Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra teoretické informatiky

Bakalářská práce

Studie kvality parametrů digitálních certifikátů na českém Internetu

Martin Černáč

Vedoucí práce: Ing. Jiří Buček

11. května 2016

Poděkování

Rád bych poděkoval Ing. Jiřímu Bučkovi za cenné rady, věcné připomínky a vstřícnost při konzultacích.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 11. května 2016 .....

Odkaz na tuto práci

Černáč, Martin. *Studie kvality parametrů digitálních certifikátů na českém Internetu*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Cílem bakalářské práce byla analýza současného stavu nasazení digitálních certifikátů využívajících šifru RSA na českém Internetu a tvorba studie kvality jejich parametrů pomocí vybraných známých útoků na šifru RSA. V práci se dále zabývám stručným teoretickým úvodem pro šifru RSA, popisem jednotlivých útoků a interpretací naměřených dat.

Klíčová slova studie, kryptografie veřejného klíče, RSA, digitální certifikát, testování síly klíče

Abstract

The purpose of this bachelor thesis was to analyse the current state of RSA-based digital certificate usage across the czech Internet spectra and to conduct a study on digital certificate parameter quality using selected well-known RSA attacks. The thesis also includes a brief introduction to the theory behind the RSA cipher, descriptions of the various attacks used and a short interpretation of gathered data.

Keywords study, public key cryptography, RSA, digital certificate, key strength testing

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Komunikace je jednou ze základních lidských potřeb. Komunikujeme spolu dnes a denně o nespočetném množství témat. Zdaleka ne všechnu naší komunikaci lze ovšem charakterizovat jako „veřejnou“. Jedná se tedy zpravidla o komunikaci soukromou – takovou, která je, lidově řečeno, „mezi čtyřma očima“. Narušení takové komunikace je krajně nežádoucí.

V dnešní době probíhá velká část mezilidské komunikace vzdáleně s pomocí moderních technologií, ať už konkrétně telefonních hovorů, elektronické pošty, nebo video přenosu – tedy převážně prostřednictvím celosvětové sítě Internet. Vzhledem k rozmanitosti dostupných služeb na této síti lze tvrdit, že velká část uživatelů sítě Internet byla už nejméně jednou webovým prohlížečem upozorněna na špatně nasazený certifikát.

Protože průměrný uživatel sítě Internet nemá příliš velké povědomí o digitálních certifikátech a o nebezpečí, kterému se vystavuje ignorováním varování o potenciálně nebezpečném komunikačním kanále, může mít jeho počínání velmi negativní dopady (například odposlechnutí citlivých údajů a jejich následné zneužití).

Z těchto důvodů jsem se rozhodl zpracovat bakalářskou práci na téma praktického nasazení digitálních certifikátů, které nám – komunikujícím – mají pomoci zprostředkovat bezpečný komunikační kanál napříč jinak veřejným médiem Internetu.

V této práci se proto zabývám popisem a implementací vybraných útoků na šifru RSA, která je běžně používána v síti Internet. Současně se tím pádem zabývám testováním kvality parametrů digitálních certifikátů, neboť právě špatně zvolené parametry, jako například moduly a tajné exponenty, rozhodují o síle šifry RSA a její schopnosti odolat známým útokům. Z důvodu velkého rozsahu sítě Internet se v práci omezím na geografickou oblast České republiky.

Cílem této práce je analyzovat aktuální stav digitálních certifikátů užívaných webovými servery na českém Internetu, a to pomocí testování kvality zvolených parametrů. Testování bude probíhat formou implementace vybraných známých útoků na šifru RSA, které budou v práci popsány. Z toho vyplývá

omezení se na digitální certifikáty s veřejným klíčem na bázi RSA, neboť je obecnou znalostí, že v četnosti užití zdaleka předčí certifikáty s veřejnými klíči tvořenými algoritmy eliptických křivek (a případnou kombinací RSA/ECDSA *dual stack*)[1]. Každý testovaný certifikát bude podroben implementovaným útokům s omezenou časovou dotací a výsledky v závěru práce sumarizovány. Společně s výsledky testování kvality parametrů bude uvedena i souhrnná četnost jednotlivých parametrů napříč zvoleným vzorkem webových serverů českého Internetu.

Kapitola 1

Analýza současné situace

Tato kapitola popisuje analýzu stavu a návrh řešení pro provedení studie.

Věnuji se rozboru současných řešení pro pasivní zajištění bezpečnosti při styku s nevalidním digitálním certifikátem. Dále debatuji praktická omezení a limitace zdrojů dat pro statistiku a výsledků studie. Následně popisuji šifru RSA[1] a vybrané známé útoky na šifru RSA, jejich asymptotické složitosti a důvody jejich začlenění (nebo naopak nezačlenění) do implementační části bakalářské práce.

1.1 Současná řešení

Nevalidní certifikáty a jejich špatné zvolené parametry mohou být vhodným zdrojem útoků cílených na uživatele bez povědomí o potenciálním nebezpečí, kterému se jejich využití pro komunikaci vystavují. Útočníkovi při úspěchu zpřístupňují celou paletu dalších možných nekalých taktik. Efektivní preventivní opatření lze provést jak na pasivní straně uživatelů, tak na aktivní straně správců webových serverů.

1.1.1 Systematické vzdělávání uživatelů

Systematické vzdělávání uživatelů o potenciálním nebezpečí není metodou pro aktivní snižování výskytu podvržených certifikátů. Jedná se ovšem o velmi efektivní metodu minimalizace napáchaných škod, dojde-li k podobnému útoku cíleného na uživatele.

Útoky s podvrženými certifikáty, kterým se uživatel rozhodne důvěřovat mohou mít dalekosáhlé dopady nejen ve firemním prostředí (citlivá firemní data), ale i u jednotlivců (například internetové bankovnictví). Ve firemním prostředí lze nejslabší bezpečnostní článek – uživatele – na takovou situaci připravit (například formou bezpečnostního školení). Poučený zaměstnanec,

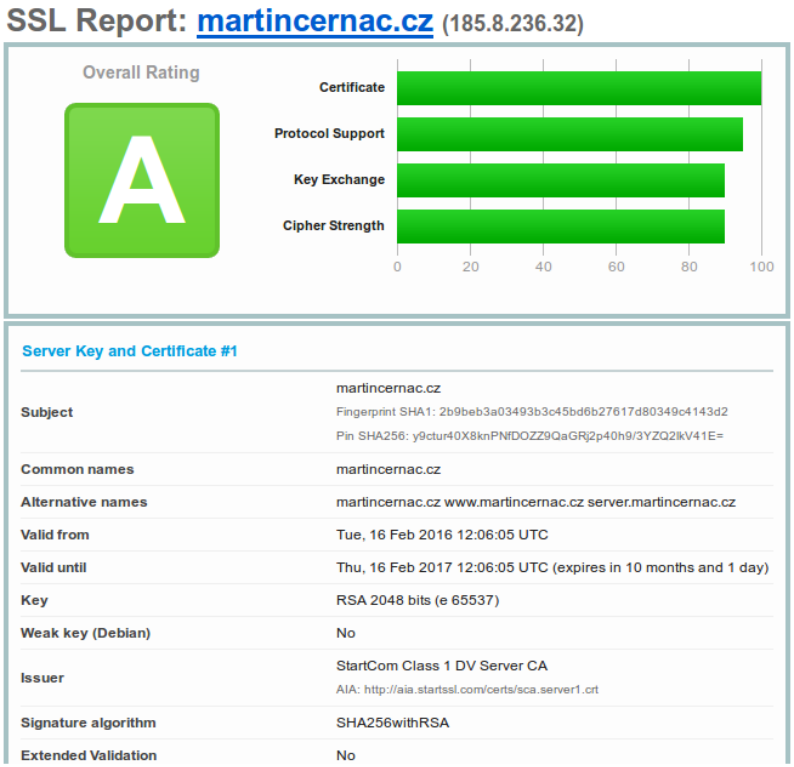
[1] Předpokládá se čtenářova znalost vybraných matematických principů, které jsou pro přehlednost uvedeny v příloze B.

který získá povědomí o možném ohrožení jeho komunikace a následně se setká s nevalidním certifikátem v prostředí, ve kterém by tomu docházet nemělo (vnitřní firemní síť, masivní globálně poskytované služby – sociální sítě, vyhledávání, e-mailové rozhraní), bude pravděpodobněji reagovat správně – takovou komunikaci přeruší, ideálně nahlásí svému správci sítě, který podnikne další kroky k nápravě.

1.1.2 Nástroje pro kontrolu

V současné době jsou k dispozici nástroje, které jsou na pokyn uživatele schopné během několika minut jednorázově zkontrolovat stav daného webového serveru a uživateli poskytnout velmi přehledný a detailní výpis zjištěných závad. Jedná se ovšem o nástroje určené spíše pro správce testovaných serverů a nejedná se o aktivní a pravidelnou kontrolu.

Mezi populární nástroje se řadí například SSL Server Test[2] (na obrázku 1.1) z nekomerčního výzkumu SSL Labs společnosti Qualys, Inc, nebo nástroj SSL Checker[3] provozovaný webovou službou SSL Shopper.



Obrázek 1.1: Snímek obrazovky vyhodnocení nasazení digitálního certifikátu nástrojem SSL Checker.

1.1.3 Automatizace procesů nasazení a obnovy digitálních certifikátů

Digitální certifikát nás má, mimo jiné, přesvědčit o tom, že jeho držitel je skutečně ten, za koho se vydává. Ne vždy je však nutné mít stejný stupeň důvěry v předložený certifikát. Potřebná důvěra k návštěvě portálu s informacemi o počasí zpravidla není zdaleka na stejné úrovni, jako potřebná důvěra před přihlášením se do internetového bankovnictví. Z toho důvodu nabízejí certifikační autority hned několik „úrovní“ ověření identity (a tím pádem výsledné důvěryhodnosti) před vystavením certifikátu, přičemž důkladnější ověření je spojeno se znatelně vyšším administračním poplatkem certifikační autority. Podoba a důslednost ověření je tedy vázána na typ vydaného certifikátu. Certifikáty lze proto kvalitativně rozdělit dle stupně ověření identity žadatele do těchto tří skupin:

DV certifikáty *Domain validated* certifikáty jsou nejběžnější formou certifikátů [4]. Jejich účelem je ověřit, že žadatel má kontrolu nad doménou, pro níž si vyžádal certifikát. Jedná se pouze o minimální ověření, obvykle za pomoci autorizačního e-mailu zaslaného na jednu z možných e-mailových adres získaných z WHOIS[2] záznamu domény, pro kterou žadatel požaduje vystavení certifikátu. Alternativní cestou může být umístění souboru vygenerovaného certifikačního autoritou se specifickým názvem a obsahem na specifikované místo dostupné z domény žadatele. Splnění některé z těchto metod dokládá, že doména je ve vlastnictví žadatele a celý proces je obvykle automatizovaný.

OV certifikáty *Organization validated* certifikáty vyžadují důkladnější ověření identity žadatele. Certifikační autorita podnikne dodatečné kroky k ověření žadatele-organizace, což vyústí ve věrohodnější certifikát než *DV* certifikát, neboť součástí certifikátu je i název organizace. Tento typ certifikátů je často nasazován společnostmi a vládními organizacemi.

EV certifikáty Vydání *Extended validation* certifikátu vyžaduje rozsáhlé ověření identity certifikační autoritou a třetím stranám proto při komunikaci poskytuje velmi vysokou úroveň důvěry. Moderní internetové prohlížeče[3] využívají informace *EV* certifikátů pro vizuální notifikaci uživatelů o důvěryhodnosti užitého certifikátu (nejčastěji formou zobrazení názvu organizace držitele na zeleném podkladu v oblasti adresního řádku).

1.1.4 Shrnutí

Při úniku citlivých dat je obvykle nejslabším článkem člověk. Právě na *člověka* by se mělo cílit při zavádění vyšší podnikové bezpečnosti. Bezpečnostní školení

[2] Označení pro databázi evidující údaje majitelů domén a IP adres.

[3] Testováno v prohlížečích Mozilla Firefox 45, Google Chrome 51 a Internet Explorer 10.

je ideální formou, která může výrazně přispět k správné reakci uživatelů.

Systémoví administrátoři mají k dispozici nástroje, které usnadňují kontrolu serverů a tím pádem posilují prevenci nasazení nevalidních certifikátů a vytváření potenciálního nebezpečí s nimi spojeného.

Z popisu jednotlivých stupňů důvěryhodnosti certifikátů plyne, že ne všechny mohou být vydávány strojově a bez přímé lidské účasti (zaměstnanců certifikační autority, zaměstnanců ověřované společnosti a dalších). Jediným typem certifikátů, který lze strojovou formou vydat je *Domain validated* certifikát. Tohoto faktu využívá nově zavedená certifikační autorita *Let's encrypt*, která jako první nabízí vydávání *DV* certifikátů zcela zdarma, strojově a automaticky – včetně softwaru pro nasazení vydaných certifikátů. Tato forma správy nasazených certifikátů garantuje jejich aktuálnost a tím pádem eliminuje problémy vzniklé neošetřenou expirací certifikátu. Neeliminuje však nízkou důvěryhodnost spojenou s *Domain validated* certifikáty (chybějící ověření identity žadatele).

1.2 Omezení a limitace statistiky

Ve fázi analýzy a návrhu řešení bylo nutné vymezit sledovanou podmnožinu webových serverů – tedy *český* Internet. Omezení pro statistiku představují zejména špatně dostupné či neexistující zdroje dat pro takovou podmnožinu. Zřejmým omezením je rychlé zastarání naměřených údajů a tím pádem neaktuálnost statistiky.

1.2.1 Český Internet

V rámci této práce se pojmem *český* Internet míní webové servery s DNS[4] záznamem v registru doménových jmen .CZ provozovaným zájmovým sdružením právnických osob CZ.NIC. Toto omezení jsem zvolil z důvodů neexistence dostupného seznamu všech webových serverů s českým obsahem. Rozhodl jsem se též zanedbat české webové servery provozované s doménovým jménem, které CZ.NIC neeviduje (například generická doménová jména *.com*, *.net* a další). Nevěnuji se doménám jiného než druhého řádu (například *fit.cvut.cz*) z důvodu neexistence dostupného zdroje dat pro provedení statistiky na doménách třetího (a vyššího) řádu.

Během analýzy jsem ovšem zjistil, že ani získání domén druhého řádu zóny .CZ není triviální, neboť zájmové sdružení právnických osob CZ.NIC mi nebylo ochotno taková data poskytnout. Politiku zpřístupnění údajů si každá ccTLD[5] autorita vytváří sama, ovšem postoj CZ.NIC, z. s. p. o. není běžnou praxí. Příkladem může být slovenský protějšek, společnost SK-NIC a.s., která každý den zveřejňuje celý obsah zóny .SK volně k nahlédnutí [5]. Obdobně

[4]Domain Name System

[5]Country code top-level domain

s daty zóny nakládá společnost Verisign, která zpřístupňuje zónové soubory generických domén[6].

Pokusil jsem se se sdružením CZ.NIC vyjednat vydání reprezentativního vzorku obsahu zóny .CZ, přičemž mi sdružení předložilo smluvní nabídku jednoho tisíce záznamů (domén), ovšem s desetiletou garancí mlčenlivosti z mé strany. Pokud by poskytnutá data během této doby unikla, vznikla by mým porušením smluvních podmínek soudně vymahatelná pokuta čítající až 1 milion korun, kterou považuji za nepřiměřenou, vzhledem k velikosti vzorku 1000 domén ze zóny obsahující přes 1,25 milionu domén[6] – tedy představující přibližně $0,8‰$ obsahu zóny. Z těchto důvodů (nereprezentativně velký vzorek, nepřiměřená smluvní pokuta) jsem se rozhodl smluvní nabídku odmítnout.

Oslovil jsem též velké české internetové vyhledávače (firmy Seznam, a.s. a Economia, a.s. provozující webové portály Centrum a Atlas), odpovědi se mi ovšem nedostalo. Byl jsem tedy donucen použít vlastních zdrojů pro výběr testovaného vzorku webových serverů. Potřebná data jsem získal z projektu Rozhled.eu, v době získání dat evidujícího přes 690 tisíc domén, tedy více než polovinu obsahu zóny .CZ.

1.2.2 CloudFlare a podobné služby

Jedním z útoků na dostupnost webových serverů jsou útoky typu DoS (případně DDoS), které daný webový server zahltí velkým počtem požadavků na obsah, který webový server nestihne obsluhovat a pro nezávislého pozorovatele se stane „nedostupným“, neboť nevyřizuje požadavky uživatele. Společnost *CloudFlare* vyvinula systém distribuované sítě, vstupující do komunikace mezi uživatelem a serverem – nápor vytvořený útokem DDoS se tedy nedostane až k webovému serveru, ale bude odfiltrován službou firmy *CloudFlare*.

Tento typ služeb má bohužel pro tuto statistiku fatální následek nedostupnosti *opravdového* digitálního certifikátu daného webového serveru, neboť z pohledu uživatele je druhou stranou právě jeden ze vstupních bodů služby *CloudFlare*. Předpokládal jsem tedy zašumění získaných dat certifikáty služeb nabízejících „ochranu“ před zahlcujícími útoky. Rozhodl jsem se je ale ve výsledných datech ponechat, neboť jejich četnost také nese jistou informační hodnotu.

1.3 Šifra RSA a digitální certifikáty

V této sekci se zaměřím na popis šifry RSA a na vymezení několika pojmů v oblasti digitálních certifikátů. Předpokládá se čtenářova znalost vybraných matematických principů, zejména modulární aritmetiky. Pro přehlednost jsou použité matematické principy popsány v příloze B.

[6] Údaj z veřejných statistik CZ.NIC[7].

1.3.1 Šifra RSA

Šifra RSA[8][7] je typickým zástupcem asymetrické šifry[8] – šifry, kde se šifrovací klíč liší od dešifrovacího klíče. Základním předpokladem bezpečnosti šifry RSA je praktická obtížnost faktorizace součinu dvou velkých prvočísel, tedy obtížnost rozložit složené číslo $n = p \cdot q$ na dva činitele p, q. Oproti tomu vynásobení dvou velkých čísel $p \cdot q = n$ je elementární úloha. Šifra RSA umožňuje použití jak pro šifrování zpráv 1.3.3, tak pro jejich podepisování 1.3.4.

1.3.2 Generování RSA klíče

Algoritmus 1. Algoritmus pro výběr RSA klíče

Výstup: modul n, veřejný klíč e a tajný klíč d;
vyber dvě různá prvočísla p, q nezávislým náhodným výběrem;
vypočítej modul n, kde $n = p \cdot q$;
vypočítej
$\varphi(n) = \varphi(p \cdot q) = \varphi(p) \cdot \varphi(q) = (p-1) \cdot (q-1) = n - (p + q - 1)$, tato hodnota je tajná;
vyber veřejný exponent e tak, že $1 < e < \varphi(n)$ a současně
$gcd(e, \varphi(n)) = 1$;
vypočítej soukromý exponent d tak, že $d \equiv e^{-1} \pmod{\varphi(n)}$, tato hodnota je tajná;

Veřejný klíč šifry představuje dvojice (n, e), kde n se nazývá modul a e veřejný[9] exponent. Soukromý klíč šifry představuje dvojice (n, d), kde n se nazývá modul a d soukromý[10] exponent. Ostatní hodnoty použité v běhu algoritmu $(\varphi(n), p, q)$ nemají další použití a nesmí být zachovány (s jejich pomocí je totiž možné vypočítat soukromý exponent d).

Praktické generování klíče je doplněno sadou doporučení a upozornění, omezující volnost výběru parametrů:

- Prvočísla p, q by měla být volena tak, aby byla přibližně stejně dlouhá. Navíc by se měla lišit alespoň v jenom z prvních 100 bitů [9].
- V současnosti nejběžnější veřejný exponent e má hodnotu $65537_{10} = 2^{16} + 1$, tedy 10000000000000001_2 [1]. Takto zvolený exponent zajišťuje efektivní šifrování[10]).
- Čísla $p - 1, q - 1$ by neměla sdílet prvočíselné faktory – ideálně pouze jeden prvočíselný faktor ($gcd(p - 1, q - 1) = 2$), zabraňuje Pollardovu $p - 1$ útoku 1.4.4.

[7] Iniciály příjmení autorů – Ron **R**ivest, Adi **S**hamir a Leonard **A**dleman.
[8] neboli šifry s veřejným klíčem
[9] neboli šifrovací
[10] neboli tajný

1.3.3 Šifrování a dešifrování

Zprávu m pomocí RSA zašifrujeme a dešifrujeme následujícími algoritmy:

Algoritmus 2. Algoritmus pro šifrování zprávy pomocí RSA
Výstup: zpráva m, veřejný klíč (n, e);
Výstup: zašifrovaná zpráva c;
ověř $0 \leq m < n$ a $gcd(m, n) = 1$;
vypočítej zašifrovanou zprávu c, kde $c \equiv m^e \pmod{n}$;

Algoritmus 3. Algoritmus pro dešifrování zprávy pomocí RSA
Výstup: zašifrovaná zpráva c, soukromý klíč (n, d);
Výstup: původní zpráva m;
vypočítej původní zprávu m, kde $c^d \equiv (m^e)^d \equiv m \pmod{n}$;

1.3.4 Podepisování

Zprávu m lze digitálně podepsat za pomoci šifry RSA. Operace je analogická jako u šifrování a dešifrování dat. Důležitou změnou je ovšem záměna klíčů pro jednotlivé operace. Chceme-li tedy vytvořit digitální podpis zprávy m, vytvoříme její otisk pomocí definované metody (běžně používané jsou hashovací funkce rodiny *SHA*[11]) a ten zašifrujeme **soukromým klíčem**. Protistrana využije našeho známého veřejného klíče k dešifrování a porovnání dešifrované hodnoty s výstupem stejné hashovací funkce aplikované na zprávu m.

1.3.5 Schéma doplnění

Schéma doplnění, neboli *Padding scheme* je způsob rozšíření zprávy tak, aby mohla být následně bezpečně zpracována. Zatímco u blokových šifer je hlavním důvodem pro použití schémata doplnění nutnost zaplnit zprávou délku bloku, v případě RSA chrání schéma doplnění instanci šifry před útoky na předvídatelnou strukturu zpráv. Šifra RSA je deterministická, proto lze na zprávy bez použití schémata doplnění použít útok *Chosen plaintext attack*. Útočník, který zachytí zašifrovanou zprávu a má znalost náležícího veřejného klíče, má možnost opakovaně šifrovat různé zprávy a porovnávat výsledek se zachycenou zprávou. Schémata doplnění jako OAEP[12][11] mimo jiné zavádí sémantickou bezpečnost, kdy útočník nedokáže rozeznat dvě zašifrované zprávy, i když zná jejich obsah. Těmto útokům schéma doplnění OAEP zabraňuje mimo jiné náhodnou částí doplnění.

[11] Secure Hash Algorithm
[12] Optimal asymmetric encryption padding

1.3.6 Digitální certifikát a jeho parametry

Digitálním certifikátem se v této práci rozumí struktura sestávající se nejméně z těchto tří součástí (parametrů):

- veřejný klíč žadatele/držitele certifikátu
- doba platnosti certifikátu
- identifikační údaje držitele certifikátu

Tato struktura je podepsána soukromým klíčem vydavatele (certifikační autorita) a tím pádem lze ověřit obsah certifikátu pomocí známého veřejného klíče certifikační autority.

1.3.6.1 Validita certifikátu

Validní certifikát je takový certifikát, který splňuje následující podmínky:

- certifikát již vstoupil v platnost, ale ještě nevypršel (parametr *doba platnosti certifikátu*)
- certifikát náleží subjektu, který se s ním prezentuje (parametr *identifikační údaje držitele certifikátu*)
- vydavatel certifikátu je námi považován za věrohodného

Při nesplnění libovolné podmínky bude certifikát v závěrečné statistice považován za nevalidní.

1.3.7 Kořenový certifikát

Z definice podpisu a digitálního certifikátu vyplývá, že nic nebrání podpisu digitálního certifikátu stejným veřejným klíčem, který je v něm obsažen. Takový certifikát se nazývá *self-signed*. Principu *self-signed* certifikátů využívají kořenové certifikáty certifikačních autorit. Jediný způsob, jak důvěřovat *self-signed* certifikátu, je považovat ho za důvěryhodný (v řetězu důvěry nelze postoupit dále).

Soubor věrohodných vydavatelů certifikátů (certifikačních autorit) a jejich kořenových certifikátů byl pro maximální přiblížení se uživatelské interakci převzat z webového prohlížeče, konkrétně Mozilla Firefox verze 45.

1.4 Útoky na RSA

V této podkapitole rozebírám vybrané útoky na šifru RSA, jejich možné použití pro testování kvality parametrů digitálních certifikátů. Věnuji se triviálním (1.4.1) i sofistikovaným (1.4.3) útokům a popisuji jejich reálné uplatnění a možnosti vyvarování se napadení.

1.4.1 Útok hrubou silou

Útok hrubou silou je triviální útok, který se snaží provést faktorizaci modulu. Naivní verze útoku se sestává z testování dělitelnosti modulu napříč množinou všech prvočísel menších nebo rovných hodnotě $\sqrt{n}$. V případě, že v průběhu útoku bude nalezeno prvočíslo p, které dělí beze zbytku modul n, instance šifry byla prolomena, neboť útočník nyní disponuje znalostí prvočísel p a $q = \frac{n}{p}$.

Algoritmus 4. Naivní faktorizace
Vstup: přirozené číslo $n > 2$;
Výstup: netriviální faktor n, nebo neúspěch (n je prvočíslo);
$a = 2$;
$b = n \pmod a$;
while $b \neq 0$ **do**
  $a =$ další prvočíslo;
  if $a > \sqrt{n}$ **then**
    return *neúspěch*;
  end
  $b = n \pmod a$;
end
return a

Praktickou limitací útoku hrubou silou je časová náročnost a nedostatek prostředků pro dokončení útoku za dobu, kdy jsou zašifrovaná data pro útočníka stále ještě informačně hodnotná. Z tohoto důvodu jsem se rozhodl útok hrubou silou neimplementovat a časovou dotaci, kterou by obdržel, využít na více sofistikované útoky na šifru RSA.

Příklad pro triviálnost útoku neuvádím.

1.4.1.1 Složitost

Rychlost algoritmu naivní faktorizace silně závisí na faktorech a velikosti vstupního čísla. Zatímco netriviální faktor velkého čísla ve tvaru 2^{128} by algoritmus nalezl při první iteraci, doba faktorizace prvočísla se stejným množstvím cifer by trvala velice dlouho, neboť algoritmus by iteroval přes všechna prvočísla menší než $\sqrt{n}$, tedy $\pi(\sqrt{n})$ iterací. Složitost takového algoritmu tedy je $O(\pi(\sqrt{n}))$, což lze aproximovat 4 do formy:

$$O\left(\frac{2\sqrt{n}}{\ln n}\right)$$

Ovšem pouze za předpokladu, že máme k dispozici seznam prvočísel menších než $\sqrt{n}$, nebo jsme schopni je v konstantním čase generovat. V opačném

případě se musíme spokojit s dělením lichými čísly a složitost tedy vzroste na:

$$O\left(\frac{\sqrt{n}}{2}\right)$$

1.4.2 Útok na příliš blízká prvočísla p a q

V typickém kryptosystému RSA, kde modul n je k-bitové číslo, tvořeno dvěma $\frac{k}{2}$-bitovými prvočísly p, q může dojít k oslabení šifry, pokud rozdíl $\Delta = |p - q|$ dvou prvočísel p a q je velmi malý. Do jisté míry se jedná o obdobou útoku hrubou silou, ovšem rozsah možných kandidátů na prvočísla p a q se podstatně zúží, zahájíme-li prohledávání v bodě $\lceil\sqrt{n}\rceil$.

Samotný útok v principu využívá Fermatovy faktorizace, pomocí které lze se znalostí předpokládaného rozsahu prvočísel faktorizovat modul téměř okamžitě za předpokladu, že $\Delta < \sqrt[4]{n}$[12]. Ve spojení s Newtonovou interpolací je možné provést faktorizace modulů, kde $\Delta < \sqrt[3]{n}$ v polynomiálním čase[13].

Z těchto důvodů zavádí standardy FIPS 186-3[14] a ANSI X9.31[9] některá opatření – například ANSI X9.31 v sekci 4.1.2 vyžaduje, aby prvočísla p a q byla volena tak, aby se lišila nejméně v jednom z jejich prvních 100 bitů.

Doporučení FIPS o délce modulu[15] garantuje velice velkou množinu možných prvočísel pro tvorbu modulu. Pravděpodobnost zvolení velmi blízkých prvočísel p a q je proto při nezávislém náhodném výběru velice malá.

1.4.2.1 Fermatova faktorizace

V 17. století Pierre de Fermat dokázal, že každé přirozené liché číslo n lze zapsat ve tvaru rozdílu čtverců, tedy $n = a^2 - b^2 = (a - b)(a + b)$, kde a, b odpovídají rozkladu n na dva faktory. Pro naše potřeby faktorizace modulu jsou přínosná pouze taková a, b, pro která platí $a \neq b$ a současně $a, b > 1$ (tyto podmínky se ovšem obecně nevztahují na Fermatovu faktorizační metodu).

Důkaz. Nechť n je liché přirozené číslo. $n = a \cdot b$, kde a, b jsou přirozená čísla taková, že $a \leq b$ a a, b jsou lichá, neboť n je liché. Nechť $c = \frac{a+b}{2}, d = \frac{a-b}{2}$ jsou přirozená čísla. Potom lze zapsat $a = c - d, b = c + d$, tedy

$$n = a \cdot b = (\frac{a+b}{2} - \frac{a-b}{2})(\frac{a+b}{2} + \frac{a-b}{2}) = (c-d)(c+d) = c^2 - d^2$$

□

Algoritmus 5. Fermatova faktorizace

Vstup: liché složené číslo $n > 0$;
Výstup: rozklad čísla n na dva netriviální činitele $(p; q)$, $p \cdot q = n$;
$a = \lceil \sqrt{n} \rceil$;
$b = \sqrt{a^2 - n}$;
while b *není celé číslo* **do**
  $a = a + 1$;
  $b = \sqrt{a^2 - n}$;
end
return $(a - b; a + b)$

Příklad V příkladu budeme aplikovat popsanou metodu na problém nalezení prvočísel p, q RSA modulu $n = 407$.

Tabulka 1.1: Ukázka algoritmu – Fermatova faktorizace modulu $n = 407$.

č. iterace	a	b
- (inicializace)	$a = \lceil \sqrt{407} \rceil = 21$	$\sqrt{21^2 - 407} = \sqrt{34} \approx 5,83$
1	$a = 21 + 1 = 22$	$\sqrt{22^2 - 407} = \sqrt{77} \approx 8,77$
2	$a = 22 + 1 = 23$	$\sqrt{23^2 - 407} = \sqrt{122} \approx 11,05$
3	$a = 23 + 1 = 24$	$\sqrt{24^2 - 407} = \sqrt{169} = 13$
$a = 24, b = 13$, $(a - b; a + b) = (24 - 13; 24 + 13) = (\mathbf{11}; \mathbf{37})$		

Na příkladu znázorněném tabulkou 1.1 jsme úspěšně faktorizovali modul $n = 407$ za pomoci 3 iterací Fermatovy faktorizační metody – nalezli jsme rozklad $(24 - 13)(24 + 13) = 11 \cdot 37 = 407$.

Algoritmus Fermatovy faktorizace ve formě útoku na blízká prvočísla p, q jsem se rozhodl implementovat, ačkoli pravděpodobnost výběru *velmi* blízkých prvočísel p, q je při nezávislém náhodném výběru k-bitových prvočísel zanedbatelná.

1.4.2.2 Složitost

Aplikujeme-li uvedenou metodu na modul, kde $\Delta < \sqrt[4]{n}$, de Weger dokázal[12], že složitost Feramtovy faktorizace je dána poměrem

$$O\left(\frac{\Delta^2}{4\sqrt[4]{n}}\right)$$

kde $\Delta = |p - q|$.

1.4.3 Útok na malý tajný exponent

Útok využívající malý tajný exponent – *Wienerův útok* – je sofistikovaný útok, jehož cílem je získání tajného exponentu d za předpokladu, že d je dostatečně

malé číslo. Volba malého tajného exponentu skýtá velkou výhodu pro výpočetně slabá zařízení – například pro čipové karty, kde dochází ke značnému urychlení procesu dešifrování přijatých zpráv, typicky pro komunikaci s terminálem, disponujícím malým veřejným exponentem, což urychluje šifrování zpráv na čipové kartě. Wiener[16] v roce 1989 poukázal na některá úskalí volby malých tajných exponentů, která mohou vést k efektivnímu odhalení tajného exponentu d.

Wiener ve své práci současně zmiňuje možnosti omezení rozsahu útoku, například zvolením prvočísel p, q tak, že $gcd(p-1, q-1)$ je vysoké, nebo volbou vyššího veřejného exponentu[13] e. Zvýšíme-li veřejný exponent e za hranici $n^{\frac{3}{2}}$, neexistuje záruka, že algoritmus řetězových zlomků nalezne řešení pro libovolně malé d. Dalším běžně užívaným mechanismem pro urychlení dešifrování je využití Čínské věty o zbytcích.

Wienerův útok pro úspěšné odhalení d vyžaduje několik okolností, zejména: $d < \frac{1}{c+1}\sqrt[4]{n}$, kde c (typicky uváděno pro $c = 2$) je nejmenší přirozené číslo, pro které platí $p < q < c \cdot p$. Například pro 2048 bitový modul ($c = 2$) útok jistě selže, bude-li mít číslo d nejméně 512 bitů.

1.4.3.1 Wienerův útok

Věta 1. [10] Nechť $n = pq$, kde $q < p < 2q$. Nechť $d < \frac{1}{3}\sqrt[4]{n}$. Se znalostí modulu n a veřejného exponentu e, kde $ed \equiv 1 \pmod{\varphi(n)}$, je možno efektivně získat tajný exponent d.

Příklad V příkladu budeme aplikovat popsanou metodu na problém nalezení soukromého exponentu se znalostí veřejného klíče $(n, e) = (1806047, 78407)$ za pomoci Wienerova útoku:

Prvním krokem je výpočet řetězového zlomku výrazu $\frac{78407}{1806047}$. Výpočet provedeme použitím Euklidova algoritmu 9:

$$\begin{aligned}
78407 &= \mathbf{0} \cdot 1806047 + 78407 \\
1806047 &= \mathbf{23} \cdot 78407 + 2686 \\
78407 &= \mathbf{29} \cdot 2686 + 513 \\
2686 &= \mathbf{5} \cdot 513 + 121 \\
513 &= \mathbf{4} \cdot 121 + 29 \\
121 &= \mathbf{4} \cdot 29 + 5 \\
29 &= \mathbf{5} \cdot 5 + 4 \\
5 &= \mathbf{1} \cdot 4 + 1 \\
4 &= \mathbf{4} \cdot 1 + 0
\end{aligned}$$

[13] Což s sebou ovšem nese negativa ve formě vyšší výpočetní náročnosti procesu dešifrování.

Algoritmus 6. Wienerův útok (Algoritmus řetězových zlomků)
Vstup: RSA veřejný klíč (n,e);
Výstup: soukromý exponent d, nebo neúspěch;

```
k = Integer[];                                   // koeficienty
z = Integer[];                                   // zbytky
j = Integer[];                                   // jmenovatele konvergent
i = 0;                                           // iterace, krok
while true do
    if i = 0 then
        k_0 = ⌊e/n⌋;
        z_0 = e (mod n);
        j_0 = 1;
    else if i = 1 then
        k_1 = ⌊n/z_0⌋;
        z_1 = n (mod z_0);
        j_1 = k_1;
    else
        if z_{i-1} = 0 then
            break;
        k_i = ⌊z_{i-2}/z_{i-1}⌋;
        z_i = z_{i-2} (mod z_{i-1});
        j_i = k_i · k_{i-1} + k_{i-2};
    i = i + 1;
end
// Nyní se pokusíme najít vhodného kandidáta pro d
c ≡ 3^e (mod n);     // Testovací zpráva 3, zašifrovaná pomocí e
forall d ∈ {j_0, j_1, ..., j_i} do
    if c^d ≡ 3 (mod n) then
        return d
end
return neúspěch
```

Nalezli jsme řetězový zlomek výrazu $\frac{e}{n} = [0; 23; 29; 5; 4; 4; 5; 1; 4]$.
Následujícím krokem je výpočet všech netriviálních konvergent výrazu $\frac{e}{n}$[14]:

$$
\begin{aligned}
[0; 23] &= 0 + \frac{1}{23} = \frac{1}{\mathbf{23}} \\
[0; 23; 29] &= 0 + \frac{1}{23 + \frac{1}{29}} = \frac{1}{\mathbf{668}} \\
[0; 23; 29; 5] &= 0 + \frac{1}{23 + \frac{1}{29 + \frac{1}{5}}} = \frac{1}{\mathbf{3363}} \\
[0; 23; 29; 5; 4] &= 0 + \frac{1}{23 + \frac{1}{29 + \frac{1}{5 + \frac{1}{4}}}} = \frac{1}{\mathbf{14120}} \\
[0; 23; 29; 5; 4; 4] &= 0 + \frac{1}{23 + \frac{1}{29 + \frac{1}{5 + \frac{1}{4 + \frac{1}{4}}}}} = \frac{1}{\mathbf{59843}} \\
[0; 23; 29; 5; 4; 4; 5] &= 0 + \frac{1}{23 + \frac{1}{29 + \frac{1}{5 + \frac{1}{4 + \frac{1}{4 + \frac{1}{5}}}}}} = \frac{1}{\mathbf{313335}} \\
[0; 23; 29; 5; 4; 4; 5; 1] &= 0 + \frac{1}{23 + \frac{1}{29 + \frac{1}{5 + \frac{1}{4 + \frac{1}{4 + \frac{1}{5 + 1}}}}}} = \frac{1}{\mathbf{373178}} \\
[0; 23; 29; 5; 4; 4; 5; 1; 4] &= 0 + \frac{1}{23 + \frac{1}{29 + \frac{1}{5 + \frac{1}{4 + \frac{1}{4 + \frac{1}{5 + \frac{1}{+ \frac{1}{4}}}}}}}} = \frac{1}{\mathbf{1806047}}
\end{aligned}
$$

Množina jmenovatelů výše uvedených konvergent

$$\{23; 668; 3363; 14120; 59843; 313335; 373178; 1806047\}$$

představuje množinu kandidátů na soukromý exponent d. Pro nalezení d (je-li obsaženo v množině kandidátů) použijeme samotnou šifru RSA – jinými slovy zašifrujeme známou zprávu za pomoci veřejného exponentu a poté se ji pokusíme dešifrovat za použití kandidáta na soukromý exponent. Dojde-li k úspěšnému dešifrování zprávy, nalezli jsme soukromý exponent d a tím pádem prolomili instanci šifry RSA. V případě, že žádný kandidát nedokázal dešifrovat testovací zprávu, útok selhal – možné příčiny jsou obsaženy v sekci 1.4.3.

[14] V implementaci Wienerova útoku se provádí kroky Euklidova algoritmu a výpočtu netriviálních konvergent současně.

V našem případě kandidát $d = 23$ úspěšně dešifroval testovací zprávu. Útok tím pádem uspěl, neboť jsme nalezli hodnotu soukromého exponentu d.

Algoritmus řetězových zlomků (Wienerův útok) budu dozajista implementovat, neboť se jedná o efektivní útok, který nevyžaduje vysokou časovou dotaci pro prohledání všech možných kandidátů na soukromý exponent d.

1.4.3.2 Složitost

Pro útok je třeba vypočítat všechny netriviální konvergenty, kterých je řádově $O(\log n)$, přičemž každý kandidát je testován v polylogaritmickém čase. Výsledná složitost je tedy polylogaritmická $O(poly(\log n))$ [16].

1.4.4 Pollardova $p-1$ metoda

Pollardova $p-1$ metoda je algoritmus pro faktorizaci čísel popsaný Johnem Pollardem v roce 1974. Stejně jako v případě Wienerova útoku 1.4.3 se jedná o specifický algoritmus[15], který při faktorizaci čísla n využívá hladkého čísla $p-1$, respektive $q-1$. Algoritmus může být velice efektivní za předpokladu, že číslo $p-1$ je B-hladké pro nějaké dostatečně malé B. V opačném případě, kdy rozklad čísla $p-1$ obsahuje pouze dva prvočinitele (číslo 2 a vysoké prvočíslo), překračuje doba běhu algoritmus útoku hrubou silou 1.4.1 [17, 1.1].

Algoritmus vychází z Malé Fermatovy věty $a^{p-1} \equiv 1 \pmod{p}$, kde p je prvočíslo a $1 < a < p$. Předpokládáme, že existuje L takové, že $p-1|L$, tedy existuje i takové, že $L = i \cdot (p-1)$. Potom můžeme využít zmíněné Malé Fermatovy věty:

$$a^L = a^{i\cdot(p-1)} = \left(a^{p-1}\right)^i \equiv 1^i = 1 \pmod{p}$$

Jinými slovy, $p|a^L - 1$, tedy nutně $\gcd(a^L - 1, n) \in n, p$. U čísla L je vhodné, aby mělo co nejvíce různých faktorů. Vhodnou volbou L je proto faktoriál (odpovídající mezi B-hladkého čísla).

Praktické nasazení Pollardovy $p-1$ metody spočívá ve spoléhání se na náhodu: úmyslně fixně zvolíme $a = 2$ a B relativně vysoké (10^5 až 10^6 [17, 1.2]) a spustíme algoritmus. Pokud byly zvoleny nevhodná prvočísla p, resp. q taková, že $p-1$ je B-hladké, algoritmus velmi efektivně tuto skutečnost odhalí. Na tento poznatek je tedy *doporučeno* dbát při generování RSA klíčů1.3.2.

Zamýšlená implementace algoritmu zahrnuje dynamickou volbu hodnoty a a meze pro hladkost čísla $p-1$, která lépe využije fixní časovou dotaci.

Příklad Pro ilustraci jsem náhodným nezávislým výběrem z 547-hladkých čísel vygenerovaných přiloženým programem vybral dvě 40-bitová čísla $p = 741030129527, q = 756077174459$. A porovnal faktorizaci pomocí algoritmu

[15] anglicky special-purpose algorithm

Algoritmus 7. Pollardova $p-1$ metoda
Vstup: Složené číslo $n, n = p \cdot q$, kde p, q jsou různá prvočísla;
Výstup: netriviální faktor p složeného čísla n;

```
a = 2;
b = 2;                                          // faktoriál v mocnině
while true do
    d = gcd(a^{b!} − 1, n) = gcd((a^{b!} mod n) − 1, n);
    if d = 1 then
        b = b + 1;
    end
    else if d = n then
        b = 2;
        a = a + 1;
        if gcd(a, n) ≠ 1 then
            return a
        end
    end
    else
        return d;
    end
end
```

Fermatovy faktorizace 1.4.2.1 s uvedeným algoritmem Pollardovy $p-1$ metody:

Tabulka 1.2: Ukázka algoritmu – Pollardova $p-1$ metoda – faktorizace modulu $n = 560275966521760946150893$.

#	a	b	d
1	2	2	$\gcd((2^{2!} \bmod n) - 1, n) = \gcd(3, n) = 1$
2	2	3	$\gcd((2^{3!} \bmod n) - 1, n) = \gcd(63, n) = 1$
3	2	4	$\gcd((2^{4!} \bmod n) - 1, n) = \gcd(16777215, n) = 1$
⋮			
315	2	316	$\gcd((2^{316!} \bmod n) - 1, n) = \gcd(5565 \ldots 1817, n) = 1$
316	2	317	$\gcd((2^{317!} \bmod n) - 1, n) = \gcd(4604 \ldots 5896, n) \notin \{n, 1\}$
$\gcd\left((2^{317!} \bmod n) - 1, n\right) = \gcd(4604 \ldots 5896, n) =$ **741030129527**			

Na příkladu znázorněném tabulkou 1.2 jsme úspěšně faktorizovali 80-bitový modul $n = 560275966521760946150893$ za pomoci 316 iterací Pollardovy $p-1$ metody. Nalezli jsme netriviální faktor $p = 741030129527$. Pro srovnání: faktorizace stejného modulu by algoritmem Fermatovy faktorizace 1.4.2.1 vyžadovala 37 809 461 iterací.

Pollardův $p-1$ algoritmus jsem se rozhodl začlenit do implementační částí bakalářské práce, protože dokáže efektivně odhalit prvočíselný rozklad modulu za předpokladu, že byly nevhodně zvoleny prvočísla p, q (nebylo dodrženo doporučení 1.3.2).

1.4.4.1 Složitost

Popsaný algoritmus funguje jako obecný algoritmus pro faktorizaci s předpokladem hladkosti čísla $p-1$, kde p je jeden z prvočíselných činitelů faktorizovaného čísla. Hodnota a může nabývat hodnot $1 < a < n$. Pro každou hodnotu a lze iterovat $1 < b \leq p-1$, kde $p = \min(p, q)$, neboť pokud $b = (p-1)! \Leftrightarrow (p-1)|b$, potom platí, že $p|a^{b!}-1$, tudíž $\gcd(a^{b!}-1, n) \in \{p, n\}$. Hodnota p může nabývat hodnot $2 < p < \sqrt{n}$.

Složitost takového algoritmu tedy nabývá hodnoty:

$$O(n \cdot \sqrt{n})$$

I přestože složitost je oproti jednoduchému útoku hrubou silou 1.4.1 o poznání vyšší, algoritmus si při splnění předpokladu hladkých čísel dokáže vystačit s $a = 2$ a $b \ll \sqrt{n}$, čímž se složitost dramaticky snižuje.

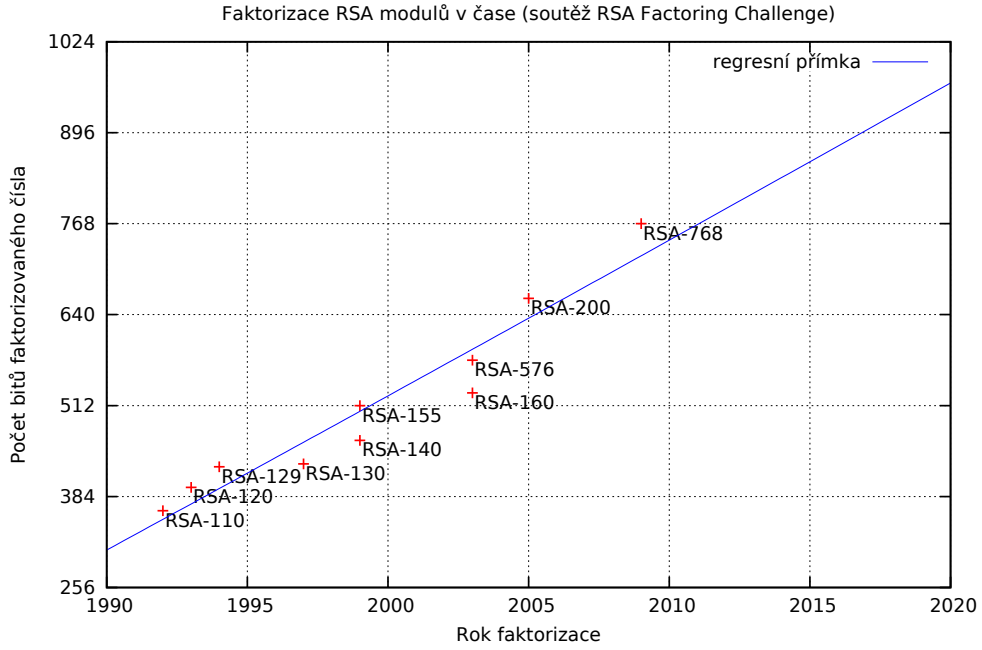
1.4.5 Shrnutí

Faktorizace modulu, a tím pádem kompromitace instance šifry RSA, je známý otevřený problém *RSA Problem*. V letech 1991-2007 probíhala soutěž *RSA Factoring Challenge*, která měla podnítit výzkum teorie čísel a praktické složitosti faktorizace velkých čísel. Vybrané úspěšné faktorizace v průběh soutěže znázorňuje graf 1.2.

Optimální (nejefektivnější) algoritmus pro faktorizaci čísel je závislý na velikosti hledaných prvočísel. Zatímco pro prvočísla do velikosti 2^{16} lze efektivně využívat předpočítanou tabulku prvočísel čítající jen několik tisíc řádek, stejné řešení nelze aplikovat pro hledání velkých prvočísel (použitých například pro 1024 bitové a vyšší RSA moduly). Nejefektivnějším v současnosti prakticky nasazeným algoritmem pro faktorizaci velkých čísel je výherce RSA Factoring Challenge (faktorizace úlohy RSA-768 – 768 bitový modul) – General number field sieve[18].

V rámci projektu *Factoring as a Service* je možné provést faktorizaci RSA 512-bitového modulu během několika jednotek hodin za pomocí platformy Amazon EC2 a to za cenu nepřevyšující 100$ USD [19]. Veškerý software s návodem pro použití je veřejně dostupný na síti GitHub. K faktorizaci je použit zmíněný paralelizovaný algoritmus GNFS.

V současné době standard FIPS doporučuje užití RSA modulu o délce 2048 nebo 3072 bitů[15], které by měly zaručit bezpečnost dat na relativně dlouhou dobu – soudě dle výsledků soutěže RSA Factoring Challenge1.2.



Obrázek 1.2: Graf úspěšných řešitelů soutěže RSA Factoring Challenge (ukončena v roce 2013) doplněný o regresní přímku.

Kapitola 2

Analýza a návrh řešení

V této kapitole popisuji analýzu požadavků řešení, rozdělení práce na logické celky, návrh struktury programu. Vysvětluji mé volby implementační platformy, dodatečných knihoven a datového úložiště.

2.1 Struktura programu

Cíl práce lze rozdělit na dvě logicky související části:

- prostudování a implementace vybraných známých útoků na šifru RSA
- otestování implementace na datech získaných z vybrané podmnožiny českého Internetu a na speciálně připravených datech

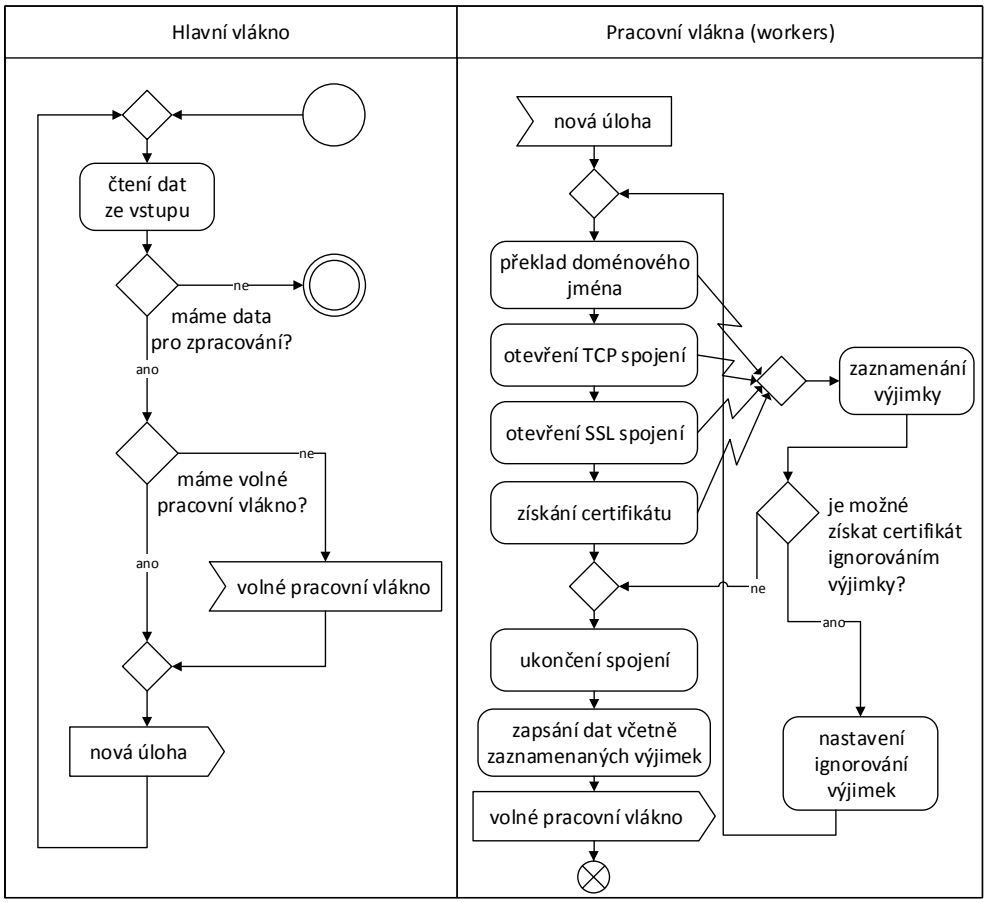
Strukturu programovací části bakalářské práce jsem se proto rozhodl rozdělit na část pro získání potřebných dat k testování (Skener 2.1.1) a část pro testování dat pomocí implementovaných útoků (Tester 2.1.2). Zvolené rozdělení práce skýtá navíc výhodu možnosti provést získání dat a jejich otestování na různých počítačích – skenovací část na počítači s rychlou přípojkou do sítě Internet a oproti tomu testovací část na výkonném počítači, který do sítě Internet vůbec být připojen nemusí.

2.1.1 Skener

Skenovací část práce má za úkol získat digitální certifikáty webových serverů, jejichž vybrané parametry budou uloženy (a později v testovací části zpracovány). Nutnými požadavky pro funkci skeneru jsou tedy:

- možnost zápisu výsledků do úložiště
- připojení do sítě Internet
- dostupný server pro překlad doménových jmen[16]

[16] DNS resolver



Obrázek 2.1: Diagram aktivit průběhu skenovací části implementace.

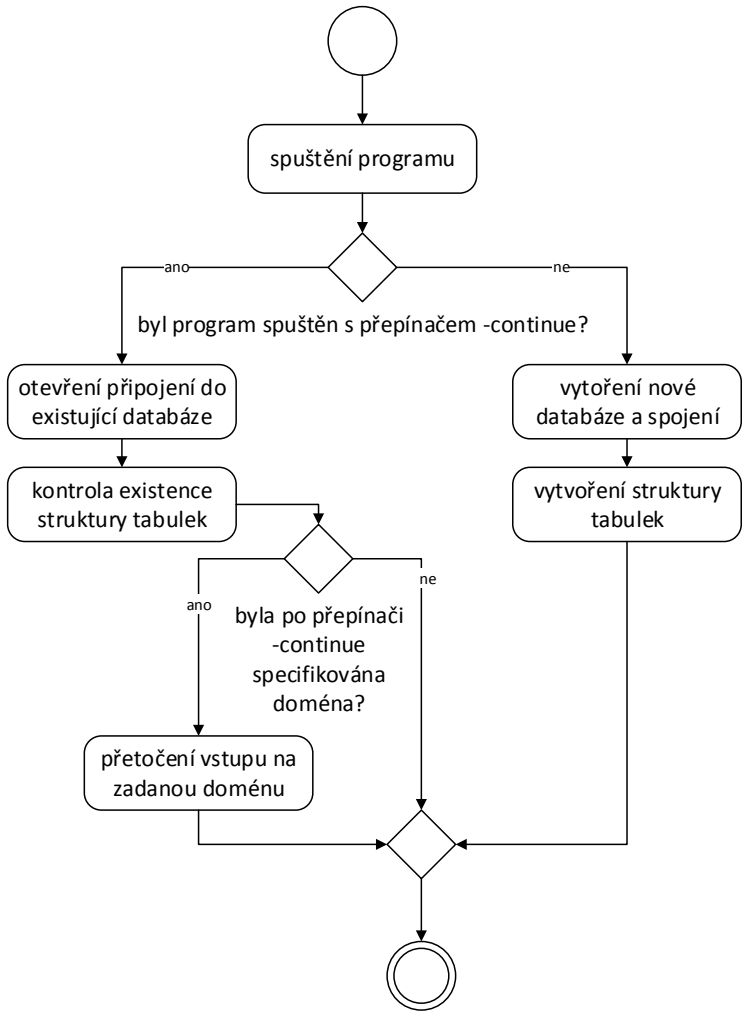
Vzhledem k relativní systémové nenáročnosti a nezanedbatelnému času potřebnému k navázání SSL[17] spojení by sekvenční mechanismus skenování byl časově velice náročný, nehledě na to, že při rozsáhlém skenování by sběr dat trpěl změnami v zóně (expirace domén). Praktickým požadavkem skenovací části úlohy je tedy paralelní běh několika spojení najednou, což velice urychlí průběh skenování.

V případě rozsáhlého skenování veřejného prostoru je vhodné informovat o takovém záměru správce používaného serveru pro překlad doménových jmen (týká se zejména drobných serverů ve správě poskytovatele internetového připojení, ale i firemních a případně školních serverů). Některé DNS servery nejsou pro skenování vhodné vůbec, neboť omezují maximální počet dotazů (příkladem DNS servery ve správě sdružení CZ.NIC, limitující zdrojovou IP adresu tazatele na přibližně 20 dotazů za vteřinu). Z těchto důvodů jsem pro

[17]Není-li uvedeno jinak, pojmem *SSL* se v této práci míní užití protokolu TLS.

skenování využíval DNS serverů ve správě společnosti Google, Inc. (v IPv4 sítí dostupné na adrese `8.8.8.8`, případně `8.8.4.4`). Ve fázi analýzy jsem se zabýval i otázkou legálnosti skenování. Vzhledem k minimální zátěži skenovaného webového serveru (spojení je ukončeno v momentě dokončení přenosu certifikátů) jsem nepředpokládal žádné právní komplikace. I přes to jsem se svým záměrem oslovil zaměstnance fakultního oddělení ICT[18], Ing. Martina Bílého, který k mému postupu neměl námitky.

S přihlédnutím na požadavky paralelizace a zaznamenávání dat bude také nutné užití úložiště, které je schopné vícevláknových operací zápisu.



Obrázek 2.2: Diagram aktivit spuštění skeneru a průběhu přípravy datového úložiště.

[18]Information and Communication Technologies

Jako kryptografický protokol pro komunikaci se servery bude použito TLS verze 1.0 a vyšší. Všechny verze protokolu SSL (1.0, 2.0, 3.0) ve světle známých útoků nebudou pro komunikaci podporovány, neboť i poslední verze SSL 3.0 byla v létě roku 2015 označena jako zastaralá a neměla by být nadále používána[20, RFC 7568].

V případě, že během procesu získání a verifikace certifikátu vyskytnou problémy (například webový server předčasně uzavře spojení, certifikát nebude platný pro dané využití, nebo pro komunikaci nebude možné použít protokol TLS), bude chyba pro úplnost zaznamenána. Bude-li se jednat o chybu, kterou lze ignorovat a získat tím obsah certifikátu (zejména neprovedení validace certifikátu), skener tak učiní, nicméně souběžně s uložením vybraných parametrů certifikátu bude zaznamenána i ignorovaná chyba.

2.1.2 Tester

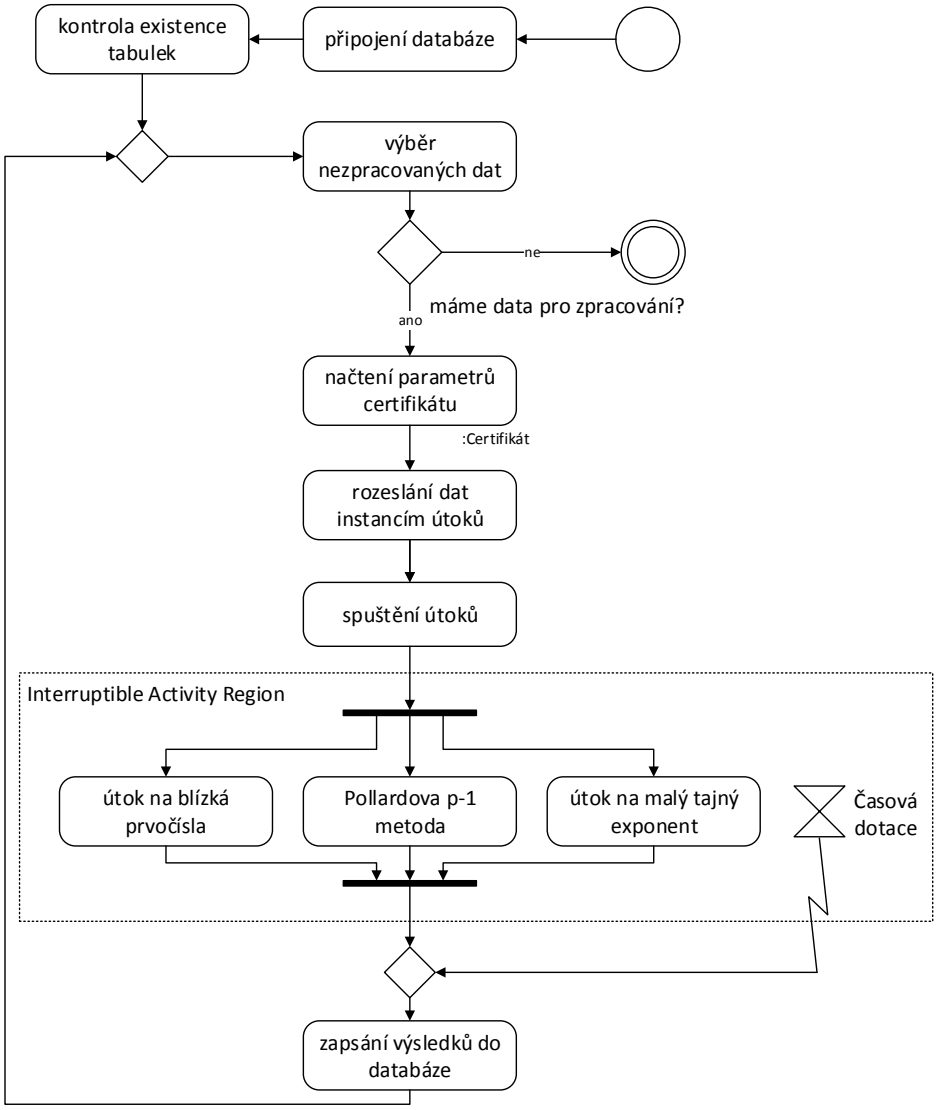
Úkolem testovací části práce je aplikovat vybrané známé útoky na šifru RSA na data získaná ze skenování, případně na data speciálně připravená data. V analytické části Útoky na RSA 1.4 jsem se věnoval popisu a analýzy přínosu implementace jednotlivých útoků, z čehož vyplynula zamýšlená implementace následujících tří útoků:

- Útok na příliš blízká prvočísla p, q pomocí Fermatovy faktorizace
- Útok na příliš nízký exponent d pomocí Wienerova algoritmu řetězových zlomků
- Faktorizace modulu pomocí Pollardovy $p - 1$ metody

Zpracovávaná data a parametry digitálních certifikátů budou vyžadovat aritmetické operace s operandy převyšující běžně dostupné datové typy – například moduly o délce tisíců bitů. Zvolená implementační platforma tedy bude muset takové operace podporovat nebo mít dostupné možnosti pro doplnění funkcionality (volně dostupné knihovny).

Každému útoku bude přidělena jednotná časová dotace. Některé útoky (konkrétně Pollardova $p - 1$ metoda) jsou záměrně implementovány tak, aby během časové dotace vypočítaly co možná největší počet méně náročných iterací, namísto jedné dlouhotrvající iterace. V případě, že implementovaný algoritmus nenalezne řešení během časové dotace, bude jeho běh přerušen a program přejde na zpracování dalších dat.

V implementaci se nebudu zabývat optimalizací a paralelizací těchto vybraných útoků, neboť se jedná o rozsáhlé téma hodné vlastní diplomové práce, jak už někteří kolegové[21] učinili. Namísto paralelizace na úrovni algoritmu jsem se tedy rozhodl paralelizovat běh několika útoků současně (z čehož plyne požadavek fixní časové dotace). Z tohoto rozhodnutí ovšem vyplývá omezení maximálního počtu vytížených jader procesoru. V současném návrhu, kdy jsou



Obrázek 2.3: Diagram aktivit průběhu testovací části implementace.

sekvenčně zpracovávány jednotlivé certifikáty, lze vytížit nejvýše tolik jader, kolik je implementováno útoků. Výhodou tohoto řešení je snazší správa časových dotací pro běh jednotlivých algoritmů, které spravuje hlavní vlákno aplikace namísto ad-hoc kontroly při každé iteraci implementovaných útoků.

Možným budoucím vylepšením je využití sofistikovanějšího plánovače úloh, například pomocí fronty dílčích úkolů, pomocí které lze dosáhnou rovnoměrnějšího vytížení dostupných výpočetních prostředků, ovšem za cenu nutnosti manipulace s daty, sdílenými napříč pracovními vlákny.

2.1.3 Shrnutí

Implementaci cíle bakalářské práce jsem rozdělil do dvou částí – části pro získání dat a části pro jejich zpracování. Obě části budou pro urychlení zpracování paralelizovány, proto bude nutné použití úložiště schopného zachování funkčnosti napříč operacemi prováděnými více vlákny[19]. Pro zpracovávaná numerická data nejsou dostatečné standardní datové typy (32-bitová, případně 64-bitová čísla) a bude proto potřeba podpora pro velká čísla a operace s nimi buď ze strany jazyka nebo za pomoci dostupných knihoven.

2.2 Datové úložiště

Požadavky na datové úložiště jsou především:

- stabilní operace na více vláknech
- kompaktnost (pro snadný přenos dat)

Možným řešením je vlastní binární formát, který zajistí kompaktnost (centrální úložiště všech dat by byl jeden soubor) a vícevláknový přístup by byl programově zajištěn na úrovni skeneru a testeru – pomocí vyrovnávací paměti a synchronizovaného zápisu do úložiště. Problémem je uzavřenost formátu, která komplikuje analýzu a interpretaci výsledků skenování a testování, neboť vlastní binární formát s nejvyšší pravděpodobností nebude kompatibilní s nástroji třetích stran.

Alternativním řešením je použití existujících řešení, která řeší problém datové stability během konkurujících zápisů – tedy software relačních databází. Relační databáze, které nabízejí transakční zpracování požadavků a řídí se dle zásad ACID[20], jsou velmi vhodnými kandidáty pro datové úložiště této práce.

Relační databáze ve většině případů (například Oracle Databse, Microsoft SQL Server, MySQL, PostgreSQL a další) používají architekturu server-klient, tedy vyžadují spuštění databázového serveru, který následně přijímá požadavky a obsluhuje datové úložiště databáze. Takové řešení by pro tuto práci nebylo vhodné, neboť by s sebou neslo řadu závislostí, komplikujících reálně uplatnění a nasazení. Výběr databázového úložiště jsem proto zúžil na *embedded*[21] databázová řešení, která nevyužívají model server-klient a tím pádem nevyžadují spuštění serveru obsluhujícího požadavky klientů. Některá výše jmenovaná databázová řešení mají své *embedded* protějšky (MySQL Embedded Server Library, Microsoft SQL Server Compact). Rozhodl jsem se ale využít řešení, které bylo od počátku stavěno na *embedded* principu a současně je široce používané. Z těchto důvodů jsem jako RSŘBD[22] rozhodl zvolit *SQLite*.

[19] jinými slovy *thread-safe* řešení

[20] Atomicity, Consistency, Isolation, Durability – česky atomicita, konzistence, nezávislost a trvanlivost

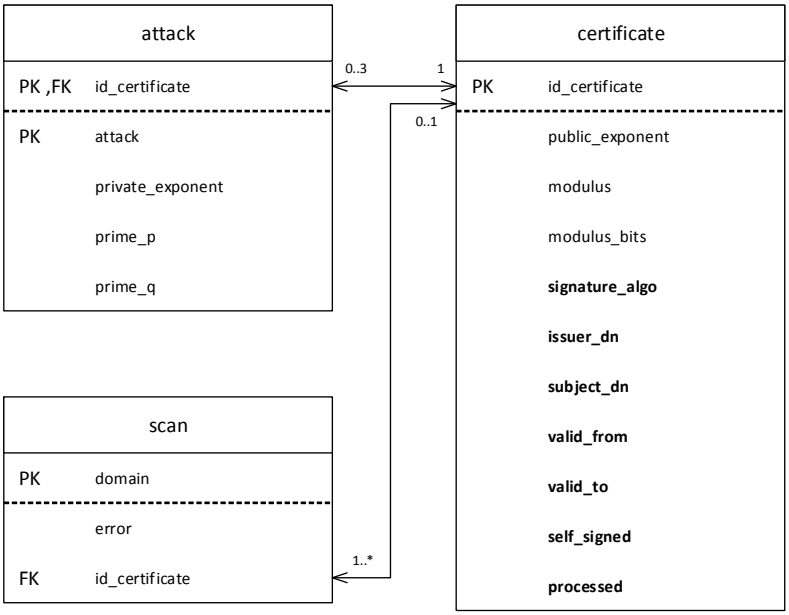
[21] vestavěná

[22] Relační systém řízení bází dat, anglicky *RDBMS*.

SQLite dodržuje soubor vlastností ACID, jedná se o *embedded* databázový systém a je pro svou nenáročnost velmi populární, zejména na mobilních (Apple iOS, Android) platformách. Pro SQLite existuje velké množství knihoven, díky kterým lze s databází komunikovat z řady programovacích jazyků, Javu nevyjímaje. Databáze není příliš striktní v definicích datových typů ukládaných hodnot – sama databáze implementuje pouze 5 datových typů (typ `NULL`, celá čísla `INTEGER`, čísla s plovoucí desetinnou čárkou `REAL`, řetězce `TEXT` a binární data `BLOB`). Pro ukládání kalendářních dat jsem zvolil formát `TEXT`, kde formát data odpovídá normě ISO8601[22] (forma `RRRR-MM-DD HH:MM:SS.SSS`).

2.2.1 Struktura datového úložiště

Databáze bude obsahovat celkem 3 tabulky, jejichž forma je zachycena v následujícím diagramu 2.4. Vzhledem k užšímu výběru datových typů a rozsahu ukládaných číselných hodnot jsou všechna pole typu `TEXT` s výjimkou těchto tří polí: `processed`, `self_signed` a `modulus_bits` (datový typ `INTEGER`).



Obrázek 2.4: Diagram struktury databázového úložiště.

2.3 Volba platformy a knihoven

Po analýze implementačních požadavků programovací části bakalářské práce jsem se zaměřil na výběr implementační platformy. Ze sumarizace 2.1.3 vy-

plývají tyto platformní požadavky:

- Podpora vícevláknových aplikací
- Podpora práce na síti
- Podpora práce s SSL/TLS a certifikáty
- Přenositelnost (skenovací a testovací části mohou běžet na rozdílných počítačích)
- Možnost výpočtů s neomezenou přesností
- Podpora práce s *thread-safe* úložištěm

Na základě těchto požadavků a mých dosavadních zkušeností jsem jako implementační platformu zvolil programovací jazyk Java. Za dobu mého bakalářského studia jsem v Javě vytvořil několik semestrálních úloh, během kterých jsem si vyzkoušel práci s většinou aspektů jazyka popsanými výše – zejména v předmětu `BI-PSI`, kdy jsem implementoval vícevláknovou síťovou aplikaci. Většinu požadavků pokryje Java bez dodatečných knihoven třetích stran. Tyto skutečnosti činní Javu velmi dobrým kandidátem pro implementační platformu této bakalářské práce.

2.3.1 Dodatečné knihovny

Během analýzy podpory výpočtů s neomezenou přesností ve zvoleném jazyce Java jsem narazil na limitaci implementace knihovny `java.math.BigInteger`, kde není implementovaná funkce odmocniny, která je potřebná pro průběh Fermatovy faktorizace. Využil jsem proto *open-source* knihovny *Google Guava*, kde jsou rozšiřující možnosti výpočtu s neomezenou přesností implementovány v knihovně `com.google.common.math.BigIntegerMath`. Knihovnu *Google Guava* jsem zvolil proto, že implementovaná funkce odmocniny je optimalizovaná a využívá řady zdokumentovaných „triků“ pro urychlení výpočtu.

Během vývoje jsem se rozhodl používat jednotkové testy, které tvoří vhodný doplněk práce pro případné pozdější změny. Do práce je proto zahrnuta i velmi rozšířená knihovna pro jednotkové testování *JUnit* a její závislost, knihovna rozšiřující syntax *Hamcrest*.

Obě implementované části bakalářské práce využívají příkazový řádek jako rozhraní pro komunikaci s uživatelem. Pro uživatelovo pohodlí aplikace obsahuje nápovědu s popisky pro jednotlivé přepínače, kterými může uživatel modifikovat chod aplikace. Za účelem ulehčení této části implementace jsem se rozhodl do projektu začlenit část knihovny *CLI* z celku *Apache Commons*, která zajišťuje formátování a výpis nápovědy s popisky přepínačů a jejich vstupní kontrolu.

Poslední z celkového počtu pěti knihoven je *SQLite* JDBC[23] ovladač, jelikož jako úložiště byla zvolena databáze *SQLite*. *SQLite* JDBC ovladač podporuje sdílení jednoho připojení mezi více vlákny a je proto vhodným nástrojem pro ukládání dat skenovací části práce.

2.3.2 Shrnutí

Jako implementační platforma byl zvolen jazyk Java, protože vyhovoval téměř všem požadavkům a mám s tímto jazykem pozitivní zkušenosti. Datové úložiště bude realizováno formou relační databáze SQLite, pro podporu vícevláknových operací, kompaktnost a usnadnění inspekce dat a výsledků. Do projektu byly dále zahrnuty následující knihovny:

- Google Guava 19.0, Apache License 2.0
- Apache Commons CLI 1.3.1, Apache License 2.0
- JUnit 4.12, Eclipse Public License 1.0
- Hamcrest 1.3, BSD 3-Clause License
- SQLite JDBC 3.8.11.2, Apache License 2.0

[23] JDBC – Java Database Connectivity

Kapitola 3

Implementace

V této kapitole popisuji realizaci implementační části práce, sestávající se z implementace vybraných útoků na šifru RSA, vybraných v analytické části Útoky na RSA 1.4. Algoritmy budou implementovány v programovacím jazyce Java, který byl v předcházející kapitole 2 zvolen jako vhodný kandidát.

3.1 Struktura projektu

Jazyk Java podporuje rozdělení projektu do takzvaných *balíčků*, které usnadňují a oddělují logicky související části projektu. Pro tuto práci jsem zvolil následující strukturu:

```
cz.cvut.fit.cernama9
├── cracker ..... balíček testovací části projektu
│   ├── attacks ..... balíček implementovaných útoků na RSA
│   │   ├── NearPrimes.java
│   │   ├── PollardPMinus1.java
│   │   ├── RSAAttack.java ..... Rozhraní útoků na RSA
│   │   └── Wiener.java
│   ├── utilites ..... Podpůrné datové struktury
│   │   ├── AttackResult.java ..... Struktura pro výsledky útoku
│   │   └── SimpleRSAPublicKey.java ..... Struktura RSA veřejného klíče
│   └── CertificateTester.java .... Hlavní třída pro testovací část projektu
└── scanner ..... balíček skenovací části projektu
    ├── CertificateDownloader.java ..... Hlavní třída pro skenování
    └── CertificateResponse.java . Datová struktura pro výsledek skenu
```

Obrázek 3.1: Logická struktura projektu.

3.2 Výjimky a ošetření chyb

Platforma Java v kódu hojně využívá konceptu výjimek, které (pokud nastanou) přeruší vykonávání kódu a vyžadují ošetření nadřazeným blokem programu. Jedná se o běžný princip předání chybového stavu v programu, který jsem použil pro validaci digitálních certifikátů, neboť součástí výjimky je i krátký popis chybového stavu. Pro potřeby této práce jsem o nasazených certifikátech uvažoval pouze jako o validních, nebo (z libovolného počtu důvodů) nevalidních. Koncept předávání chybového stavu výjimkou v kombinaci s velkým zanořením zdroje výjimky v kódu s sebou nese obtížné ignorování a zotavení se z výjimky za účelem nalezení případných dalších důvodů nevalidního digitálního certifikátu, neboť dochází k přerušení sekvenčního průběhu kódu v místech, která nejsou cizími objekty přístupná. V případě, že při skenování nastane některá z výjimek, kterou lze za účelem získání certifikátu ignorovat (dle diagramu 2.1), skener naváže spojení znovu, tentokrát ovšem bez jakýchkoliv validací certifikátu – díky čemuž je možné zaznamenat jak problém s validitou, tak i konkrétní certifikát.

Diagram 3.2 zachycuje sekvenční možné vyvolání výjimek.

Zakázáním validace certifikátu lze dosáhnout vlastní implementací třídy podporující rozhraní `HostnameVerifier` a `TrustManager`.

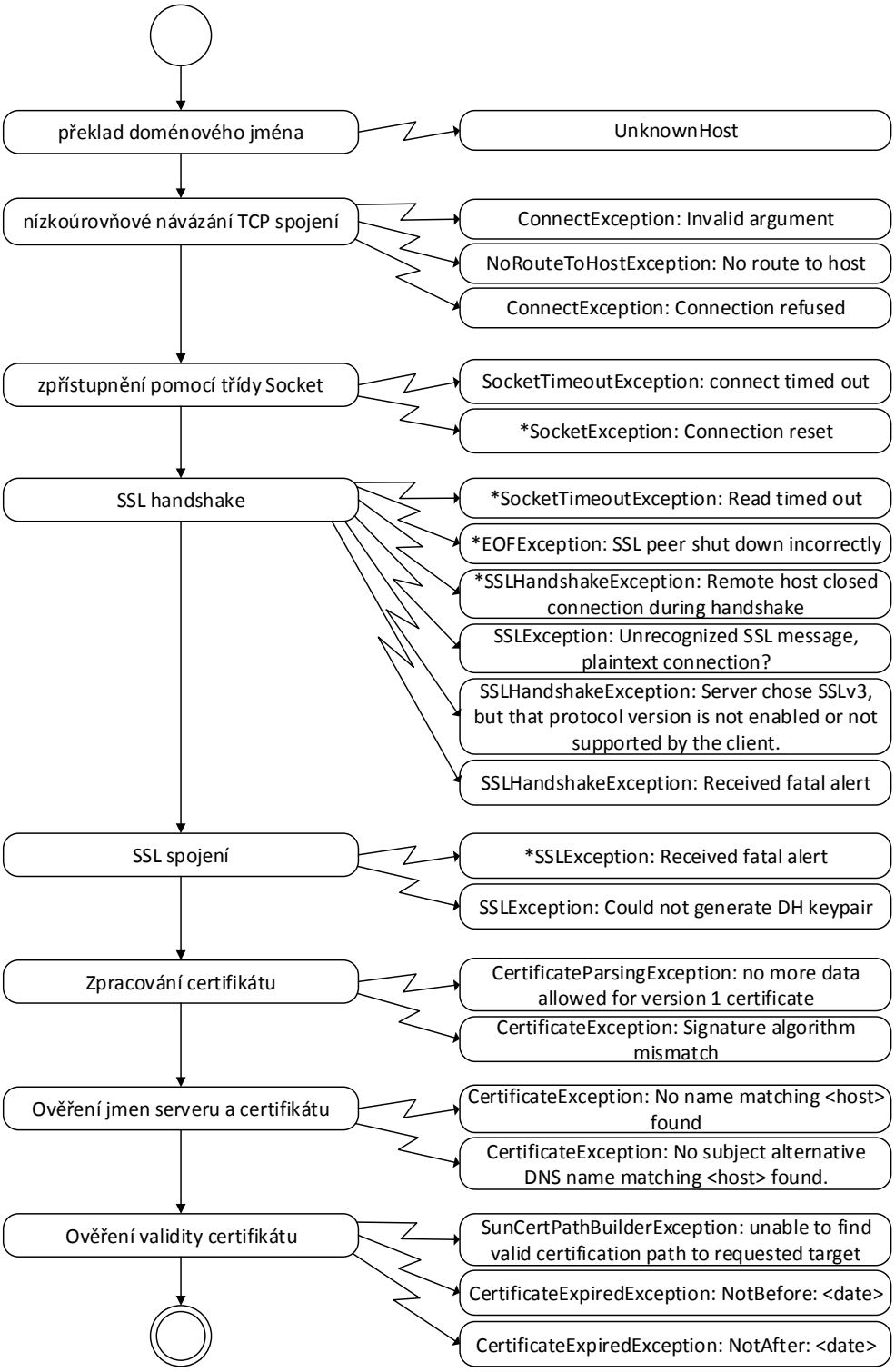
Implementace třídy podporující rozhraní `HostnameVerifier`, která jako validní označí každý certifikát nezávisle na shodě doménových jmen, lze docílit triviálně, neboť je třeba implementovat pouze jednu její metodu `verify`. Jazyk Java v takovém případě povoluje velmi expresivní vyjádření pomocí anonymní třídy zapsané lambda výrazem, který pomocí metody `setHostnameVerifier` třídy `HttpsURLConnection` vždy vyhodnotí shodu doménových jmen a efektivně tím ignoruje validaci doménového jména serveru a certifikátu.

```
//connection is an instance of HttpsURLConnection
connection.setHostnameVerifier((hostname, session) -> true);
```

Ukázka kódu 3.1: Nastavení ignorování validace jména webového serveru.

Implementace třídy, která označí každá certifikát jako důvěryhodný, lze analogicky dosáhnout implementací třídy podporující rozhraní `TrustManager`. Pro potřeby této práce se jedná o instanci `X509TrustManager`. V kódu existující implementace je hojně využito zmíněného konceptu výjimek. Pro efektivní ignorování ověření důvěryhodnosti certifikátu tedy postačí žádné výjimky negenerovat. Je ovšem nutná tvorba nového kontextu `SSLContext`, který bude použit pro spojení, u kterých nastaly problémy s validitou certifikátu.

```
SSLContext sc = SSLContext.getInstance("SSL");
sc.init(null, new TrustManager[]{new X509TrustManager()
{
  public X509Certificate[] getAcceptedIssuers()
  {
    return null;
```



Obrázek 3.2: Diagram sekvenční možného vyvolání výjimek. Výjimky označené hvězdičkou mohou být vyvolány i v dalších fázích procesu.

```
  }
  public void checkClientTrusted(X509Certificate[] certs, String
      authType) { }
  public void checkServerTrusted(X509Certificate[] certs, String
      authType) { }
}}, null);
```

Ukázka kódu 3.2: Nastavení ignorování validace důvěryhodnosti certifikátu.

3.2.1 Význam výjimek

Tato část kapitoly popisuje jednotlivé vyvolané výjimky a příčiny jejich vyvolání. V případě potřeby je doplněn krátký komentář.

`UnknownHostException: <host>`
Nepodařilo se nalézt DNS záznam domény (v tomto případě substituovanou textem `<host>`). Před zahájením skenování jsem sice provedl pročištění dat od vypršených domén, ale i prodleva jednoho dne znamenala několik vypršených domén.

`ConnectException: Invalid argument`
Nepodařilo se otevřít připojení s cílem, protože cíl je špatně zadán – v průběhu měření jsem nalezl doménu `taxikolin.cz`, jejíž DNS záznam typu `A` (obsahující IPv4 adresu) nesl hodnotu `0.18.120.100`. Taková IPv4 adresa samozřejmě není platným cílem.

`NoRouteToHostException: No route to host`
Doména má platný DNS záznam, ale k cíli se nepodařilo připojit – webový server má pravděpodobně výpadek.

`ConnectException: Connection refused`
Webový server pro danou doménu nenaslouchá na TCP portu 443 protokolu HTTPS.

`SocketTimeoutException: connect timed out`
Webový server pro danou doménu sice naslouchá na TCP portu 443, ale neodpovídá (zvolená doba maximálního čekání je 5 vteřin).

`SocketException: Connection reset`
Webový server spojení neočekávaně ukončil.

`SocketTimeoutException: Read timed out`
Webový server pro danou doménu sice naslouchá na TCP portu 443, ale přestal komunikovat (zvolená doba maximálního čekání je 10 vteřin).

`EOFException: SSL peer shut down incorrectly`
Webový server spojení neočekávaně ukončil.

`SSLException: Unrecognized SSL message, plaintext connection?`
Server sice naslouchá na TCP portu 443, ale nekomunikuje pomocí protokolu HTTPS. Zpravidla se jedná o špatně nakonfigurovaný server, který nemá zapnutou podporu pro SSL.

`SSLHandshakeException: Server chose SSLv3, but that protocol version is not enabled or not supported by the client.`
Webový server není schopen komunikovat pomocí bezpečného kryptografického protokolu TLS a podporuje pouze zastaralý protokol SSL 3.0.

`SSLHandshakeException: Received fatal alert: <alert>`
Během SSL spojení nastala chyba. Zpravidla se jednalo o špatně nakonfigurované SNI (za zástupné `<alert>` bylo dosazeno `unrecognized_name`).

`SSLHandshakeException: Remote host closed connection during handshake`
SSL spojení bylo ukončeno ve fázi vyjednávání (handshake)

`SSLException: Could not generate DH keypair` Webový server si vyžádal užití Diffie-Hellman klíče v rozsahu, který Java nepodporuje. V současné době je limit 2048 bitů.

`CertificateParsingException: no more data allowed for version 1 certificate`
Nesprávně vygenerovaný certifikát.

`CertificateException: Signature algorithm mismatch`
Nesprávně vygenerovaný certifikát – nejedná se o indikovaný typ podpisu, příkladem je doména `batontech.cz`.

`No name matching <host> found`
Certifikát nenáleží serveru, který se s ním identifikuje (zástupné `<host>`). Certifikát neobsahuje záznam o alternativních doménových jménech. Jedná se zpravidla o výchozí certifikáty velkých webhostingových společností.

`No subject alternative DNS name matching <host> found.`
Analogický problém k `No name matching <host> found` – v tomto případě ale certifikát obsahuje doplněk o alternativních doménových jménech. Ani jedno z nich ovšem není shodné s doménovým jménem, jehož certifikát byl vyžadován.

`SunCertPathBuilderException: unable to find valid certification path to requested target`

Certifikát není podepsaný důvěryhodnou certifikační autoritou. Jedná se o self-signed certifikát, nebo je webový server špatně nakonfigurován a nepřikládá případně mezilehlé (intermediate) certifikáty nutné k ověření důvěryhodnosti podpisu.

`CertificateExpiredException: NotAfter: <date>`

Předložený certifikát je sice vystaven pro dané doménové jméno, je podepsán důvěryhodnou certifikační autoritou, ale jedná se o vypršený certifikát – jeho platnost skončila v dobu zastoupenou symbolem `<date>`.

`CertificateExpiredException: NotBefore: <date>`

Analogicky k předchozímu `NotAfter`, jedná se ovšem o certifikát, který v platnost ještě nevstoupil – stane se tak až v době zastoupené symbolem `<date>`.

3.3 Výkonnost implementovaných řešení

Obě implementované části (skener a tester) byly paralelizovány za účelem rychlejšího dokončení jejich činnosti. U každé části toho bylo dosaženo jiným způsobem. Zatímco u skenování trvá dlouhou dobu navázání spojení a přenos certifikátů, u testovací části se jedná o výpočetně náročnou úlohu po stránce procesorového času. Z tohoto důvodu má smysl u skenovací části uvažovat o několika desítkách vláken pro skenování a zaznamenávání výsledků, oproti tomu u testovací části byl zvolen konzervativnější počet vláken – pro každý typ útoku jedno samostatné vlákno. Implementovány byly celkem 3 útoky a ve spojení s hlavním vláknem, které zajišťuje přísun dat a řízení časové dotace, se ve výsledku jedná celkem o 4 vlákna. Testovací část jsem na vzorku dat spouštěl na svém osobním počítači, jehož procesor má 2 fyzická jádra, na kterých dokáže provozovat celkem až 4 vlákna. Dostupné výpočetní prostředky jsem tedy zcela vytížil.

3.3.1 Testování výkonnosti testeru

Pro otestování počtu iterací, kterých je implementace na testovacím hardwaru během fixní časové dotace 600 vteřin schopna dosáhnout, jsem provedl několik měření, které shrnuje tabulka 3.1. Jedná se o průměrné hodnoty z měření 10 instancí dat pro každou velikost modulu. Testování výkonnosti jsem provedl na svém osobním počítači s procesorem Intel Core i5-2410M, za použití 64-bitové Oracle JRE[24] verze 1.8.0_66-b17.

[24] Java Runtime Environment

Tabulka 3.1: Dosažené počty iterací za fixní časovou dotaci 600 vteřin. Hodnoty označené hvězdičkou znamenají dokončení všech výpočtů v rámci časové dotace.

Velikost modulu	Průměrné počty dosažených iterací		
	NearPrimes	PollardPMinus1	Wiener
512 bitů	174 504 526	15 156	*11
1024 bitů	95 041 910	7 296	*12
2048 bitů	31 024 026	2 303	*13
4096 bitů	10 198 390	1 790	*14

3.3.2 Testování rychlosti skenování

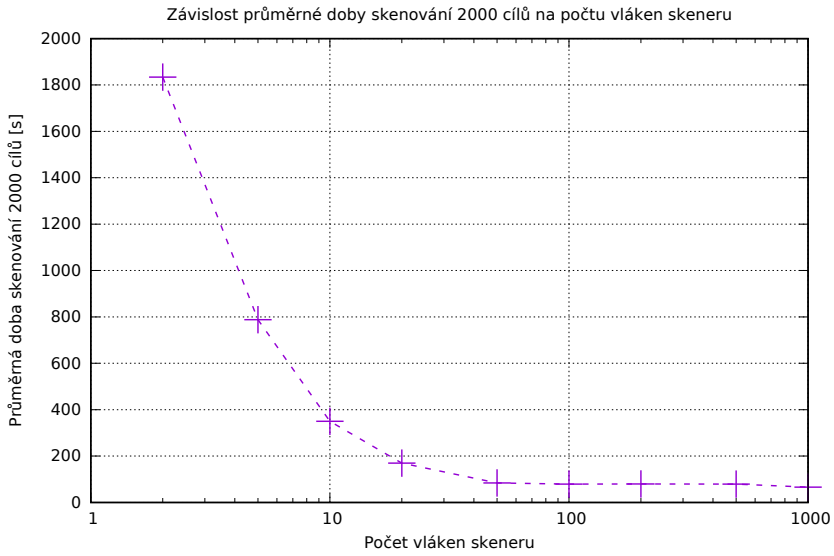
Na rychlost dokončení síťového skenování má velký vliv kvalita připojení do sítě Internet a počet zvolených vláken, která paralelně zpracovávají vstupní seznam domén. Následující graf 3.3 zachycuje rychlost dokončení skenování na vzorku obsahujícím 2000 domén v závislosti na počtu použitých vláken. Z grafu je patrné, že vyšší počet vláken skeneru přináší velké urychlení až do okamžiku, kdy se limitujícím faktorem stává prostupnost přípojky do sítě Internet a případná limitace počtu požadavků na server pro překlad doménových jmen. Velmi vysoké počty vláken (500 a více vláken) v situaci znázorněné grafem již nepřinášejí další urychlení práce programu, naopak s sebou přinášejí negativa v podobě nutné správy velkého počtu vláken a tím pádem zvýšené využití systémových prostředků bez kýženého efektu urychlení skenování.

3.4 Poznatky a problémy implementační fáze

V této podkapitole jsem shrnul a okomentoval některé vhodné implementační volby, ale také problémy, které jsem odhalil až při implementaci. Tato část textu tedy nese značnou informační hodnotu pro případné následníky této práce, kteří by čelili stejným problémům, jakým jsem čelil já sám.

3.4.1 Java KeyStore

Java KeyStore (zkráceně JKS) je kompaktní repozitář obsahující certifikáty. Sada vývojových nástrojů jazyka Java (zkráceně JDK) obsahuje sadu nástrojů (zejména `keytool`) pro práci s tímto binárním úložištěm, jehož obsah je navíc povinně chráněn heslem. V praxi lze tento formát použít například jako soubor důvěryhodných kořenových certifikátů početných certifikačních autorit, což je současně důvod jeho užití v rámci této práce. Soubor kořenových certifikátů jsem za pomoci nástroje `certutil` ze sady nástrojů Mozilla NSS dne 17. dubna 2016 vyexportoval z prohlížeče Mozilla Firefox 45 a za pomoci již zmíněného nástroje `keytool` je naimportoval do nově vytvořeného JKS repozitáře, který je později načten aplikací dle ukázky kódu:



Obrázek 3.3: Graf zachycující rychlost skenování oproti počtu vláken skeneru. Vodorovná osa používá logaritmické měřítko.

```
//Get the path to bundled keystore containing CA root certificates
final URL keystorePath = this.getResource("/keystore.jks");
//Setup bundled truststore
System.setProperty("javax.net.ssl.trustStore",keystorePath.getPath());
System.setProperty("javax.net.ssl.trustStorePassword", "changeit");
```

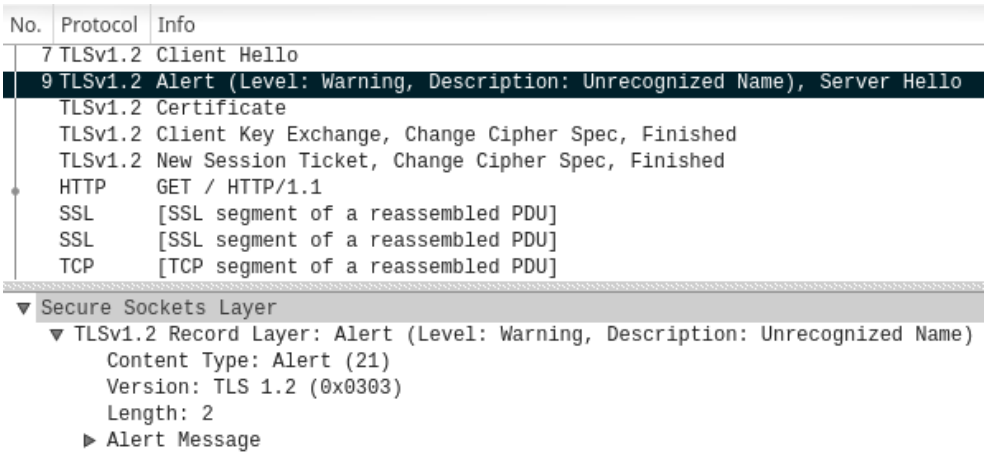
Ukázka kódu 3.3: Nastavení JKS repozitáře.

3.4.2 Podpora Server Name Indicator

Technologie Server Name Indicator (zkráceně SNI) je rozšířením TLS protokolu, ve kterém klient přímo definuje doménové jméno webového serveru, jehož certifikát vyžaduje [23, RFC 4366]. Díky tomu může server na jedné IP adrese a TCP portu předkládat různé certifikáty, což umožňuje provoz více HTTPS serverů s různými certifikáty na jedné IP adrese. Podpora této technologie byla do Javy zavedena již ve verzi jazyka 7. Implementace v knihovně JSSE[25] je ovšem velmi striktní a některé chyby, které prohlížeče ignorují, považuje za fatální selhání a ukončí SSL spojení. Jedná se o komunikaci se serverem, který technologii SNI sice podporuje, ale je špatně nakonfigurován. Příkladem může být web Fakulty informačních technologií ČVUT – během návštěvy portálu `fit.cvut.cz` skrze protokol HTTPS v prohlížeči Mozilla Firefox 45 jsem zachytával síťový provoz a na přiloženém snímku obrazovky je vidět odpověď

[25] Java Secure Socket Extension

špatně nastaveného webového serveru, kterou ale Mozilla Firefox ignoruje – narozdíl od JSSE implementace v Javě.



Obrázek 3.4: Snímek obrazovky zachycené síťové komunikace se špatně nastaveným webovým serverem při přístupu pomocí HTTPS na fit.cvut.cz (zvýrazněný tok s chybovou hláškou).

Java dovoluje před první použitím knihovny JSSE vypnout podporu SNI, která problém v mnoha případech (včetně zmíněného fakultního webu) vyřeší, v jiných případech ovšem problémy vyvolá (servery, které bez klientské podpory SNI nedostanou informaci, jaký certifikát klientovi odeslat). Z tohoto důvodu jsem do práce začlenil odchycení výjimky třídy `SSLProtocolException` s obsahem `handshake alert: unrecognized_name`, při jejímž vyvolání dojde k uzavření původního spojení a vytvoření nového spojení, které využívá upravenou třídu `SSLSocketFactory`, která SNI dočasně (po dobu jednoho spojení) zakáže.

3.4.3 Zapnutí podpory zastaralých šifer

Výchozí bezpečnostní konfigurace platformy Java nepovoluje navázat spojení se serverem, který se prezentuje certifikátem s podpisem na bázi šifry MD2, nebo certifikátem s RSA veřejným klíčem menším než 1024 bitů. Zároveň zakazuje užití SSL 3.0, šifry RC4 a Diffie-Hellman klíčů o velikosti menší než 768 bitů. V práci jsem tyto bezpečnostní omezení obešel, aby nezkreslovala výsledná naměřená data:

```
//accept all certificates
System.setProperty("jdk.certpath.disabledAlgorithms", "");
System.setProperty("jdk.tls.disabledAlgorithms", "");
```

Ukázka kódu 3.4: Zrušení bezpečnostních omezení.

3.4.4 Jednoznačná identifikace certifikátu

Struktura datového úložiště, která byla popsána v části 2.2, obsahuje pole s názvem `id_certificate`. Struktura certifikátu sama o sobě obsahuje identifikující položku sériové číslo, její hodnota je ovšem v rukou certifikační autority. V případě nedůvěryhodných self-signed certifikátů je zcela v režii tvůrce těchto certifikátů. Z tohoto důvodu nepoužívám informaci o sériovém čísle jako jednoznačnou identifikaci certifikátu, namísto toho využívám techniku ve webových prohlížečích známou jako *fingerprint* – jedná se o otisk `DER` formy certifikátu pomocí zvolené hashovací funkce (nejčastěji SHA-1 nebo SHA-256) – v případě této implementace se jedná o SHA-1. Hexadecimální reprezentace otisku představuje pro účely této práce jednoznačný identifikátor certifikátu.

Kapitola 4

Testování

Zadání bakalářské práce požaduje otestování implementovaných útoků na speciálně připravených certifikátech. K jejich vzniku je třeba vytvořit speciální, záměrně oslabenou instanci šifry RSA, která bude náchylná k prolomení pomocí jednotlivých implementovaných útoků. V této kapitole proto popisuji proces nalezení záměrně slabých parametrů pro šifru RSA, jejich aplikaci, vytvoření standardního formátu klíčů a následně i certifikátů.

4.1 Tvorba testovacích certifikátů

V rámci tvorby záměrně slabých klíčů jsem se rozhodl vytvářet klíče délky 2048 bitů dle doporučení FIPS 186-3[14] – a také proto, že taková délka klíče je nejčastější[1]. Nalezené parametry jsem do standardní podoby ve formátu `PEM` snadno převedl za pomocí modulu `Crypto.PublicKey.RSA` jazyka Python – program je obsažen na přiloženém CD.

4.1.1 Příliš blízká prvočísla p, q

Nalezení záměrně oslabené instance šifry RSA s velmi malým rozdílem zvolených prvočísel je triviální. Instanci jsem vytvořil vygenerováním 1024-bitového prvočísla p pomocí nástroje `openssl prime` z příkazové řádky[26]:

```
$ openssl prime -generate -bits 1024
160830782557826086255408586193761971121442118372472145500301652795033947227766219916749120960126418797758852156198232377201612482273170099423459869540494384165050088833965082562383206437322563648927080789814938427704305067225148235449446755732287855514474365565009970367161518797526617298237257651887629848979
```

Ukázka kódu 4.1: Vygenerování 1024-bitového prvočísla pomocí nástroje openssl prime.

[26] Úvodní znak `$` naznačuje, že se jedná o příkaz v příkazové řádce – není součástí příkazu.

Následně jsem využil metody `nextProbablePrime` z knihovny `java.math.BigInteger` jazyka Java pro nalezení nejbližšího vyššího prvočísla:

```
BigInteger p = new BigInteger("1608 ... 8979"),
           q = p.nextProbablePrime();
```

Ukázka kódu 4.2: Nalezení nejbližšího vyššího prvočísla v jazyce Java.

Získané prvočíslo q:

```
160830782557826086255408586193761971121442118372472145500301652795033947227766219916749120960126418797758852156198232377201612482273170099423459869540494384165050088833965082562383206437322563648927080789814938427704305067225148235449446755732287855514474365565009970367161518797526617298237257651887629849699
```

Uvedená prvočísla p, q se v desítkovém zápise liší v posledních 4 číslicích. Ve dvojkovém zápise se čísla liší pouze v 11. nejnižším bitu, tedy v 1013. nejvyšším bitu oproti doporučení specifikace [9, ANSI X9.31], které doporučuje nejméně jeden rozdílný bit v prvních 100 nejvyšších bitech čísel.

Poté jsem postupoval od druhého kroku standardního postupu uvedeného v části Generování RSA klíče 1.3.2.

Výsledkem je veřejný klíč, který nejeví žádné známky úmyslného narušení bezpečnosti, ale jeho modul lze téměř okamžitě faktorizovat pomocí Fermatovy faktorizace.

4.1.2 Příliš nízký tajný exponent

Veřejný klíč, který je napadnutelný Wienerovým útokem, s vysokou pravděpodobností nebude obsahovat veřejný exponent se standardní doporučovanou[24, NIST 800-78-4] hodnotou 65537. Při generování zranitelného klíče jsem opět postupoval dle části Generování RSA klíče 1.3.2, ovšem u výběru exponentů jsem využil *originálního* postupu pro generování RSA klíčů[8], kde se napřed zvolí tajný exponent a veřejný exponent se na jeho základě vypočítá. Jako tajný exponent jsem zvolil číslo 4242424242424241, jehož multiplikativní inverzí modulo $\varphi(n)$ je 2047-bitové číslo e, použité jako veřejný exponent.

Veřejný exponent e:

```
909136631468229673932136502656844894881717843477915616897643323592249712983227122901966012949051044989659264397931335831705605339050819714740456468417284953808346773963685943552242508430737074481594731321435753201674807530608772321557385159042529623835636900782854378944510682338188180872932473905883091373293009412262085246872288743610319528573192862774616552631220142963586374927446404612 85
```

Výsledkem je veřejný klíč, který ovšem už na první pohled jeví známky nestandardního způsobu vytvoření, neboť veřejný exponent má namísto obvyklé hodnoty 65537 nezvykle velkou velikost srovnatelnou s modulem.

4.1.3 Hladká čísla $p-1, q-1$

Poslední záměrně oslabený klíč je tvořen prvočísly p, q, pro která platí, že $p-1$, případně $q-1$, jsou hladká čísla. Pro vygenerování takových čísel jsem vytvořil krátký program v jazyce PHP, který využívá knihovny pro práci s velkými čísly GMP (GNU Multiple Precision). Program je přiložen na doprovodném CD.

Algoritmus 8. Generování prvočísel nevhodných pro RSA
Výstup: 1024-bitové prvočíslo p, kde $p-1$ je 1000-hladké;
$primes = \{2, 3, 5, \ldots, 997\}$; `// Prvočísla menší než 1000`
$p = 2$; `// testované číslo p`
while ***true*** **do**
  if BitLength$(p+1) <$ *1024* **then**
    $X =$ náhodný prvek $\in primes$;
    $p = p \cdot X$;
  else if BitLength$(p+1) =$ *1024* $\wedge$ $p+1$ *je prvočíslo* **then**
    return $p+1$;
  else
    $p = 2$; `// musíme začít znovu`
end

Pomocí uvedeného algoritmu jsem vygeneroval dvě 1024-bitová čísla p, q, pro která platí, že $p-1$, respektive $q-1$, mají řadu malých prvočinitelů.

Jako příklad uvádím prvočíslo p:

```
89972811204097954593630349958521842930960606864238468354055552344532903039401242183603663650582292075770329638598535458282987761002642617391857756125563737303910856065646500534915514773178769835910449858523286767236978944128056013112895632074362786896740722740122431636447471273647859747500587153238569367501
```

a prvočíselný rozklad $p-1$:

$$\begin{aligned} p-1 \;=\; & 2^2 \cdot 3 \cdot 5^4 \cdot 7^3 \cdot 11^3 \cdot 13 \cdot 17^6 \cdot 19^3 \cdot 29 \cdot 31 \cdot 37^4 \cdot 41^2 \cdot 47 \cdot 53^2 \cdot 59^3 \\ & \cdot\; 61^3 \cdot 67^6 \cdot 71 \cdot 73^2 \cdot 79^2 \cdot 83^3 \cdot 89^3 \cdot 97 \cdot 103 \cdot 107^2 \cdot 109^2 \cdot 113^2 \cdot 127^3 \\ & \cdot\; 131^2 \cdot 137^3 \cdot 139^2 \cdot 149^4 \cdot 151^3 \cdot 157^4 \cdot 163 \cdot 167 \cdot 173^2 \cdot 179^3 \cdot 181^5 \\ & \cdot\; 191^3 \cdot 193^2 \cdot 197^4 \cdot 199^4 \cdot 223^2 \cdot 227^4 \cdot 229 \cdot 239 \cdot 241 \cdot 251^2 \cdot 257^2 \\ & \cdot\; 263 \cdot 269^3 \cdot 271 \cdot 277^3 \cdot 281^2 \cdot 283^4 \cdot 307 \cdot 311^3 \cdot 313^3 \cdot 317^5 \cdot 331 \\ & \cdot\; 337 \cdot 347 \cdot 349 \cdot 353 \end{aligned}$$

Číslo $p-1$ je tedy 353-hladké. Analogickým postupem jsem vygeneroval prvočíslo q.

4.1.4 Těžké instance certifikátů

Způsobem popsaným v předešlých třech podkapitolách jsem také vygeneroval záměrně obtížnější, ale stále velice slabé instance certifikátů.

4.1.4.1 Příliš blízká prvočísla p, q

Míra úspěšnosti Fermatovy faktorizace ve fixní časové dotaci přímo závisí na blízkosti prvočísel p, q. Proto k tvorbě dostatečně obtížné, ale stále velice slabé instance stačilo zvolit prvočísla dostatečně vzdálená. Vybral jsem 1024-bitová prvočísla, jejichž rozdíl je roven přibližně hodnotě 20^{158}. Instance vyžadovala 34 121 345 iterací pro úspěšnou faktorizaci modulu, což na testovacím hardwaru odpovídá přibližně 12 minutám procesorového času.

4.1.4.2 Příliš nízký tajný exponent

Standardní časová dotace je z efektivní povahy Wienerova útoku dostatečná pro dokončení všech dílčích výpočtů. Vytvořil jsem tedy 2048-bitový RSA klíč, jehož tajný exponent má velikost 520 bitů. Tím pádem tajný exponent přesahuje hranici nejvyššího možného zranitelného exponentu o několik jednotek bitů a Wienerův útok na této instanci neuspěje, i když během několika minut na testovacím hardwaru prověří všechny kandidáty.

4.1.4.3 Hladká čísla $p - 1, q - 1$

Obtížnější nalezení prvočinitelů p, q při použití Pollardovy $p - 1$ metody je zajištěno vyšší mezí B hladkosti čísel $p-1$, respektive $q-1$. Pro instanci, kterou není možné na testovacím hardwaru v časové dotaci 600 vteřin vypočítat, jsem zvolil mez $B = 1009$. Instance vyžadovala 4314 iterací pro faktorizaci modulu, tedy téměř dvojnásobek standardní časové dotace, což na testovacím hardwaru odpovídá přibližně 20 minutám procesorového času.

4.1.5 Podepsání certifikátů

Pro dosažení maximální věrohodnosti reálné situace jsem ze speciálně vybraných parametrů vytvořil standardní strukturu certifikátu X.509 (formát `PEM`), abych je mohl nasadit na testovacích doménách. V době testování společnost StartCom Ltd. rozšířila nabídku vystavení DV certifikátů 1.1.3 v omezeném počtu žadatelům zdarma. Této možnosti jsem využil a na své osobní webové stránce `martincernac.cz` vytvořil tyto subdomény třetího řádu:

- `pollard.martincernac.cz`
- `wiener.martincernac.cz`[27]

[27] Nefunguje v prohlížečích Internet Explorer 11 a Mozilla Firefox 45, detaily jsem popsal v části Nalezená chyba v knihovně Mozilla NSS 4.1.6.

- `near-primes.martincernac.cz`

Současně jsem vytvořil subdomény pro obtížné instance certifikátů a pojmenoval subdomény analogicky s předponou `hard`:

- `hard-pollard.martincernac.cz`
- `hard-wiener.martincernac.cz`[28]
- `hard-near-primes.martincernac.cz`

Následně jsem vytvořil nové certifikáty s parametrem CN[29] odpovídajícím jednotlivým doménám. Jako RSA klíče jsem použil předem speciálně vytvořené slabé, snadno napadnutelné klíče. Pro domény s předponou `hard` jsem parametry volil tak, aby je na testovacím hardwaru nebylo možné v časové dotaci kompromitovat. Pro vytvořené certifikáty jsem pomocí nástroje `openssl` vytvořil Certificate signing request (CSR), tedy požadavek pro podepsání certifikátu certifikační autoritou ve formě souboru obsahujícího požadované parametry certifikátu a veřejný klíč. Tyto CSR jsem zaslal certifikační autoritě StarCom Ltd., která mi obratem vystavila **podepsané, důvěryhodné, velmi snadno napadnutelné certifikáty**. Tyto certifikáty s dobou platnosti jeden rok jsem nasadil na výše zmíněné subdomény a zařadil do obsahu přiloženého CD.

4.1.6 Nalezená chyba v knihovně Mozilla NSS

V průběhu testování jsem si nechal společností StarCom Ltd. vystavit podepsané certifikáty s velmi slabými RSA klíči. V případě speciálního klíče, zranitelného Wienerovým útokem 1.4.3, jsem do veřejného klíče dosadil nezvykle dlouhý veřejný exponent. Při testování nasazení certifikátu se mi v prohlížeči Mozilla Firefox 45 nepodařilo otevřít SSL spojení, neboť prohlížeč zahlásil fatální chybu `SEC_ERROR_BAD_SIGNATURE`. Prohlížeč Mozilla Firefox využívá funkcí multiplatformní kryptografické knihovny NSS[30]. V rámci hledání příčiny této chyby jsem si opatřil zdrojové kódy NSS verze 3.23 z oficiálního repozitáře mozilla.org, přičemž jsem zjistil, že chyba nastane při pokusu o zápis do právě zahajovaného SSL spojení (funkce `PR_Write`, pomocí NSS nástroje `vfyserv`). Knihovna je ovšem schopna certifikáty získat a načíst. Pomocí přepínače `-c` zmíněného nástroje se mi je podařilo uložit do souborů. Ověření takto získaných certifikátů pomocí nástroje `vfychain` proběhne úspěšně – certifikát tedy rozhodně nemá neplatný podpis, jak naznačuje fatální chyba `SEC_ERROR_BAD_SIGNATURE`.

[28] Stejně jako subdoména `wiener.martincernac.cz` nefunguje v některých prohlížečích.
[29] Common Name, subjekt certifikátu
[30] Mozilla Network Security Services

Spojení se nepodaří navázat ani v prohlížeči Internet Explorer verze 11, ten ale přesnou příčinu uživateli nesděluje. Protože se nejedná o software s dostupným zdrojovým kódem, nemohl jsem příčinu chyby blíže lokalizovat.

V ostatních testovaných prohlížečích (Google Chrome 50.0.2661.94, Chromium 49.0.2623.108, Opera 36.0.2130.65) a jiném softwaru, který nepoužívá knihovnu Mozilla NSS, je certifikát zpracován správně a spojení je úspěšně navázáno. Tuto skutečnost jsem otestoval ve zmíněných prohlížečích a nástrojích `openssl s_client` 1.0.2g, `wget` 1.17.1 a `curl` 7.47.0.

Vytvořil jsem chybové hlášení, které jsem zaslal do aplikace Bugzilla (informační systém cílený na chyby v projektech společnosti Mozilla). Vývoj řešení chyby sleduji, doposud[31] se mému hlášení nedostalo žádné pozornosti. Hlášení je dostupné on-line [25].

4.2 Testování implementovaných algoritmů

Testování implementace jsem prováděl během vývoje, a to za pomoci jednotkových testů, které jsem vytvořil na základě běžných zadání. Během implementace jsem nalezl a vytvořil několik zadání, která testují části algoritmů, kde by algoritmus měl skončit neúspěchem, nebo testují části algoritmů, do kterých se velmi často v rámci časové dotace algoritmus nedostane.

4.2.1 Útok na příliš blízká prvočísla

Algoritmus Fermatovy faktorizace je implementován v podobě obecného faktorizačního algoritmu. Je tím pádem schopen faktorizovat všechna přirozená čísla, která lze použít jako modul pro šifru RSA. Cenou za obecnost algoritmu je vysoká asymptotická složitost pro čísla p, q, která si blízká nejsou. Vzhledem k nastavené fixní časové dotaci jsem do algoritmu nezačleňoval mez, přes kterou již algoritmus faktorizaci neprovádí. O včasné ukončení se postará hlavní vlákno programu po vypršení časové dotace. Implementovaný algoritmus nemá speciální zadání, která by bylo nutno testovat.

4.2.2 Útok na příliš nízký tajný exponent

Implementovaný útok na malý tajný exponent 1.4.3 obsahuje formulovanou limitaci, kdy pro tajný exponent d a modul n musí platit $d < \frac{1}{3}\sqrt[4]{n}$. V případě, že není tato nerovnost splněna, nelze garantovat nalezení správného kandidáta na hodnotu d. Příkladem může být následující veřejný klíč $(187, 107)$ a přidružený soukromý klíč$(187, 3)$, kdy i přes volbu nejmenšího možný tajného exponentu, není možné takový tajný exponent pomocí algoritmu řetězových zlomků nalézt, neboť $3 \not< \frac{1}{3}\sqrt[4]{187} \approx 1,232$.

[31] V době psaní této části práce – 1. května 2016.

4.2.3 Pollardova $p-1$ metoda

Pollardova $p-1$ metoda si v popsaném algoritmu 1.4.4 ve většině případů vystačí s proměnnou $a = 2$. Cílem otestování tohoto algoritmu tedy byla zadání, která vynutí přechod proměnné a k vyšším hodnotám než počáteční hodnotě $a = 2$.

Sestavení zadání, které nelze vyřešit s $a = 2$ Algoritmus při výpočtu nejvyššího společného dělitele při $a = 2$ kontroluje soudělnost modulu s čísly $2^k - 1$. Pro sestavení zadání tedy postupujeme obráceně. Snažíme se najít modul, který lze zapsat ve tvar $2^k - 1$ a který současně má pouze dva prvočíselné činitele, které nelze zapsat ve tvaru $2^{k'} - 1$. Ručním otestováním několika k lze nalézt $n = 2^{11} - 1 = 2047 = 23 \cdot 89$, kde ani jeden z prvočíselných činitelů nelze zapsat ve tvaru $2^{k'} - 1$.

Takové zadaní si během 103 iterací vynutí zvýšení hodnoty a z počáteční hodnoty $a = 2$ až na $a = 12$, kdy konečně $\gcd\left((12^{4!} \bmod n) - 1, n\right) = \gcd(534, n) = 89$.

4.3 Výsledky skenování

Měření jsem provedl dne 22. dubna 2016 a čase 16:47 až 17:53 SELČ. Během tohoto měření jsem se od 100 000 českých webových serverů pokusil získat jejich certifikáty a zkontrolovat jejich validitu. Případné chyby jsem zaznamenal a tabulka 4.5 shrnuje jejich četnost. Získaná data, včetně seznamu cílů (domén), lze nalézt na přiloženém CD.

Z dat vyplývá velmi tristní situace, kdy ve vzorku 8% všech aktivních doménových jmen v zóně .CZ **pouze 1338 domén ze vzorku 100 000 domén má platný a správně nasazený digitální certifikát**. Mezi těmito doménami se nachází velké vyhledávače (`centrum.cz`, `bing.cz`), internetové obchody (`tsbohemia.cz`, `datart.cz`, `bazos.cz`), webové stránky veřejných institucí (`bis.cz`, `czu.cz`) i velké komerční projekty (`zbozi.cz`, `o2videoteka.cz`, `csfd.cz`) a další. I přesto, že se jedná o náhodný výběr, díky velkému rozsahu se nejedná pouze o neznámé webové stránky s nulovou návštěvností.

Zaznamenal jsem celkem 61 939 webových serverů, které předložily digitální certifikát – bylo ovšem zaznamenáno pouze 7725 unikátních certifikátů. Tato skutečnost je zapříčiněna zejména velkými hostingovými společnostmi, které ve výchozím nastavení vždy předkládají stejný certifikát – který samozřejmě není validní (nenáleží předkládajícímu subjektu, není podepsán důvěryhodnou certifikační autoritou). Největší podíl v naměřených datech na tomto jevu má společnost WEDOS Internet, a.s., jejíž certifikát pro neexistující doménu `wedos.ws` a řadu jejích subdomén byl předložen 4970 různými

webovými servery. Několik dalších často[32] předložených certifikátů a jejich držitelů je zachyceno v tabulce 4.1.

Tabulka 4.1: Často užité nevalidní certifikáty – uvedeny jsou nálezy alespoň v 1% skenovaných domén.

držitel (parametr CN)	*self-signed*	četnost
webhosting.wedos.ws	ne	4970
*.savana-hosting.cz	ne	2050
*.web4u.cz	ne	1406
*.pipni.cz	ne	1292
vmm02.farma.gigaserver.cz	ano	1219
*.vshosting.cz	ne	1062

Z naměřených dat jsem dále vyčetl několik dalších statistických poznatků (četnosti zvolených parametrů).

Tabulka 4.2: Četnost užitých algoritmů pro podpis struktury certifikátu.

Algoritmus podpisu certifikátu	Absolutní četnost	Relativní četnost
SHA256withRSA	4495	58,20%
SHA1withRSA	2786	36,07%
MD5withRSA	227	2,94%
SHA256withECDSA	119	1,54%
SHA512withRSA	94	1,22%
SHA1withDSA	1	0,01%
SHA1withECDSA	1	0,01%
SHA384withRSA	1	0,01%

Dle dat z tabulky 4.2 je pouhých 121 certifikátů tvořeno podpisem bez použití šifry RSA. Z těchto 121 certifikátů s jiným typem podpisu je 109 tvořeno certifikáty zmíněné služby CloudFlare 1.2.2 a jedná se tím pádem o lehké zašumění dat. Je také patrný pozitivní trend ústupu tvorby podpisu pomocí MD5withRSA, který již není považován za kryptograficky bezpečný.

Překvapujícím zjištěním dat z tabulky 4.3 je velký nález klíčů o délce 1024 bitů, které by dle doporučení [15] neměla být používána. Takových klíčů bylo nalezeno více než dvakrát tolik nežli klíčů o velikosti 4096 bitů.

Veřejné exponenty se zmíněného FIPS doporučení drží přesněji – byly nalezeny pouze 4 certifikáty s jiným než běžným veřejným exponentem 65537, a to sice s hodnotou 17.

Nezávisle na celkové validitě certifikátu bylo v čas dokončení skenování nalezeno 6136 certifikátů (tedy přibližně 79%), které jsou z pohledu doby platnosti v pořádku (vstoupily v platnost a současně nevypršené). Z ostatních

[32] alespoň 1000 nálezů

Tabulka 4.3: Četnost délky modulů RSA veřejných klíčů certifikátu.

Délka modulu v bitech	četnost	Relativní četnost
2048	5840	76,81%
1024	1210	15,91%
4096	527	6,93%
512	10	0,13%
3072	10	0,13%
2432	3	0,04%
1040	1	0,01%
3120	1	0,01%
4092	1	0,01%
4095	1	0,01%

1589 certifikátů celkem 3 certifikáty nevstoupily v platnost a ostatních 1586 je vypršených.

Celkem 2219 certifikátů (přibližně 29%) je podepsáno obsaženým veřejným klíčem – jedná se o takzvané *self-signed* certifikáty.

Tabulka 4.4: Četnost certifikátů jednotlivých certifikačních autorit s alespoň 100 certifikáty.

Certifikační autorita	Absolutní četnost	Relativní četnost
GeoTrust	1325	17,15%
RapidSSL	1100	14,24%
COMODO	780	10,10%
StarCom	675	8,74%
Let‘s Encrypt	646	8,36%
Thawte	297	3,84%
Go Daddy	166	2,15%
AlphaSSL	129	1,67%

4.3.1 Anomálie

Z měření vyplynulo několik anomálií. Mnohé z nich pravděpodobně vznikly nepozorností osoby, která certifikáty vytvářela. Jedná se například o certifikát, který vstoupí v platnost až po své expiraci (záměna hraničních údajů platnosti)[33]. Zajímavým nálezem byl též certifikát, kde jako identifikátorem držitele je uvedena hvězdička (certifikát je tedy zamýšlen jako univerzální – platný pro jakékoliv doménové jméno)[34].

[33] V přiložených datech například certifikát `e368a74a36f97b27634dd3262daa1b8c2c3c4985`.

[34] V přiložených datech například certifikát `71a381132c322410b0a4931aa931194e52411c2f`.

Tabulka 4.5: Četnost výskytu chyb ve vzorku 100 000 domén. Pro každou doménu je evidována jen jedna chyba.

Popis problému	Četnost
CertificateException: No subject alternative DNS name matching <host> found.	38 663
CertificateException: No name matching<host> found	20 744
ConnectException: Connection refused	14 757
SocketTimeoutException: connect timed out	13 702
NoRouteToHostException: No route to host	3 046
SSLException: Unrecognized SSL message, plaintext connection?	2 691
SocketException: Connection reset	1 765
SSLHandshakeException: Remote host closed connection during handshake	1 533
SunCertPathBuilderException: unable to find valid certification path to requested target	984
SSLHandshakeException: Received fatal alert: handshake_failure	246
SocketTimeoutException: Read timed out	123
UnknownHostException: <host>	103
CertificateExpiredException: NotAfter: <date>	93
CertificateParsingException: no more data allowed for version 1 certificate	83
SSLException: Could not generate DH keypair	64
SSLHandshakeException: Server chose SSLv3, but that protocol version is not enabled or not supported by the client.	22
SSLHandshakeException: Received fatal alert: unrecognized_name	19
SSLException: Received fatal alert: protocol_version	18
SSLException: Received fatal alert: close_notify	2
EOFException: SSL peer shut down incorrectly	1
ConnectException: Invalid argument	1
CertificateException: Signature algorithm mismatch	1
SSLException: Received fatal alert: unexpected_message	1
Celkem webových serverů s nesprávně nasazeným digitálním certifikátem	**98 661**

4.4 Výsledky testování kvality

Výsledkem rozsáhlého skenování 4.3 je několik tisíc různorodých certifikátů. Pokud bych všem 7225 certifikátům ponechal standardní fixní časovou dotaci (600 vteřin), trvalo by testování kvality certifikátů více než 1200 hodin. Z praktických důvodů jsem tedy provedl náhodný výběr 100 certifikátů, kterým jsem pro otestování útoku ponechal časovou dotaci 600 vteřin. Celkem jsem tedy testování kvality vyhradil přes 16 hodin procesorového času na 4 vláknech aplikace. Tyto certifikáty jsou v přiloženém výstupním SQLite databázovém souboru označeny hodnotou `1` ve sloupci `processed`. Díky návrhu funkce aplikace, která zpracované certifikáty po uplynutí časové dotace tímto příznakem označí, lze fázi testování kvality přerušit a navrátit se později se zanedbatelnou časovou ztrátou.

Pomocí implementovaných útoků se mi ovšem nepodařilo kompromitovat žádné instance RSA veřejných klíčů. Tuto skutečnost přikládám na vinu malé časové dotaci. Z praktických důvodů ovšem není možné plošně přidělit časovou dotaci v měřítku hodin, neboť ani ta by neměla garantovanou šanci na úspěch útoku. V případě cíleného útoku na jednu konkrétní instanci šifry RSA by bylo možné v závislosti na délce modulu využít projektu *Factoring as a Service*, s pomocí kterého je možné provést faktorizaci RSA 512-bitového modulu během několika jednotek hodin za pomocí masivní paralelizace na platformě Amazon EC2. Takové prostředky jsem ovšem během bakalářské práce neměl k dispozici.

Zadání bakalářské práce vyžaduje otestování na speciálně vytvořené množině certifikátů, které mají úmyslně oslabené vybrané parametry. Tvorbu těchto certifikátů jsem popsal v kapitole 4.2, kde se zmiňuji i o jejich nasazení na několik veřejně dostupných domén třetího řádu. Výsledky tohoto testování jsou úspěšné, během časové dotace došlo ke kompromitace všech veřejných klíčů získaných ze záměrně oslabených parametrů a certifikátů tak podlehly testování kvality. Výstupní SQLite databázový soubor obsahující zjištěná data je taktéž přiložen na doprovodném CD.

Provedl jsem rovněž testování kvality na certifikátech s parametry zvolenými tak, aby je nebylo možné na testovacím hardwaru kompromitovat během standardní časové dotace 600 vteřin. U klíčů zranitelných na Pollardovu $p-1$ metodou a Fermatovu faktorizaci je to z důvodu přílišně náročnosti instancí dat, v případě Wienerova útoku se jedná o limitaci algoritmu. Výstupní SQLite databázový soubor je rovněž přiložen na doprovodném CD.

Z neúspěšného pokusu o kompromitaci skutečně nasazených digitálních certifikátů plyne pozitivní, ale nepřekvapivé pozorování – s **běžně**[35] dostupnými prostředky není možné v rozumném čase kompromitovat instanci šifry RSA, i přes užití modulu, který svou délkou zdaleka nesplňuje současná doporučení. Lze však kompromitovat instanci šifry, která byla záměrně oslabena.

[35] služba Amazon EC2 není chápána jako *běžný* prostředek

Některá oslabení navíc nejsou na pohled z veřejného klíče viditelná. V případě, že se však při použití šifry RSA u volby parametrů jako je délka klíče a velikost veřejného exponentu budeme držet doporučených množin hodnot, nehrozí nám v dohledné době reálné nebezpečí prolomení instance šifry.

Závěr

V rámci této bakalářské práce jsem analyzoval řadu známých útoků na šifru RSA a vybrané útoky jsem implementoval. Vytvořil jsem software pro shromáždění testovacích dat z českého Internetu a implementované algoritmy jsem na této množině dat otestoval. Útoky jsem podrobil také testování krajních případů a speciálně připravených instancí dat, k úspěchu těchto útoků náchylných. Zjištěné poznatky jsem shrnul a formuloval tvorbou krátké statistiky.

Čtenáři byl poskytnut přehled problémů spojených s praktickým nasazením digitálních certifikátů a představena současná řešení těchto problémů. Pro správné porozumění vybraných útoků na šifru RSA byl čtenář dále obeznámen se základními principy RSA, jejími možnostmi užití a limitacemi. Byly popsány vybrané útoky na šifru RSA, zvoleny útoky pro implementaci a shrnuty implementační volby pro strukturu programů a omezení z nich plynoucí. Vytvořených nástrojů bylo využito k oskenování reprezentativního vzorku webových serverů na českém Internetu a výsledky shrnuty. Tento proces by úspěšně zopakován pro testovací množinu oslabených instancí dat. Všechny stanovené požadavky se mi tedy podařilo v práci splnit.

Znepokojivým pozorováním je jednoduchost vytvoření velmi slabé instance šifry RSA (například pomocí příliš blízkých prvočísel p, q) a následné podepsání certifikátu obsahujícího takový RSA klič důvěryhodnou certifikační autoritou (díky automatizaci procesu vydávání *Domain Validated* certifikátů).

Přínosem této práce jsou vytvořené nástroje, které ulehčují správu webových serverů, zejména díky rychlému skenování rozsáhlých sítí a snadnému vyhodnocení výsledků díky užitému databázovému řešení. S pomocí těchto nástrojů lze též získat podklady pro přehled stavu nasazení a kvality digitálních certifikátů na veřejných sítích.

Osobním přínosem této práce je hlubší porozumění tématice šifry RSA a analyzovaných útoků na ni. Zajímavé pro mne též bylo shrnutí četnosti nálezů vybraných parametrů digitálních certifikátů a určení celkového procenta správně nasazených certifikátů v rámci testovaného vzorku.

Možná rozšíření práce zahrnují implementaci dalších útoků na šifru RSA,

zejména pokročilých metod faktorizace modulu, například metody kvadratického síta. Případným vylepšením může být paralelizace v současnosti implementovaných algoritmů pro testování kvality parametrů digitálních certifikátů. Zajímavé poznatky by mohlo přinést skenování a vyhodnocení kvality parametrů na větší testovací množině, ideálně zóně s dostupným seznamem aktivních domén – například na zóně .SK.

Literatura

[1] Kario, H.: July 2015 scan results [online]. jul 2015, [cit. 2016-04-14]. Dostupné z: https://securitypitfalls.wordpress.com/2015/07/29/july-2015-scan-results/

[2] Qualys, Inc.: SSL Server Test [online]. 2016, [cit. 2016-05-01]. Dostupné z: https://www.ssllabs.com/ssltest/

[3] SSL Shopper: SSL Checker [online]. 2016, [cit. 2016-05-01]. Dostupné z: https://www.sslshopper.com/ssl-checker.html

[4] Kemmerer, C.: DV OV and EV Certificates [online]. jul 2015. Dostupné z: https://www.ssl.com/article/dv-ov-and-ev-certificates/

[5] SK-NIC, a.s.: Seznam všech aktivních domén zóny .SK [online]. 2016, [cit. 2016-05-01]. Dostupné z: https://sk-nic.sk/documents/domeny.txt

[6] VeriSign, Inc.: Zone Files For Top-Level Domains (TLDs) - Verisign. [online]. apr 2016, [cit. 2016-04-24]. Dostupné z: https://www.verisign.com/en_US/channel-resources/domain-registry-products/zone-file/index.xhtml

[7] CZ.NIC, z. s. p. o.: Počet .cz domén - Statistiky CZ. [online]. apr 2016, [cit. 2016-04-16]. Dostupné z: https://stats.nic.cz/stats/number_of_domains/

[8] Rivest, R.; Shamir, A.; Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM*, ročník 21, č. 2, 1978: s. 120–126.

[9] American National Standards Institute: *ANSI X9.31-1998: Digital Signatures using Reversible Public Key Cryptography for the Financial Services Industry (rDSA)*. 1998.

[10] Boneh, D.: Twenty years of attacks on the RSA cryptosystem [online]. *Notices of the American Mathematical Society (AMS)*. Dostupné z: https://crypto.stanford.edu/~dabo/papers/RSA-survey.pdf

[11] Mihir Bellare, P. R. (editor): *Optimal Asymmetric Encryption - How to encrypt with RSA*, *Lecture Notes in Computer Science*, ročník 950, 1995.

[12] de Weger, B.: *Cryptanalysis of RSA with Small Prime Difference [online]*. 2002, [cit. 2016-04-11]. Dostupné z: https://pdfs.semanticscholar.org/c10a/86303ed615941bc1062e3512bdb3f59a5070.pdf

[13] Robert Erra, C. G.: The Fermat factorization method revisited [online]. jun 2009, [cit. 2016-04-13]. Dostupné z: http://eprint.iacr.org/2009/318.pdf

[14] STANDARDS, F. I. P.: FIPS PUB 186-3: Digital Signature Standard (DSS) [online]. jun 2009. Dostupné z: http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf

[15] Recommendation for Key Management: Part 3 - Application-Specific Key Management Guidance [online]. apr 2014. Dostupné z: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57Pt3r1.pdf

[16] Wiener, M. J.: Cryptanalysis of Short RSA Secret Exponents [online]. aug 1989, [cit. 2016-04-12]. Dostupné z: https://www.cits.ruhr-uni-bochum.de/imperia/md/content/may/krypto2ss08/shortsecretexponents.pdf

[17] Charest, A.-S.: Pollard's p-1 and Lenstra's factoring algorithms [online]. oct 2005, [cit. 2016-04-24]. Dostupné z: http://www.math.mcgill.ca/darmon/courses/05-06/usra/charest.pdf

[18] Kleinjung, T.: *Factorization of a 768-bit RSA modulus [online]*. 2010, [cit. 2016-04-11]. Dostupné z: http://eprint.iacr.org/2010/006

[19] Valenta, L.; Cohney, S.; Liao, A.; aj.: Factoring as a Service [online]. Cryptology ePrint Archive, Report 2015/1000, 2015, [cit. 2016-04-24]. Dostupné z: http://eprint.iacr.org/2015/1000.pdf

[20] Barnes, R.; Thomson, M.; Pironti, A.; aj.: Deprecating Secure Sockets Layer Version 3.0 [online]. RFC 7568 (Proposed Standard), Červen 2015, [cit. 2016-04-30]. Dostupné z: http://www.ietf.org/rfc/rfc7568.txt

[21] Kováč, P.: *Faktorizace velkých čísel pomocí knihovny GMP*. Diplomová práce, České vysoké učení technické v Praze, Fakulta elektrotechnická, 2007.

[22] ISO: *ISO 8601:1988. Data elements and interchange formats — Information interchange — Representation of dates and times [online].* 1988, 14 s., [cit. 2016-05-01]. Dostupné z: http://www.iso.ch/cate/d26780.html

[23] Blake-Wilson, S.; Nystrom, M.; Hopwood, D.; aj.: Transport Layer Security (TLS) Extensions [online]. RFC 4366 (Proposed Standard), Duben 2006, obsoleted by RFCs 5246, 6066, updated by RFC 5746. Dostupné z: http://www.ietf.org/rfc/rfc4366.txt

[24] National Institute of Standards and Technology: Cryptographic Algorithms and Key Sizes for Personal Identity Verification [online]. may 2015, [cit. 2016-05-01]. Dostupné z: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-78-4.pdf

[25] Černáč, M.: 1268235 – SEC_ERROR_BAD_SIGNATURE when a RSA certificate has a very large public exponent [online]. apr 2016, [cit. 2016-05-01]. Dostupné z: https://bugzilla.mozilla.org/show_bug.cgi?id=1268235

[26] Piacentini, M.: DB Browser for SQLite.

[27] Velebil, J.: *Diskrétní matematika: Text k přednášce [online].* Katedra Matematiky FEL ČVUT, 2007, [cit. 2016-04-15]. Dostupné z: ftp://math.feld.cvut.cz/pub/velebil/y01dma/dma-notes.pdf

Příloha **A**

Uživatelská příručka

V této příloze se nachází informace pro obsluhu programů, jejich požadavků pro správný běh a instrukce k případnému překladu zdrojových kódů do formy spustitelné aplikace. Popsána je i ukázka použití výsledných programů. Doporučeným nástrojem pro zobrazení a práci s databázovými soubory SQLite je *open-source* software *DB Browser for SQLite*, dříve znám jako *SQLite Database Browser*. Tento software je dostupný z oficiálního repozitáře na síti GitHub [26] a ve vybraných distribucích (Ubuntu) rovněž jako balíček.

A.1 Požadavky pro překlad a běh

Zdrojové kódy jsou napsány v jazyce Java a ke svému běhu tedy vyžadují přítomnost JRE[36] platformy Java. Přiložené spustitelné aplikace byly otestovány na platformě Oracle JRE 8, konkrétně verze 1.8.0_66-b17.

Pro překlad zdrojových kódů je rovněž vyžadována sada vývojových nástrojů pro platformu Java, tedy JDK[37]. Přiložené spustitelné aplikace byly přeloženy s pomocí Oracle JDK 8, konrkténě verze 1.8.0_66.

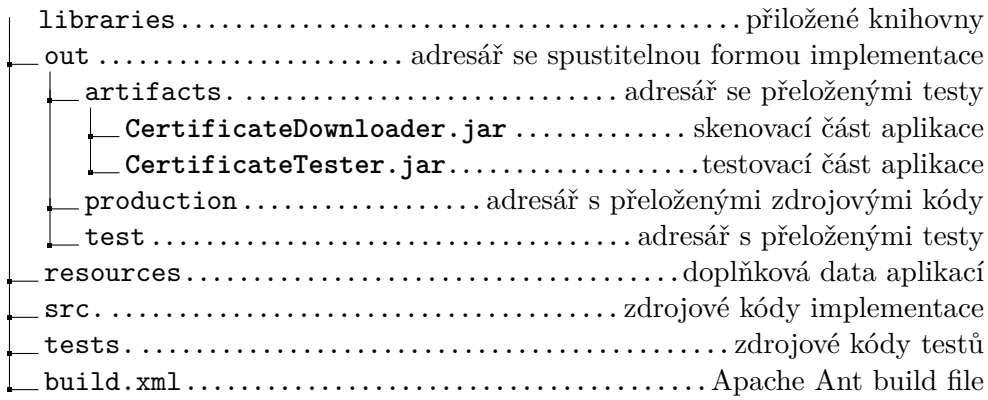
A.2 Překlad zdrojových kódů

Pro překlad zdrojových kódů je použito *open-source* řešení Apache Ant, které provede kompilaci zdrojových kódů a společně s knihovnami je zabalí do formy dvou spustitelných aplikací. Jedná se o multiplatformní aplikaci, jejíž požadavky pro běh jsou identické s potřebami překládané aplikace. Po spuštění příkazu[38] `$ ant` v příkazové řádce dojde k překladu dle definic ze souboru `build.xml` a k rozšíření projektové struktury o složku `out`. Novou adresářovou strukturu znázorňuje diagram A.1.

[36] Java Runtime Environment

[37] Java Development Kit

[38] Úvodní znak `$` naznačuje, že se jedná o příkaz v příkazové řádce – není součástí příkazu.



Obrázek A.1: Adresářová struktura projektu.

A.3 Spuštění aplikací a popis přepínačů

Přeložené aplikace se spouští z příkazové řádky a to následujícím způsobem:

```
$ java -jar CertificateDownloader.jar -output vystup.db -threads 25
```

Ukázka kódu A.1: Ukázka spuštění skenovací aplikace.

Analogicky k příkladu v ukázce kódu A.1 lze spustit testovací část aplikace `CertificateTester.jar`.

Aplikace si s sebou nesou nápovědu v případě zadání neplatné kombinace přepínačů. Tato stručná nápověda dokáže uživatele nasměrovat správným směrem. Podrobnější popis přepínačů je uveden v této kapitole.

Notace `<soubor>` značí cestu k souboru, `<číslo>` značí přirozené číslo.

A.3.1 Skenovací část aplikace

`-input <soubor>`

Tento přepínač umožňuje specifikovat textový soubor, obsahující doménová jména oddělená řádky, ze kterého se budou cíle pro skenování číst. V případě, že přepínač nebude použit, předpokládá se, že cíle skenování budou dostupné na standardním vstupu programu.

`-output <soubor>`

Jedná se o vyžadovaný přepínač, který jako argument nese cestu k souboru databáze SQLite, do kterého budou zapsány výsledky skenování. V kombinaci s přepínačem `-continue` musí daný soubor existovat, v případě vynechání přepínače `-continue` soubor naopak existovat nesmí, protože tím zahajujeme nové sezení pro skenování.

`-threads <číslo>`

Nepovinný přepínač umožňuje nastavit počet vláken, které budou paralelně

provádět skenování cílů. Nebude-li přepínač použit, bude použita výchozí hodnota 50 vláken.

`-continue <poslední zpracovaný záznam>`

S pomocí tohoto přepínače můžeme obnovit přerušené skenování, případně doplnit již existující databázový soubor o nové záznamy. Přepínač má nepovinný argument poslední oskenované domény – v případě, že bude atribut specifikován, vstupní data budou ignorována až do nálezu specifikované hodnoty, další záznamy doménových jmen již budou standardně zpracovány. Přepínač dále způsobí otevření existujícího (namísto vytvoření nového) výstupního souboru, uvedeného v argumentu přepínače `-output`.

A.3.2 Testovací část aplikace

`-data <soubor>`

Jedná se o vyžadovaný přepínač, jehož parametrem je soubor databáze SQLite, který byl vytvořen jako výstup ze skenovací části aplikace. Do databáze bude průběžně zapisováno – případné úspěchy útoků a označení zpracovaných certifikátů. Chod aplikace je tedy možné přerušit a znovu spustit později, případně je i možné přenést datový soubor na zcela jiný počítač a pokračovat v testování kvality parametrů tam.

`-timeout <číslo>`

Nepovinný přepínač umožňuje nastavit časovou dotaci pro zpracování jednoho certifikátu v jednotkách sekund. Nebude-li přepínač použit, bude použita výchozí hodnota 600 vteřin.

Příloha **B**

Matematické pojmy

Informace pro tuto část bakalářské práce jsem čerpal z vlastních poznámek z přednášek předmětu `BI-ZDM` a dalších zdrojů, zejména však ze skript Jiřího Velebila[27], ve kterých se nachází důkazy níže uvedených vět.

Definice 1 (Dělitelnost)**.** *Nechť a, b jsou celá čísla. Řekneme, že nenulové číslo $a, a \neq 0$ dělí číslo (je dělitelem čísla) b, pokud existuje takové celé číslo k, pro které platí $b = k \cdot a$. Zapisujeme $a|b$.*

Věta 2 (Dělení se zbytkem)**.** Nechť $a \in \mathbb{Z}$ a $d \in \ltimes^+$. Pak existují $q \in \mathbb{Z}$ a $r \in \ltimes$ taková, že $a = qd + r$ a $0 \leq r < d$. Čísla q (celočíselný podíl) a r (zbytek po dělení a číslem d) jsou jednoznačně určena.

Definice 2 (Prvočíslo)**.** *Nechť p je přirozené číslo. Řekneme, že p je prvočíslo, pokud existují právě 2 různá přirozená čísla, která dělí číslo p.*

Definice 3 (Složené číslo)**.** *Nechť n je přirozené číslo. Řekneme, že n je složené číslo, pokud existují alespoň 3 různá přirozená čísla, která dělí číslo n.*

Věta 3. Každé přirozené číslo $n, n > 1$ je buď prvočíslo, nebo složené číslo.

Definice 4 (Největší společný dělitel)**.** *Nechť a, b jsou celá čísla, z nichž alespoň jedno je nenulové ($a \neq 0 \vee b \neq 0$) Největší společný dělitel $gcd(a, b)$ je největší přirozené číslo d takové, které dělí obě čísla a, b – tedy $d|a$ a současně $d|b$. Pro $a = b = 0$, definujeme $gcd(0, 0) = 0$.*

Definice 5 (Nesoudělnost)**.** *Čísla a, b nazveme nesoudělná, když $gcd(a, b) = 1$. V opačném případě ($gcd(a, b) \neq 1$) jsou čísla soudělná.*

Definice 6 (Základní věta aritmetiky)**.** *Každé přirozené číslo $n, n > 1$ lze vyjádřit jako součin mocnin prvočísel:*

$$n = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_k^{\alpha_k}$$

kde $p_1 < p_2 < \ldots < p_k$ *jsou prvočísla,* $\alpha_1, \alpha_2, \ldots, \alpha_k$ *jsou přirozená čísla a* $k > 0$.

Tento zápis se nazývá prvočíselný rozklad čísla n *nebo prvočíselnou mocninnou faktorizací čísla* n.

Definice 7 (B-Hladké číslo)**.** *Číslo nazveme* B*-hladké, obsahuje-li jeho prvočíselný rozklad pouze prvočísla* $\leq B$*, přičemž* $B \in \ltimes$.

Algoritmus 9. Euklidův algoritmus

Vstup: celá čísla a, b, pro která platí $a \geq b > 0$;
Výstup: $gcd(a, b)$;
$t = 0$;
while $b \neq 0$ **do**
  $t = b$;
  $b = a \pmod{b}$;
  $a = t$;
end
return a

Definice 8 (Prvočíselná funkce)**.** *Prvočíselná funkce* $\pi : \ltimes^+ \mapsto \ltimes^+$ *pro* $\pi(n)$ *značí počet prvočísel* p *takových, že* $p \leq n$.

Věta 4 (Čebyševova nerovnost)**.** [27, B.1.6] Existují kladná reálná čísla A, B taková, že $0 < A < 1$ a $B > 1$ a platí nerovnost:

$$A \cdot \frac{n}{\ln n} \leq \pi(n) \leq B \cdot \frac{n}{\ln n}$$

pro všechna $n \geq 2$.

Definice 9 (Eulerova funkce)**.** *Eulerova funkce* $\varphi : \ltimes^+ \mapsto \ltimes^+$ *pro* $\varphi(n)$ *značí počet přirozených čísel* k *takových, že* $1 \geq k \geq n$ *a* $gcd(k, n) = 1$*. Dle konvence* $\varphi(1) = 1$.

Eulerova funkce má několik vlastností, kterých budeme v některých částech práce (zejména 1.3) využívat:

- Protože každé prvočíslo p má pouze 2 různé dělitele ($p|p$, $1|p$), platí $\varphi(p) = p - 1$.
- Nechť $m_1, m_2 \in \ltimes$, taková, že $gcd(m_1, m_2) = 1$. Potom platí $\varphi(m_1 \cdot m_2) = \varphi(m_1) \cdot \varphi(m_2)$.

V sekci 1.3, v části zabývající se generováním klíčů využijeme kombinaci těchto vlastností, a to sice:

$$n = p \cdot q, \text{ kde } p, q \text{ jsou prvočísla} : \varphi(n) = \varphi(p \cdot q) = (p - 1) \cdot (q - 1)$$

Vyčíslení vysoké hodnoty n pro $\varphi(n)$ je problémem faktorizace n, následován aplikací uvedených vlastností Eulerovy funkce. Nalezení prvočíselného rozkladu hodnoty n, která vznikla jako produkt několika vysokých prvočísel, je s běžně dostupnými výpočetními prostředky velmi náročná úloha, jejíž podrobnosti jsou popsány v části 1.4.1.

Věta 5. [27, A.1.1] Každý zlomek $\frac{a}{b}$, kde a, b jsou kladná přirozená čísla, lze zapsat ve tvaru

$$\frac{a}{b} = q_1 + \frac{1}{q_2 + \frac{1}{q_3 + \ldots \frac{1}{q_n}}}$$

kde $q_1, \ldots, q_n$ jsou přirozená čísla, $q_n \neq 0$. Čísla q_n nazýváme koeficienty.

Definice 10 (Řetězový zlomek)**.** *[27, A.1.3] Zlomek na pravé straně rovnosti [Věta 5] se nazývá řetězový zlomek čísla $\frac{a}{b}$ a budeme jej značit $[q_1; q_2; \ldots; q_n]$.*

V implementaci Wienerova útoku 1.4.3 se k výpočtu koeficientů řetězových zlomků využívá Euklidův algoritmus 9, který využijeme i v následujícím příkladu:

Příklad V příkladu budeme aplikovat popsanou metodu na problém nalezení řetězového zlomku výrazu $\frac{31}{407}$ za pomoci Euklidova algoritmu:

$$\begin{aligned} 31 &= 0 \cdot 407 + 31 \\ 407 &= \mathbf{13} \cdot 31 + 4 \\ 31 &= \mathbf{7} \cdot 4 + 3 \\ 4 &= \mathbf{1} \cdot 3 + 1 \\ 3 &= \mathbf{3} \cdot 1 + 0 \end{aligned}$$

Nalezli jsme tedy řetězový zlomek výrazu $\frac{407}{31} = [13; 7; 1; 3]$.

Příloha C

Seznam použitých zkratek

ACID Atomicity, Consistency, Isolation, Durability

ANSI American National Standards Institute

ccTLD Country code top-level domain

DDoS Distributed Denial of Service

DER Distinguished Encoding Rules

DNS Domain Name System

DoS Denial of Service

DSA Digital Signature Algorithm

ECDSA Elliptic Curve Digital Signature Algorithm

FIPS Federal Information Processing Standards

HTTPS HTTP s SSL/TLS

HTTPS Hypertext Transfer Protocol

ICT Information and Communication Technologies

IP Internet Protocol

IPv4 Internet Protocol version 4

JDBC Java Database Connectivity

JDK Java Development Kit

JKS Java KeyStore

JRE Java Runtime Environment

JSSE Java Secure Socket Extension

OAEP Optimal asymmetric encryption padding

OCSP Online Certificate Status Protocol

PEM Privacy-enhanced Electronic Mail

RDBMS Relational database management system

RFC Request for Comments

RSŘBD Český ekvivalent zkratky RDBMS

RSA Rivest, Shamir, Adleman (příjmení autorů šifry)

SHA Secure Hash Algorithm

SSL Secure Sockets Layer

SQL Structured Query Language

TCP Transmission Control Protocol

TLS Transport Layer Security

Příloha D

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `res` ..... doplňková data
  - `cert` ..... speciálně vytvořené certifikáty a tajné klíče
  - `scan` ..... vstupní, naměřená a zpracovaná data
  - `truststore` ..... použité důvěryhodné kořenové certifikáty
  - `util` ..... použité pomocné programy
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text a zadání práce
  - `BP_Cernac_Martin_2016.pdf` ..... text práce ve formátu PDF
  - `zadani.pdf` ..... zadání práce ve formátu PDF