TECHNICKÁ UNIVERZITA V KOŠICIACH
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

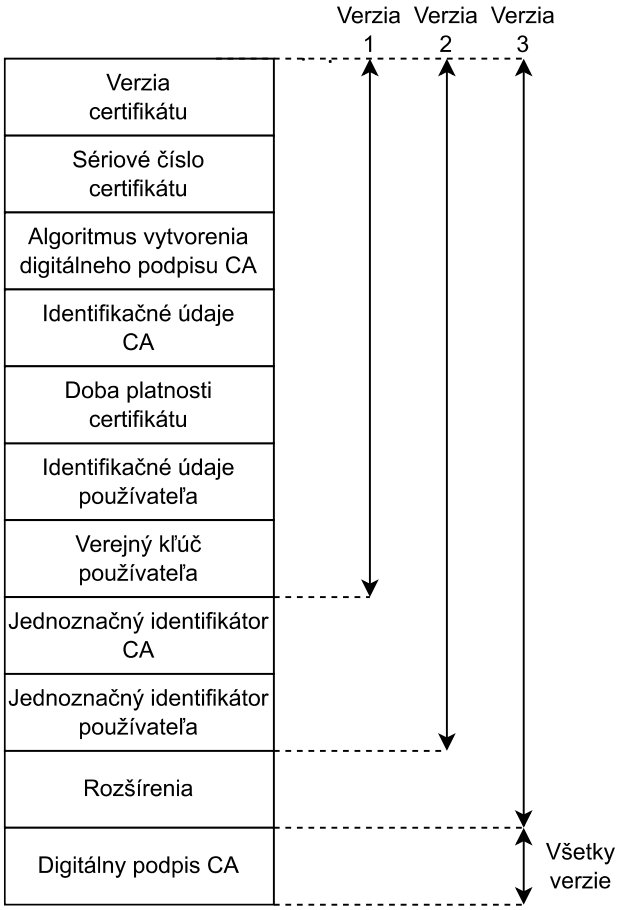
KLIENT-SERVER ZABEZPEČENÁ KOMUNIKÁCIA

12.09.2023 **Bc. Martin Janitor**

Obsah

1. Základné pojmy

Certifikát - digitálny dokument, ktorý overuje identitu osoby, organizácie alebo zariadenia. Poskytuje spôsob, ako bezpečne vymieňať informácie cez sieť tým, že sa vytvára dôvera medzi komunikujúcimi stranami. Certifikáty sa bežne používajú na zabezpečenú komunikáciu, autentifikáciu a digitálne podpisy.

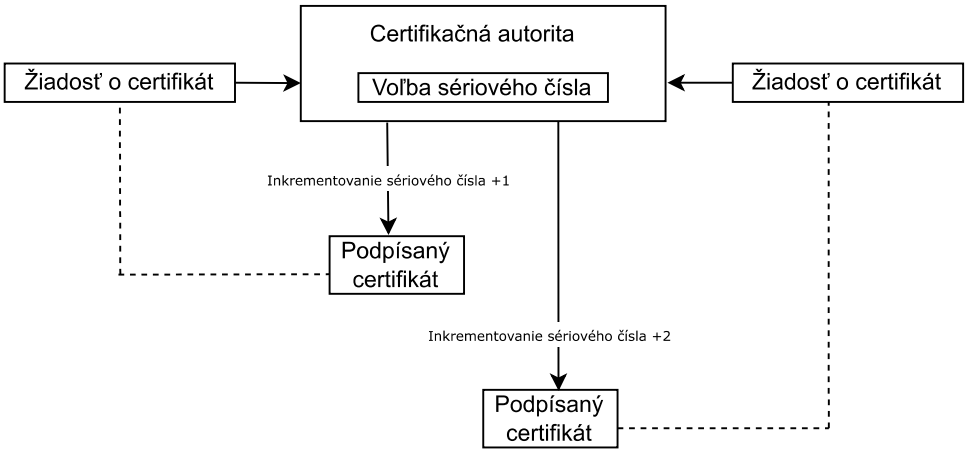


Obr.1 *Obsah premenných v certifikáte*

Sériové číslo - nezáporné celé číslo, ktoré sa priradí k certifikátu počas procesu vytvárania certifikátu. Sériové číslo je jedinečný identifikátor priradený ku každému certifikátu na odlíšenie od ostatných vydaných rovnakou certifikačnou autoritou (CA). Keď je certifikát predložený dôveryhodnej strane, táto strana môže použiť sériové číslo na overenie podpisu, že certifikát je platný a vydávaný dôveryhodnou CA.

V certifikáte sériové číslo volí CA, ktorá certifikát vydala. Sériové číslo môže byť dlhé najviac 20 bajtov. V OpenSSL knižnici sa sériové číslo vygeneruje do *.srl* súboru. OpenSSL štandardne generuje sériové číslo pomocou jednoduchej inkrementálnej schémy. Začína náhodnou počiatočnou hodnotou a pre každý nový certifikát ju inkrementuje. Počiatočná hodnota sa často odvíja od kombinácie informácií špecifických pre systém, ako je aktuálny čas alebo náhodný *seed*, aby sa zabezpečila jedinečnosť.

Princíp priradenia sériového čísla je uvedený v prílohe *c. 6.5*

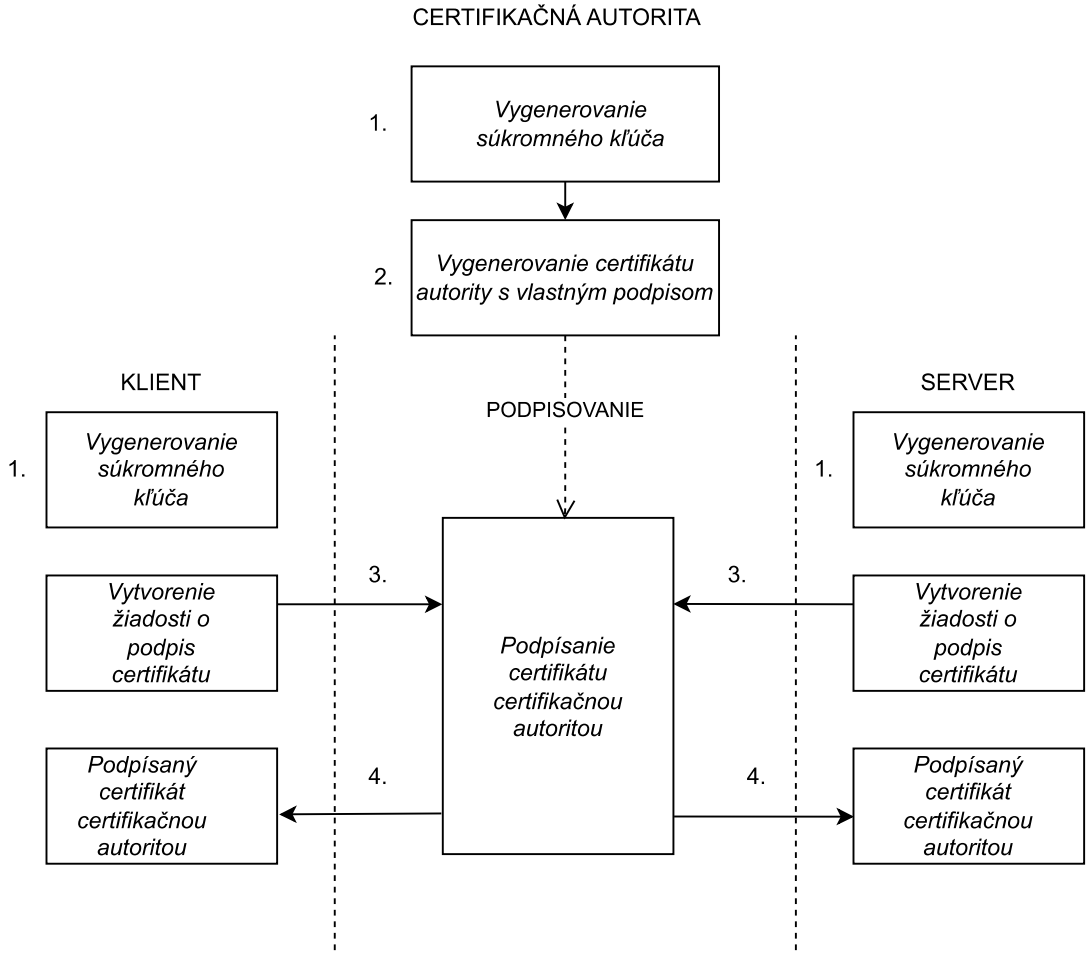


Obr .2 *Voľba a priradenie sériového čísla certifikačnou autoritou – Žiadosť o certifikát je zaslaná certifikačnej autorite, certifikačná autorita inkrementuje sériové číslo a podpíše žiadateľov certifikát s inkrementnutím sériovým číslom a odošle žiadateľovi, nasledujúci žiadateľ bude disponovať sériovým číslom vyšším o 1 ako predošlý žiadateľ.*

2. Vygenerovanie certifikátov

Štruktúra aplikácie OpenSSL pozostáva z nasledujúceho vzoru: ***openssl command [options]***

1. **openssl** - názov aplikácie
2. **command** - OpenSSl poskytuje množstvo možnosti pre prácu s kryptografickými algoritmami, príkazom *command* sa vyberie konkrétna možnosť pre prácu s daným algoritmom. Výpis všetkých možnosti, ktoré ponúka OpenSSL knižnica je možné vypísať zadaním príkazu: *openssl list -commands*
3. **options** - Prislúchajúce možnosti, nastavenia k danému commandu. Výpis všetkých nastavení k danému commandu, ktoré ponúka OpenSSL knižnica je možné vypísať zadaním príkazu: *openssl [command] –help*



Obr.3 *Proces generovania certifikátov – Vygenerovanie certifikátu s vlastným podpisom (certifikačná autorita), ktorá následne podpisuje certifikáty žiadateľom (klient a server)*

2.1. Vygenerovanie súkromného kľúča [Krok c.1 Obr.3]

Príklad [ECC]*: openssl ecparam -genkey -name prime256v1 -out myCA.key*

Príklad [RSA]: *openssl genrsa -out myCA.key 1024*

2.1.1. Prepínače špecifické pre ECC

ecparam: command ktorý poskytuje prácu s eliptickými krivkami

-genkey: pomocou nastavených parametrov [*options*] alebo načítaných parametrov sa vygeneruje súkromný kľuč

-name [typ_krivky]: Určuje typ eliptickej krivky s ktorou sa bude pracovať. Zoznam všetkých dostupných kriviek je možné vypísať príkazom: *openssl ecparam -list_curves*

2.1.2. Prepínače špecifické pre RSA

genrsa: command ktorý poskytuje generovanie RSA kľúčov

-[číslo]: veľkosť výstupného kľúča v bitoch

2.1.3. Všeobecné prepínače

-out [názov_súboru]: názov predstavuje názov súboru kde sa zapíšu požadované údaje

Výpis súkromného kľúča je možné zobraziť pomocou príkazu: *openssl ec -in example.key -text -noout [Príklad uvedený v prílohe 6.1]*

2.2. Vygenerovanie certifikátu certifikačnej autority s vlastným podpisom [Krok c.2 Obr.3]

Príklad: *openssl req -x509 -config certificate-authority-options.conf -new -nodes -key myCA.key -sha256 -days 365 -out myCA.pem*

req: command req sa používa na vytvorenie a manipuláciu s certifikátom vo formáte *PKCS#10*

-x509: Keď je špecifikovaná voľba "-x509", OpenSSL generuje samopodpísaný certifikát namiesto žiadosti o podpísanie certifikátu (*CSR*). Samopodpísaný certifikát je certifikát, kde subjektom certifikátu je aj vydávateľ certifikátu. Obvykle sa používa v prípadoch, keď nie je k dispozícii certifikačná autorita (*CA*).

-config [názov_súboru.conf]: umožňuje zadať alternatívny konfiguračný súbor *[Príklad uvedený v prílohe 6.4]*.

-new*:* vygeneruje sa nová žiadosť o certifikát

-notes: vytvorený súkromný kľúč, nebude uložený v šifrovanom formáte

-key [názov_súboru]: názov súboru obsahujúci súkromný kľúč, ktorý sa má použiť v žiadosti o certifikát

-[hashovacia_funkcia]: Pri generovaní žiadosti o vydanie certifikátu hashovacie funkcie slúžia na výpočet odtlačku správy (hash), ktorý sa následne podpíše súkromným kľúčom uchovávaným u žiadateľa o certifikát. Podpísaný odtlačok správy sa potom pridá do samotnej žiadosti o certifikát a pošle sa certifikačnej autorite (CA) na overenie a vydanie certifikátu. Výpis všetkých dostupných hashovacích funkcií je možné zobraziť príkazom: *openssl dgst -list*

-days [číslo]: určuje počet dní pre ktoré bude certifikát platný

-out [názov]: Názov predstavuje názov súboru kde sa zapíšu požadované údaje

Výpis certifikátu je možné zobraziť pomocou príkazu: *openssl ec -in example.pem -text -noout [Príklad uvedený v prílohe 6.2]*

2.3. Vytvorenie žiadosti o podpis certifikátu certifikačnou autoritou [Krok c.3 Obr.3]

Príklad: *openssl req -config options.conf -new -key server.key -out server_ziadost.csr*

Výpis žiadosti o podpis certifikátu je možné zobraziť pomocou príkazu: *openssl req -in example.csr -text -noout [Príklad uvedený v prílohe 6.3]*

2.4. Vygenerovanie podpísaného certifikátu certifikačnou autoritou [Krok c.4 Obr.3]

Príklad: *openssl x509 -req -in server_ziadost.csr -CA myCA.pem -CAkey myCA.key -CAcreateserial -out server.pem -days 365 -sha256 -extfile server.ext*

x509: Používa na rôzne operácie s X.509 certifikátmi, konvertovanie certifikátov do rôznych foriem, podpisanie žiadosti o certifikát, spracovanie a všeobecnú manipuláciu

-in [názov_súboru]: určuje vstup, z ktorého sa má čítať žiadosť o certifikát

-CA [názov_súboru]: určuje certifikát "CA", ktorý sa má použiť na podpisovanie

-CAkey: Nastavuje súkromný kľúč CA, ktorým sa má podpísať certifikát

-extfile [názov_súboru.ext]: umožňuje špecifikovať súbor s rozšíreniami certifikátu, ktoré majú byť pridané alebo upravené v certifikáte

-CAcreateserial: vygenerovanie nového jedinečného sériového čísla pre CA certifikát

3. Projekty [komunikácia SSL/TLS]

Primárna zložka obsahuje dva typy projektov:

1. **CLIENT_SERVER_SECURE** – Realizuje šifrovanú klient-server komunikáciu s využitím rôznych implementácii aplikovania soketov pre operačné systémy *Linux* a *Windows*.
2. **CLIENT_SERVER_SECURE_BIO** - Realizuje šifrovanú klient-server komunikáciu s využitím BIO abstrakcie aplikovania soketov pre operačné systémy Linux a Windows.

Projekt *CLIENT_SERVER_SECURE* bol prevzatý z webovej adresy uvedenej v zdroji c.19, ktorý bol následne značne modifikovaný. Projekt *CLIENT_SERVER_SECURE_BIO* bol vytvorený na základe projektu *CLIENT_SERVER_SECURE* s úpravami, ktoré využívajú BIO abstrakciu.

3.1. Štruktúra Projektu

Kompletná hierarchická štruktúra projektov je uvedená v súbore ***program_structure.svg***

Pre funkčnosť daného projektu je nutné zachovať danú štruktúru projektu.

3.2. Vygenerovanie certifikátov

- Projekt umožňuje vygenerovať dva typy certifikátov: **RSA** a **ECC** certifikáty
- V adresári *CERIFICATEs* sa nachádzajú priečinky RSA a ECC, v ktorých sú uložené skripty na generovanie certifikátov podľa príslušného typu: **gen_cert_ECC.bat** alebo **gen_cert_RSA.bat** pre vygenerovanie certifikátov.

3.3. Kompilácia projektov

1. Prvým krokom pre úspešnú kompiláciu projektu je nutné vygenerovať súkromné kľúče a certifikáty podľa predošlého kroku 3.2
2. Presunutie vygenerovaných súborov:

   client.key a **client.pem** do adresára *CLIENT_SERVER_SECURE/CLIENT* alebo CLIENT_SERVER_SECURE_BIO/CLIENT podľa aktuálneho projektu
   server.key a **server.pem** do adresára *CLIENT_SERVER_SECURE/SERVER* alebo *CLIENT_SERVER_SECURE_BIO/ SERVER* podľa aktuálneho projektu
   myCA.pem do adresára *CLIENT_SERVER_SECURE* alebo *CLIENT_SERVER_SECURE_BIO* podľa aktuálneho projektu

3.3.1. Kompilácia zdrojových súborov

- V adresári *CLIENT_SERVER_SECURE* alebo *CLIENT_SERVER_SECURE_BIO* sa nachádza súbor **makefile**, ktorého spustením sa preložia zdrojové súbory ***client.c*** a ***server.c*** na základe ktorých sa následne vygenerujú spustiteľne súbory ***client_run*** a ***server_run***.

- V adresári *CLIENT_SERVER_SECURE* alebo *CLIENT_SERVER_SECURE_BIO* sa nachádzajú aj súbory ***comp_client.bat*** a ***comp_server.bat***, ktoré umožňujú jednotlivú kompiláciu zdrojových súborov ***client.c*** alebo ***server.c***

3.3.2. Spustenie požadovaného projektu

- Prepnutie sa da požadovaného adresára podľa typu projektu *CLIENT_SERVER_SECURE* alebo *CLIENT_SERVER_SECURE_BIO*
- Ako prvý musí byť spustený ***server*** a následne ***klient***.

3.3.2.1. Spustenie servera

a) Spustenie **bat** súboru **start_server.bat** [Windows]
b) Sever bude spustený na localhoste (127.0.0.1)
c) Server bude využívať portové číslo 5000

```
[+] Use TLS server method.
[*] Server's certificat and private key loaded from file.
[+] Server's private key match public certificat !
[+] Server listening on the 5000 port...
```

Obr.4 *Výpis po spustení servera v CMD s využitím skriptu start_server.bat*

3.3.2.2. Spustenie klienta

a) Spustenie **bat** súboru **start_client.bat** [Windows]
b) Klient bude realizovať komunikáciu na adrese localhostu (127.0.0.1)
c) Klient bude využívať portové číslo 5000

```
[+] Use TLS method.

CA certificate loaded
Certificate attached.
[+] Cipher used : TLS_AES_256_GCM_SHA384
[+] Server certificates :
        Subject: /C=US/ST=Fake State/L=Fake Locality/O=Fake Company/CN=local.dev
        Issuer: /C=US/ST=Fake State/L=Fake Locality/O=Fake Company/CN=local.dev
[+] Server certificates X509 is trust!
[+] Server data received : Enchante ClientName, je suis ServerName.
```

Obr.5 *Výpis po spustení klienta v CMD s využitím skriptu start_klient.bat*

```
[+] Use TLS server method.
[*] Server's certificat and private key loaded from file.
[+] Server's private key match public certificat !
[+] Server listening on the 5000 port...
[+] Connection [127.0.0.1:55732]
[+] Cipher used : TLS_AES_256_GCM_SHA384
[+] Client certificates :
        Subject: /C=US/ST=Fake State/L=Fake Locality/O=Fake Company/CN=local.dev
        Issuer: /C=US/ST=Fake State/L=Fake Locality/O=Fake Company/CN=local.dev
[+] Client data received : ClientName
```

Obr.6 *Výpis zo servera po prijatí správy od klienta v CMD*

4. Funkcie využívané v zdrojových kódoch realizujúce šifrovanú komunikáciu klient-server

Kompletná funkcionálna hierarchická štruktúra projektu, ktorá zobrazuje aj poradie volaní jednotlivých funkcií sa nachádza v súbore ***klient_server_SSL_schematic.svg***

4.1. Všeobecné funkcie

TLS_method() - funkcia, ktorá vráti ukazovateľ na štruktúru "SSL_METHOD", ktorá predstavuje protokol *TLS (Transport Layer Security).* Štruktúra "SSL_METHOD" definuje súbor metód, ktoré možno použiť na vytvorenie novej relácie *SSL/TLS*. Tieto metódy zahŕňajú funkcie na spracovanie šifrovania, dešifrovania, výmeny kľúčov a overovania certifikátov. Pre server sa využíva funkcia *TLS_server_method()* a pre klienta sa využíva funkcia *TLS_client_method().*

SSL_CTX_new() - funkcia, ktorá vytvára nový objekt *SSL_CTX (Secure Sockets Layer Context)*, ktorý sa používa na uchovávanie nastavení *SSL/TLS* pre server alebo klienta. Objekt SSL_CTX obsahuje súbor metód *SSL/TLS*, ako aj informácie o certifikátoch, privátnych kľúčoch a šifrovacích algoritmoch, ktoré sa použijú počas SSL/TLS handshaku.

SSL_new() - používa sa na vytvorenie nového objektu SSL. Objekt SSL reprezentuje *SSL/TLS* pripojenie a obsahuje konfiguračné informácie súvisiace so zabezpečenou komunikáciou.

SSL_CTX_load_verify_locations() - funkcia, ktorá sa používa na špecifikovanie súboru alebo adresára obsahujúceho dôveryhodné certifikáty *CA (Certificate Authority),* ktoré bude *SSL/TLS* klient používať na overenie certifikátu servera počas *SSL/TLS* handshaku. Počas *SSL/TLS* handshaku server predstaví svoj digitálny certifikát klientovi na overenie svojej identity. Klient potom overí identitu servera kontrolou certifikátu servera voči zoznamu dôveryhodných certifikátov CA. Ak sa certifikát servera dá overiť voči jednému z dôveryhodných certifikátov CA, *SSL/TLS* spojenie sa vytvorí.

***SSL_CTX_use_certificate_file*()** - slúži na načítanie digitálneho certifikátu servera do objektu SSL_CTX.

***SSL_CTX_use_PrivateKey_file*()** - Pre použitie súkromného kľúča pre SSL/TLS komunikáciu musí byť kľúč načítaný do objektu SSL_CTX. Funkcia načíta súkromný kľúč zo súboru a uloží ho do objektu SSL_CTX pre neskoršie použitie.

SSL_CTX_check_private_key() - slúži na overenie, či súkromný kľúč zodpovedá certifikátu načítaného v SSL kontexte.

SSL_get_peer_certificate() - slúži na získanie digitálneho certifikátu X.509 odoslaného počas *SSL/TLS* handshaku od protistrany.

socket() - používa na vytvorenie nového socketu. Socket je koncový bod komunikácie, ktorý umožňuje procesom komunikovať medzi sebou, buď na rovnakom počítači alebo cez sieť.

Funkcia socket() prijíma tri argumenty:

- Prvý argument určuje komunikačnú doménu. *PF_INET* je najbežnejšia doména a používa sa pre Internet Protocol (IP) verzie 4.
- Druhý argument určuje typ socketu. *SOCK_STREAM* je spoľahlivý, používa sa pre TCP (Transmission Control Protocol) spojenia.
- Tretí argument je protokol. Zvyčajne je to nastavené na 0, čo umožňuje systému vybrať vhodný protokol na základe domény a typu socketu.

Konštanta *AF_INET* sa používa spolu s konštantou *PF_INET* na špecifikovanie protokolu IPv4 pre socket. Táto kombinácia konštánt je bežne používaná, pretože poskytuje podporu pre sieťové protokoly IPv4.

***SSL_set_fd*()** – používa sa na priradenie SSL/TLS objektu existujúceho deskriptoru súboru (*fd*) pre socket.

SSL_write() - slúži na zápis dát do *SSL/TLS* pripojenia. Funkcia realizuje operáciu zápisu do socketu, ktorá zašifruje dáta pomocou protokolu *SSL/TLS* pred ich odoslaním do siete. Dôležité je poznamenať, že funkcia *SSL_write()* nezaručuje, že všetky dáta budú odoslané v jednom volaní. Ak nie sú odoslané všetky dáta, aplikácia musí *SSL_write()* volať opäť s neodoslanými dátami, kým sa neodoslú všetky dáta alebo sa nevyskytne chyba.

SSL_read() - slúži na čítanie dát zo *SSL/TLS* pripojenia. Funkcia realizuje operáciu čítania zo socketu, ktorá dešifruje dáta pomocou protokolu *SSL/TLS* po ich prijatí zo siete. Dôležité je poznamenať, že funkcia *SSL_read()* nezaručuje, že všetky žiadané dáta budú prečítané v jednom volaní. Ak nie sú prečítané všetky dáta, aplikácia musí *SSL_read()* volať opäť s požadovaným miestom v buffery, kým nebudú prečítané všetky dáta alebo sa nevyskytne chyba.

SSL_shutdown() - používa sa na ukončenie *SSL/TLS* spojenia. Je zvyčajne používaná v prípade, že klient alebo server chce spojenie ukončiť.

SSL_CTX_set_verify() - používa sa na nastavenie režimu overovania pre overovanie certifikátov v SSL/TLS kontexte. Režim overovania určuje, ako sa certifikáty *SSL/TLS* overujú počas handshake procesu.

4.2. Funkcie špecifické pre rolu klienta

SSL_get_verify_result() - slúži na získanie výsledku overovania digitálneho certifikátu protistrany počas *SSL/TLS* handshaku. Klient overuje dôvernosť certifikátu a overuje, že certifikát nebol sfalšovaný a bol vydávaný dôveryhodnou autoritou. Funkcia *SSL_get_verify_result()* môže byť volaná na strane klienta po vytvorení *SSL/TLS* pripojenia na získanie výsledku overenia certifikátu protistrany.

gethostbyname() - je funkcia, ktorá slúži na prevod názvu hostiteľa (*hostname*) napr. www.google.sk na jeho príslušnú IP adresu. Funkcia berie ako argument reťazec obsahujúci názov hostiteľa a vracia informácie o hostiteľovi, vrátane jeho IP adresy.

connect() – používa sa na vytvorenie spojenia so vzdialeným socketom alebo serverom cez sieť, čo umožňuje odosielanie a prijímanie dát medzi týmito dvoma bodmi. Obvykle sa používa so socketom, ktorý bol vytvorený pomocou funkcie socket() a po nastavení adresy vzdialeného užívateľa .

SSL_connect() – používa sa na iniciovanie *SSL/TLS* handshaku so vzdialeným serverom. Vytvára zabezpečené, šifrované spojenie medzi klientom a serverom pomocou protokolu *SSL/TLS*. Počas *SSL/TLS* handshaku si klient a server vyjednávajú parametre šifrovaného spojenia, ako sú šifrovacie algoritmy a použité kľúče. Keď je handshake dokončený, klient a server môžu komunikovať bezpečne cez šifrované spojenie.

SSL_set_mode() - používa sa na nastavenie režimu prevádzky pre *SSL/TLS* pripojenie. Táto funkcia umožňuje povoliť alebo zakázať konkrétne vlastnosti protokolu *SSL/TLS*.

4.3. Funkcie špecifické pre rolu servera

bind() - v sieťovom programovaní socketov slúži na priradenie lokálnej adresy (IP adresa a číslo portu) k socketu. Toto je nevyhnutné predtým, ako môžete použiť socket na odosielanie alebo prijímanie dát.

listen() - funkcia v sieťovom programovaní socketov sa používa na nastavenie socketu do pasívneho režimu a čakanie na prichádzajúce spojenia. Po tom, ako je socket priradený k lokálnej adrese pomocou funkcie *bind()*, môže sa použiť funkcia *listen()* na uchovávanie prichádzajúcich požiadaviek na spojenie, kým nie sú prijaté dáta pomocou funkcie *accept()*.

accept() - je sieťová funkcia používaná pri socketoch, ktorá slúži akceptovanie prichádzajúceho pripojenia od vzdialeného klienta. Keď je vytvorený socket a viazaný na port, počúva prichádzajúce pripojenia od vzdialených klientov. Keď sa klient pripojí k tomuto socketu, na strane servera sa zavolá funkcia *accept()* a vytvorí sa nový socket na spracovanie komunikácie s týmto klientom. Tento nový socket sa potom používa na výmenu údajov medzi klientom a serverom. Funkcia *accept()* blokuje vykonávanie programu, kým nepríde nové spojenie od klienta.

SSL_accept() - Keď sa klient pripojí k serveru pomocou *SSL/TLS*, na strane servera sa zavolá funkcia *SSL_accept()*, aby sa inicializoval *SSL/TLS* handshake. Server pošle svoj *SSL/TLS* certifikát klientovi, ktorý ho overí. Klient potom pošle svoj certifikát (ak je potrebný), a server ho tiež overí. Potom si obe strany vymenia kryptografické kľúče a dohodnú sa na sade šifrovacích algoritmov, ktoré sa použijú počas zvyšku relácie.

4.4. Funkcie realizujúce komunikáciu nezávisle od operačného systému

BIO_new_connect() – vytvára nový objekt *BIO (Basic Input/Output)*, ktorý reprezentuje sieťové pripojenie na vzdialený server. Má dva argumenty: reťazec obsahujúci názov hostiteľa alebo IP adresu vzdialeného servera a reťazec obsahujúci číslo portu, na ktorý sa má pripojiť.

BIO_new_accept() – vytvára nový objekt BIO a viaže ho na socket, ktorý môže byť použitý na prijímanie prichádzajúcich spojení od klientov. Parametrom je reťazec reprezentujúci číslo portu na ktorom ma server počúvať komunikáciu.

SSL_set_bio() **–** nastavuje objekty vstupu/výstupu (I/O) BIO pre objekt *SSL (Secure Sockets Layer).*

BIO_free_all() **–** uvoľňuje všetku pamäť, ktorá je spojená s objektom BIO

BIO_do_accept() –používa sa na prijatie prichádzajúcich spojení na BIO objekt, ktorý bol vytvorený pomocou funkcie *BIO_new_accept()*. Pri volaní *BIO_do_accept()* sa blokuje a čaká na príchod nového spojenia.

BIO_pop() **-** používa sa na odstránenie najvrchnejšieho objektu BIO zo zásobníka.

5. Využité nástroje (verzie)

5.1. Obraz BPS [Virtual Box]

Operačný systém: Windows 7 Super Lite by R-ALMODARIS 32-bit

GCC prekladač: gcc (MinGW-W64 x86_64-ucrt-posix-seh, built by Brecht Sanders) 13.1.0

OpenSSL knižnica: OpenSSL 3.0.8 7 Feb 2023

5.2. Operačné systémy:

Windows 10 Home 64-bit

Kali GNU/Linux 2023.1 6.0.0-kali6-amd64

5.3. GCC prekladač:

Windows - gcc (MinGW-W64 x86_64-ucrt-posix-seh, built by Brecht Sanders) 12.2.0, GCC 13.2 posix winlibs rel.1

Linux - gcc (Debian 12.2.0-14) 12.2.0

5.4. OpenSSL knižnica:

Windows, Linux: OpenSSL 3.1.1 30 May 2023, OpenSSL 3.1.2

6. Prílohy

6.1. Zobrazenie súkromného kľúča

Vygenerované pomocou príkazu: *openssl ec -in myCA.key -text -noout*

```
read EC key
Private-Key: (256 bit)
priv:
    e5:d5:ba:55:15:3f:ac:d1:5f:f0:ae:7d:50:07:42:
    f4:21:35:27:5d:3e:a3:ab:0a:6f:de:1d:35:49:2a:
    5e:64
pub:
    04:3b:c9:68:18:2b:0c:84:82:fb:76:15:d7:64:5a:
    eb:1a:99:78:d3:4e:bd:2c:c8:1e:ea:ab:0e:87:fb:
    9b:9d:25:85:6c:81:3b:b6:bf:a3:53:d6:aa:5e:c4:
    33:55:a7:be:b4:23:e1:eb:9f:37:6c:e8:ee:d4:2c:
    77:fb:a7:39:b6
ASN1 OID: prime256v1
NIST CURVE: P-256
```

Obr.7 *Premenné uvedené v štruktúre súkromného kľúča [ECC]*

6.2. Zobrazenie PEM certifikátu

Vygenerované pomocou príkazu: *openssl x509 -in myCA.pem -text*

```
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number:
            01:79:f6:49:9c:65:d8:95:f5:82:97:b3:e4:c9:d8:f0:2c:13:09:05
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C = US, ST = Fake State, L = Fake Locality, O = Fake Company, CN = local.dev
        Validity
            Not Before: Jul  5 07:10:41 2023 GMT
            Not After : Jul  4 07:10:41 2024 GMT
        Subject: C = US, ST = Fake State, L = Fake Locality, O = Fake Company, CN = local.dev
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:3b:c9:68:18:2b:0c:84:82:fb:76:15:d7:64:5a:
                    eb:1a:99:78:d3:4e:bd:2c:c8:1e:ea:ab:0e:87:fb:
                    9b:9d:25:85:6c:81:3b:b6:bf:a3:53:d6:aa:5e:c4:
                    33:55:a7:be:b4:23:e1:eb:9f:37:6c:e8:ee:d4:2c:
                    77:fb:a7:39:b6
                ASN1 OID: prime256v1
                NIST CURVE: P-256
    Signature Algorithm: ecdsa-with-SHA256
    Signature Value:
        30:46:02:21:00:b5:43:f1:86:76:e7:f5:c9:25:10:62:e2:25:
        b3:d4:2e:2a:a8:58:11:90:11:3d:9a:d5:06:30:70:cf:7e:e3:
        e1:02:21:00:ab:65:cf:2b:dc:ef:20:47:ce:19:88:5a:d3:9f:
        17:b1:65:95:f7:7a:37:ea:a5:e5:87:df:fd:e3:8a:a0:5d:85
```

Obr.8 *Premenné uvedené v štruktúre PEM certifikátu*

6.3. Zobrazenie žiadosti o podpis certifikátu

Vygenerované pomocou príkazu: *openssl req -in klient_ziadost.csr -text -noout*

```
Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: C = US, ST = Fake State, L = Fake Locality, O = Fake Company, CN = local.dev
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (256 bit)
                pub:
                    04:14:ca:59:57:6e:6e:2f:f9:63:d9:e4:e3:86:0a:
                    37:fb:91:3f:5d:55:7c:cb:26:bb:1e:69:1c:b7:be:
                    3e:72:19:96:88:91:e3:e6:c0:29:3b:f9:f9:0c:5f:
                    51:7a:2d:7e:48:b7:ea:0d:2c:36:19:9f:0d:4d:32:
                    c7:a4:ec:ac:84
                ASN1 OID: prime256v1
                NIST CURVE: P-256
        Attributes:
            (none)
            Requested Extensions:
    Signature Algorithm: ecdsa-with-SHA256
    Signature Value:
        30:46:02:21:00:ff:66:5c:5b:43:d3:b6:cd:45:0e:ac:78:0f:
        91:c9:7d:31:84:14:a8:90:e7:ef:ae:2f:89:58:67:ae:7c:2b:
        05:02:21:00:ae:65:ca:ac:0e:04:0b:4a:e2:fb:b8:dd:bd:e0:
        e0:a0:99:4c:81:9d:db:07:55:9e:ca:02:58:e4:a6:58:c6:a4
```

Obr.9 *Premenné uvedené v štruktúre žiadosti o podpis certifikátu*

6.4. Opis extensions a konfiguračného súboru

6.4.1. Konfiguračný súbor

Konfigurácia OpenSSL vyhľadá hodnotu *openssl_conf* v predvolenej sekcii a vezme ju ako názov sekcie, ktorá špecifikuje, ako nakonfigurovať ľubovoľné moduly v knižnici. V OpenSSL sa **.conf** súbor odkazuje na konfiguračný súbor, ktorý sa používa na definovanie rôznych nastavení a parametrov pre aplikácie v OpenSSL. Tento súbor sa zvyčajne používa na prispôsobenie knižníc v OpenSSL.

.conf súbor má špecifický formát a obsahuje sekcie, *options* a *values*. Prvky, ktoré sa zvyčajne nachádzajú v konfiguračnom súbore OpenSSL:

Sekcie - Konfiguračný súbor je rozdelený na rôzne sekcie, pričom každá je označená názvom v hranatých zátvorkách [ ]. Sekcie pomáhajú organizovať súvisiace nastavenia a možnosti. Príklady sekcií: providers, ssl_configuration, tls_system_default.

Options, values - Každá sekcia obsahuje rôzne možnosti a príslušné hodnoty. Možnosť je špecifikovaná kľúčovým slovom a priradenou hodnotou. Príklad**:** RSA.Certificate = server-rsa.pem

```
[req]
prompt = no
distinguished_name = req_distinguished_name

[req_distinguished_name]
C = US
ST = Fake State
L = Fake Locality
O = Fake Company
# OU = Org Unit Name
# emailAddress = info@example.com
CN = local.dev
```

Obr.10 *Ukážka konfiguračného súboru -options.conf*

6.4.2. Rozšírený konfiguračný súbor

V OpenSSL sa súbor **.ext** používa ako konfiguračný súbor pre rozšírené nastavenia. Zvyčajne sa používa s nástrojom príkazového riadka OpenSSL na špecifikovanie ďalších možností a nastavení na generovanie alebo manipuláciu s kryptografickými certifikátmi.

Súbor **.ext** obsahuje súbor možností, ktoré definujú požadované rozšírenia pre certifikát alebo žiadosť o podpis certifikátu (*CSR*). Tieto rozšírenia poskytujú ďalšie informácie alebo funkcie nad rámec základných, čo umožňuje väčšie prispôsobenie a flexibilitu.

Opis základných možností:

1. **authorityKeyIdentifier**: určuje, ako identifikovať verejný kľúč vydavateľa certifikátu. V tomto prípade obsahuje identifikátor kľúča a rozlišovacie meno (*DN*) vydavateľa.
2. **basicConstraints**: určuje, či je certifikát certifikačnou autoritou (*CA*) a či môže podpisovať iné certifikáty
3. **keyUsage**: definuje, ako sa môže používať kľúč, ktorý je spojený s certifikátom

```
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
subjectAltName = @alt_names

[alt_names]
DNS.1 = dev.mergebot.com
DNS.2 = dev.mergebot.com.192.168.1.19.xip.io
```

Obr.11 *Ukážka configuračného súboru server.ext*

6.5. Sériové číslo

Princíp generovania sériového čísla:

1. Vygeneruje sa certifikát s vlastným podpisom a vlastným sériovým číslom [*certifikačná autorita*]
2. Certifikačná autorita zvolí sériové číslo, ktoré bude poskytovať žiadateľom o podpis certifikátu a uloží hod do súboru s príponou ***.srl***
3. Certifikačná autorita poskytne podpísaný certifikát žiadateľovi s aktuálnym sériovým číslom uloženým v súbore s príponou ***.srl***
4. Certifikačná autorita inkrementuje sériové číslo v súbore s príponou ***.srl***
5. S každým novým žiadateľom sa proces opakuje od kroku c.3

```
C:\Users\Martin\Desktop\PROJECT_SSL_TLS\CERIFICATEs\RSA>openssl x509 -in myCA.pem -text
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number:
            16:55:16:f4:c7:ce:b2:a0:99:9d:4d:de:13:09:13:29:a2:59:79:23
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, ST = Fake State, L = Fake Locality, O = Fake Company, CN = local.dev
        Validity
            Not Before: Aug  9 15:37:30 2023 GMT
            Not After : Aug  7 15:37:30 2028 GMT
        Subject: C = US, ST = Fake State, L = Fake Locality, O = Fake Company, CN = local.dev
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
```

Obr.12 *Zobrazenie certifikátu certifikačnej autority*

```
C:\Users\Martin\Desktop\PROJECT_SSL_TLS\CERIFICATEs\RSA>openssl x509 -in server.pem -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            15:20:ad:0e:5e:cd:26:08:ee:52:a9:4b:38:cd:5a:99:5b:4b:73:8b
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, ST = Fake State, L = Fake Locality, O = Fake Company, CN = local.dev
        Validity
            Not Before: Aug  9 15:37:30 2023 GMT
            Not After : Aug  7 15:37:30 2028 GMT
        Subject: C = US, ST = Fake State, L = Fake Locality, O = Fake Company, CN = local.dev
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
```

Obr.13 *Zobrazenie certifikátu servera*

```
C:\Users\Martin\Desktop\PROJECT_SSL_TLS\CERIFICATEs\RSA>openssl x509 -in client.pem -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            15:20:ad:0e:5e:cd:26:08:ee:52:a9:4b:38:cd:5a:99:5b:4b:73:8c
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, ST = Fake State, L = Fake Locality, O = Fake Company, CN = local.dev
        Validity
            Not Before: Aug  9 15:37:30 2023 GMT
            Not After : Aug  7 15:37:30 2028 GMT
        Subject: C = US, ST = Fake State, L = Fake Locality, O = Fake Company, CN = local.dev
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
```

Obr.14 *Zobrazenie certifikátu klienta*

6.6. Doplňujúce zdroje k SSL/TLS komunikácii a certifikátom

Informácie k tvorbe konfiguračného súboru:
[1] https://www.openssl.org/docs/man3.0/man5/config.html
[2] https://www.openssl.org/docs/man3.0/man5/x509v3_config.html

Informácie k SSL/TLS:
[3] https://www.ibm.com/docs/en/ibm-mq/7.5?topic=ssl-overview-tls-handshake
[4] https://www.sans.org/reading-room/whitepapers/protocols/ssl-tls-beginners-guide-1029
[5] https://owasp.org/www-chapter-london/assets/slides/OWASPLondon20180125_TLSv1.3_Andy_Brodie.pdf
[6] https://tls13.xargs.org/
[7] https://blog.chromium.org/2021/03/a-safer-default-for-navigation-https.html
[8] https://www.digiater.nl/openvms/doc/alpha-v8.3/83final/BA554_90007/ch04s03.html

[9] https://aticleworld.com/ssl-server-client-using-openssl-in-c/

Nastavenie Networking v prostredí Virtual-Boxu:
[10] https://www.dedoimedo.com/computers/virtualbox-network-sharing.html
[11] https://www.nakivo.com/blog/virtualbox-network-setting-guide/

Informácie k BIO štruktúre:
[12] https://github.com/openssl/openssl/tree/master/demos/bio
[13] https://www.digiater.nl/openvms/doc/alpha-v8.3/83final/BA554_90007/ch04s03.html?fbclid=IwAR2goybeRALkiD1Y_gB769GnNVCC0KgGimfkOkOyzwpcWTxOdtYiV3pCzSY
[14]http://odl.sysworks.biz/disk$axpdocsep021/opsys/vmsos731/vmsos731/6661/6661pro.htm?fbclid=IwAR1i86q9M8aFbVYg-_Yoeir8DHOxnsTffGmPl8J5NMDMRTd-NDOHW-x1uzE#a369115153
[15] https://developer.ibm.com/tutorials/l-openssl/

Informácie k certifikátom:
[16] https://superuser.com/questions/126121/how-to-create-my-own-certificate-chain
[17] https://www.openssl.org/docs/man1.1.1/man1/x509.html
[18] http://www.steves-internet-guide.com/ssl-certificates-explained/
[19] https://www.asafety.fr/projects-and-tools/c-client-serveur-ssl-tls-multiplateformes-avec-openssl/