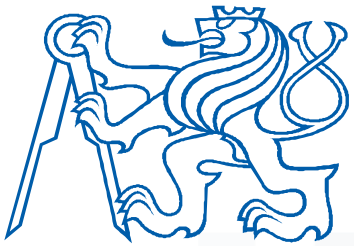
České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra telekomunikační techniky

A7B32KBE – 11.přednáška

SSL/TLS, DTLS

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



Obsah

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

SSL - Secure Socket Layer

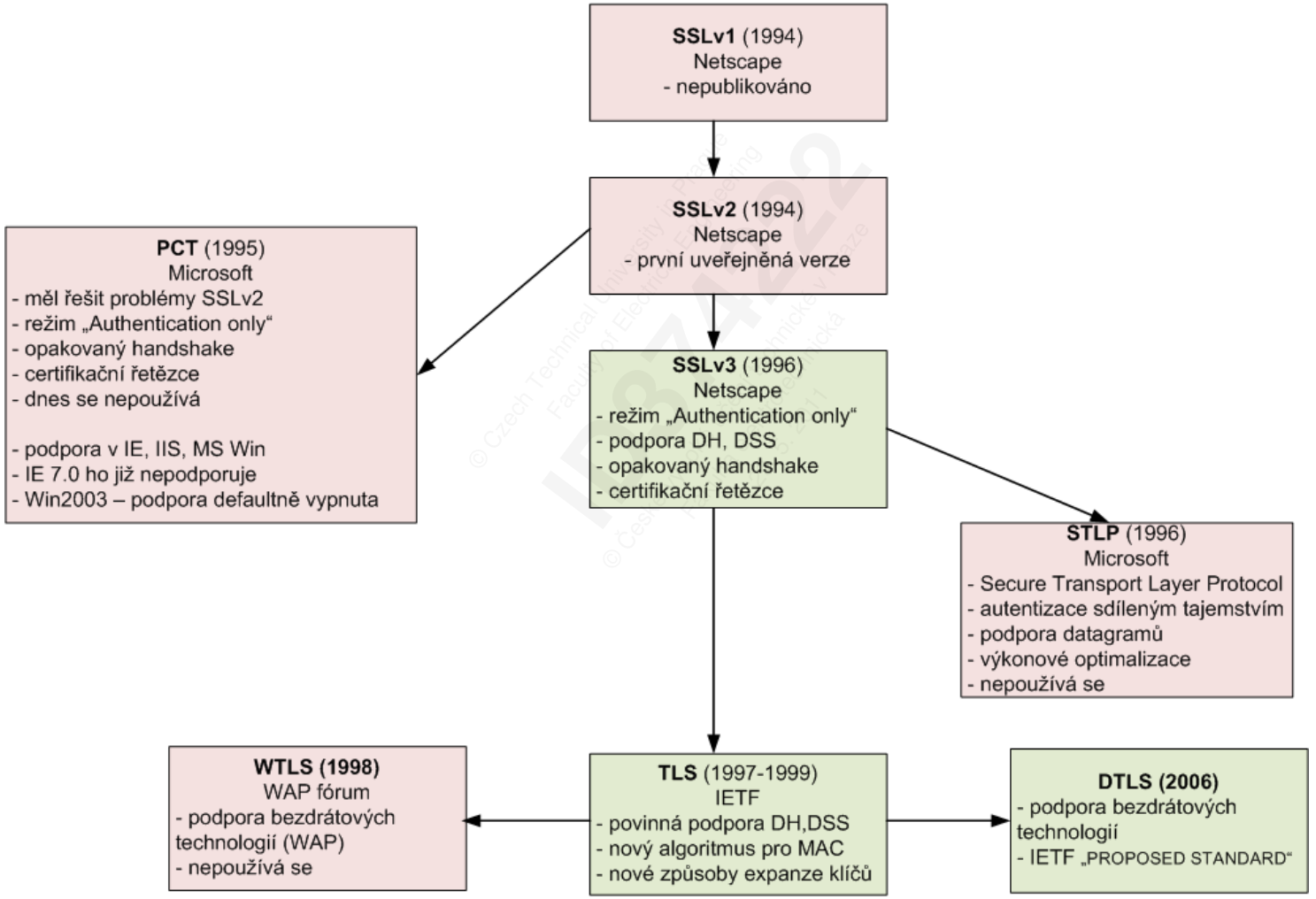
- vyvinut firmou Netscape
- nadstavba TCP/IP
- umístěna mezi aplikační a transportní vrstvou
- zajišťuje bezpečnou komunikaci mezi dvěma uzly
  - utajení přenášených informací
  - integrita přenášených zpráv
  - autentizace komunikujících stran
- dobrá interoperabilita
- snadná rozšiřitelnost o nové 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
- OpenSSL – www.openssl.org
- aktuální stabilní verze – 1.0.0d
- do 09/2000 se nesmělo používat v USA v RSA režimu (algoritmus RSA byl patentově chráněn)
- podporuje SSLv2, SSLv3 a TLSv1
- zdarma pro nekomerční i komerční použití

Bezpečnostní chyba v OpenSSL knihovně

- vznikla 17.9.2006
- odhalena 13.5.2008
- týká se verzí 0.9.8c-1 až 0.9.8g-9
- týkala se všech distribucí vycházejících z Debianu (včetně Ubuntu), OpenSSH, OpenVPN, Openswan, DNSSEC, klíče pro X.509 certifikáty,Tor, xrdp, …
- při ladění byla odstraněna část kódu využívající /dev/urandom jako zdroj entropie pro generování náhodných čísel
- generátor náhodných čísel vycházel pouze z čísla procesu
- omezení entropie generátoru způsobilo, že místo 2^{128} klíčů se generovalo pouze 2^{15} různých klíčů
- opraveno

Vývoj SSL/TLS



Vývoj SSL/TLS

PCT – Private Communication Technology

- částečně kompatibilní varianta SSLv2 řešící bezpečnostní problémy SSLv2
- existence režimu bez šifrování, pouze s autentizací
- vyšší výkon
- podpora současně silné autentizace a slabého šifrování (U.S. omezení vývozu šifrovacích algoritmů)

STLP – Secure Transport Layer Protocol

- modifikace SSLv3
- podpora datagramů (UDP)
- nové formy autentizace

Vývoj SSL/TLS

SSLv3

- měl obsahovat jen opravy bezpečnostních problémů SSLv2
- nakonec je SSLv3 hodně odlišný od SSLv2
- podporuje mnohem více kryptografických algoritmů
- podpora zřetězených certifikátů

TLS – Transport Layer Security

- standardizován IETF - RFC 2246
- v hlavičce se TLS označuje jako SSL verze 3.1
- kosmetické změny oproti SSLv3
- **není kompatibilní**
- povinná podpora DH, DSS, 3DES
- RSA nepovinné - kvůli patentům (vypršely v 1998)
- zdržení díky standardizaci X.509 (nelze schválit standard závisející na neschváleném standardu)

SSL – základní informace

Šifrování

- asymetrické šifrování – sestavení klíčů relace, autentizace
- symetrické šifrování – vlastní šifrování datového kanálu
    - blokové šifry – DES, 3DES, (AES)
    - proudové šifry – RC4

Integrita

- MD-5
- SHA-1

Autentizace

- kombinace jméno/heslo (RADIUS)
- kombinace jméno/token (SecureID)
- digitální certifikáty (ITU X.509)

SSL – základní informace

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

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 (hodnota nemusí být definována)

Cipher Specification - protokolová svita (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

Relace vs. Spojení

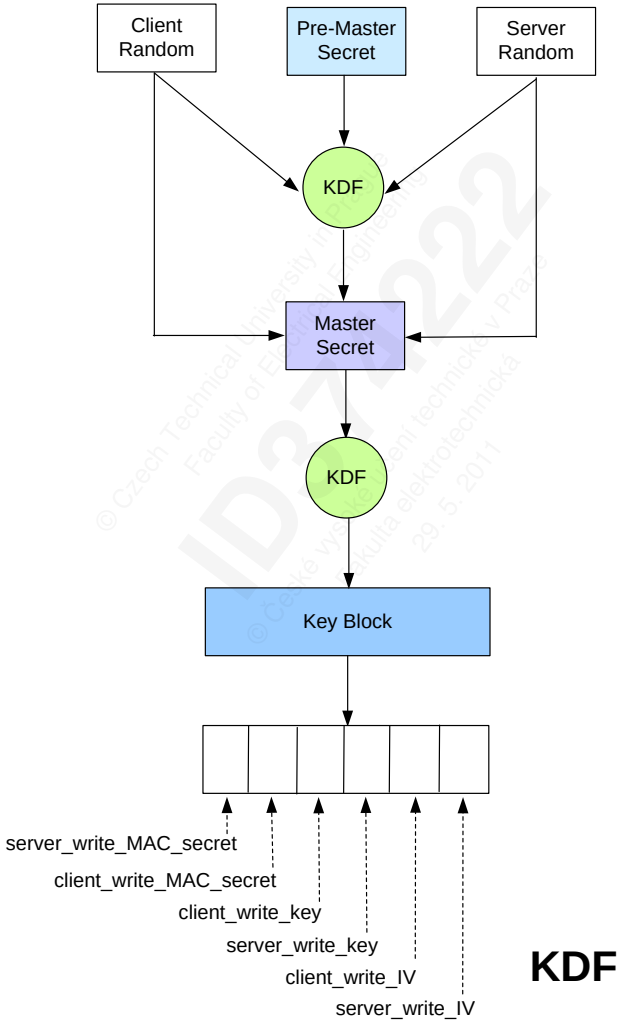
Initialization vectors – obsahuje inicializační vektor, pokud je použita bloková šifra v režimu CBC. Je inicializován protokolem SSL Handshake

Sequence number – klient i server udržují samostatná 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

SSL – základní informace



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)

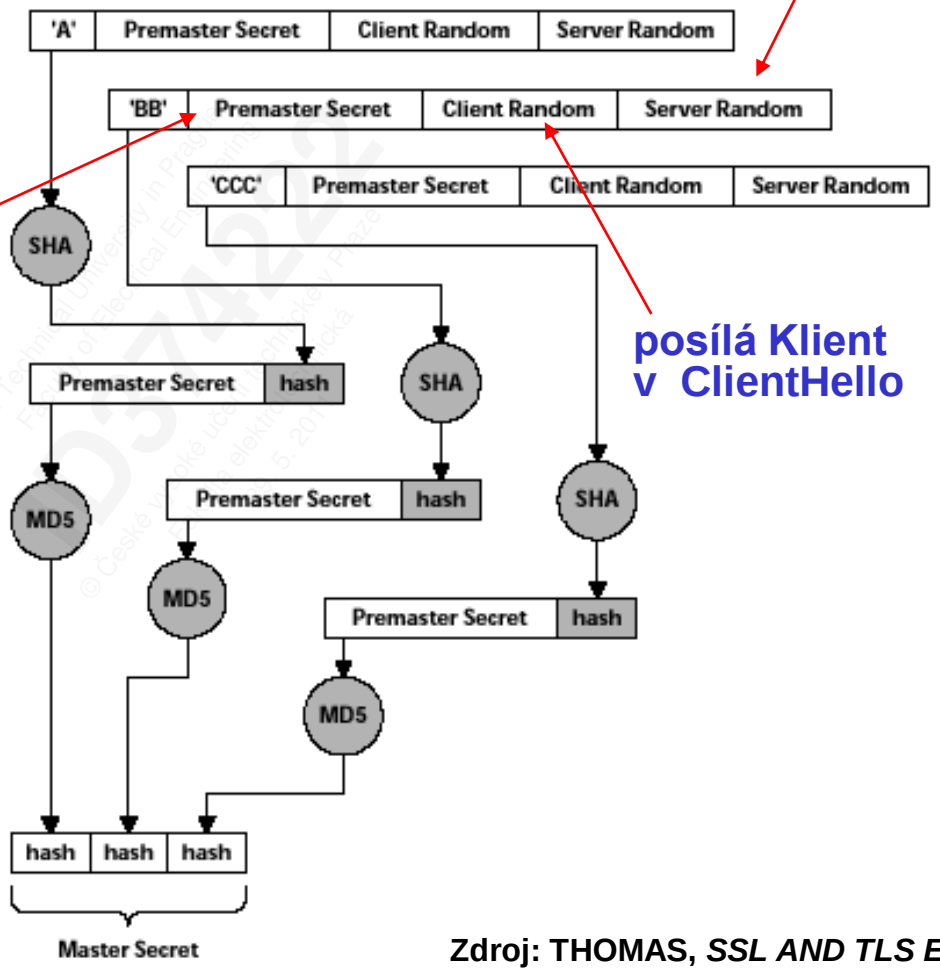
posílá Server
v ServerHello

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):

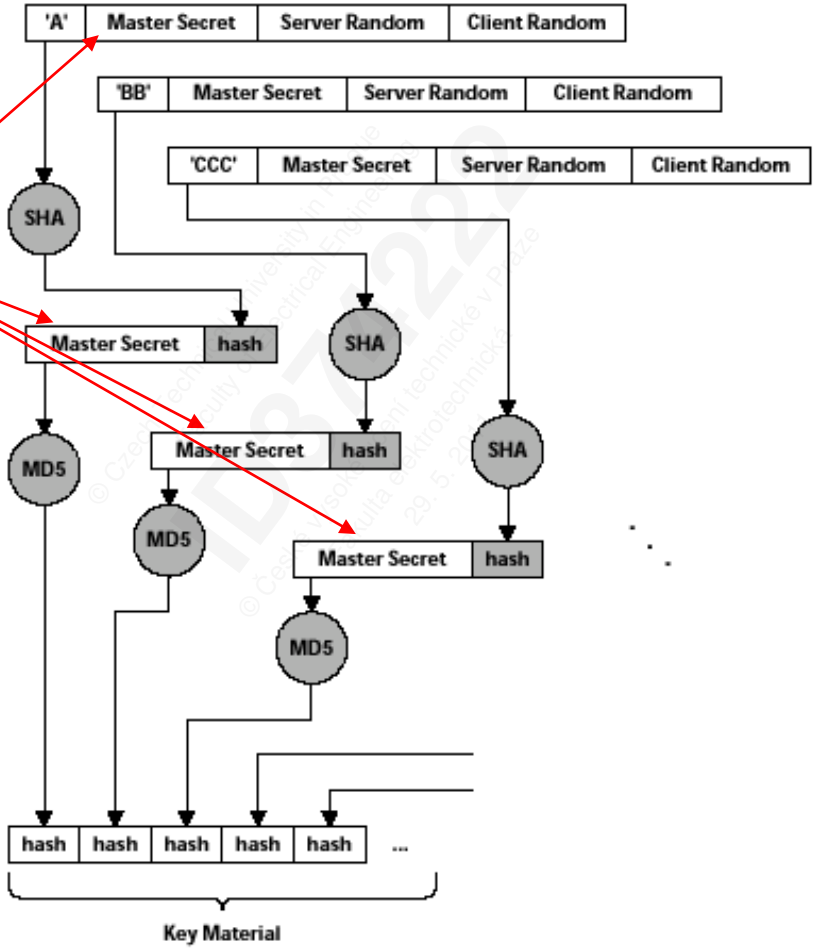
```
key_block =
      MD5(master_secret + SHA(`A' + master_secret +
         ServerRandom + ClientRandom)) +
      MD5(master_secret + SHA(`BB' + master_secret +
          ServerRandom +  ClientRandom)) +
      MD5(master_secret + SHA(`CCC' + master_secret +
            ServerRandom + ClientRandom))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

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).

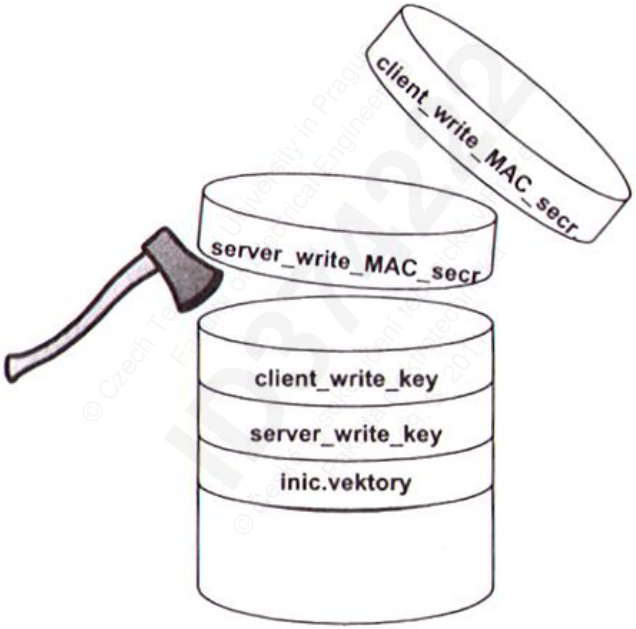
Odvození klíčů z Master Secret

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

Již zmíněný MASTER_SECRET



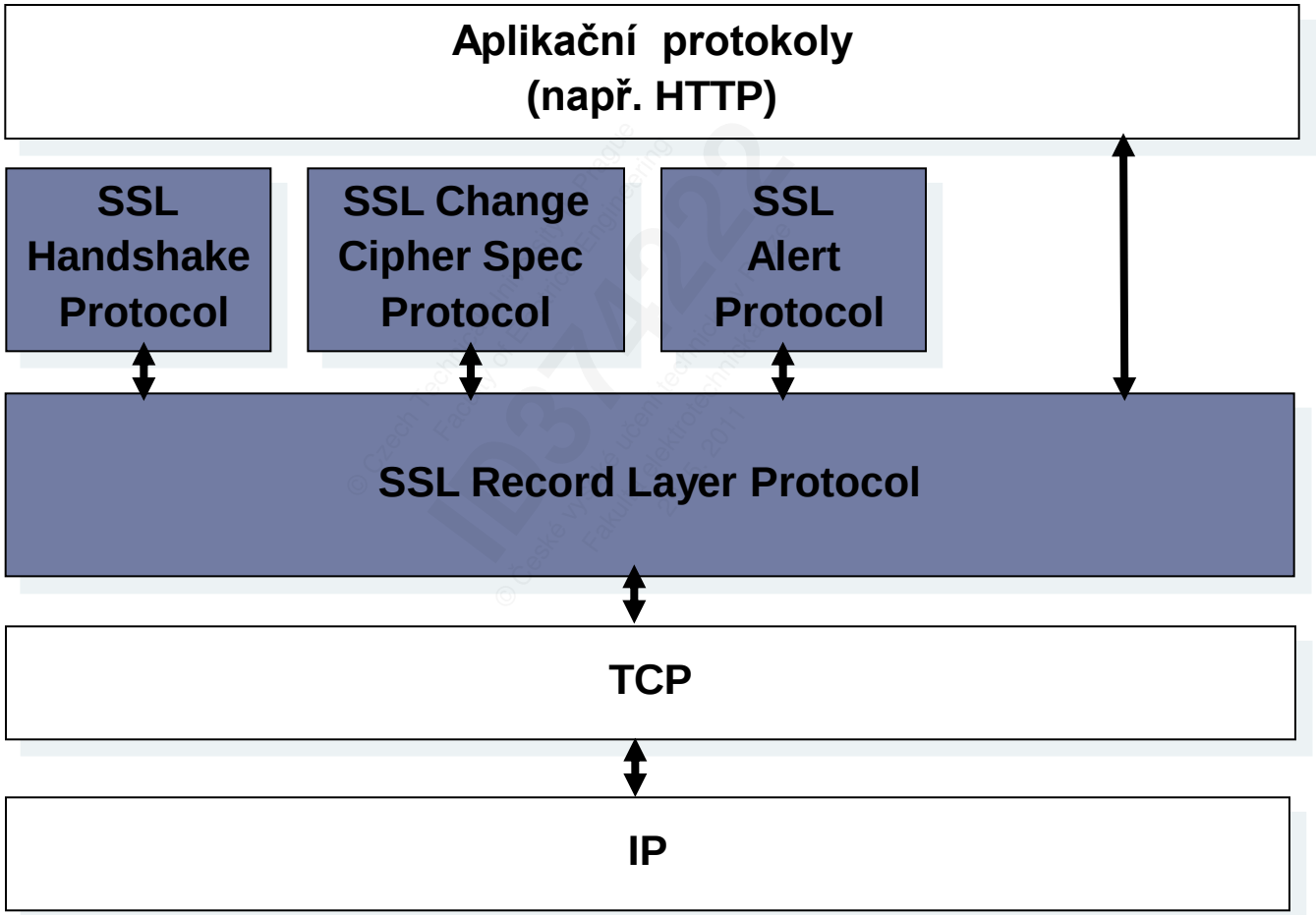
ZdrojE: THOMAS, *SSL AND TLS ESSENTIALS*



Získávání jednotlivých klíčů z balíku dat

Obrázek z knihy **Velký průvodce protokoly TCP/IP: Bezpečnost** (Libor Dostálek a kol.)

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

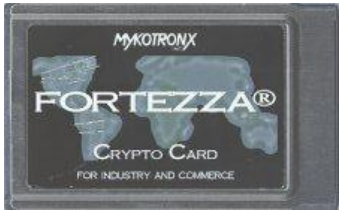
5) zašifruje datové bloky

- 2 možné typy algoritmů

proudové (**stream**) - nepotřebuje doplňování velikosti a záznam může být použit 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.

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



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

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:

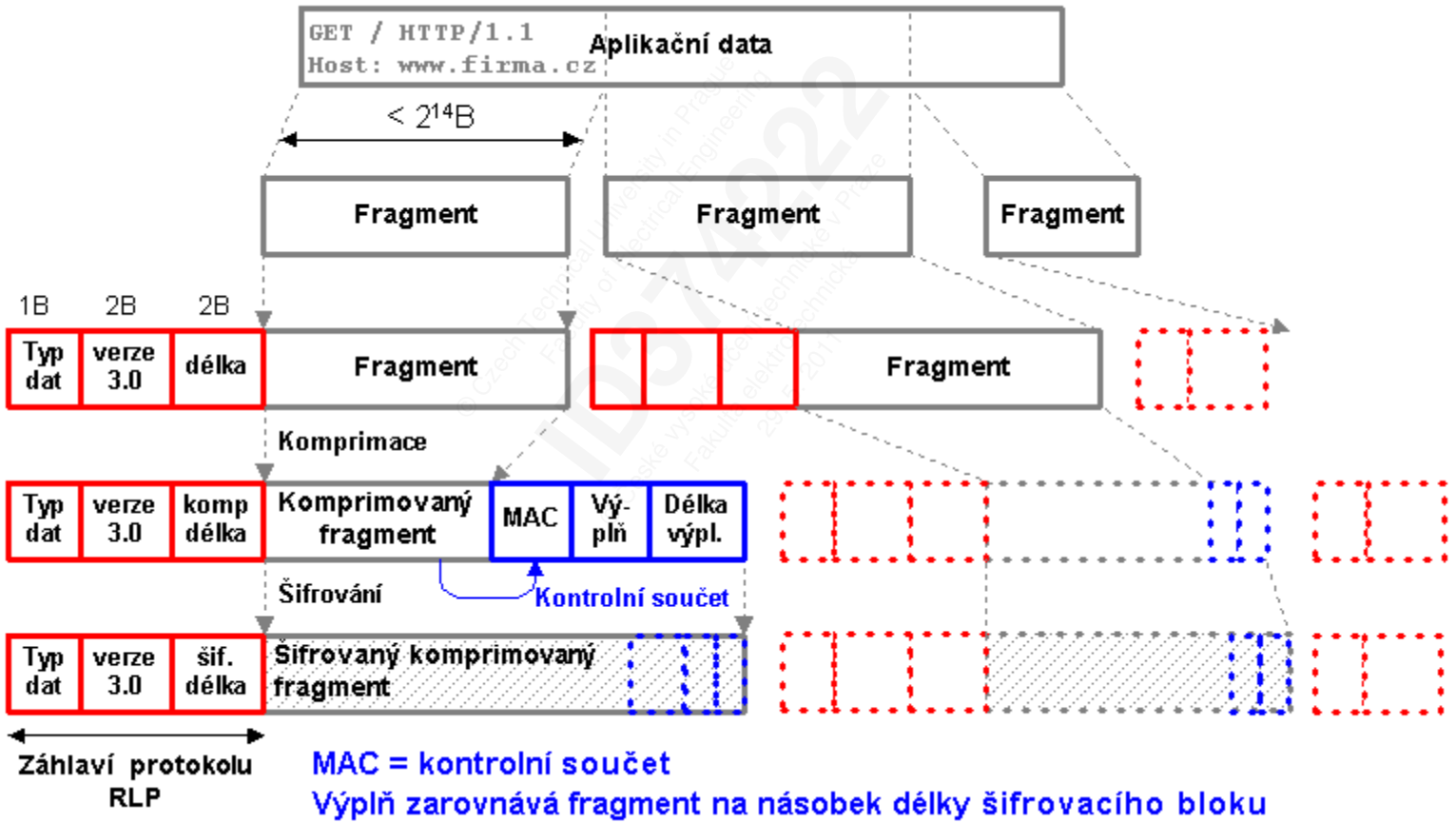
MAC-DATA = H(MAC-WRITE-SECRET || OPAD || H(MAC-WRITE-SECRET || IPAD, SEQUENCE-NUMBER || type || length ||fragment))

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

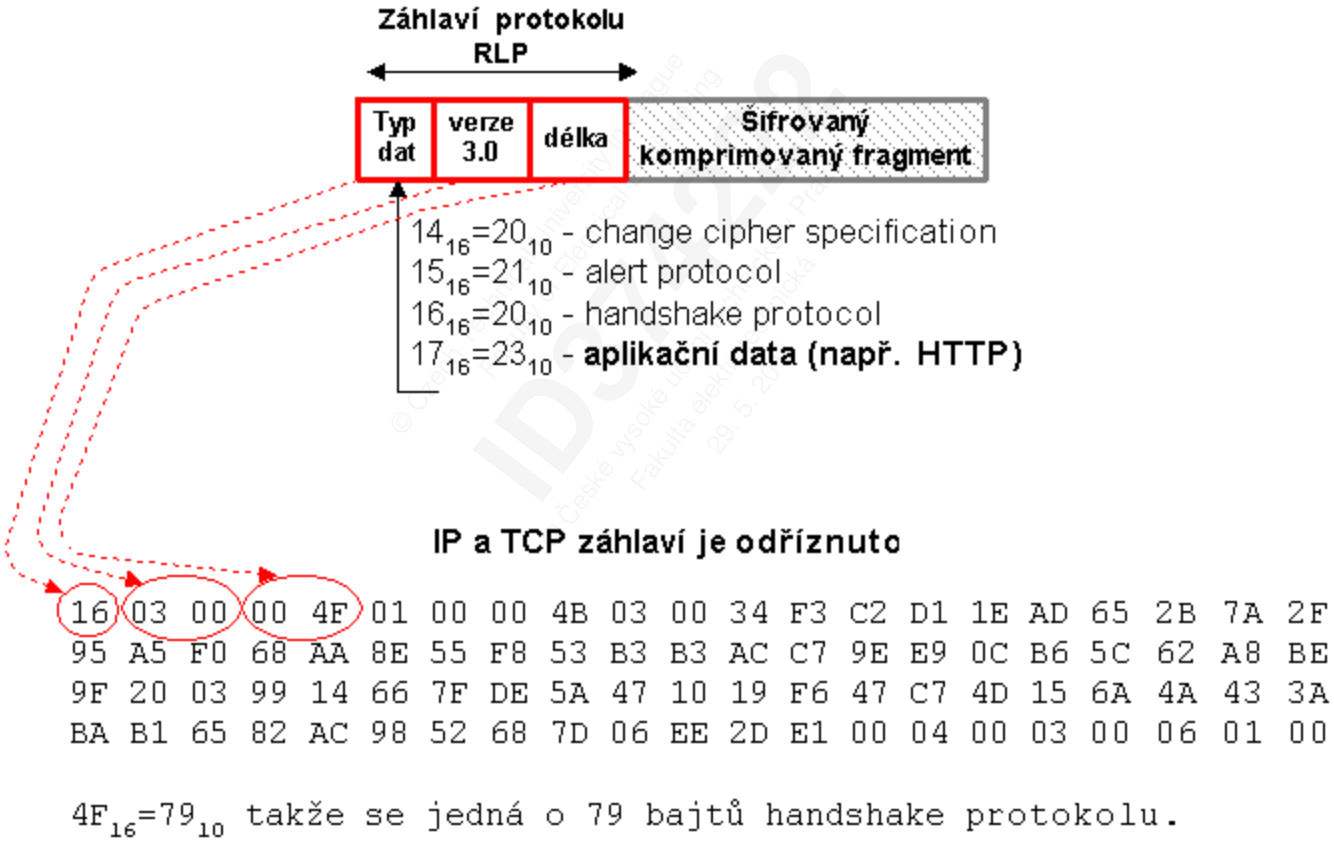
7) přidá hlavičku RLP před datový fragment
8) odevzdá data transportnímu protokolu TCP

RLP - úprava aplikačních dat



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

RLP - záhlaví



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

Délka - délka 2B - určení délky fragmentu

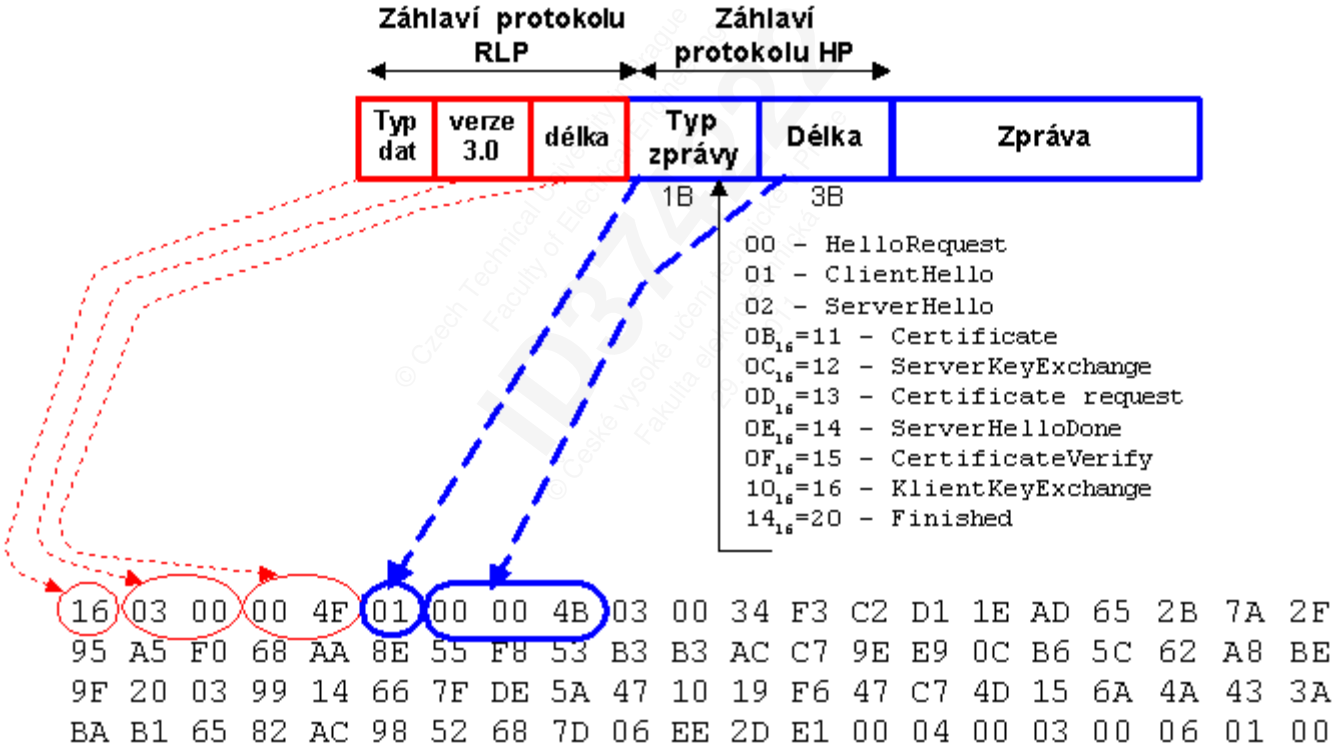
SSL Handshake Protokol

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)

Handshake Protocol



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

Dva způsoby komunikace pomocí HP:

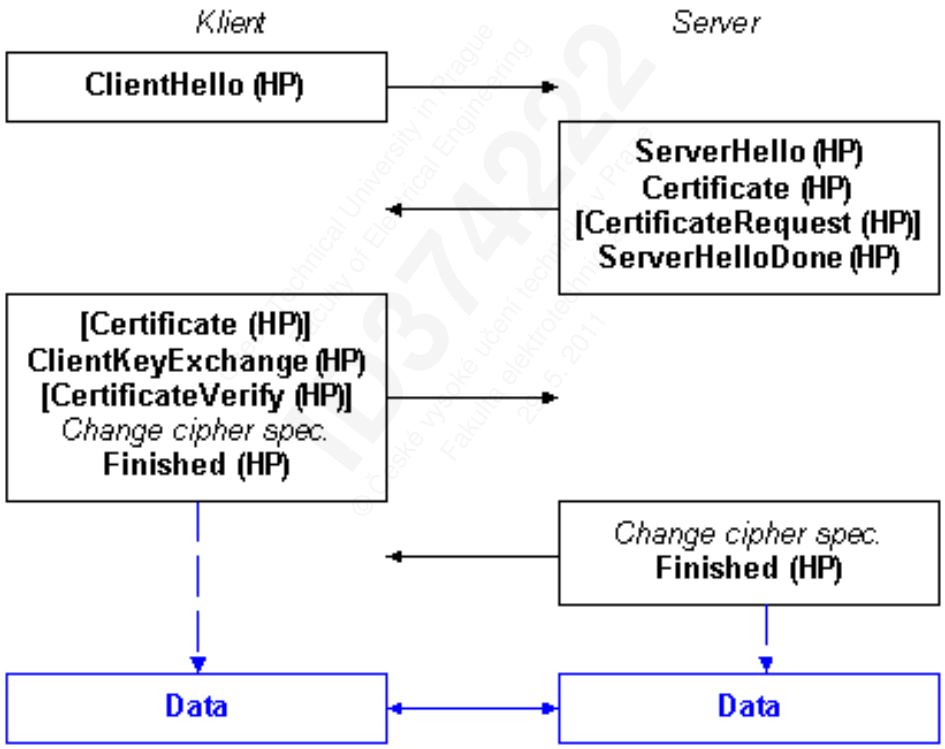
navazování nové relace (session)

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

obnova existující relace

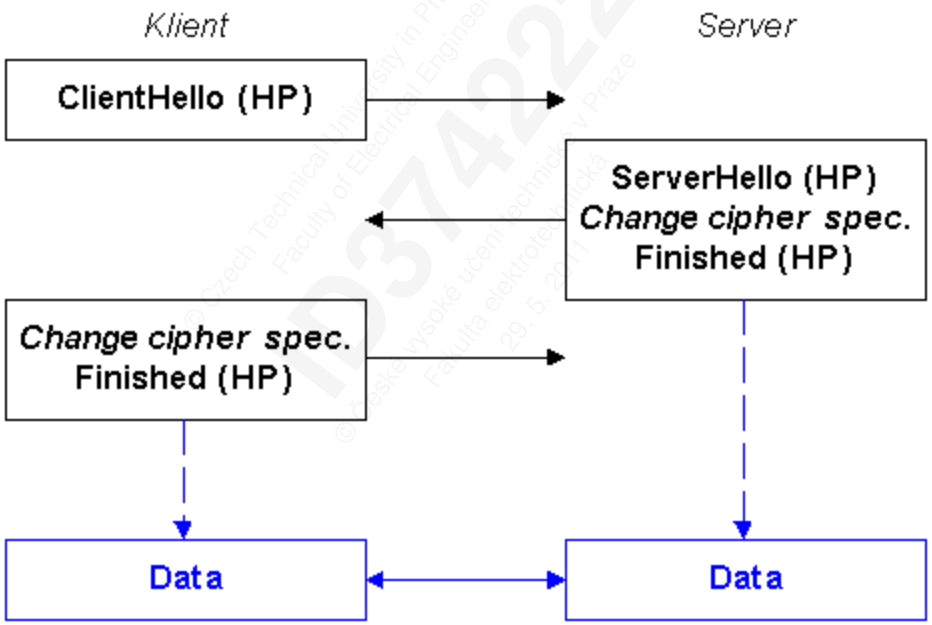
- pokud již existuje relace a je potřeba nové spojení
- není nutné provádět úplný handshake
- stačí zkrácená verze využívající existující session ID

Handshake Protokol - sestavení nové relace



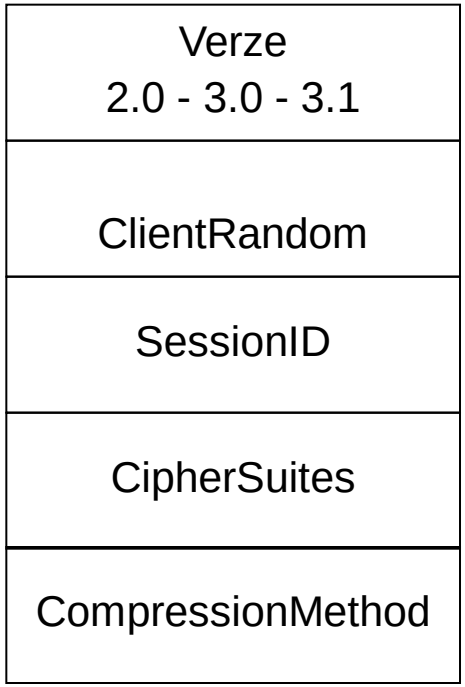
http://www.cpress.cz/knihy/tcp-ip-bezp/Cd-II/CD-ssl/ssl.htm

Handshake Protokol - obnovení existující relace



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

HP - zpráva Client Hello



client_version – 2B - nejvyšší verze podporovaná klientem

client_random - aktuální čas (4B) + 28B náhodných dat

session_id - 1B + prázdné, pokud klient chce vytvořit nové ID relace nebo ID existující relace, pokud chce klient vytvořit nové spojenív rámci existující relace

cipher_suites – 2B + 2B pro každou sadu

- obsahuje seznam kryptografických algoritmů podporovaných klientem seřazený jak je preferuje klient (šifrovací sada se skládá z šifrovacího algoritmu, způsobu výměny klíčů, algoritmu pro MAC (hash. funkce), velikosti IV a parametrů pro volbu klíčů

compression_methods – 1B + 1B na každou metodu - seznam klientem podporovaných způsobů komprese

Vybrané kryptografické sady pro SSL v3.0

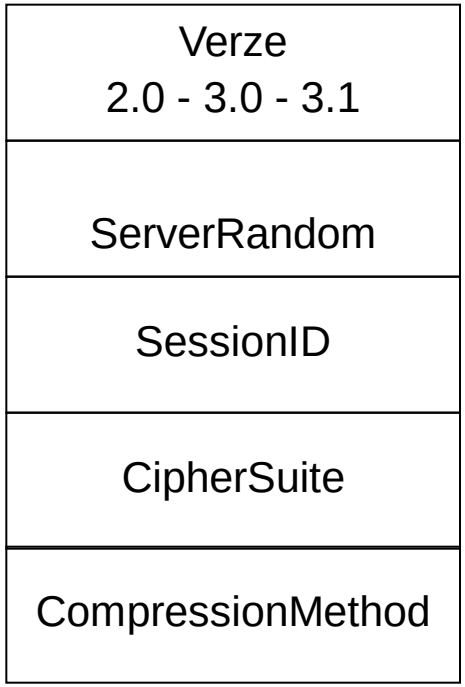
```
CipherSuite SSL_NULL_WITH_NULL_NULL                          = 00 00
    - je používána zejména na počátku komunikace
CipherSuite SSL_RSA_WITH_NULL_MD5                            = 00 01
CipherSuite SSL_RSA_WITH_NULL_SHA                            = 00 02
CipherSuite SSL_RSA_WITH_RC4_128_MD5                         = 00 04
CipherSuite SSL_RSA_WITH_RC4_128_SHA                         = 00 05
   ...
CipherSuite SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA                = 00 13
CipherSuite SSL_DHE_RSA_WITH_DES_CBC_SHA                     = 00 15
CipherSuite SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA                = 00 16
CipherSuite SSL_DH_anon_EXPORT_WITH_RC4_40_MD5               = 00 17
CipherSuite SSL_DH_anon_WITH_RC4_128_MD5                     = 00 18
CipherSuite SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA            = 00 19
CipherSuite SSL_DH_anon_WITH_DES_CBC_SHA                     = 00 1A
CipherSuite SSL_DH_anon_WITH_3DES_EDE_CBC_SHA                = 00 1B
CipherSuite SSL_FORTEZZA_KEA_WITH_FORTEZZA_CBC_SHA           = 00 1D
CipherSuite SSL_FORTEZZA_KEA_WITH_RC4_128_SHA                = 00 1E
```

- v SSL 3.0 je definováno celkem 28 kryptografických sad algoritmů

Vybrané kryptografické sady 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
...
```

HP - zpráva Server-Hello



server_version – 2B - min(nejvyšší verze podporovaná klientem, nejvyšší verze podporovaná serverem)

server_random – 32B - aktuální čas + náhodná data. Náhodná data musí být nezávislá na náhodných datech klienta

session_id – 1B + ID relace zvolené serverem, pokud klient chce obnovit starou relaci (server ověří, zda-li je možné relaci obnovit;pokud ano, odpoví ID relaci a proces sestavení spojení dál pokračuje). Pokud chce klient vytvořit novou relaci, server vygeneruje nové ID relace

cipher_suite – 2B - jedna zvolená kryptografická sada (volí jí server na základě nabídky od klienta – **standard neříká jak!**)

compression_method – 1B - serverem zvolená kompresní metoda (obvykle NULL)

HP – zpráva Certificate

Certificate

- posílá server klientovi
- vyžadován pro každý způsob výměny klíčů (kromě anonymní DH)
- obsahuje jeden certifikát X.509 nebo celý řetěz certifikátů (až ke kořenové CA)
- může obsahovat
  - veřejný RSA klíč určený k šifrování nebo
  - veřejný RSA nebo DSS klíč určený k pouze k podepisování nebo
  - pevné DH parametry

HP – zpráva Server_key_exchange

- je odeslán pouze tehdy, pokud certifikát neobsahuje dostatečné informace k výměně klíčů (např.: certifikát obsahuje pouze klíč RSA k podepisování)
- zpráva může obsahovat
  - veřejný RSA klíč (exponent a modul), nebo
  - parametry DH (p, g - veřejné DH hodnoty), nebo
  - parametery systému Fortezza
- zpráva je zajištěna digitálním podpisem
  - pokud je použit DSS: je podepsán hash (pomocí SHA-1) z (client_random | server_random | server_params)
  - pokud je použito RSA: MD5 hash a SHA-1 hash z (client_random | server_random | server_params) jsou zřetězeny a zašifrovány soukromým RSA klíčem

HP – zprávy Certificate request a Server_hello_done

Certificate_request

- odesílá Server pokud je potřeba, aby Klient autentizoval sám sebe
- specifikuje, jaký typ certifikátu je vyžadován (rsa_sign, dss_sign, rsa_fixed_dh, dss_fixed_dh, …)

Server_hello_done

- informuje Klienta, že Server ukončil svou úlohu při výměně klíčů
- po odeslání Server čeká na odpověď od klienta
- Klient by měl ověřit, že Server poskytl platný certifikát a že parametry spojení (vybrané Serverem) jsou přijatelné

HP – Certificate a Client_key_exchange

Client_Certificate

- posílá se pouze, pokud si ho vyžádá server
- může obsahovat
  - veřejný RSA nebo DSS klíč vhodný pouze k podepisování, nebo
  - pevné DH parametry

Client_key_exchange

- odesílá se vždy (pokud je použit DH algoritmus s pevnými klíči je prázdný)
- může obsahovat
  - pre-master secret zašifrovaný pomocí RSA, nebo
  - jednorázovou hodnotu parametrů DH schématu klienta, nebo
  - parametery systému Fortezza pro výměnu klíčů
- obsahuje *PreMasterSecret* zašifrovaný veřejným klíčem Serveru

HP - Certificate_verify

Certificate_verify

- odesílá se pouze pokud klient poslal certifikát
- zajišťuje autentizaci klienta
- obsahuje podepsaný hash všech předchozích zpráv protokolu handshake
  - pokud je použit DSS: je podepsán hash zprávy (pomocí SHA-1)
  - pokud je použito RSA: MD5 hash a SHA-1 hash ze zpráv jsou zřetězeny a zašifrovány soukromým RSA klíčem

MD5(master_secret || pad2 || MD5(handshake_messages || master_secret || pad1))

SHA(master_secret || pad_2 || SHA(handshake_messages || master_secret || pad1))

HP - Finished

- Odesílá se okamžitě po zprávě change_cipher_spec
- První zpráva, která používá nově sjednané algoritmy, klíče, IV atd.
- Používá se k ověření faktu, že se výměna klíčů a autentizace povedla
- obsahuje MD5 a SHA-1 hash všech předchozích zpráv:

  MD5(master_secret || pad2 || MD5(handshake_messages || sender || master_secret | pad1)) || SHA-1(master_secret | pad2 || SHA-1(handshake_messages || sender || master_secret || pad1))

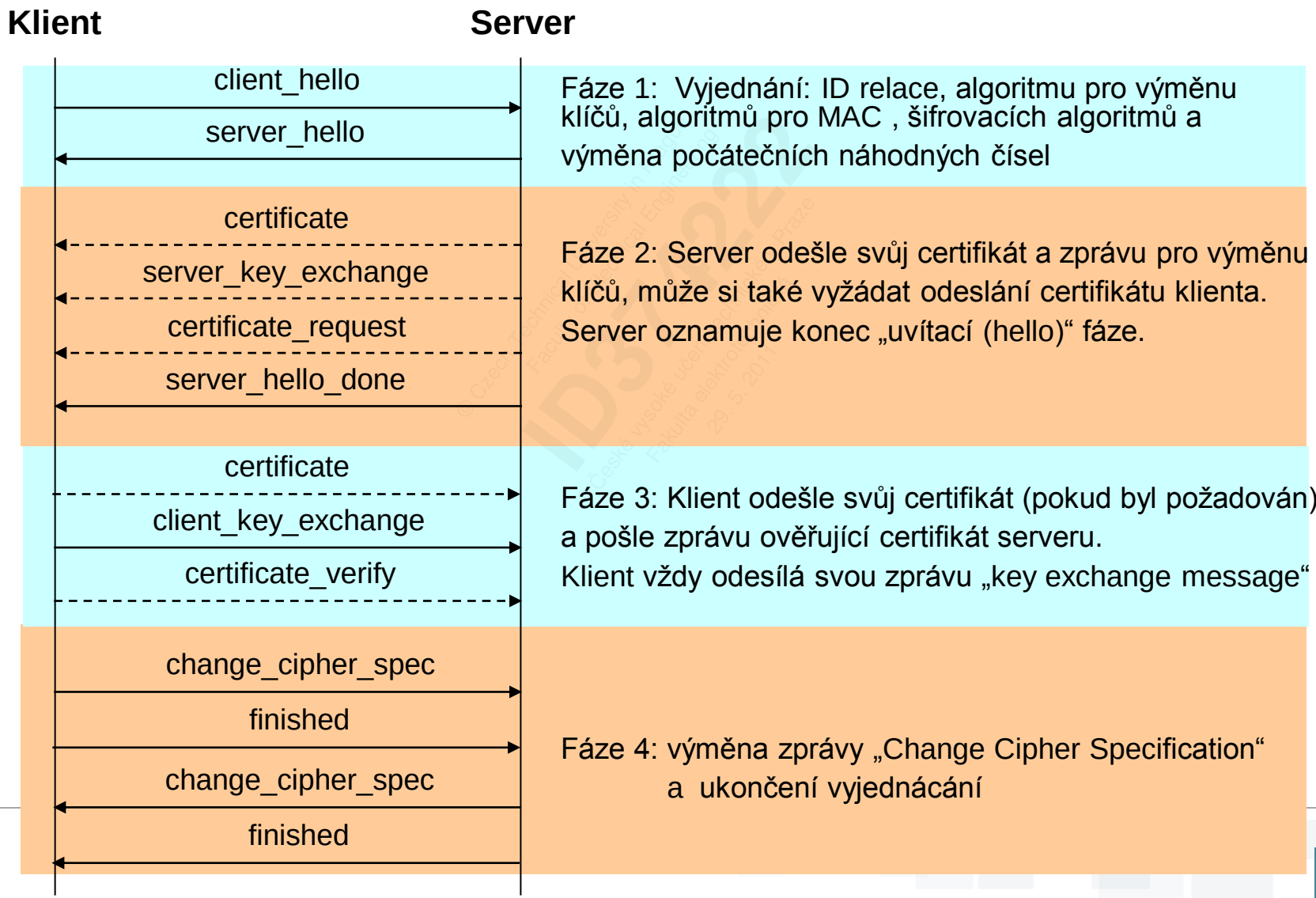
  kde “sender” je kód, který identifikuje jestli je odesílatel Server nebo Klient (klient: 0x434C4E54; server: 0x53525652)

- Toto umožní příjemci kontrolu, zda-li nebyla data během handshake fáze neoprávněně modifikována.

HP – zpráva HelloRequest

- prázdná zpráva
- málo obvyklé
- Server vyzývá Klienta, aby mu zaslal zprávu ClientHello a odstartoval tak výměnu klíčů

SSL Handshake Protokol



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

RSA s autentizací klienta – verze 2

- server ve zprávě `server certificate` odešle svůj veřejný podepisovací klíč RSA nebo DSA klíč
- server ve zprávě `server key exchange` odešle jednorázový RSA veřejný klíč
- klient ve zprávě `client certificate` odešle svůj veřejný podepisovací klíč (algoritmus RSA nebo DSA)
- 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

Permanentní DH bez autentizace klienta

- server ve zprávě `server_certificate` odešle své konstantní DH parametry podepsané CA
- zpráva `server_key_exchange` se neposílá
- klient ve zprávě `client_key_exchange` posílá své parametry pro jednorázovou DH výměnu
- zprávy `client_ certificate` a `certificate_verify` se neposílají

- kryptografické sada začíná na SSL_DH_…

Výměna klíčů - alternativní

Permanentní DH s autentizací klienta

- server ve zprávě `server_certificate` odešle své konstantní DH parametry podepsané CA
- zpráva `server_key_exchange` se neposílá
- klient ve zprávě `client_certificate` odešle své konstantní DH parametry
- zpráva `client_key_exchange` se posílá, ale je prázdná
- zpráva `certificate_verify` se neposílá

- kryptografické sada začíná na SSL_DH_…

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_…

Výměna klíčů - alternativní

Jednorázová DH s autentizací klienta

- server odešle ve zprávě `server_certificate` svůj podepsaný veřejný klíč (RSA nebo DSA)
- server ve zprávě `server_key_exchange` odešle veřejné hodnoty pro jednorázovou DH výměnu
- klient odešle ve zprávě `client_certificate` svůj podepsaný veřejný klíč (algoritmus RSA nebo DSA)
- klient ve zprávě `client_key_exchange` odešle veřejné hodnoty pro jednorázovou DH výměnu
- klient ve zprávě `certificate_verify` odešle podpis všech svých předchozích zpráv
- kryptografické sada začíná na SSL_DH_RSA_with… nebo SSL_DH_DSS_with…

Výměna klíčů - alternativní

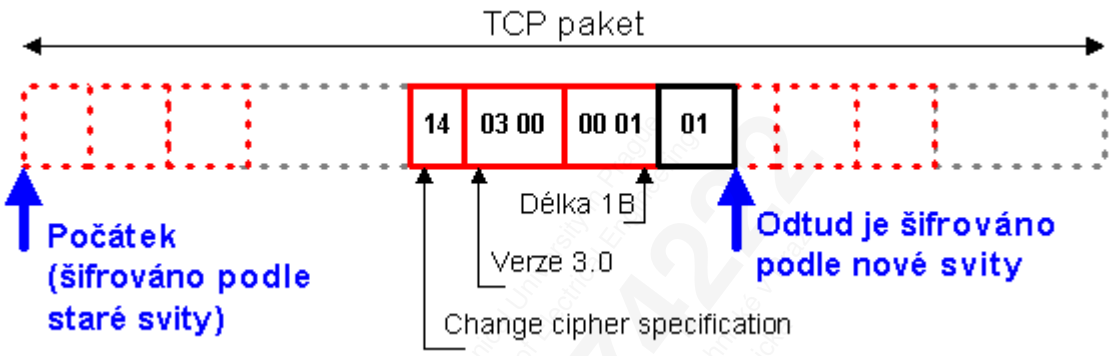
Anonymní DH bez autentizace klienta

- zpráva `server_certificate` se neposílá
- server pošle (nepodepsané) jednorázové parametry DH protokolu ve zprávě `server_key_exchange`
- klient ve zprávě `client_key_exchange` odešle jednorázové veřejné hodnoty DH protokolu
- zprávy `client_certificate` a `certificate_verify` nejsou odeslány
- server i klient vygenerují jednorázové parametry pro DH algoritmus a odešlou je (neprobíhá zde žádné ověření jejich původu)
- kryptografické sada začíná na SSL_DH_anon_...

Anonymní DH s autentizací klienta

- není povolena

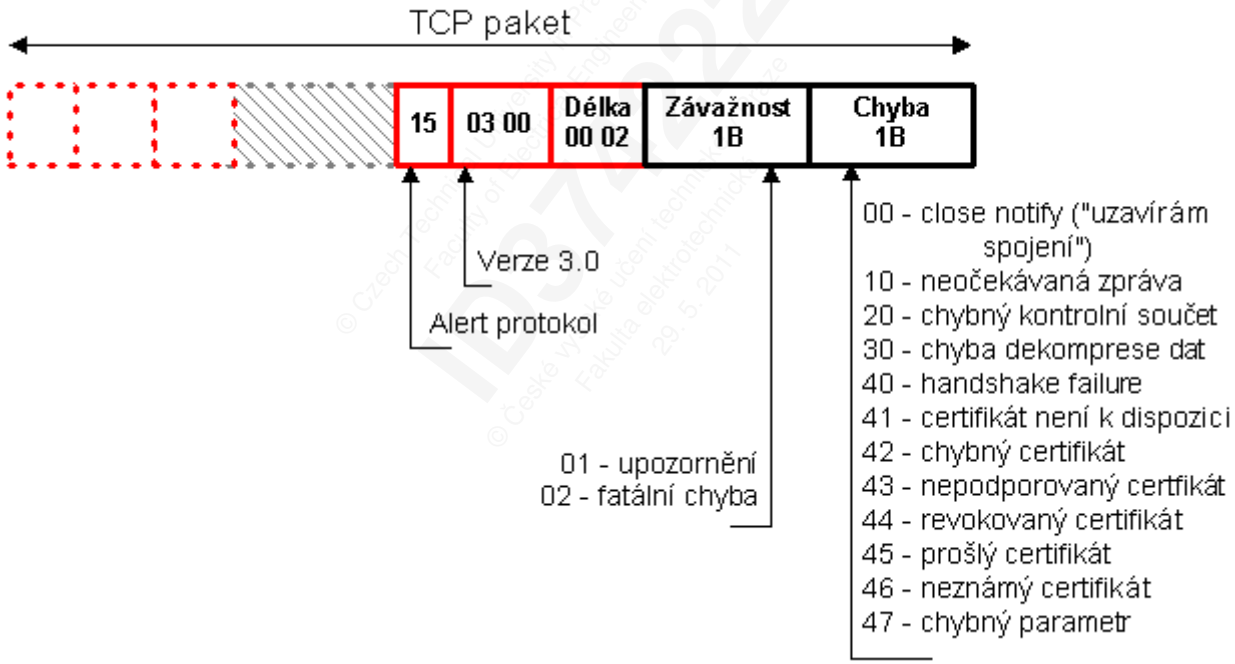
CCSP - Change Cipher Specification Protocol



- 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
- v AP není podporována zpráva s kódem 01 41
  - 01 – informační
  - 41 – no_certificate
- 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
- nejprve sestrojíme expanzní funkci P_hash(secret,seed)

  $P\_hash(secret, seed) = HMAC\_hash(secret, A_1 \| seed) \| HMAC\_hash(secret, A_2 \| seed) \| HMAC\_hash(secret, A_3 \| seed) \| \ldots$

  kde

  $$A_0 = seed$$
  $$A_i = HMAC\_hash(secret, A_{i-1})$$

  $$PRF(secret, label, seed) = P\_MD5(secret\_left, label \| seed) \oplus P\_SHA(secret\_right, label \| seed)$$

  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 1.1

- 04/2006
- RFC 4346
- několik menších vylepšení v oblasti bezpečnosti
- srozumitelnější popis činnosti
- implicitní IV je nahrazen explicitním IV (ochrana proti útokům na CBC)
- chyby související s výplní bloků mají jiný kód (bad_record_mac alert místo decryption_failed)
- předčasné ukončení spojení nezpůsobí fatální chybu

- nové poznámky o útocích na TLS
- řada vyjasnění a drobných redakčních zásahů

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-CGM)
- 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
  - pro virtual hosting
  - u SSL 2.0/3.0 TLS 1.0 /1.1 musí mít každý HTTP s podporou SSL/TLS server unikátní IP adresu
- 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

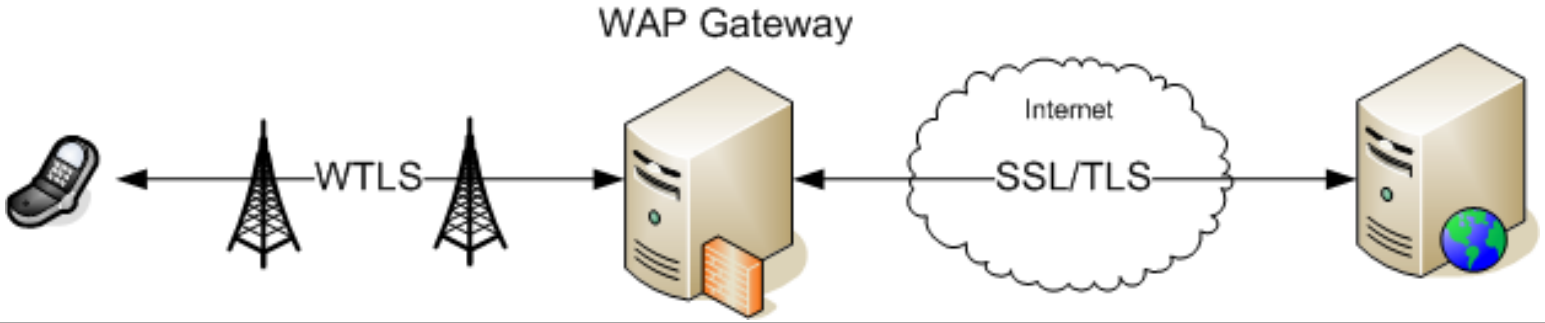
- RFC5054 - 9 protokolových sad s SRP
- autentizace klienta při TLS handshaku jménem/heslem nešifrovaným kanálem
- heslo nelze odposlechnout
- založeno na DH …řeší i výměnu sdíleného tajemství

Protokoly využívající SSL/TLS - příklad

Aplikační protokol	Číslo TCP portu
HTTPS (HTTP Secure)	443
SMTPS (SMTP over SSL/TLS)	465
NNTPS (NNTP over SSL/TLS)	563
LDAPS (LDAP over SSL/TLS)	636
FTPS (FTP control over SSL/TLS)	989
FTPS-data (FTP data over SSL/TLS)	990
TELNETS (Telnet over SSL/TLS)	992
IMAPS (IMAP4 over SSL/TLS)	993
POP3S (POP3 over SSL/TLS)	995

WTLS - Wirelles Transport Layer Security

- zabezpečuje spojení mezi mobilním zařízením (klient) a WAP bránou
- zajišťuje: utajení přenášených dat, integritu, autentizace (pomocí certifikátů), ochranu před replay útoky
- skládá se z :
  - WTLS Handshake protokolu
  - WTLS Alert protokolu
  - WTLS Change Cipher Specification protokolu
  - WTLS Record Layer protokolu
- velké rozdíly oproti TLS

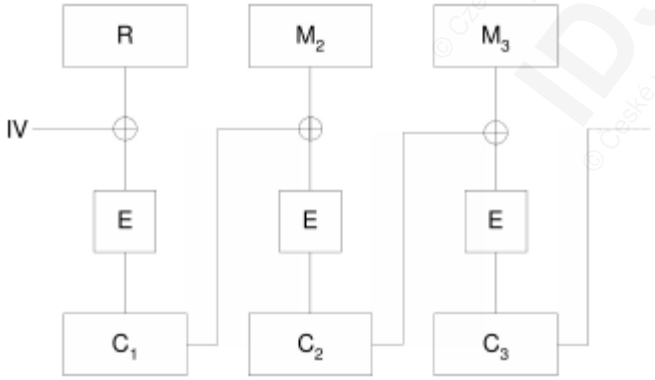


Datagram Transport Layer Security

- 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

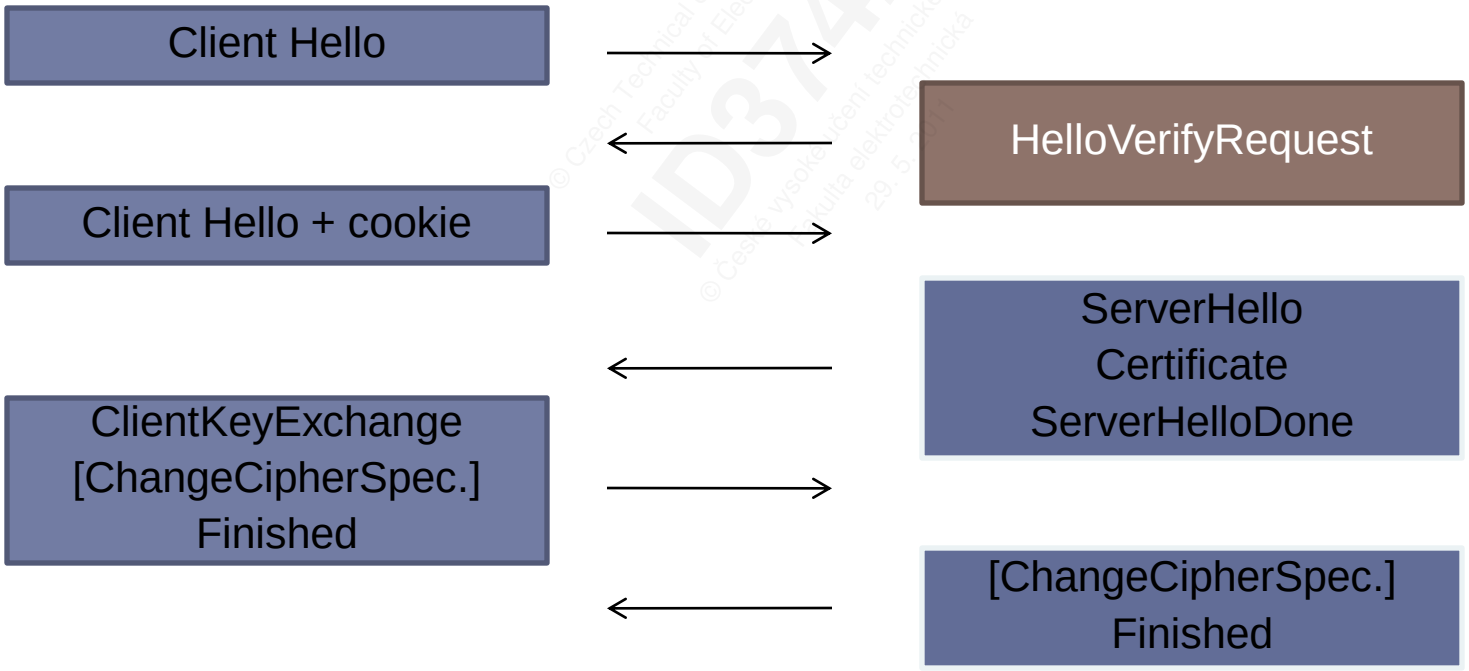
- nelze použít kryptografické sady z TLS 1.0
  - udržují vztah mezi jednotlivými záznamy a to vyžaduje jejich bezpečné doručení (není u DTLS garantováno kvůli UDP)
- lze použít CBC režim s explicitní definicí IV
  - definovaný v TLS 1.1 a AES-CM



CBC s explicitním IV

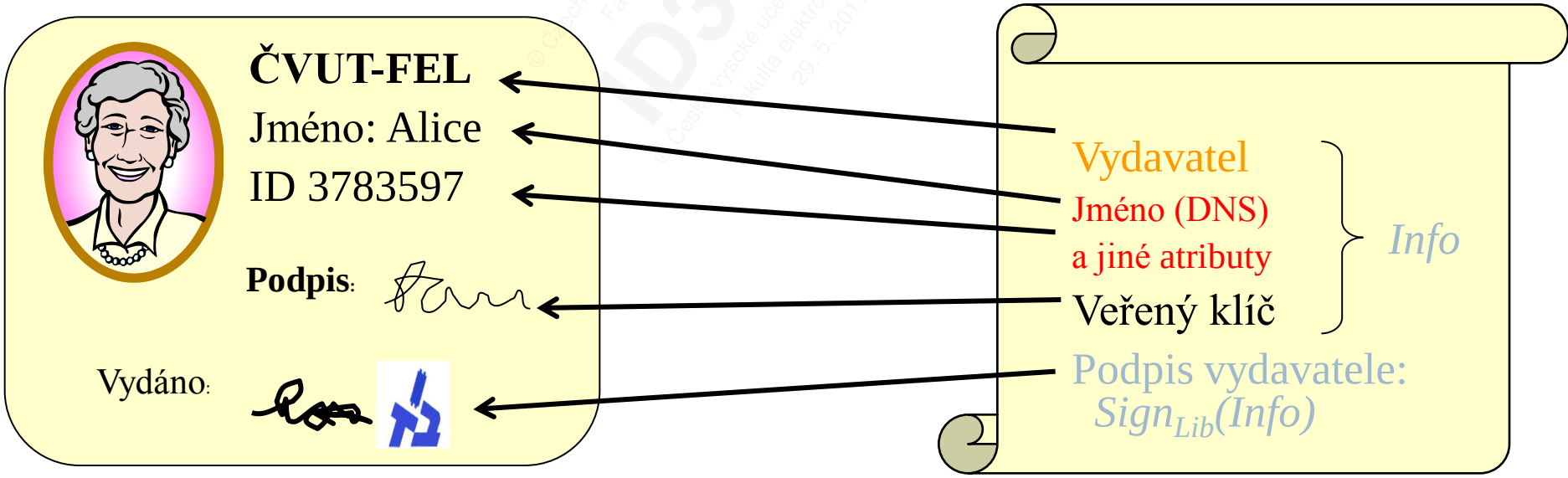
DTLS

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

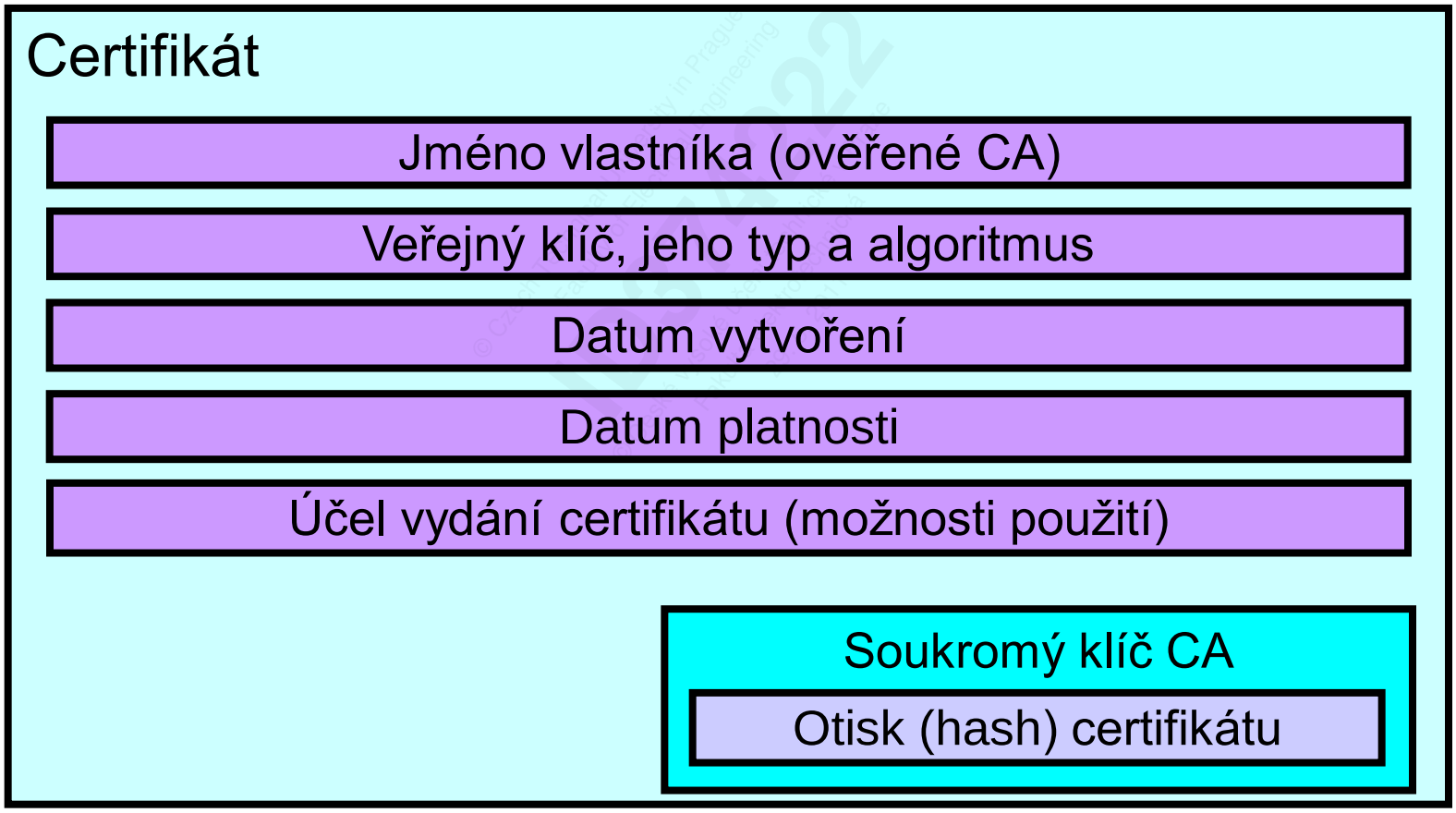


Certifikát

- Ekvivalent k OP nebo cestovnímu pasu
- Přiřazuje veřejný klíč ke jménu nebo jiným atributům držitele klíče (např. DNS jména u webových stránek)
- Podepsáno důvěryhodnou stranou (Issuer / Certifikační Aurtorita)
- Umožňuje důvěřující straně (Bob, client) ověřit jméno a a atributy vlasnítka (Alice, web site)

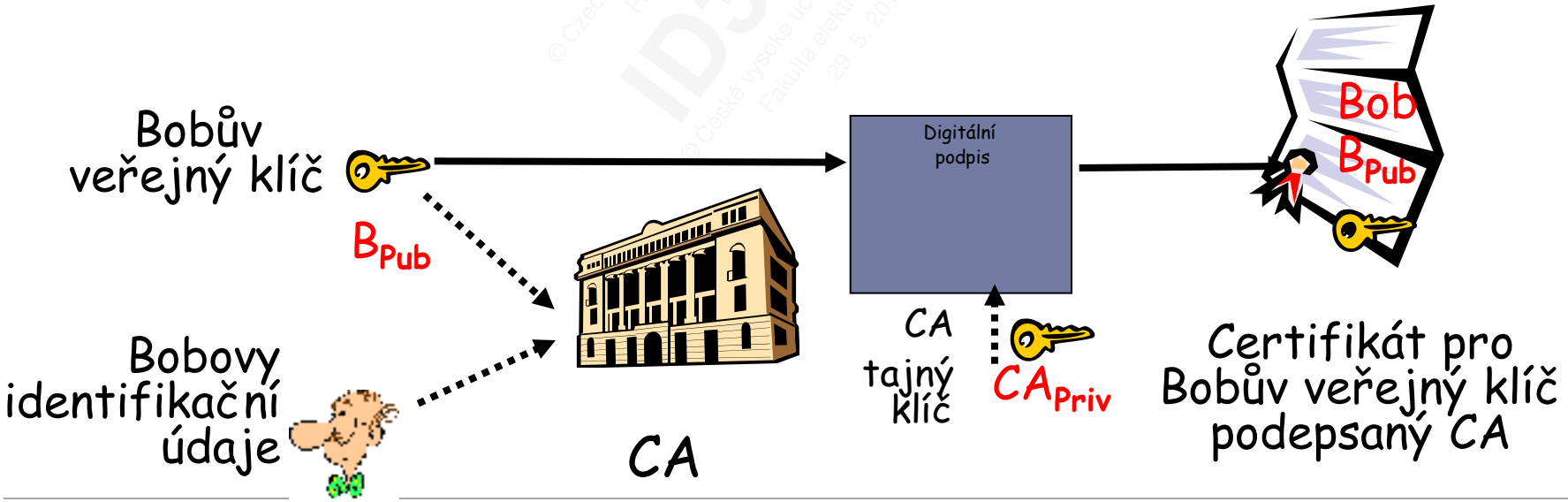


V současné době používá SSL certifikáty veřejných klíčů podle doporučení ITU X.509.



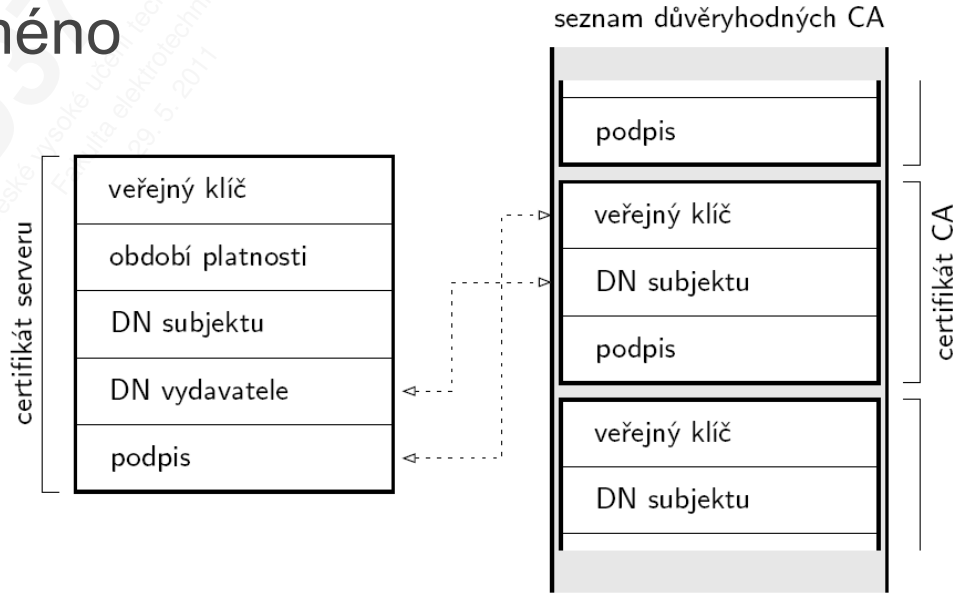
Certifikační Autorita

- Certifikační autorita (CA): svazuje veřejný klíč (např. B_{Pub}) s identifikátorem (např. jméno: `Bob`).
- Bob (osoba, server) registruje svůj veřejný klíč B_{Pub} u CA.
  - Bob přesvědčí CA, že se jmenuje Bob, odešle B_{Pub}
  - CA vytvoří certifikát svazující jméno“Bob” s jeho veřejným klíčem
  - Certifikát je digitálně podepsán CA – CA tím říká “B_{Pub} je `Bobův veřejný klíč”



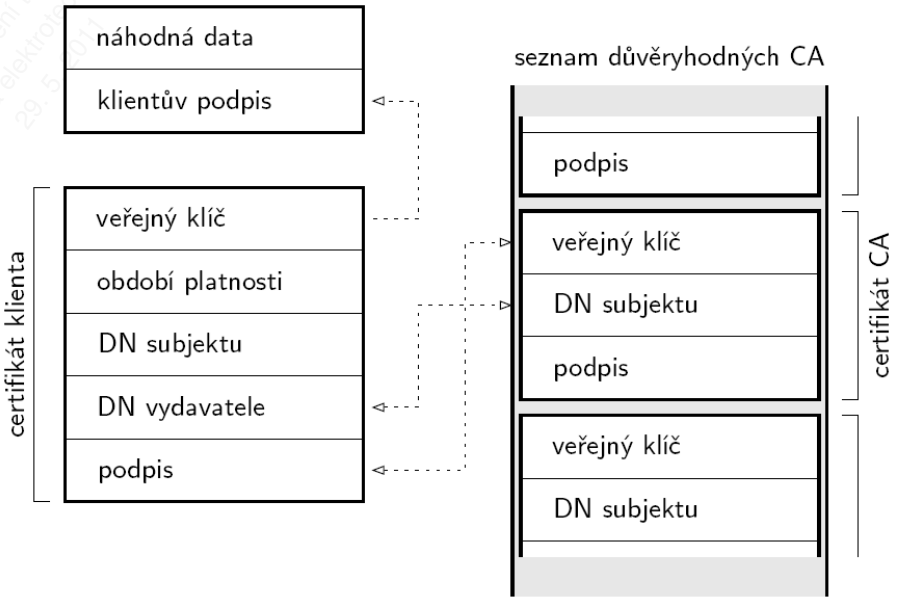
Autentizace serveru - automaticky ověřovaná pole

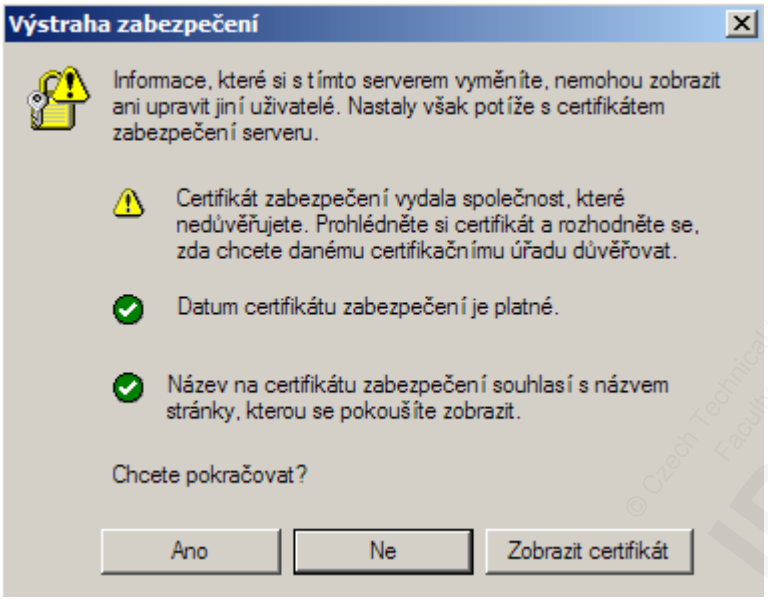
- ověření doby platnosti certifikátu
- je vydávající CA důvěryhodná?
- lze veřejným klíčem
- CA ověřit pravost certifikátu?
- odpovídá doménové jméno uvedené v certifikátu skutečnému jménu serveru?



Autentizace klienta - kroky

- lze klientův podpis ověřit pomocí jeho veřejného klíče?
- je certifikát klienta platný?
- je vydávající CA důvěryhodnou CA?
- lze veřejným klíčem CA ověřit pravost podpisu vydavatele?
- je klientův certifikát uveden v seznamu klientských certifikátů?
- je autentizovaný klient autorizován pro přístup k prostředkům?



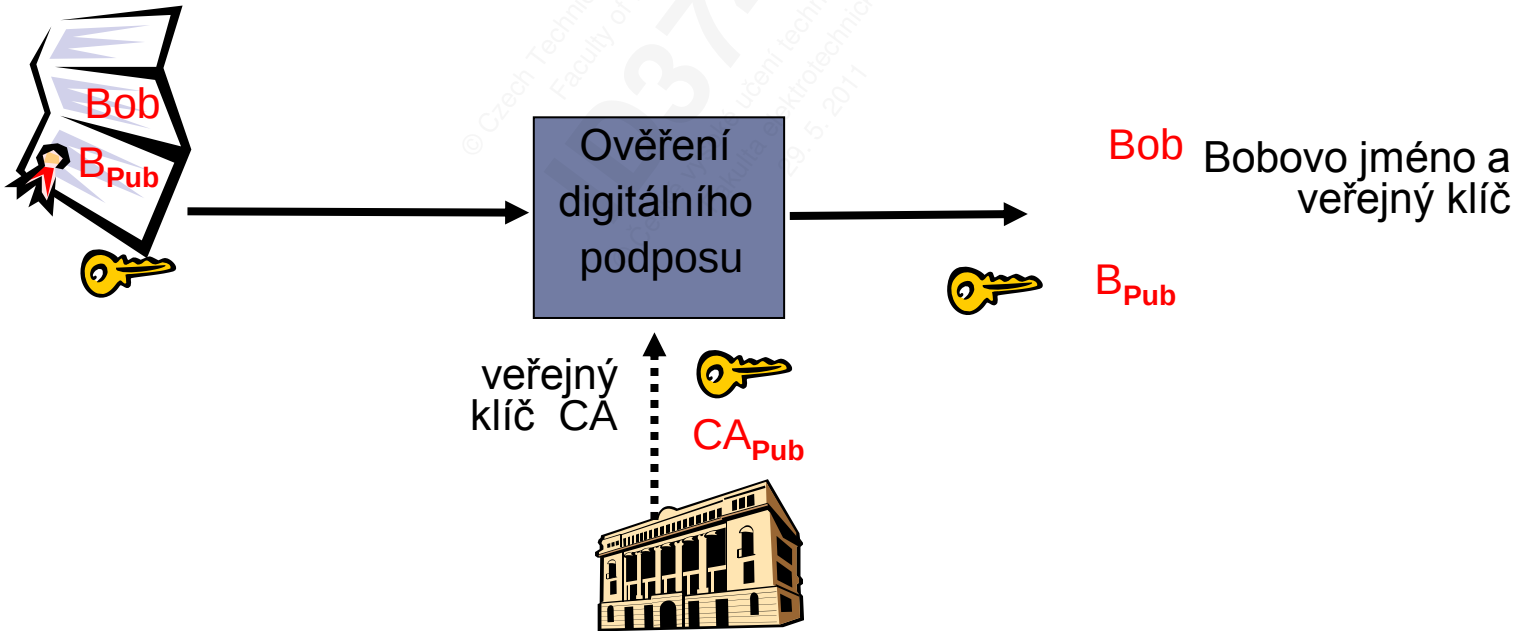


Certifikát - potvrzuje totožnost toho, kdo se jím prokazuje.
Certifikáty vydává Certifikační Autorita (CA).
CA - důvěryhodná pro obě strany

Použití certifikátu

Pokud chce Alice použít Bobův veřejný klíč (buďto k poslání zašifrované zprávy nebo k ověření Bobova podpisu) musí:

- získat Bobův certifikát (od Boba nebo odjinud)
- použít veřejný klíč CA k ověření platnosti Boba certifikátu
- po ověření získat z certifikátu Bobův veřejný klíč



Dotazy