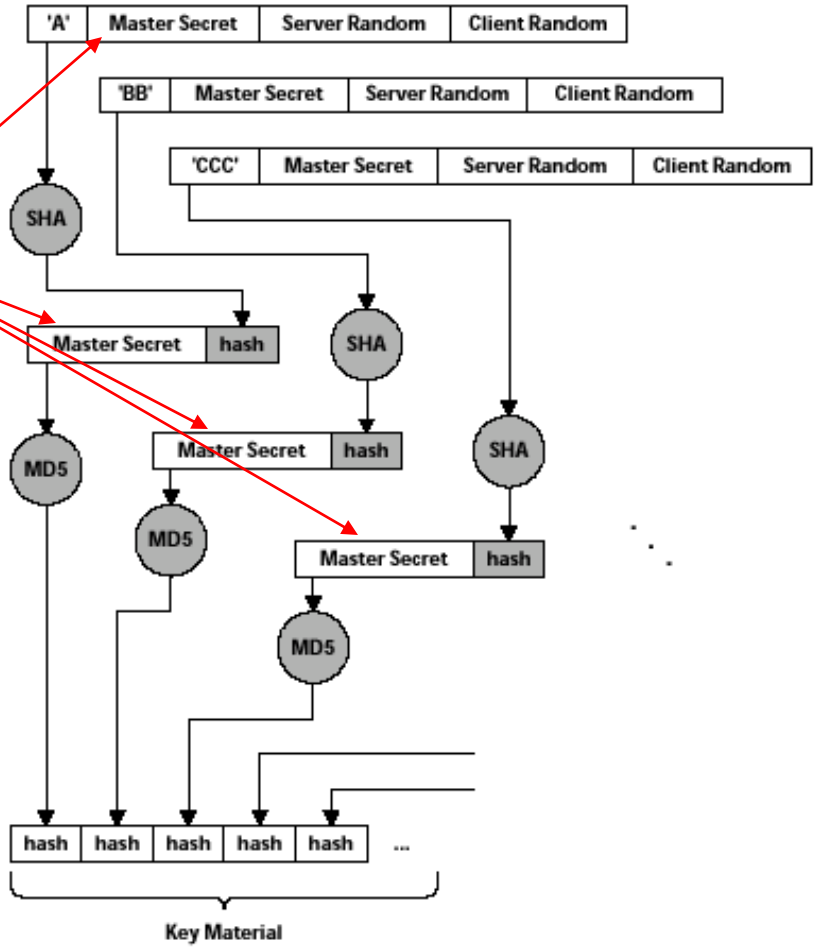
6. Nyní obě strany mohou spočíst balík klíčů (key_block):

```
key_block =
     MD5(master_secret + SHA(`A' + master_secret +
        ServerRandom + ClientRandom)) +
     MD5(master_secret + SHA(`BB' + master_secret +
         ServerRandom +  ClientRandom)) +
     MD5(master_secret + SHA(`CCC' + master_secret +
           ServerRandom + ClientRandom))
```

7. Z balíku se postupně oddělí data tvořící tajemství pro výpočet kontrolního součtu klienta (*client_write_MAC_secret*), pak data tvořící tajemství pro výpočet kontrolního součtu serveru (*server_write_MAC_secret*), pak šifrovací klíč klienta (*client_write_key*), šifrovací klíč serveru (*server_write_key*) a nakonec inicializační vektory (IV).

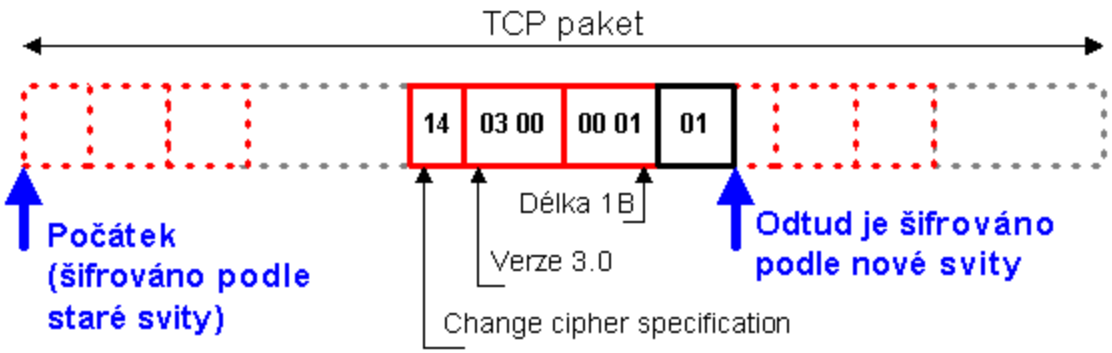
Odvození klíčů z Master Secret

Stejné jako když se počítal MASTER_SECRET, až na to, že zde je použit místo PRE_MASTER_SECRET Již zmíněný MASTER_SECRET



ZdrojE: THOMAS, *SSL AND TLS ESSENTIALS*

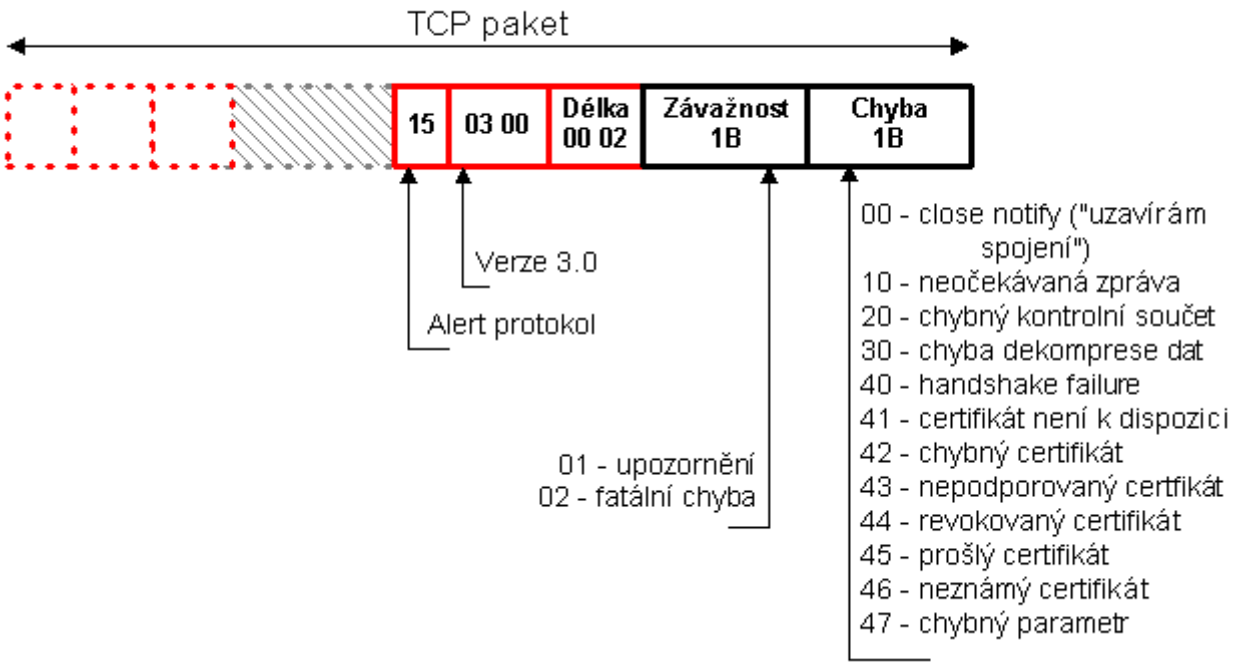
CCSP - Change Cipher Specification Protocol



- velmi jednoduchý protokol
- jediná funkce
- informace o přechodu na novou šifrovací sadu

http://www.cpress.cz/knihy/tcp-ip-bezp/Cd-II/CD-ssl/ssl.htm

SSL – Alert Protocol



http://www.cpress.cz/knihy/tcp-ip-bezp/Cd-II/CD-ssl/ssl.htm

SSL Alert Protocol

- signalizace chybových hlášení
- analogie ICMP v TCP/IP
- dvě úrovně zpráv
  - informační (komunikace může pokračovat)
    11 v SSL , 23 v TLS
  - kritická chyba (spojení je ukončeno)

Příklad kritických chyb:

- Bad_record_mac - špatná hodnota MAC
- Decompression_failure - délka zprávy po dekompresi překročila 2^{14}B
- Handshake_failure - chyba při vyjednávání parametrů
  Unexpected_message – přišla nečekávaného typu
  - Bad_record_MAC – špatný kryptografický kontrolní součet
  - Handshake_failure – selhalo vyjednání algoritmu
  - Illegal_parameter – neplatný parametr

SSL Alert Protokol

Příklad informačních chyb:

- Certificate_expired - certifikát vypršel
- Certificate_revoked - certifikát byl zrušen tím, kdo jej vystavil
- Unsupported_ certificate - nepodporovaný typ certifikátu
- Close_notify -
- No_certificate -
- Bad_certificate -
- Unsupported_certificate -

V případě kritické chyby je :

- ukončeno aktuální spojení
- ID relace je zneplatněno → nelze vytvořit nové spojení v rámci této relace (existujících spojení se to nedotkne)

Rozdíly mezi SSL a TLS

- nové chybové zprávy
- není podporován algoritmus Fortrezza
- jiné doplnění bloku (SSL - doplňky jsou vždy nejmenší, TLS – doplňky jsou volitelně velké do 255B)
- jiný algoritmus pro MAC
- rozdíly jsou minimální ale přesto nejsou TLS 1.0 a SSL 3.0 kompatibilní
- změny v AP

- rozdíly ve zprávách *Certificate_verify*, *Finished*
- označení TLS je kompromis
  - MS chtěl PCT, Netscape chtěl SSL
  - TLS se nelíbilo ani jednomu

Rozdíly SSL a TLS

- nová pseudonáhodná funkce PRF používaná ve zprávě Finished
- nejprve sestrojíme expanzní funkci P_hash(secret,seed)

  P_hash(secret, seed) = HMAC_hash(secret, A_1 seed) || HMAC_hash(secret, A_2 || seed) || HMAC_hash(secret, A_3 || seed) || …

  kde

  A_0 = seed

  A_i = HMAC_hash(secret, A_{i-1})

  PRF(secret, label, seed) = P_MD5(secret_left, label || seed) ⊕P_SHA(secret_right, label || seed)

  label – identifikující ASCII string

  seed – náhodné číslo, které se používá při inicializaci systému

  secret – tajné heslo používané v HMAC
- kryptografické sady začínají na TLS_

Rozdíly SSL a TLS

- zpráva **Finished** obsahuje hash ve tvaru
  PRF(master_secret, “client finished”, MD5(handshake_messages) | SHA(handshake_messages))
- rozdíly v kryptografických výpočtech
  - pre-master secret se počítá stejně jako u SSL, ale
  - master secret: PRF(pre_master_secret, “master secret”, client_random | server_random)
  - key block: PRF(master_secret, “key expansion”, server_random | client_random)
- 1999 – RFC 2712, podpora Kerberosv5
- 2005 – RFC 4132, podpora algoritmu Camelia
- 2006 – RFC 4492, podpora ECDH, ECDHE, ECDSA
- 2008 – RFC 5288, podpora AES-GCM
- 2011 – draft-kanno-tls-camellia-01 – podpora dalších 42 sad s algoritmem Camellia

TLS 1.2

- 08/2008
- RFC 5246
- jiná PRF
  - náhrada kombinace hashovacích funkcí MD5/SHA-1 za jedinou - SHA-256
  - náhrada kombinovaného MD-5/SHA-1 hashe za jediný hash
- podpora nových šifrovacích režimů (AES-GCM)
- podpora AES přidána přímo do standardu TLS
- větší kontrola verzí zpráv HP
- sloučení kryptografických sad využívajících AES z externích RFC

TLS 1.2

TLS rozšíření

- RFC 4366, 4492,4680,4681,5054,5077,5081
- posílají se v ClientHello
- rozšíření server_name_indication (SNI)
  - pro virtual hosting s podporou TLS
  - u SSL 2.0/3.0 musí mít každý HTTP s podporou SSL/TLS server unikátní IP adresu
  - existovalo již u TLS 1.0 jako RFC 3546
- rozšíření Max_Fragment_Length
  - možnost změny max. velikosti fragmentu
  - někdy se hodí pracovat s fragmenty přesné délky
  - podporované hodnoty 2^9 , 2^{10} , 2^{11} , 2^{12}
- rozšíření client_certificate_url
  - klient pošle URL certifikátu místo samotného certifikátu

TLS 1.2

- rozšíření user_mapping
  - obecná podpora výměny dalších parametrů pomocí rozšířených ClientHello zpráv
- rozšíření cert_type
  - podpora OpenPGP certifikátů (kromě X.509)
  - výchozí hodnota 0 (= X.509)
  - RFC 5081
  - připraveno i na podporu dalších typů

SRP – Secure Remote Password protokol

- RFC5054 - 9 protokolových sad s SRP
- autentizace klienta při TLS handshaku jménem/heslem nešifrovaným kanálem
- heslo nelze odposlechnout
- založeno na DH …řeší i výměnu sdíleného tajemství

Navázání SSL/TLS spojení z pohledu aplikace

- Vyhrazený port – pouze šifrovaná komunikace

Aplikační protokol	Číslo TCP portu
HTTPS	443
SMTPS	465
LDAPS	636
FTPS (FTP control)	989
FTPS-data (FTP data)	990
IMAPS	993

- Standardní port aplikace (80,25,143,…)
  - zabezpečená i nezabezpečená komunikace
  - příkaz STARTTLS
  - MiTM útok typu SSLStripppping – odstranění zprávy „250 STARTTLS“ vyžadující navázání zabezpečeného spojení

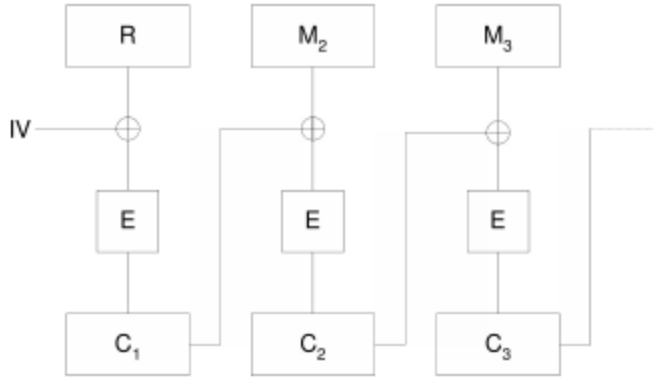
DTLS - Datagram Transport Layer Security

- DTLS 1.0 - 04/2006
- RFC 4347
- součást OpenSSL od verze 0.9.8
- vychází z TLS, poskytuje obdobné služby
- používá UDP (obecně datagramově orientovaný)
- řazení paketů, ztrátu datagramů, nemožnost přenést datagram větší než …→ neřeší DTLS, ale aplikace
- vhodný pro aplikace citlivé na zpoždění (VoIP) a tunelované aplikace (VPN)

- DTLS 1.2 - 01/2012
- RFC 6347

DTLS

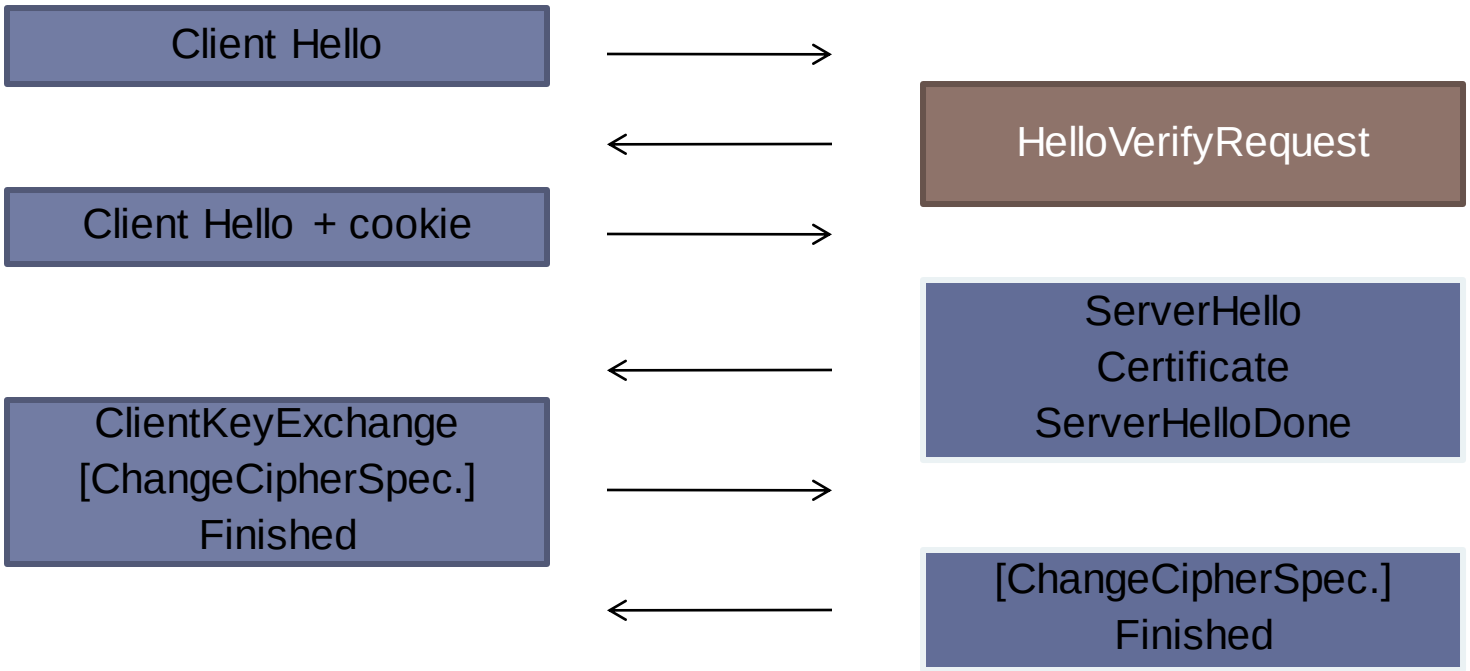
- nelze použít kryptografické sady z TLS 1.0
  - udržují vztah mezi jednotlivými záznamy a to vyžaduje jejich bezpečné doručení (není u DTLS garantováno kvůli UDP)
- lze použít CBC režim s explicitní definicí IV
  - definovaný v TLS 1.1 a AES-CM
  - není náchylný na BEAST (Browser Exploit Against SSL/TLS) útok



CBC s explicitním IV, R-blok náhodných dat

DTLS

- ochrana handshake protokolu proti DoS útokům
- zpráva *HelloVerifyRequest*
- výměna *cookies* (příjemce je schopen přijmout a zpracovat data na udávané IP adrese)



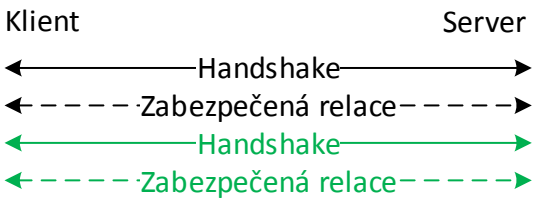
Útoky na SSL/TLS

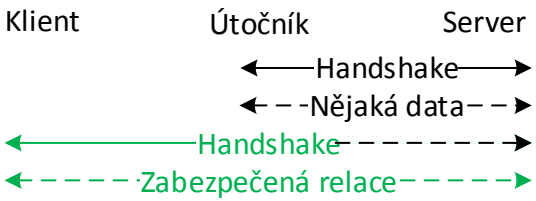
- 2011 – BEAST
  - Browser Exploit Against SSL/TLS
  - SSLv2, SSLv3, TLS1.0
  - Implicitní IV pro CBC

- Padding Oracle
  - Systematická chyba v návrhu SSL/TLS „MAC-then-encrypt“
    - 2013 – Lucky Thirteen
    - 2014 – POODLE (Padding Oracle On Downgraded Legacy Encryption)
      - Funguje pouze na SSL3.0 -> nejprve downgrade z TLS na SSL
  - Řešení – komplexní režimy typu AES-GCM nebo přístup “encrypt-then-MAC“ (RFC 7366)

Útoky na SSL/TLS

- 2013 – Renegotiation attack
  - Re-negotiation = nový handshake v rámci existujícího zabezpečeného spojení
  - Nový zabezpečené spojení nahradí předchozí





  - Počáteční provoz a klientská komunikace jsou považovány za pocházející ze stejného kontextu

- Od 2016 by měli být šifrovací sady s RC4 defaultně vypnuty (IE, FF, Chrome) – domluva MS, Mozilla Foundation, Google

Útoky na SSL/TLS

- 2013 – CRIME (Compression Ratio Info-leak Made Easy)
  - zneužití webových cookies používaných protokoly HTTPS a SPDY v kombinaci s kompresí na úrovni TLS/HTTP/SPDY
  - umožňuje session hijack; obrana – vypnout kompresi
- 2013 - BREACH
  - Útočník přeruší uživateli TCP relaci (příznak FIN)
  - Server neobdrží pokyn k ukončení SSL/TLS relace (ta dále existuje) a útočník může přes prohlížeč oběti dále komunikovat zabezpečeným kanálem s cílovým serverem
  - Zamezí odhlášení uživatele z TLS relace
- 2014 – FREAK
  - OpenSSL
  - MITM, downgrade attack
  - Dojednání TLS spojení s 512b klíči

Útoky na SSL/TLS

- 2015 – Logjam
  - Downgrade attack
  - Vynucuje použití „exportních“ 512b grup pro Diffie-Hellmanův protokol

- CCSP injection
- 2016 – DROWN
  - cross-protocol útok
  - SSLv2 je stále zapnuto na cca 1/3 HTTPS serverů

Dotazy