Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Automatizovaná kontrola stavu certifikátů X.509

David Král

Bakalářská práce
2021

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2018/2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**David Král**
Osobní číslo:	**I16107**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Automatizovaná kontrola stavu certifikátů X.509**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem bakalářské práce je vytvořit softwarový nástroj pro automatizovanou kontrolu stavu X.509 certifikátů. Základním kritériem je datum platnosti certifikátu, volitelně lze kontrolovat revokační seznamy a další mechanismy platnosti. V teoretické části bude popsán základní principy kryptografie založené na veřejných klíčích (PKI) a související certifikáty X.509. Dále bude popsán vlastní návrh aplikace pro kontrolu platnosti certifikátů. Vytvářený nástroj musí být schopen načíst X.509 certifikát a ověřit jeho platnost (dle časových razítek) a informovat o blížící se konci platnosti. Nástroj musí být schopen prohledávat vybrané části souborového systému na lokálním počítači a obdobně prohledávat vzdálené systémy s využitím protokolu SSH/SCP. Nástroj musí podporovat autentizaci ke vzdálenému systému pomocí jména/hesla a s využitím veřejného klíče. Konfigurace nástroje bude uložena v jednoduchém textovém souboru (INI, CSV, YAML, ...). Nástroj by měl být platformě nezávislý a podporovat Windows a Linux.

Rozsah pracovní zprávy: **40 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

KNUDSEN, Jonathan. *Java cryptography*. Sebastopol, Calif.: O'Reilly, c1998. Java series (O'Reilly & Associates). ISBN 978-1565924024
RHEE, Man Young. *Internet security: cryptographic principles, algorithms, and protocols*. Hoboken, NJ: J. Wiley, 2003. ISBN 9780470852859
JSch – Java Secure Channel JCraft online 1998-2016 [cit. 2018-10-23]. Dostupné z: http://www.jcraft.com/jsch/

Vedoucí bakalářské práce: **Ing. Roman Diviš**
Katedra softwarových technologií

Datum zadání bakalářské práce: **31. října 2018**
Termín odevzdání bakalářské práce: **12. května 2019**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Lukáš Čegan, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 14. prosince 2018

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 14. 5. 2021

David Král

PODĚKOVÁNÍ

Rád bych poděkoval Ing. Romanu Divišovi především za ochotu, vstřícnost, trpělivost, cenné rady a konzultace během zpracování této práce. Dále bych rád poděkoval celému akademickému sboru fakulty elektrotechniky a informatiky, který mi během studia předával znalosti, které jsem mohl při psaní práce využít.

ANOTACE

Bakalářská práce se zabývá implementací softwarového nástroje pro automatizovanou kontrolu stavu certifikátů standardu X.509. Dále se práce zaměřuje na vysvětlení základních principů asymetrické kryptografie. Jsou představeny mechanismy, které jsou tímto typem kryptografie využívány, jejich bezpečnost a slabiny. Teoretická část také popisuje strukturu a obvyklou správu samotného certifikátu. V praktické části je vyvíjen zmíněný nástroj pro kontrolu platnosti certifikátů X.509. Čtenáři jsou seznámeni jak se softwarovými prostředky, využitými pro implementaci, tak i s jednotlivými částmi zdrojového kódu.

KLÍČOVÁ SLOVA

kryptografie, certifikát, veřejný klíč

ANNOTATION

The bachelor's thesis focuses on the implementation of a software tool for automated checking of the status of X.509 certificates. Furthermore, the work explains the basic principles of asymmetric cryptography. The mechanisms used by this type of cryptography, their security and weaknesses are introduced. The theoretical also describes the structure and usual administration of the certificate itself. In the practical part is developed the mentioned tool for checking the validity of X.509 certificates. Readers are acquainted with the software resources used for implementation and with parts of the source code as well.

KEYWORDS

cryptography, certificate, public key

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK

ACME	Automated Certificate Management Environment
CA	Certifikační Autorita
CRL	Certificate Revocation List
CTL	Certificate Trusted List
DSA	Digital Signature Algorithm
DPV	Delegated Path Validation
DPD	Delegated Path Discovery
DV	Domain Validated certificate
ECC	Elliptic Curve Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
EFF	Electronic Frontier Foundation
GCHQ	Government Communications Headquarters
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IPv4	Internet Protocol version 4
ITU	International Telecommunication Union
JSch	Java Secure Channel
NIST	National Institute of Standards and Technology
OCSP	Online Certificate Status Protocol
PKI	Public Key Infrastructure
RSA	Rivest–Shamir–Adleman
SFTP	SSH File Transfer Protocol
SSH	Secure Shell
TLS	Transport Layer Security
UML	Unified Modeling Language
URI	Uniform Resource Identifier
YAML	YAML Ain't Markup Language

ÚVOD

Téma bakalářské práce Automatizovaná kontrola stavu certifikátů X.509 se zabývá kryptografií a bezpečností komunikace, protože kryptografie patří mezi nejaktuálnější témata v oboru informačních technologií.

Kryptografie je vědní obor, který se zabývá utajováním obsahu zpráv za pomocí šifer. Šifra je metoda převodu zprávy na podobu, která je čitelná pouze se znalostí šifrovacího klíče. Počátky kryptografie sahají až do pátého století před naším letopočtem, do období Řecko-Perských válek. Už tehdy bylo potřeba chránit vojenskou komunikaci před nepřáteli. S plynoucí dobou se kryptografie informací zdokonalovala právě díky četným válkám a potřebě šifrování zpráv jako ochrana při získání zprávy nepřítelem. Nejednou tak kryptografie ovlivnila běh lidských dějin. Zvláště pak ve 20. století, během obou světových válek se kryptografie dočkala velkého rozmachu.

V dnešní době již dávno není šifrování jen vojenskou záležitostí. S rostoucí mírou používání elektronické komunikace pro stále více důležitější účely samozřejmě roste potřeba tyto informace chránit před lidmi, kteří by jejich získání dokázali využít ve svůj prospěch na úkor našeho neprospěchu. Asymetrická kryptografie neboli kryptografie využívající veřejné klíče, na kterou je zaměřena tato bakalářská práce se dnes využívá nejen pro šifrování zpráv, ale také například pro ověření elektronického podpisu a prokázaní tak autora nebo vlastníka dat.

Jak už bylo zmíněno výše, v první části práce budou čtenáři seznámeni se základními informacemi o kryptografii, budou popsány a vysvětleny základní principy a mechanismy asymetrického šifrování a bude rozebrána struktura certifikátu X.509 využívaného právě v tomto typu šifrování. Ve druhé části práce bude představena vlastní implementaci aplikace pro kontrolu platnosti certifikátů X.509. V práci budou uvedeny veškeré softwarové prostředky, které byly použity při tvorbě aplikace a detailně popsány jednotlivé části zdrojového kódu.

1 ASYMETRICKÁ KRYPTOGRAFIE

1.1 Asymetrická šifra

1.1.1 Základní představení

Asymetrická šifra funguje na principu dvou klíčů. Zpráva, která je zašifrována klíčem prvním lze dešifrovat pouze klíčem druhým a naopak. Síla asymetrické šifry tkví v tom, že ze znalosti prvního klíče nelze odvodit klíč druhý. Použití dvou klíčů je zásadní rozdíl od symetrické šifry, která pro zašifrování a dešifrování zpráv sdílí jeden tentýž klíč. Vzhledem k tomu, že u některých šifer jsou operace šifrování a dešifrování zaměnitelné, nehovoříme o dvojici klíčů jako o šifrovacím klíči a dešifrovacím klíči, ale jako o klíči veřejném a soukromém [1, s. 25].

1.1.2 Historie a vývoj

Pro vznik asymetrické šifry, tak jak ji známe dnes, bylo zapotřebí v minulosti nejméně šesti lidí, které je při popisu zrodu asymetrické šifry vhodné jmenovat. Ještě před asymetrickou šifrou bylo nutné vždy pro šifrovanou komunikaci mezi dvěma stranami podstoupit složitou distribuci sdíleného klíče. Prvními, kdo vyřešili problém distribuce klíčů metodou, která dokázala předat sdílený tajný klíč přes nezabezpečený kanál, byli matematici Martin Hellman, Whitfield Diffie a Ralph Merkle [2, s. 183].

Al Cimino [2] ve své knize obecně popisuje výměnu klíčů mezi třemi fiktivními postavami. Těmi postavami jsou Alice s Bobem, kteří si chtějí zasílat zprávy a Eva, která je ve své roli odposlouchává. První myšlenku dvojitého zámku pro výměnu sdílených klíčů popisuje jako odeslání Alicí schránky s klíči zamčené visacím zámkem. Bob po obdržení schránky ale nemá klíč k odemknutí, a proto pouze zamkne schránku svým visacím zámkem a odešle schránku zpět Alici. Alice odemkne svůj visací zámek a schránku zamčenou už pouze Bobovým zámkem pošle zpět Bobovi. Bobovi nic nebrání, aby odemknul svůj zámek a získal tak obsah schránky. V kryptografickém světě ale tato metoda nefunguje, protože k získání původního textu je zapotřebí zašifrovaný text nejprve dešifrovat Bobovým klíčem a až poté Aliciným [2, s. 183].

V roce 1976 byl výše zmíněnou trojicí mužů představen koncept exponenciální výměny klíčů, založený na jednosměrné funkci, veřejném a soukromém klíči. Od tohoto roku bylo ostatními matematiky představeno spoustu dalších algoritmů založených na principu veřejného klíče, ale ne všechny byly spolehlivé. Spousta z nich je sice spolehlivých a bezpečných do dnes, ale nejsou příliš praktické. Jen pár algoritmů je bezpečných a praktických zároveň [3, s. 161].

Mezi nejznámější algoritmy patří RSA, ElGamal, Schnorr a ECC, přičemž za zmínku rozhodně stojí uvést zbylá tři jména, jejichž první písmena v příjmení tvoří právě název RSA algoritmu.

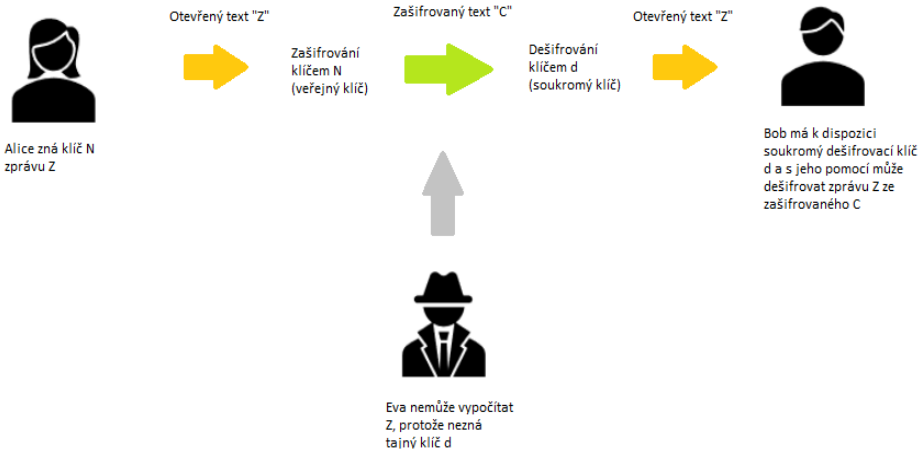
Tuto šifru s jednosměrnou matematickou funkcí jako první představili počítačoví vědci Ron Rivest, Adi Shamir a matematik Leonard Adleman v roce 1978 [2, s. 187].

Ačkoli zásluhy za vytvoření šifrování s PKI připadli Diffiemu, Hellmanovi a Merkleovi a později výše zmíněné trojici Rivestovi, Shamirovi a Adlemanovi, na konci 90. let odtajnila GCHQ materiály, ze kterých vyplívá, že s myšlenkou kryptografie založené na veřejném klíči přišel jejich zaměstnanec James Ellis již 60.letech. Protože, ale Ellis nebyl matematik, s uvedením jeho myšlenky do praxe zápolili jiní vědci z GCHQ bezúspěšně další tři roky. Poté do GCHQ nastoupil mladý absolvent Cambridgeské univerzity Clifford Cocks, který problém vyřešil za méně než půl hodiny a objevil tak algoritmus známý později jako RSA [2, s. 189].

1.1.3 Principy a mechanismy

Dnešní asymetrické šifrování funguje na principu veřejného a soukromého klíče. Pokud je záměrem někomu předat šifrovanou zprávu, je nutné ji zašifrovat jeho veřejným klíčem, který je, jak už jeho název napovídá veřejný, a tudíž ho může vlastnit kdokoli. To ovšem ničemu nevadí protože, veřejným klíčem se data pouze šifrují a pro dešifrování je potřeba znát soukromý klíč. Soukromý klíč však vlastní pouze osoba, která poskytla ostatním svůj veřejný klíč a soukromý klíč s nikým nesdílí. Na obrázku 1 představuje Alice odesílatele zprávy zašifrované Bobovým veřejným klíčem a Bob příjemce, který zprávu dešifruje pomocí svého soukromého klíče. Eva zde nese roli hackera, který odposlouchává komunikaci.

Al Cimino [2] ve své knize asymetrické šifrování vysvětluje obecně na schránkách a visacích zámcích. Pokud budeme znovu uvažovat komunikaci mezi Alicí a Bobem, tak Bob veřejně poskytuje zámky, které může zamknout kdokoli. Alici tedy stačí pouze zámek zaklapnout. Odemknout zámek však může pouze Bob, protože jen on má klíč od zámků. Analogií visacího zámku, který stačí pouze zaklapnou je ve světě počítačů jednosměrná funkce [2, s. 186]. „*V matematice jsou jednosměrné funkce takové, pro které platí, že vypočítat funkční hodnotu ze vstupní hodnoty je proveditelné snadno, ale vypočítat ze znalosti funkční hodnoty příslušnou vstupní hodnotu je velmi obtížné až prakticky nemožné.“* [2, s. 184]



Obrázek 1:Asymetrická šifra – zpracováno dle [2] s. 185

RSA

Princip RSA šifry využívá jednosměrnou funkci a je založen na Eulerově větě. Krom toho je RSA vhodným nástrojem jak pro šifrování dat, tak pro podepisování dokumentů [4]. Web Algoritmy.net [4] rozděluje komunikaci mezi uživateli A a B pomocí RSA do tří fází. V první fázi probíhá výpočet klíčů následovně:

„A si zvolí dvě náhodná velká prvočísla p_a a q_a a jejich součin $p_a \times q_a$ označí jako n_a. Následně si zvolí číslo e_a takové, že je nesoudělné s $\varphi(n_a) = \varphi(p_a \times q_a) = (p_a - 1) \times (q_a - 1)$, kde $\varphi(n)$ je hodnota Eulerovy funkce pro n (zároveň platí $e_a < \varphi(n_a)$). Nakonec vypočítá číslo d_a, které je multiplikativní inverzí čísla e_a. Dvojici (n_a, d_a) si ponechá – jedná se o soukromý klíč. B postupuje analogicky a vytvoří si také veřejný a soukromý klíč.“[1] [4]

Druhou fázi přirozeně představuje šifrování zprávy, kde uživatel A chce zaslat uživateli B zprávu z a šifruje ji pomocí veřejného klíče B_{vk} jako $x = z^{e_b} \, mod(n_b)$. Ve třetí fázi B přijímá zprávu a dešifruje ji pomocí vzorce $z = x^{d_b} \, mod(n_b)$ [4]. Délka šifrovacích klíčů se u RSA pohybuje od 1024 do 4096 bitů, přičemž 2048 bitů je délka šifrovacího klíče, která je aktuálně považována za ještě bezpečnou [1, s. 26].

[1] Podrobný rozbor algoritmu je mimo rámec této práce. Je možné jej dohledat například v publikaci „Internet security: cryptographic principles, algorithms and protocols“ [3], strana 165-172.

ElGamal

Algoritmus ElGgamal byl navržen egyptským inženýrem Taherem Elgamalem v roce 1985. Stejně jako RSA je ElGamal vhodný jak pro šifrování zpráv, tak digitální podpisy. Na rozdíl od RSA, které je založeno faktorizaci, ElGamal využívá obtížnosti výpočtu diskrétního logaritmu [3, s. 172]. Zjednodušeně se dá říct, že i když známe g^a a g^k, je extrémně složité vypočítat g^{ak} [5].

Rhee [3] ve své knize popisuje algoritmus ElGamal tak, že nejprve je nutné si zvolit prvočíslo p a dvě náhodná čísla g a x, tak aby platilo, že $g < p$ a zároveň $x < p$, kde x představuje soukromý klíč. Náhodné číslo g je primitivní základ modulu p. Veřejný klíč je tedy definován čísly y, g a p, kde $y = g^x \ (mod\ p)$. Dalším krokem pro zašifrování zprávy m, kde platí, že $0 < m \leq p - 1$, je nejprve nutné vybrat náhodné číslo k, takové, že největší společný dělitel čísel k a $p - 1$ se rovná 1. Zašifrovanou zprávu zle vyjádřit jako dvojici (r, s) následovně:

$$r \equiv g^k \ (mod\ p)$$

$$s \equiv \left(y^k m \ (mod\ p)\right) (m \ (mod\ p - 1))^2$$

Pro rozšifrování zprávy m, je potřeba vydělit s a rx, tak aby:

$$s \div r^x \ \equiv m \ (mod\ p - 1)$$

Velká nevýhoda kryptografického systému ElGamal je, že šifrovaný text je po zašifrování dvojnásobně větší než původní text. Tato slabina způsobuje, že Elgamal není tolik využívaný jako například RSA. I tak je ale tento systém zastoupen v mnoha standardech a programech jako alternativa k ostatním algoritmům podporujícím asymetrickou kryptografii [6, s. 31].

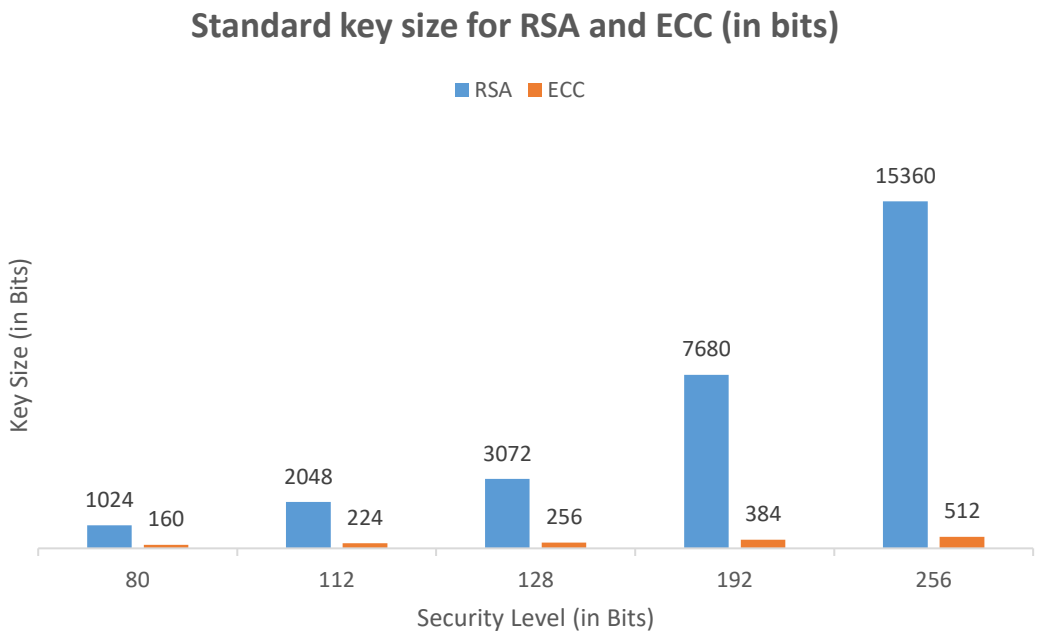
ECC

ECC je alternativou k výše zmíněným algoritmům RSA a ElGamal a v kryptografii se pod touto zkratkou označují schémata založená na eliptických křivkách. Systém představili v roce 1985 Neal Koblity a Victor Miler. Podle Klímy [7] ECC umožňuje šifrování a digitální podpis s kratšími klíči a v mnoha ohledech dosahuje lepšího poměru cena/výkon. Možnosti použití kratších klíčů je docíleno tím, že problém výpočtu diskrétního logaritmu eliptické křivky je podstatně složitější než problém výpočtu samotného diskrétního logaritmu, který například využívá ElGamal. Některé algoritmy založené na PKI, jako například

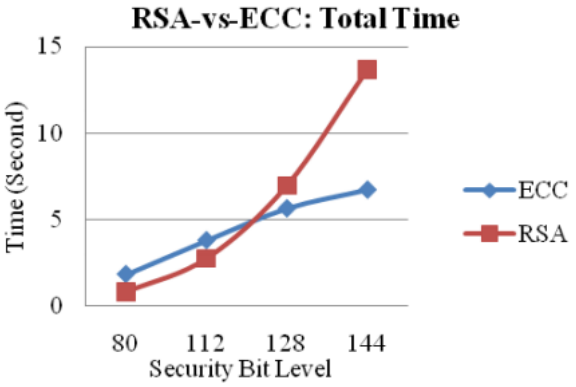
[2] Podrobný rozbor algoritmu je mimo rámec této práce. Je možné jej dohledat například v [3], strana 172/179.

Diffie-Hellman, ElGamal a Schnorr, však mohou být implementovány pomocí eliptických křivek nad konečnými poli [3, s. 187-195].

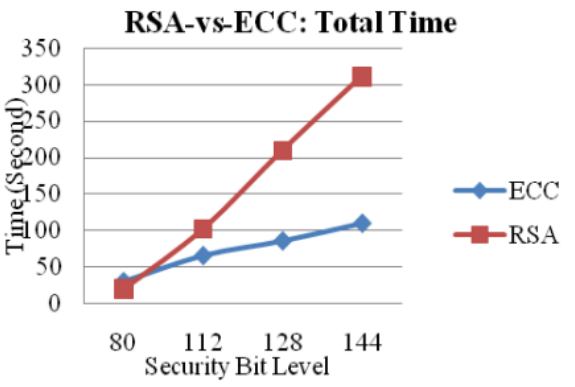
Srovnání kryptografických systémů RSA a ECC je možné si prohlédnout na grafech číslo 2 a 3. K pochopení grafů je nejprve potřeba si vysvětlit graf číslo 1. Zmíněný graf popisuje doporučenou bezpečnostní úroveň v bitech dle institutu NIST. Například pro 80bitovou úroveň zabezpečení, NIST doporučuje pro RSA klíč dlouhý 1024 bitů a pro ECC 160 bitů. Po tomto vysvětlení lze pozorovat rozdíly při šifrování a dešifrování systémů RSA a ECC. Dle měření prováděného Mahtoem a Yadavem [8] lze vidět, že při vstupu 8 bitů (graf č. 2) je RSA při celkové čase, kdy se měřilo jak šifrování i dešifrování, rychlejší až na hranici mezi bezpečnostní úrovní 112 a 128 bitů. Od 128 bitů je rychlejší ECC. Při vstupu 256 bitů (graf č. 3) je prakticky od začátku rychlejší ECC. Za zmínku rozhodně stojí připomenout, že ECC při výpočtech, dle grafu č. 1, využívá několikanásobně kratší šifrovací klíč oproti RSA. Autoři měření konstatují, že ačkoli je RSA velice efektivní nástroj v šifrování, ECC je efektivnější a bezpečnější nástroj než právě zmíněné RSA [8].



Obrázek 2:Bitová úroveň zabezpečení doporučená NIST (zpracováno dle předlohy) [8]



Obrázek 3: Vývoj časové náročnosti zašifrovaní a dešifrování při velikosti vstupních dat 8 bitů [8]



Obrázek 4: Vývoj časové náročnosti zašifrovaní a dešifrování při velikosti vstupních dat 256 bitů [8]

1.1.4 Využití v praxi

Na rozdíl od symetrické kryptografie, která je hojně využívána pro lokálně uložená elektronická data, je asymetrická kryptografie využívána především pro šifrování při komunikaci. Díky dvojici klíčů, ve které má každý svou úlohu v šifrování a dešifrování, řeší asymetrická kryptografie problém distribuce sdíleného klíče, který data šifruje i dešifruje. Ačkoli distribuce veřejného klíče má také svá úskalí, jejichž řešení práce popisuje v následujících kapitolách.

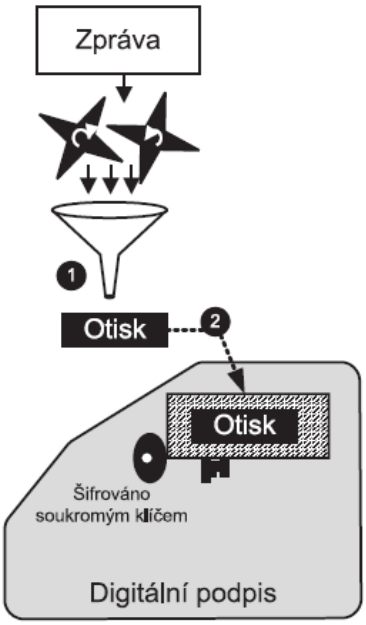
Asymetrické šifrování má v zásadě dvojí využití podle toho, v jakém pořadí je dvojice klíčů použita. Pokud je nejprve použit veřejný klíč k zašifrování dat a poté soukromý pro dešifrování, jedná se o použití, jehož cílem je zajištění důvěrnosti přenášených dat. V druhém případě je pořadí použití klíčů opačné. Využití je především pro určení totožnosti odesílatele (digitální podpis) a zaručení integrity dat [9].

1.2 Digitální podpis

Jak již bylo výše zmíněno, digitální podpis je založen na principu asymetrické šifry, tedy šifry založené na soukromém a veřejném klíči, a slouží k zajištění pravosti dokumentů. Podepsání dokumentu digitálním podpisem se provádí obecně ve dvou krocích:

1. Spočítá se otisk dokumentu, takzvaný hash.
2. Otisk se následně šifruje soukromým klíčem uživatele, který podpis vytváří.

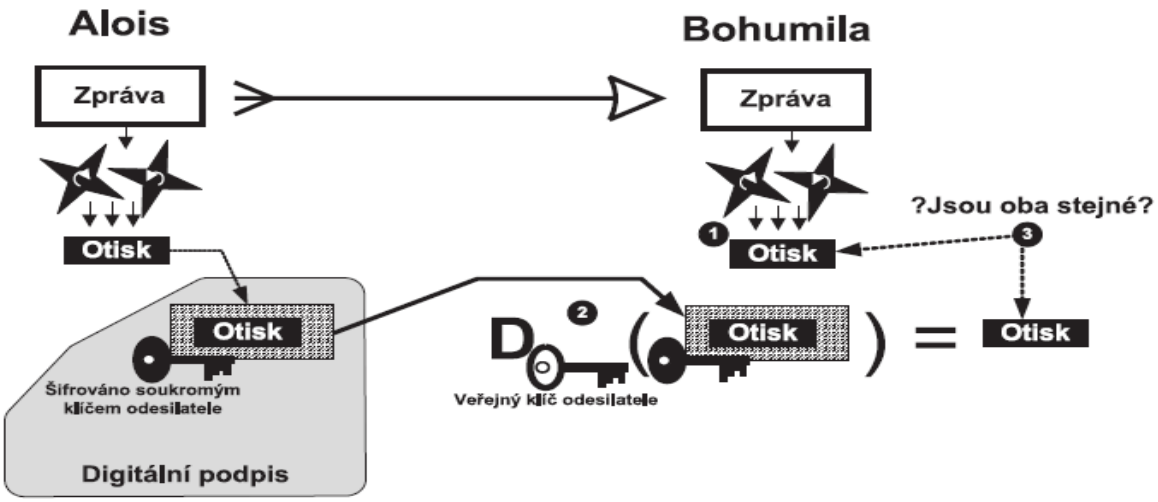
Celý proces je znázorněn na obrázku 2, kde jsou i vyznačeny popsané body. Výsledný otisk zprávy zašifrovaný soukromým klíčem je poté možno nazývat digitálním podpisem [1, s. 27].



Obrázek 5: Digitální podpis – tvorba [1, s. 27]

Na obrázku 3 je znázorněno ověřování digitálního podpisu. Ověřování probíhá ve třech následujících krocích:

1. Příjemce zprávy samostatně spočte otisk z přijaté zprávy.
2. Poté příjemce dešifruje digitální podpis odesilatele jeho veřejným klíčem. Tím získá původní otisk zprávy od odesilatele.
3. Příjemce porovná otisk získaný v bodě 1 s otiskem získaným v bodě 2. Pokud se tyto otisky rovnají, příjemce má jistotu, že zpráva je opravdu podepsána odesílatelem. Zároveň je takto prokázáno, že integrita zprávy nebyla narušena [1, s 27].



Obrázek 6: Digitální podpis – verifikace [1, s. 28]

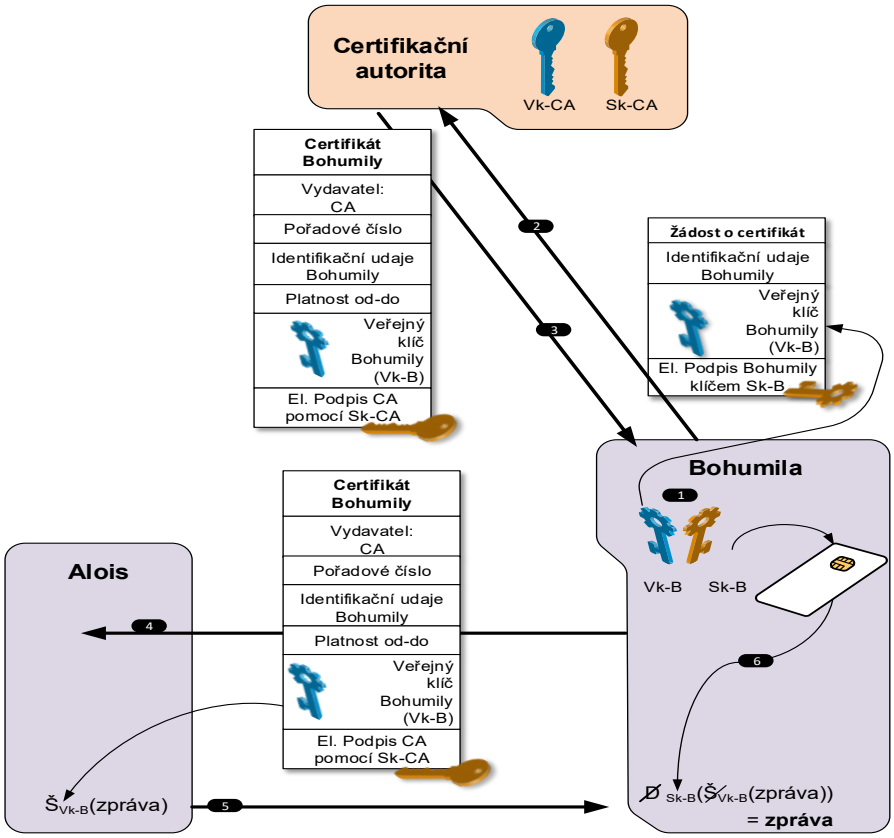
Mezi nejznámější algoritmy pro vytváření digitálního podpisu samozřejmě patří již zmíněné RSA a ElGamal, ale také algoritmy jako DSA, ECDSA, Rabin a další. Většina algoritmů je modifikací již nějakého známého algoritmu. Například DSA je modifikací šifrovacího algoritmu ElGamal a přináší vylepšení například v podobě kratší délky podpisu navzdory stejné bezpečnostní úrovni. ECDSA je potom modifikací DSA, která využívá eliptických křivek [10].

Digitální podpis zažívá velký rozmach a dá se předpokládat, že v budoucnosti bude tento trend pokračovat. Techniky digitálního podepisování se v praxi používají v řadě aplikací. Mezi nejpoužívanější patří emailová komunikace, finanční transakce nebo podepisování různých dokumentů, například pro státní správu [3, 208-209].

I digitální podpis založený na asymetrické kryptografii má své riziko. Jedná se o důvěryhodnost veřejného klíče. Pokud si opět představíme komunikaci mezi Aloisem a Bohumilou, kterou odposlouchává Eva, může nastat situace, kdy si Alois vyžádá veřejný klíč Bohumily, aby ji mohl zaslat zprávu šifrovanou jejím veřejným klíčem a zároveň ji pošle svůj veřejný klíč, aby Bohumila mohla ověřit Aloisův digitální podpis. Komunikaci ale odposlouchává Eva, která tyto klíče zachytí a Aloisovi i Bohumile podvrhne svoje veřejné klíče. Alois i Bohumila tedy budou vlastnit Eviny veřejné klíče v domnění, že Alois vlastní Bohumilin veřejný klíč a Bohumila vlastní Aloisův. Nyní může Eva bez povšimnutí číst a měnit zprávy zasílané mezi Aloisem a Bohumilou a zároveň je může i podepisovat.

Takovýmto situacím by mělo zabránit využití certifikační autority. Certifikační autorita je důvěryhodná třetí strana, jejíž veřejný klíč je veřejně známý a považovaný za důvěryhodný. V takovéto situaci Bohumila nezasílá svůj veřejný klíč rovnou Aloisovi, ale jak je vidět na obrázku 4, podává žádost o certifikát k certifikační autoritě, ve které uvádí svoje identifikační údaje, svůj veřejný klíč a celou žádost podepíše svým digitálním podpisem. Certifikační autorita vytvoří dle žádost certifikát obsahující veřejný klíč Bohumily, podepíše ho svým soukromým klíčem a zašle ho Bohumile. Bohumila poté již nezasílá Aloisovi pouze veřejný klíč, ale certifikát podepsaný certifikační autoritou, který si může Alois u certifikační autority ověřit. Obdobně postupuje při sdílení svého veřejného klíče Alois [11].

Žádost o certifikát a zejména certifikát samotný jsou datové struktury, které jsou definovány standardy. Certifikát popisovaný v této práci definuje standard X.509 [11].



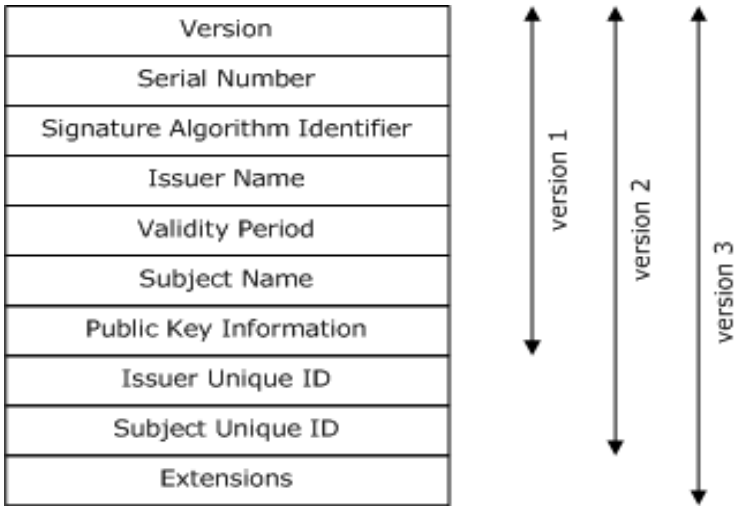
Obrázek 7: Asymetrická kryptografie – komunikace za pomocí certifikátů [1, s.57]

1.3 Certifikáty X.509

1.3.1 Obecně, struktura certifikátů

„*Certifikát je digitálně podepsanou datovou strukturou, jejíž základní součástí je veřejný klíč držitele certifikátu.*“ [1, s. 58] Certifikát je mnoha publikacích přirovnáván k průkazu totožnosti, jelikož má spoustu shodných atributů. Tato práce se zabývá certifikátem, který je definován standardem X.509 a proto se i tímto přívlastkem označuje i samotný certifikát – certifikát X.509. Certifikát X.509 byl vyvinut ve třech verzích. První verze byla poprvé představena v roce 1988, verze 2 v roce 1993 a verze 3 v roce 1996 [3, s. 223].

Před samotným rozborem struktury certifikátu je potřeba zmínit, že veškerý popis certifikátu X.509 v této práci se bude týkat poslední verze certifikátů, verze 3, která byla vydána Mezinárodní telekomunikační unií (ITU), avšak rozdíly v mezi strukturami verzí 1 až 3 si lze prohlédnou na obrázku 5.



Obrázek 8: Struktura certifikátu X.509 - verze 1-3 [12]

Jak lze vidět na obrázku, verze 3 disponuje těmito poli: [1, s. 60-63]

- **Version** – Označení verze certifikátu. Toto pole může nabývat hodnot 0-2, což může být z počátku trochu matoucí, ale označení je jasné, hodnota 0 označuje verzi 1, 1 označuje verzi 2 a hodnota 2 označuje verzi 3.
- **Serial Number** – Sériové číslo je definováno jako unikátní kladné číslo, které je certifikátů přiřazeno vydávají certifikační autoritou (CA). Číslo je unikátní pro danou CA. Stejné číslo může mít teoreticky certifikát vydaný jinou CA, protože teprve Serial Number společně s Issuer Name jednoznačně identifikují certifikát.
- **Signature Algorithm Identifier** – Algoritmus podpisu označuje identifikátor objektu (*AlgorithmIdentifier)*, který specifikuje kombinaci algoritmů, které byly použity pro vytvoření digitálního podpisu certifikátu certifikační autoritou. Jedná se o hashovací algoritmus pro výpočet otisku, například SHA-1, a asymetrický šifrovací algoritmus, kterým je poté otisk šifrován, například RSA.
- **Issuer Name** – Issuer neboli vydavatel, specifikuje toho, kdo certifikát vydal. Obsahuje však datový objekt *RelativeDistinguishedName,* který specifikuje relativní jedinečné jméno, které je tvořeno dílčími informacemi o subjektu jako jsou například jméno, název země, adresa, email nebo například název společnosti. Tyto informace jsou vždy uloženy ve dvojici, kterou vždy tvoří identifikátor objektu a hodnota. Například: *Country = CZ*
- **Validity Period** – Toto pole obsahuje dobu platnosti. Jedná se o časový interval, po který je daný certifikát platný.

- **Subject Name** – Subject označuje držitele certifikátu a obsahuje stejný datový objekt jako Issuer Name, tedy *RelativeDistinguishedName.* Toto pole musí být jedinečné v rámci jedné certifikační autority.
- **Public Key Information** – Toto pole obsahuje sekvenci dvou objektů, prvním z nich je *SubjectPublicKeyInfo*, ve kterém je veřejný klíč držitele certifikátu, a druhým z nich je již známý *AlgorithmIdentifier*, který stejně jako u algoritmu podpisu označuje použité algoritmy, tentokrát ale pro šifrování veřejného klíče.
- **Issuer Unique ID, Subject unique ID** – Tyto pole obsahují identifikační hodnotu vydavatele a držitele certifikátu, která by byla jednoznačná napříč všemi certifikačními autoritami. Pole jsou na obrázku, protože představovala rozšíření certifikátu u verze 2. Ve verzi 3 se však nepoužívají a veškerá rozšíření jsou skryta pod polem Extensions.
- **Extensions** – Extensions neboli rozšíření slouží k uložení informací, které se do základní položek certifikátu už nevešly. V praxi je zcela běžné, že aplikace ne všem rozšířením rozumí. Tento problém je vyřešen tím, že u každé položky je nastavena závažnost rozšíření. Pokud je závažnost u položky nastavena na hodnotu TRUE, je dané rozšíření označeno jako kritické a pokud ho aplikace nedokáže zpracovat, musí daný certifikát odmítnout. Pokud je hodnota nastavena na FALSE, tak rozšíření není kritické a aplikace ho může v případě, kdy ho nedokáže zpracovat, ignorovat. Seznam standardních rozšíření i s popisem je možné prostudovat v tabulce číslo 1 a 2.

	Rozšíření certifikátu	**V certifikátech CA**	**V certifikátech koncových uživatelů**
Standardní internetová rozšíření	**Identifikátor klíče úřadu** (Authority Key Identifier)	**Povinné ve všech certifikátech, které nejsou kořenové. Nesmí být označeno jako závažné**	
	Identifikátor klíče předmětu (Subject Key Identifier)	**Povinné; nesmí být závažné**	**Mělo by být; nesmí být závažné**
	Použití klíče (Key Usage)	**Povinné v certifikátech, pomocí kterých se verifikuje el. Podpis certifikátů a CRL; mělo by být závažné**	
	Rozšířené použití klíče	**Je povinné pro některé certifikáty (TSA, DVCS apod.)**	
	Platnost soukromého klíče (Private Key Usage Period)		
	Certifikační politiky, též někdy Zásady certifikátu (Certificate Policies)	**Volitelné**	
	Mapování zásad (Policy Mapping)	**Jestliže se použije, pak by mělo být závažné.**	
	Subject Directory Attributes	**Jestliže se použije, pak nesmí být závažné**	
	Alternativní jméno úřadu (Issuer Alternative Name)	**Nemělo by být závažné, aplikace jej musí rozeznávat**	

	Základní omezení (Basic Constraints)	**Musí být použito, a to jako závažné**	
	Omezení jmen (Name Constraints)	**Je-li použito, pak jako závažné**	
	Omezení politik (Policy Constraints)	**Je-li použito, mělo by být závažné**	
	Distribuční místa seznamu odvolaných certifikátů (CRL Distribution Points)	**Nemělo by být závažné**	
	Omezení (Any-Policy)	**Je-li použito, pak jako závažné**	
	Nejčerstvější seznam CRL (Freshest CRL)	**Nesmí být závažné**	
Privátní internetová rozšíření	**Přístup k informacím úřadu** (Authority Information Access)	**Nesmí být závažné**	
	Přístup k informacím předmětu (Subject Information Access)	**Nesmí být závažné**	
Kvalifikované certifikáty	**Biometrické informace** (Biometric Information)	**Nesmí být závažné**	
	Qualified Certificate Statements	**Může i nemusí být závažné**	
Microsoft	**Název šablony certifikátu** (Certificate Template Name)		

Tabulka 1: Certifikát X.509, rozšíření [1, s.64]

V praxi je možné se také setkat s pojmem kvalifikovaný certifikát. Je to typ certifikátu, který slouží k náhradě podpisu psaného rukou digitálním podpisem. Kvalifikovaný certifikát nemá jinou strukturu než běžný certifikát, ale jeho podstata tkví v tom, že slouží k jednoznačné identifikaci držitele a CA vždy zná konkrétní osobu, která certifikát vydala [1, s. 73]. Co se týká životního cyklu certifikátu, dal by se rozdělit do pěti fází, které Dostálek popisuje následovně. [1, s. 74]

1. Vytvoření žádosti o certifikát,
2. vydání certifikátu,
3. platnost certifikátu – Po vydaní nemusí být certifikát automaticky platný. Certifikát je platný v určeném časovém intervalu a jeho platnost končí buď vypršením nebo odvoláním certifikátu,
4. expirace nebo odvolání certifikátů.
    a. Expirace certifikátu – přirozené vypršení platnosti certifikátu,

b. odvolání certifikátu – Odvolání certifikátu je možné ještě před vypršení platnosti certifikátu. Odvolání certifikátu je zajištěno tak, že CA certifikát zveřejní na seznamu odvolaných certifikátů (CRL).

1.3.2 Principy validace certifikátů a podmínky platnosti

Podmínky platnosti

Podmínky platnosti u certifikátu jsou v zásadě dvojí. První z podmínek je datum platnosti certifikátu. Ověření je v tomto případě triviální. Aktuální datum, které reprezentuje čas, kdy certifikát ověřujeme, musí být v časovém intervalu, který získáme z certifikátu. Druhá podmínkou je, že certifikát nesmí být odvolán, tedy zneplatněn. Informace o zneplatněných certifikátech jsou vystavovány certifikačními autoritami, které daný certifikát vydaly, na revokačním seznamu (CRL).

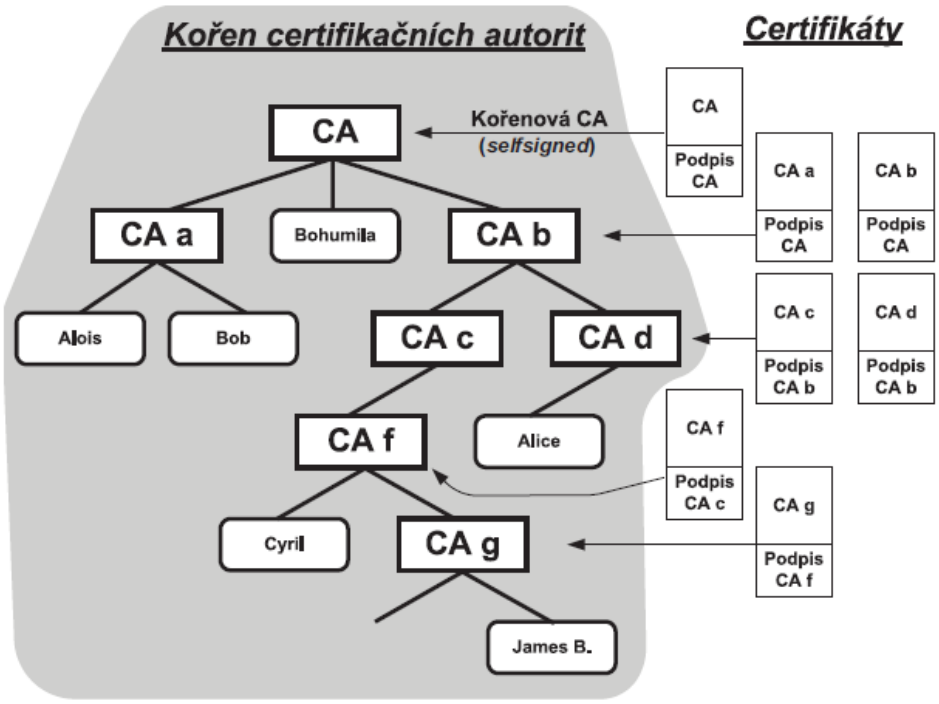
Získání a kontrola CRL však není jediná metoda ověření, zda byl certifikát zneplatněn. V dnešní době je efektivnější využití OCSP protokolu, díky kterému lze ověřit, zda byl certifikát zneplatněn i bez stahování CRL. OCSP servery většinou provozuje CA, která daný certifikát vydala. Níže je však popsán klasický postup kontroly certifikátu pomocí CRL.

Informace o CRL bývají z pravidla uloženy v rozšířeních certifikátu.

- V položce *CRL Distribution Points* by měly být uvedeny URI, na kterých je zveřejněný CRL.
- V položce *Freshest CRL* je potom uvedeno URI, kde je publikováno přírůstkové CRL.
- Položka *Authority Information Access* může obsahovat URI, na kterém naslouchá OCSP server [1, s. 109].

Certifikační cesta

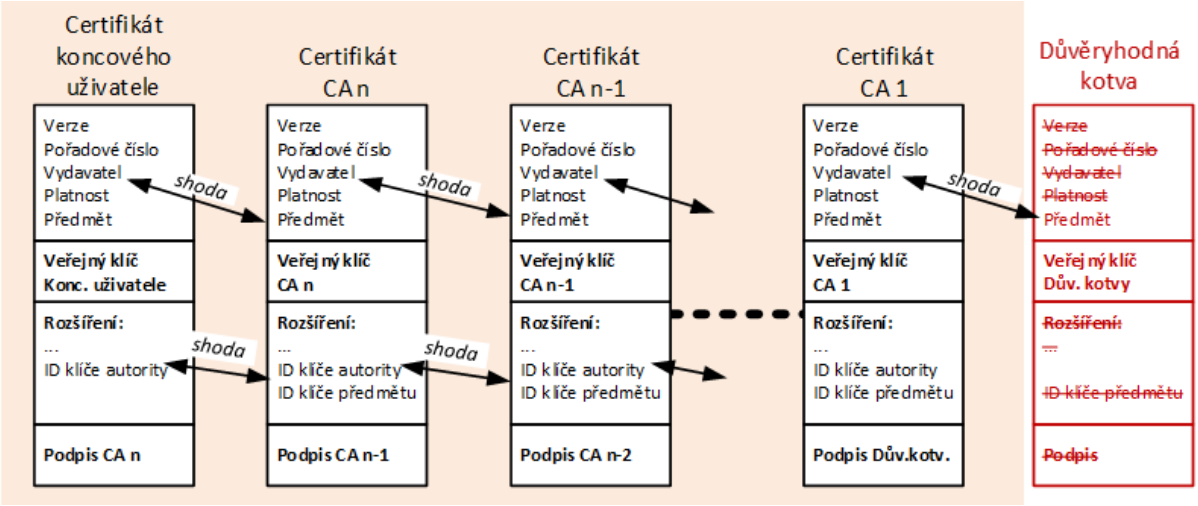
V praxi se lze setkat s pojmem kořenová certifikační autorita. To je CA, která nad sebou již žádnou nařízenou CA nemá, svůj certifikát si podepisuje sama a je obecně uznávanou za důvěryhodnou CA. Ne všechny CA, jsou však kořenové CA. Některé CA jsou podřízené kořenové CA, což znamená, že jejich certifikáty si nepodepisují samy, ale vydala jim je právě jejich nadřízená CA, která však pořád nemusí být kořenovou CA, klidně může být podřízená jiné nadřízené CA. Takové větvení certifikačních autorit se nazývá stromem certifikačních autorit [11]. Pro lepší představu zobrazeno na obrázku 6. Na obrázku je také vidět, že kořenová CA, může vydávat certifikáty jak podřízeným CA, tak i koncový uživatelům.



Obrázek 9: Strom certifikačních autorit [1, s.98]

Pokud budeme dle obrázku č. 6 ověřovat například certifikát Cyrila, potřebujeme znát a ověřovat všechny certifikáty v certifikační cestě až ke kořenové CA. Tento problém je řešen takzvanou důvěryhodnou kotvou. Tímto pojmem je označen certifikát, který nemusí být kořenový, ale je prohlášen za důvěryhodný a certifikáty v certifikační cestě se poté ověřují pouze k nejbližší důvěryhodné kotvě [11]. Pokud bychom na obrázku č. 6 označili za důvěryhodnou kotvu certifikát certifikační autority *CA b* a chtěli ověřit certifikát Cyrila, je tedy potřeba nejprve sestavit certifikační cestu.

Tvorba certifikační cesty je znázorněna na obrázku 7. Tedy od koncového uživatele se postupuje až k důvěryhodné kotvě. V certifikátu koncového uživatele je vybrána položka *Vydavatel* a položka *ID klíče autority* a poté se hledá certifikát, který má totožné hodnoty, akorát v polích *Předmět* a *ID klíče předmětu.* Položky s ID jsou zde kvůli tomu, že certifikační autority mohou mít více certifikátů a pole *Vydavatel* tak nemusí být unikátní, zatím co pole *ID klíče autority* bude vždy odkazovat na jedinečný *ID klíč předmětu.* Samotné ověřování pak probíhá ve směru od důvěryhodné kotvy ke koncovému uživateli. Tedy na obrázku č. 7 se nejdříve bude ověřovat certifikát *CA 1*, přičemž ověřování tkví v ověření podpisu předešlého certifikátu (v tomto případě důvěryhodné kotvy) a také ve zkontrolování různých rozšíření certifikátu [11].

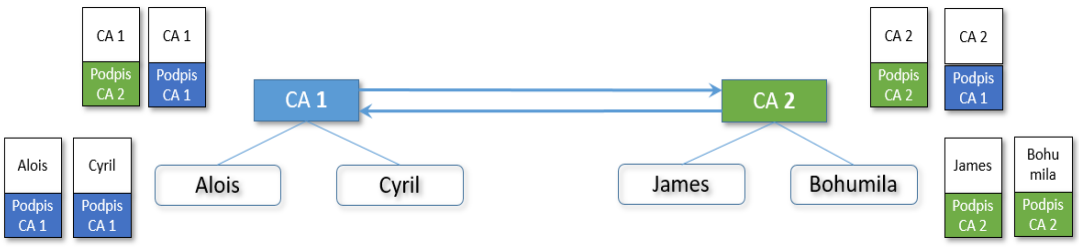


Obrázek 10: Certifikační cesta [1, s.99]

Důvěra mezi CA

Může nastat situace kdy spolu budou chtít šifrovaně komunikovat dva uživatelé, kteří mají každý vydaný certifikát u jiné certifikační autority a tyto autority nejsou v jednom stromu certifikačních autorit. Dostálek [1, s. 100] ve své knize zmiňuje čtyři možné řešení takového problému.

1. Prvním řešením je, že si uživatel přidá certifikační autoritu, které důvěřuje uživatel druhý, mezi své důvěryhodné kotvy a bude jí důvěřovat také. Z bezpečnostního hlediska se nejedná o příliš efektivní řešení.
2. Dalším řešením je takzvaná křížová certifikace. Tuto metody si lze prohlédnout na obrázku 8, kde dvě certifikační autority tvoří dva samostatné stromy. Budeme nyní uvažovat že obě CA, jsou kořenovými CA, tudíž si každá podepsala sama sobě svůj certifikát. Křížová certifikace zjednodušeně znamená, že si CA navzájem zažádají o vydání certifikátu. Každá z nich pak bude mít dva certifikáty.



Obrázek 11: Křížová certifikace mezi CA [1, s.102]

Pokud situaci popíšeme na obrázku 8, CA1 má po křížové certifikaci vlastní podepsaný certifikát a zároveň certifikát podepsaný CA2. Stejně tomu bude i u CA2. Pokud by tedy chtěl Cyril komunikovat s Bohumilou, po křížové certifikaci

se zařadí certifikát CA1 podepsaný od CA2 do certifikační cesty. Stejně tomu bude i z pohledu druhé strany, pokud uvažujeme obousměrnou komunikaci.

3. Řešení dva pomocí křížové certifikace je funkční, ale v případě komunikace mezi vícero samostatnými certifikačními autoritami zcela neefektivní. Pro tento případ je zapotřebí mezi CA vytvořit most (Bridge). Takto se označuje certifikační autorita, kterou společně vybudují CA, které spolu chtějí komunikovat. Tato společná CA se tedy pak pro zainteresované stává jakousi nadřazenou CA, která ostatním vydá certifikáty a tím vytvoří spojnici při tvorbě certifikační cesty.
4. Posledním zmiňovaným řešením je CTL (Certificate Trusted List). Jedná se seznam důvěryhodných certifikátů nebili důvěryhodných kotev, který je podepsaný certifikační autoritou, které se obecně důvěřuje. Například v Evropské unii existuje nařízení eIDAS, které specifikuje podmínky vydávání určitých certifikátů. Zároveň existuje dohoda zemí Evropské unie, podle které předávají členské země seznam[3] důvěryhodných certifikátů (kotev), pro vytvoření CTL.

Proces validace

V předchozích kapitolách byly základně představeny všechny dílčí části validace certifikátu X.509. Proces validace tedy v základu tvoří tři hlavní části.

- Validace datumu platnosti certifikátu.
- Ověření statusu certifikátu, potažmo není-li certifikát revokován. Buď pomocí protokolu OCSP nebo pomocí CRL.
- Sestavení certifikační cesty až po důvěryhodnou kotvu a ověření podpisů všech certifikátů v této cestě.

Protože řešit validaci certifikátů jednotlivě u každého uživatel není zrovna efektivní (například pro podnikový intranet), RFC-3379 zavádí dvě strategie (protokoly), jak situaci řešit. DPV a DPD.

- DPV – Umožňuje využívat k ověření DPV server. Server ověří certifikát sám a klientovi vrací výsledek ověření.
- DPD – Server DPD poskytuje klientovi informace potřebné k ověření certifikátu. Samotné ověření certifikátu se provádí lokálně u klienta [1, s. 94].

[3] Seznam důvěryhodných certifikátů je k nahlédnutí na webové stránce: https://webgate.ec.europa.eu/tl-browser/#/

2 SYSTÉMOVÁ SPRÁVA CERTIFIKÁTŮ

2.1 Ruční správa certifikátů

2.1.1 Úkony spojené s manuální správou certifikátů

Ruční správa certifikátů s sebou nese značnou zátěž. O to větší, pokud se spravovaná infrastruktura informačních technologií stále rozšiřuje a stává se tak její údržba pro správce čím dál složitější. Moderní systémy vyžadují mnohdy provoz 24/7/365 a digitální certifikáty jsou tohoto provozu nedílnou součástí. Pokud je infrastruktura spravována manuálně, je nutné udržovat si tabulky s umístěními a stavy jednotlivých certifikátů a obecně věnovat správě certifikátů značný čas [13].

Studie ukazují, že správa certifikátů je často zanedbávána. Ve studii z roku 2020 sponzorované společností Keyfactor, provedl Ponemon Institute průzkum u 596 odborníků na IT bezpečnost ve Spojených státech a zjistil, že 74 % respondentů uvedlo, že mají zkušenosti s neočekávanou nefunkčností serverů, způsobenou digitálními certifikáty. Dále zjistili, že 46 % respondentů se domnívá, že nejvyšší prioritou při minimalizaci rizika nezajištěných digitálních identit, je znát datum expirace certifikátu [14].

Keyfactor [15] popisuje správu životního cyklu certifikátu jako vlastní obor, který se shoduje s tématikou PKI, ale disponuje vlastní sadou pravidel a protokolů. Obor je zaměřen na zjišťování, správu a monitorování digitálních certifikátů. Jakmile jsou certifikáty certifikační autoritou vydány, je potřeba, aby byly po celou dobu platnosti řádně spravovány. Správa životního cyklu certifikátu se dá rozdělit na sedm základních částí představených níže: [15]

- **Zjištění certifikátů** – V této fázi je potřeba zkontrolovat celou síťovou infrastrukturu, aby se zjistilo, kde jsou jednotlivé certifikáty nainstalovány a aby se ověřilo, zda jsou certifikáty implementovány správně. Tímto procesem se eliminují neznámé certifikáty nebo certifikáty, které nejsou korektně implementovány a mohly by se tak stát slabými články bezpečností infrastruktury.
- **Generování certifikátů / Žádost o certifikátu u CA** – Při tvorbě síťové infrastruktury je třeba dbát na správnou konfiguraci certifikátů. Při zasílání formuláře na certifikační autoritu je potřeba vědět o jaký certifikát mám zájem a jaké vlastnosti by měl certifikát obsahovat (různé rozšíření a podobně). Je potřeba mít také již vygenerovaný soukromý a veřejný klíč. Veřejný klíč je samozřejmě s formulářem poskytnut certifikační autoritě pro vydaní certifikátu s tímto klíčem.

- **Analýza** – Certifikátů je celá řada. Čím je větší rozmanitost typů certifikátů v infrastruktuře, tím složitější je jejich celková správa. Je dobré certifikáty analyzovat a mít přehled například kde a jaké se používají šifrovací algoritmy nebo popřípadě znát umístění stahovaných CRL.
- **Dohled nad certifikáty** – Při velkém počtu certifikátů je vhodné ukládat a spravovat všechny certifikáty na centralizovaných serverech, distribuovaných systémech a cloudových platformách.
- **Monitorování certifikátů** – Certifikáty je z pohledu správce potřeba stále monitorovat, aby byl případně certifikát včas obnoven před jeho expirací a nedošlo tak k nechtěné nefunkčnosti nějaké aplikace či služby.
- **Validace** – Každý certifikát je platný pouze po omezenou dobu. Tato doba je dána časovým intervalem platnosti certifikátu. Někdy ale okolnosti zapříčiní zneplatnění certifikátu ještě před koncem časového intervalu platnosti. Mezi tyto okolnosti může například patřit změna názvu společnosti, změna pracovního vztahu zaměstnance s organizací. Tyto situace je potřeba hlídat, aby byl certifikát stále validní. Popřípadě je nutné certifikát obnovit či odvolat.
- **Revokace** – U certifikátu X.509 je revokace neboli odvolání certifikátu zprostředkováno tak, že certifikát je vystaven na seznamu odvolaných certifikátů (CRL). CRL vystavuje každá certifikační autorita. Nevýhoda CRL tkví v jeho periodě zveřejňování. Certifikát může být sice ihned zařazen na CRL, ale nejbližší plánované zveřejnění CRL může být třeba až za hodinu, za den nebo klidně i za týden – záleží na nastavení. Stav certifikátu se považuje za změněný, pokud je certifikát zrušen (*Revoked*) nebo pozastaven (*Hold*).
    - Zrušení certifikátu – Tento stav nastává, pokud se například zjistí, že certifikační autorita vydala certifikát nesprávně, nebo pokud existuje podezření na prolomení soukromého klíče, nebo pokud klient poruší zásady stanovené certifikační autoritou. Nejběžnějším důvodem je ale ztráta soukromého klíče uživatelem. V takovém případě je certifikát nevratně odvolán a zařazen na CRL.
    - Pozastavení certifikátu – tento stav může být použit k dočasnému zneplatnění certifikátu. Například v situaci, kdy si majitel soukromého klíče není jist jeho ztrátou. Pokud je klíč nalezen, certifikát se odebere z CRL a je opět platný.

2.1.2 Rizika

Manuální správa certifikátů nese značná rizika. Největším rizikem při manuální správě je výpadek systému nebo služeb z důvodu expirace certifikátu. Při manuální správě si správce musí hlídat, zda jsou všechny certifikáty aktuální a validní, což je u zvyšujících se počtu zařízení a tím i certifikátů stále složitější a přináší to tak riziko stavu nefunkčního systému způsobeného lidským faktorem. I když řada společností nabízí v oblasti kybernetické bezpečnosti služby automatizované správy certifikátů, některé organizace stále spoléhají na ruční postupy monitorování certifikátů založené na tabulkách s informacemi o stavu certifikátu [15].

Mezi další rizika patří například neschopnost reagovat na nutnou výměnu většího počtu certifikátů z důvodu nějakého rozsáhlého kryptografického incidentu [14].

2.1.3 Nevýhody

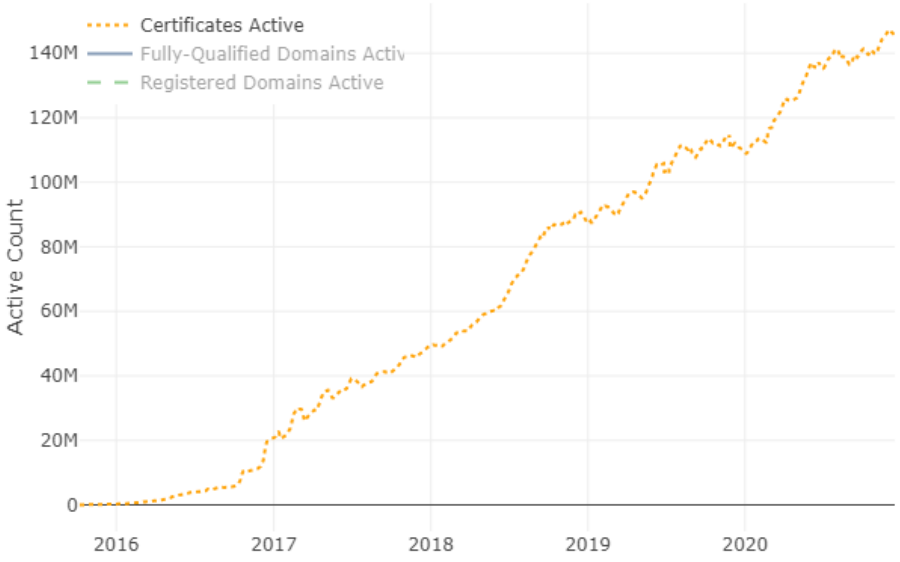
Samotné nevýhody ruční práce správy certifikátů vyplývají z předchozích podkapitol, proto již zde budou pouze shrnuty. Hlavními nevýhodami jsou tedy především složitá správa certifikátů prostřednictvím tabulek. Tabulky k udržení stavu certifikátů skrývají riziko v tom, že uživatel neupozorňují na expiraci certifikátu či jeho revokaci. Dále je uživatel nucen veškeré úkony související s životním cyklem certifikátu obstarávat ručně. Jedná se například o podávání žádostí k certifikační autoritě, obnova certifikátů, správná konfigurace a os ní spojená obecná znalost problematiky certifikátů. Tato potřebnost znalosti problematiky může některé základní uživatele omezovat. Tyto všechny procesy zastávají značnou časovou náročnost, která by se však dala eliminovat zavedením automatizace, kterou řada společností nabízí.

2.2 Automatické vydávání certifikátů

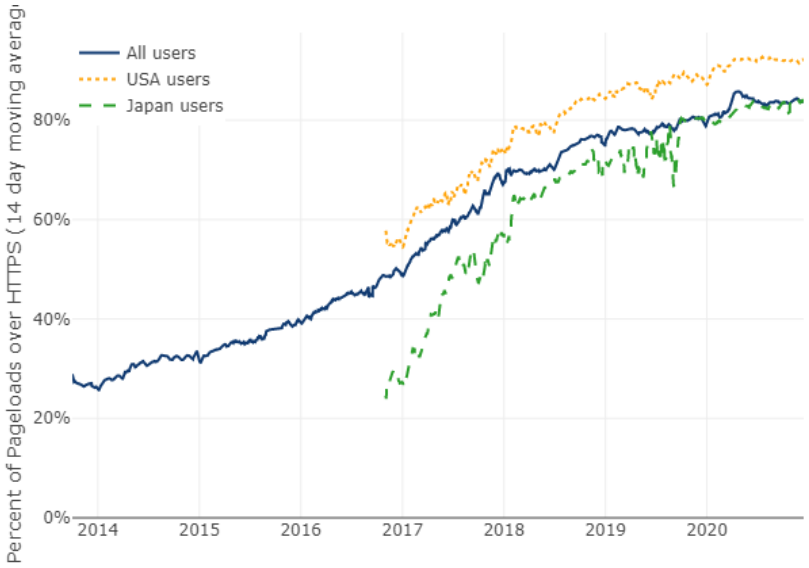
2.2.1 Projekt Let's Encrypt – základní informace

V roce 2010 mezinárodní nezisková organizace EFF (Electronic Frontier Foundation), zabývající se právní ochranou a poradenstvím při technologický soudních přích, oskenovala celý adresní prostor IPv4, aby zjistila, jaké se kde používají certifikáty. Při této příležitosti byly zjištěny poměrně alarmující stavy. Sken odhalil, že té době existovalo zhruba tisíc různých certifikačních autorit, které jsou všechny důvěryhodné, přestože v oficiálním seznamu jich bylo okolo 60. Byly také odhaleny zásadní technologické přešlapy v konfiguraci webových serverů nebo dokonce přešlapy samotných certifikačních autorit při vydávání certifikátů. V roce 2010 nebylo šifrování na internetu tolik rozšířené, a proto začala EFF apelovat na největší poskytovatele webových služeb, aby začali implementovat do svých produktů šifrování [16].

V reakci na toto zjištění později EFF ve spolupráci s Mozillou a Michiganskou univerzitou, spustila projekt Let’s Encrypt, jehož cílem je automatizovat a eliminovat problémy při zavádění TLS. Let’s Encrypt by mělo platit za ukázkovou certifikační autoritu, která je transparentní, plně automatická a zdarma. Celý projekt se dá rozdělit na tři moduly. První z nich se jmenuje Boulder a jedná se o vlastní automatickou certifikační autoritu. Druhým modulem je protokol nesoucí název ACME a zde se jedná o protokol pro automatizaci vydávání certifikátů. Posledním modulem je ACME klient nazvaný Letsencrypt, jehož úkolem je zautomatizovat konfiguraci TLS a převést nešifrované webové stránky na šifrované během pár kliknutí [16].
Let's Encrypt bylo spuštěno v listopadu 2015 a od té doby jeho popularita počet vydaných certifikátů pořád vzrůstá. Na grafu č. 4 si lze prohlédnout celkový vývoj v počtu aktivních certifikátů vydaných certifikační autoritou Let's Encrypt. Graf č. 5 ukazuje aktuální procentuální statistiku webových stránek načtených prohlížečem Firefox pomocí protokolu HTTPS. Graf znázorňuje jak veškeré uživatele prohlížeče, tak pro porovnání uživatele pouze z USA nebo Japonska. Ze statistik Let's Encrypt lze také vyčíst, že certifikační autorita nyní vydává stabilně přes 1,5 miliónu certifikátů denně [17]. Z těchto prezentovaných statistik lze snadno vyvodit, že projekt je úspěšný a jeho cíle, zajistit jednoduché šifrování na internetu pro všechny, se postupně naplňují.



Obrázek 12: Let's Encrypt, počet aktivní certifikátů [17]



Obrázek 13: Let's Encrypt – Firefox, procentuální využití HTTPS [17]

2.2.2 Principy fungování a postupy získání certifikátu

Co se týká principů, Let's Encrypt je založeno na plné transparentnosti. To znamená, že veškeré certifikáty vydané touto autoritou jsou vystaveny veřejně. To je zabezpečeno tak, že certifikáty vlastní sériová čísla, která tvoří posloupnost. Úplná transparentnost je poté zajištěna veřejným vystavením logů komunikace žadatelů o certifikáty. Při této transparentnosti by neměli vznikat jakékoli pochybnosti o tom, za jakých okolností byl daný certifikát vydán, protože si každý může lohy prohlédnout. Zároveň je veškerý vývoj v jazyce Go veřejně dostupný na GitHubu[4] [16]. Další z principů zajišťuje vydávání pouze certifikátů DV pro TLS. Důvodem je, že cílem této CA není vystavovat certifikáty u kterých je potřeba ověření totožnosti žadatele, ale jít názorným příkladem svými postupy ostatním CA a zjednodušit vydávání takových certifikátů, aby byly dostatečně důvěryhodné pro validaci prohlížečem a zároveň, aby byl celý proces dostatečně nízkonákladový a bylo tedy možné službu poskytovat zdarma [16].

Postup získání certifikátu je následující. [16]

- **Registrace** – Získání účtu, ke kterému jsou následně přiřazeny validace a certifikáty.
- **Validace držení domény** – Vzhledem k vydávání pouze DV certifikátů, autorita neověřuje informace o vlastníkovi, ale pouze potvrzení, že žadatel je oprávněn o certifikát pro konkrétní doménu žádat. Ověřuje se vystavením na určité cestě HTTPS/HTTP serveru, plus Let's Encrypt ještě k ověření používá certificate transparency, pro kontrolu, zda na požadované doménové jméno již není

[4] Dostupné z: https://github.com/letsencrypt

vystavený certifikát. Pokud již na požadované jméno vystavený certifikát je, CA požaduje důkaz o držení privátního klíče současného certifikátu.

- **Vygenerování certifikátu** – Po úspěšné validaci je vygenerována sada klíčů a vytvořena žádost o certifikát. CA žádost příjme a certifikát vytvoří.
- **Revokace certifikátu** – Předchozím krokem bylo získání certifikátu úspěšně dokončeno. Tento krok zde autor uvádí pro případ, kdy by bylo potřeba certifikát revokovat. Postup je zcela automatický, stačí pouze odeslat na CA identifikátor certifikátu a CA po ověření, že jste oprávněni s tímto certifikátem manipulovat, certifikát revokuje.

2.3 Automatizovaná kontrola certifikátů

2.3.1 Principy

Automatizovaná kontrola certifikátů logicky vychází z nevýhod ruční správy certifikátů. Můžeme se tedy pokusit o definování funkcionalit, kterými by měl kvalitní nástroj pro správu certifikátu disponovat.

- Automatické vyhledávání certifikátů v síťové infrastruktuře.
- Přenesení a uspořádání všech certifikátů do jednoho centrálního systému.
- Upozornění na blížící se expiraci certifikátu.
- Zefektivnění a automatizace procesů instalace a obnovování certifikátů.
- Automatizace při revokaci certifikátu.

2.3.2 Vhodné nástroje

Nástrojů poskytujících automatizovanou správu certifikátů je mnoho, zde je uveden pouze stručný výběr: [18]

- Amazon Web Services (AWS),
- Digicert,
- Global Sign,
- Certbot,
- Let's Encrypt /Bitnami HTTPS Configuration Tool,
- SecureW2.

3 NÁSTROJ PRO KONTROLU CERTIFIKÁTŮ X.509

3.1 Cíle a požadavky nástroje

Hlavním cílem praktické části toto práce byla implementace aplikace pro kontrolu certifikátu X.509. Primární uživatelským požadavkem je validace data platnosti certifikátu, ale během zpracování certifikátu probíhá mimo jiné i validace certifikátu skrze revokační seznamy a kontrola certifikační cesty až po kořenovou certifikační autoritu. Nástroj by měl být schopen prohledávat servery a nacházet certifikáty zadané v konfiguračním souboru. Z toho plyne také požadavek na načítání konfiguračního souboru. Načítaný textový soubor psaný v jazyce YAML a obsahuje konfiguraci v podobě časové filtru, filtru přípon certifikátů, defaultních autorizačních hodnot a seznam serverů s cestami k certifikátům a autorizačními metodami pro přístup na vzdálený server.

Nástroj by měl být platformě nezávislý, takže je možno přistupovat k certifikátům jak na operačním systému Windows, tak i na vzdálených Linuxových systémech. Pro přístup na vzdálené servery aplikace podporuje autentizaci skrze uživatelského jména a hesla, v tomto případě buď načteného z konfiguračního souboru nebo přímo interaktivně při připojování se na server, nebo pomocí veřejného klíče.

3.2 Technologie zvolené při implementaci

Objektově orientovaný programovací jazyk Java

Pro implementaci nástroje byl zvolen objektově orientovaný programovací jazyk Java a to především kvůli jeho širší znalosti oproti ostatním programovacím jazykům, díky využívání jazyka k implementaci semestrálních prací v průběhu celého studia. Déle kvůli jeho rozšířenosti a přenositelnosti. Java jako vyšší programovací jazyk disponuje všemi potřebnými funkcionality, ať už skrze základní nebo externí knihovny, pro implementaci zmíněného nástroje, a proto je pro tento účel zcela vhodný.

YamlBeans

V projektu je použita externí knihovna YamlBeans verze 1.11, která slouží ke z pracování konfiguračního YAML souboru. YamlBeans usnadňuje serializaci a deserializaci objektů Java do a z YAML souboru, což je formát dat, který je lehce čitelný i pro člověka [19]. V tomto projektu je pomocí této knihovny realizováno načtení a namapování YAML souboru na připravený Java objekt.

Java Secure Channel

Druhou externí knihovnou použitou v projektu je Java Secure Channel verze 0.1.55 od JCraft, která je zde využita pro zabezpečené připojení ke vzdáleným serverům skrze SSH protokol. Díky JSch lze snadno implementovat bezpečné vzdálené přihlášení a následný přenos dat, v tomto konkrétním případě certifikátů X.509.

Maven

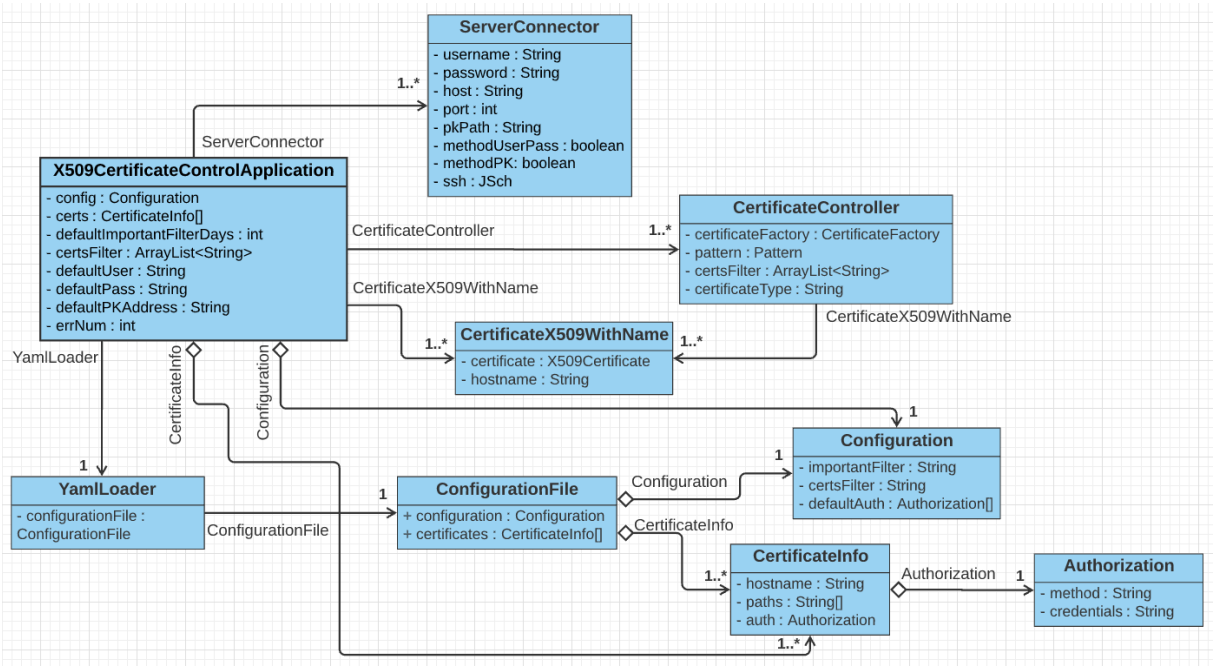
Maven je nástroj, který lze použít pro vytváření a správu libovolných projektů založených na jazyce Java [21]. Maven je schopný vytvořit build projektu a odkazovat se přitom do svých repozitářů na externí knihovny skrze dependencies. Aplikace pro kontrolu certifikátů X.509 se právě skrze Maven odkazuje na výše zmíněné knihovny YamlBeans a JSch.

IntelliJ IDEA

V neposlední řadě je potřeba zmínit vývojové prostředí, ve kterém byl celý projekt vyvíjen. InteliJ IDEA jedno z nejrozšířenějších vývojových prostředí, které se těší velké oblibě u vývojářů. Ačkoli autor využíval během studia převážně prostředí NetBeans, pro implementaci bylo nakonec zvoleno prostředí InteliJ IDEA verze 2019.2.4 z důvodu možnosti přehledné práce s Maven nástrojem a také z důvodu motivace naučit se s novým prostředím, které je hojně využíváno v praxi.

3.3 Popis návrhu aplikace

Návrh aplikace je nejlépe čitelný z UML diagramu zobrazujícího všechny třídy aplikace.



Obrázek 14: UML class diagram aplikace

Životní cyklus aplikace lze rozdělit do několika základních fází.

1. Načtení konfiguračního YAML souboru,
2. paralelní procházení seznamu serverů získaných z konfiguračního souboru,
3. připojování se ke vzdálenému systému,
4. načtení certifikátu a následná validace platnosti,
5. výpis certifikátů do konzole.

3.3.1 Načtení konfiguračního YAML souboru

Při spuštění aplikace dojde k načtení cesty ke konfiguračnímu souboru primárně z parametru použitého při spuštění. Pokud je aplikace spuštěna bez parametru, načte cestu k výchozímu konfiguračnímu souboru, který je uložený v adresáři projektu. Tento výchozí konfigurační soubor lze snadno upravovat nebo jej lze použít jako vzor pro tvorbu nového konfiguračního souboru, který bude následně aplikaci předán parametrem.
O načtení YAML souboru se stará vytvořená třída *YamlLoader*, která pomocí metody *load()* načte konfigurační soubor do předem připravené instance třídy *ConfigurationFile*. Samotné namapování souboru do připravené třídy má na starost *YamlReader* z knihovny YamlBeans.

3.3.2 Paralelní procházení seznamu serverů

Po načtení konfiguračního souboru, nastavení filtrů a pole s informace o cestách k certifikátům, lze jednoduchým for cyklem skrze třídu *CompletableFuture* spustit pro každý server nové vlákno zpracovávající metodu *checkCertificate()*.

3.3.3 Připojení ke vzdálenému serveru

Pokud se certifikáty nenačítají z lokálního uložiště je potřeba navázat zabezpečené spojení se vzdáleným serverem. Toto má na starost vytvořená třída *ServerConnector*, které jsou potřebné atributy nastaveny v konstruktoru a samotné spojení se poté vytváří skrze bezparametrickou metodu *connect()*. Jak lze vidět níže, metoda si pomocí instance JSch, pojmenované *ssh,* načte soubor *known_hosts* z adresáře *<user>\.ssh* aktuálního uživatele. Tento soubor slouží k jako zdroj uložených veřejných klíčů známých serverů. Při pokusu o připojení na server se tento seznam projde a pokud systém server nezná nebo byl klíč změněn zeptá se uživatele, zda chce klíč tohoto serveru přidat do zmíněného souboru. Při dalším pokusu o navázání spojení je již server označen jako známý a spojení se za předpokladu správného uživatelského jména a hesla vytvoří. Metoda *connect()* vrací objekt třídy *ChannelSftp*, jež představuje komunikační kanál připojený k serveru pomocí SFTP protokolu.

3.3.4 Načtení a validace certifikátu

Po úspěšném vytvoření spojení se serverem je komunikační kanál parametrem předán do instance třídy *CertificateController*, kde je s jeho pomocí prohledán adresář zadaný cestou z konfiguračního souboru. Pomocí metody *loadCertificateFromRemoteServer()* jsou dle filtru přípon certifikátů načteny do ArrayListu všechny vyhovující certifikáty. U každého certifikátu se poté skrze metodu *createCertChain()* sestaví tzv. certifikační cesta, která obsahuje seznam certifikátů až po kořenový certifikát certifikační autority, který je podepsaný sám sebou a tudíž již nemá nadřazený certifikát kterým by byl podepsán.

Tento řetěz certifikátů je parametrem předán metodě *validateKeyChain()*, která prochází veškeré certifikáty v seznamu a ověřuje jejich platnost. Zejména zda nejsou revokovány a zda je platný jejich nadřazený certifikát.

3.3.5 Výpis certifikátů do konzole

Po validaci certifikátů je vytvořen seznam certifikátů, který je již v hlavní třídě aplikace *X509CertificateControlApplication* procházen for-each cyklem jednotlivé certifikáty zapisovány metodou *validateDateAndWriteCertificateToConsole()* proměnné *log* a ta je následně vypsána do konzole.

3.4 Popis vstupního konfiguračního souboru

Vstupní konfigurační soubor je typu YAML. YAML je zkratka pro YAML Ain't Markup Language, což v překladu znamená „YAML není značkovací jazyk“. Ačkoli tedy YAML není značkovacím jazykem, dokáže skvěle plnit funkci standardu pro serializaci strukturovaných dat pro všechny programovací jazyky, stejně jako soubory typu XML nebo JSON. Je samozřejmě strojově čitelný a zároveň je skvěle čitelný i pro lidi. YAML soubor je schopen pracovat jak s řetězci, čísly, časovými informacemi, tak i s logickými hodnotami true, false a hodnotou null. Dále disponuje možností vkládání poznámek, sekvencí (polí) nebo mapou, která data ukládá vždy ve dvojici klíč a hodnota. Jednotlivé datové struktury je možno jakkoli kombinovat pomocí vnořování. Zanoření o jeden řád je v YAML souboru definováno vždy dvěma mezerami. Při použití tabulátoru nebude YAML soubor validní.

Konfigurační soubor pro vyvíjený nástroj popisuje následující struktura souboru, na které si lze prohlédnout neměnné názvy klíčů map pro YAML soubor a jejich hodnot, které jsou reprezentovány regulárními výrazy:

```
configuration:
  importantFilter: "expiration < \d+ days"
  certsFilter: "(.pem|.cer|.crt|.der)+(, (.pem|.cer|.crt|.der))*"
  defaultAuth:
    - method: userpass
      credentials: "\w([._-](?![._-])|\w)*\w:\w([._-](?![._-])|\w)*\w"
    - method: publickey
      credentials: [\w\.-\/]*
certificates:
( - hostname: (https?:\/\/)?[\w\.-]+
  paths:
    (- "[\w\.-\/]*")+
  auth:
    method: (userpass|publickey|interactive)
    credentials: "\w([._-](?![._-])|\w)*\w:\w([._-](?![._-])|\w)*\w")*
```

Uvedený regulární výraz složený z různých struktur lze vyložit dle definovaných pravidel YAML souboru. Základní strukturou je mapa obsahující dva prvky. První prvek má klíč *configuration* a hodnotou je další mapa, která obsahuje tentokrát tři prvky. První dva mají jako klíč název filtru a hodnotou je řetězec popisující filtr. Třetí prvek je má klíč *defaulAuth* a hodnotou je sekvence dvou prvků představují zase mapu a každá z map má dva prvky, kde první z nich má klíč „metod“ a klíč druhého prvku je *credentials*.

Když se dostaneme zpět na první úroveň vnoření, druhý prvek mapy je tvořen klíčem *certificates* a hodnotou je další mapa obsahující prvky s klíči *hostname*, *paths* a auth“. První dva prvky mají v hodnotě textový řetězec, ale u posledního prvku s klíčem *auth* tvoří hodnotu další mapa obsahující znovu dva prvky s klíči *method* a *credentials* hodnotami typy řetězců.

Po dosazení dle regulárního výrazu dostaneme konfigurační soubor, který vypadá například jako YAML soubor uvedený níže:

```yaml
configuration:
  importantFilter: "expiration < 60 days"
  certsFilter: ""
  defaultAuth:
    - method: userpass
      credentials: "root:root"
    - method: publickey
      credentials: ./private.key
certificates:
  - hostname: localhost
    paths:
      - "c:\\program files\\apache\\"
      - "c:\\certs\\"
  - hostname: remote.server.tld
    paths:
      - "/etc/pki"
    auth:
      method: interactive
  - hostname: 192.168.100.1
    paths:
      - "/etc/pki"
    auth:
      method: userpass
      credentials: "root:toor"
  - hostname: 192.168.100.2
    paths:
      - "/etc/pki/tls/certificate.crt"
    auth:
      method: publickey
      credentials: ./private.key
  - hostname: 192.168.100.3
    paths:
      - "/etc/pki"
      - "/etc/tls"
```

3.5 Ukázka použití nástroje

Vzhledem k faktu, že nástroj nedisponuje žádným grafickým rozhraním, lze za ukázku považovat pouze náhled na vstupní konfigurační soubor a následný výstup nástroje. V testovacím konfiguračním souboru je filtr pro upozorněni na expiraci nastaven na méně než 60 dnů k nadcházejícímu datu expirace. Filtr pro přípony certifikátů je nastaven na pro přípony .pem, .cer, .crt a .der. Výchozí hodnoty pro přístup pomocí uživatelského jména a hesla jsou root a root a pro přístup pomocí privátního klíče je potřebný klíč uložen v adresáři projektu.

Dále jsou zde uvedeny dvě certifikační cesty. První z nich odkazuje na vzdálený server *fei-as.upceucebny.cz* do složky */certs/* uživatele *st52529*. Druhá odkazuje na localhost do adresáře *test_certs* v adresáři bakalářské práce.

Níže je k nahlédnutí testovací konfigurační soubor pouze se skrytým heslem pro uživatele st52529.

```
configuration:
  importantFilter: "expiration < 60 days"
  certsFilter: ".pem, cer, .crt, .der"
  defaultAuth:
    - method: userpass
      credentials: "root:root"
    - method: publickey
      credentials: ./private.key
certificates:
  - hostname: fei-as.upceucebny.cz
    paths:
      - "/certs/"
    auth:
      method: userpass
      credentials: "st52529:rowing67"
  - hostname: localhost
    paths:
      - "C:\\Users\\David\\Documents\\UPce\\3.rocnik\\bakalarka\\test_certs"
```

Na vzdáleném serveru *fei-as.upce.ucebny.cz* se nachází platný testovací certifikát s názvem „SectigoRSAClientAuthenticationandSecureEmailCA.crt“ s datem expirace 1. 1. 2031. Na místním počítači se ve složce *testovaci_certifikaty* nachází pět certifikátu:

- „alice.crt“ s platností do 25. 4. 2022 a platnou certifikační cestou,
- „512b-dsa-example-cert.der“ s platností do 21. 8. 2017,
- „user.crt“ s platností do 13. 1. 2021,
- „postsignum_vca3_ocsp1.cer“ s platností do 18. 5. 2021 a bez rozšíření definující CRL seznamy.

Obrázek 15: Výpis testovacího běhu aplikace, obrázek z archivu autora

Test proběhl 13. 5. 2021. Na výpisu je viditelné, že vše proběhlo v pořádku. U všech certifikátů bylo vypsáno, zda jdou časově validní a pokud ne, tak byl udán počet dní do nebo od expirace. Pokud certifikát neprošel validací certifikační cesty nebo CRL seznamu, byla tato informace u certifikátu vypsána také.

ZÁVĚR

Hlavním cílem této bakalářské práce byla implementace softwarového nástroje pro automatizaci kontroly stavu certifikátu X.509. Sekundárními cíli pak bylo popsání základních principů asymetrické kryptografie, představení mechanismů, které asymetrickou kryptografii využívají, popsání struktury a správy certifikátu standardu X.509 a v neposlední řadě popsání implementace zmíněného nástroje pro kontrolu certifikátů.

Základní principy asymetrické kryptografie byly popsány v první kapitole, kde byla nejen nastíněna historie, vývoj šifrování a zrod asymetrické šifry, ale také zde byly představeny a rozebrány konkrétní algoritmy využívající tuto technologii. Uvedené algoritmy byly vysvětleny na příkladech z praxe. První kapitola také představila základní informace o certifikátech standardu X.509, jejich strukturu, principy validace a podmínky platnosti. Vysvětleno zde bylo i fungování digitálního podpisu, jeho vytvoření a životní cyklus. Dále se zde práce věnovala procesu validace certifikátu, a to zejména validaci od koncového certifikátu až po důvěryhodnou kotvu nebo kořenový certifikát certifikační autority.

Druhá kapitola se věnovala správě certifikátů, a to jak ruční správě, která je v mnoha institucích stále preferovaná, tak i automatizované správě. U manuální správy byly představeny rizika a nevýhody na rozdíl od automatizované správy, kde naopak byly vyzdviženy výhody implementace takovéto správy. Dále pak byla druhá kapitola zaměřena na automatické vydávání certifikátů pomocí projektu Let's Encrypt, jeho historické počátky a aktuální progres s narůstajícím potencionálem do budoucnosti. Na konci této kapitoly byly představeny konkrétní nástroje pro automatizovanou správu certifikátů, které již v dnešní době u spousty informačně progresivních společností fungují.

Poslední třetí kapitola byla zaměřena na popis implementace vlastní aplikace pro automatizovanou kontrolu stavu certifikátů X.509. Aplikace je nyní schopná načíst z konfiguračního souboru data potřebná k připojení se na vzdálené systémy, tam vyhledávat certifikáty, provést validaci a podat uživateli informace o aktuálním stavu certifikátů. Aplikace je tedy vhodná jako pomocný nástroj pro jakéhokoli uživatele, který má na starosti manuální správu certifikátů a je nucen jejich expiraci a stav kontrolovat ručně.

Nástroj byl vyvíjen dle smluveného zadání. Bylo zde implementováno načítání YAML souboru skrze externí knihovnu YamlBeans, bylo využito paralelního běhu vláken pro zpracování certifikátů dle jednotlivých serverů a v neposlední řadě zde byla prověřena práce s certifikáty a jejich validací. Dále byly představeny technologie použité při vývoji nástroje, životní cyklus aplikace a následně popis návrhu aplikace s popisem nejdůležitějších procesů a znázornění tříd

skrze UML diagram. Na závěr poslední kapitoly byl proveden test a ukázka funkčnosti aplikace na testovacích certifikátech, která proběhla úspěšně a tím byl hlavní cíl práce naplněn.

POUŽITÁ LITERATURA

[1] DOSTÁLEK, Libor, Marta VOHNOUTOVÁ a Miroslav KNOTEK. *Velký průvodce infrastrukturou PKI a technologií elektronického podpisu*. 2., aktualizované vydání Brno: Computer Press, 2009. 542 s. ISBN 978-80-251-2619-6.

[2] CIMINO, Al. *Příběh kryptologie: od starověkých šifer po kvantovou kryptografii*. Přeložil Marek ČTRNÁCT. Praha: Dobrovský, 2018. Knihy Omega. 215 s. ISBN 978-80-7390-887-4

[3] RHEE, Man Young. *Internet security: cryptographic principles, algorithms and protocols*. Chichester: Wiley, 2003. 405 s. ISBN 0-470-85285-2.

[4] ALGORITMY.NET. Algoritmus RSA. *Algoritmy.net* [online]. [cit. 2020-11-07]. Dostupné z: https://www.algoritmy.net/article/4033/RSA

[5] SAMANVAYAPANDA. ElGamal Encryption Algorithm. *GeeksforGeeks* [online]. Noida: GeeksforGeeks, 2018 [cit. 2020-11-09]. Dostupné z: https://www.geeksforgeeks.org/elgamal-encryption-algorithm/

[6] FULAJTÁR, David. *Asymetrická šifrovací schémata* [online]. Praha, 2014. Bakalářská práce. Bankovní institut vysoká škola Praha, Katedra informatiky a kvantitativních metod. Vedoucí práce Ing. Vladimír Beneš, Ph.D. Dostupné z: https://is.ambis.cz/th/p3de6/Fulajtar_David_Bakalarska_prace_final.pdf

[7] KLÍMA, Vlastimil, Tomáš ROSA. Kryptologie pro praxi – principy ECC. In: *Crypto-World.info* [online]. c2003-2013 Crypto-World.info, říjen 2005 [cit. 2020-11-09]. Dostupné z: http://crypto-world.info/klima/2005/ST_2005_10_12_13.pdf

[8] MAHTO, Dindayal, Dilip Kumar YADAV. RSA and ECC: A Comparative Analysis. In: *Research India Publications* [online]. Delhi: Research India Publications, 2017 [cit. 2020-11-12]. Dostupné z: https://www.ripublication.com/ijaer17/ijaerv12n19_140.pdf

[9] TRUSCHKA, Jakub. Asymetrická kryptografie v praxi. *SystemOnLine.cz* [online]. Brno: SystemOnLine.cz, [cit. 2020-11-12]. Dostupné z: https://www.systemonline.cz/it-security/asymetricka-kryptografie-v-praxi.htm

[10] LYASOTA, Zhanna. A Guide to Digital Signature Algorithms. *DZone* [online]. DZone, 28. 8. 2018 [cit. 2020-11-12]. Dostupné z: https://dzone.com/articles/digital-signature-1

[11] DOSTÁLEK, Libor. PKI a elektronický podpis 02 Certifikáty a certifikační autority. In: *Youtube* [online]. 22. 4. 2020 [cit. 2020-11-13]. Kanál uživatele Libor Dostalek. Dostupné z: https://www.youtube.com/watch?v=vgnUADpHGHg&t=21s

[12] MICROSOFT. X.509 Public Key Certificates. In: *Microsoft* [online]. Microsoft, 31. 5. 2018 [cit. 2020-11-15]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/seccertenroll/about-x-509-public-key-certificates

[13] REMME. Certificate Lifecycle Management: Manual vs. Automated. In: *Remme* [online]. Remme Capital LTD, 1. 3. 2019 [cit. 2020-12-05]. Dostupné z: https://remme.io/blog/certificate-lifecycle-management-manual-vs-automated

[14] YACKEL, Ryan. Manual vs. Automated Certificate Management | Keyfactor. In: *Security Boulevard* [online]. Security Boulevard, 4. 8. 2020 [cit. 2020-12-05]. Dostupné z: https://securityboulevard.com/2020/08/manual-vs-automated-certificate-management-keyfactor/

[15] KEYFACTOR. What is Certificate Management? In: *Keyfactor* [online]. Keyfactor, c2020 [cit. 2020-12-05]. Dostupné z: https://info.keyfactor.com/what-is-certificate-management#guidance

[16] CALETKA, Ondřej. LinuxDays 2015 - Pojďme šifrovat aneb ACME, továrna na certifikáty – Ondřej Caletka. In: *Youtube* [online]. 13. 10. 2015 [cit. 2020-12-13]. Kanál uživatele LinuxDays. Dostupné z: https://www.youtube.com/watch?v=BPoS4Ocj9lo&feature=emb_logo

[17] LET'S ENCRYPT. Let's Encrypt Stats. *Let's Encrypt* [online]. San Francisco: Let's Encrypt, [cit. 2020-12-13]. Dostupné z: https://letsencrypt.org/stats/

[18] LUDIN, Jake. The Best Certificate Management & Generation Tools. *SecureW2* [online]. SecureW2, c2020 [cit. 2020-12-15]. Dostupné z: https://www.securew2.com/blog/the-best-certificate-management-generation-tools/

[19] ESOTERICSOFTWARE. YamlBeans – README.md. In: *GitHub* [online]. GitHub, Inc c2021 [cit. 2021-05-11]. Dostupné z: https://github.com/EsotericSoftware/yamlbeans

[20] JCRAFT. JSch. In: *JCraft* [online]. Sendai, Japan: JCraft c1998-2016 [cit. 2021-05-11]. Dostupné z: http://www.jcraft.com/jsch/

[21] APACHE MAVEN PROJECT. Introduction. In: *Apache Maven Project* [online]. The Apache Software Foundation c2002–2021 [cit. 2021-05-11]. Dostupné z: https://maven.apache.org/what-is-maven.html