Univerzita Pavla Jozefa Šafárika v Košiciach

Prírodovedecká fakulta

PROBLÉM FAKTORIZÁCIE V ASYMETRICKEJ KRYPTOGRAFII

BAKALÁRSKA PRÁCA

Študijný program:	Informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Ústav informatiky
Vedúci záverečnej práce:	RNDr. Rastislav Krivoš-Belluš, PhD.

Košice 2016

Ján Kotrady

Poďakovanie

Poďakovanie patrí môjmu vedúcemu práce RNDr. Rastislav Krivoš-Belluš, PhD. za odborné rady a pripomienky, ktoré si veľmi vážim a ktoré mi pomohli pri príprave tejto práce.

Namiesto tejto strany vložte
zadanie z informačného systému
podpísané vedúcim ústavu!

Abstrakt

Cieľom práce je analyzovať použitie problému faktorizácie v asymetrickej kryptografii. Práca popisuje jednotlivé faktorizačné algoritmy, analyzuje ich časovú zložitosť a porovnáva faktorizačné algoritmy v kontexte asymetrickej kryptografie implementovaním vybraných faktorizačných algoritmov. Práca je venovaná Fermatovej faktorizácii, Pollard $p - 1$ a Pollard ρ faktorizačným algoritmom. Vybrané faktorizačné algoritmy sú porovnávané na vopred pripravenej vzorke dát so špecifickými vlastnosťami. Okrem klasických metód faktorizácie sa práca zaoberá aj faktorizáciou algoritmom najväčšieho spoločného deliteľa. Táto časť práce poukazuje na stále existujúci problém chybne generovaných kľúčov a zaznamenáva signifikantné výsledky faktorizácie algoritmom najväčšieho spoločného deliteľa na reálnej vzorke verejných modulov. V práci sa nám podarilo faktorizovať až 66 modulov, 1024 až 2048 bitového kryptografického systému RSA, čo stále znamená veľké bezpečnostné riziko kryptografického systému RSA.

Kľúčové slova: faktorizácia, RSA, algoritmus, zložitosť

Abstract

The aim of this work is to analyze the usage of factorization problem in asymmetric cryptography. The thesis describes factorization algorithms, analyze their time complexity and compares factorization algorithms in the context of asymmetric cryptography, implementing selected factorization algorithms. The thesis is dedicated to the Fermat factorization, Pollard $p-1$ and Pollard ρ factorization algorithm. Selected factorization algorithms are compared with the previously prepared data sample with specific properties. Apart from classical factorization method, the work also deals with factorization with greatest common divisor algorithm. This section highlights the still existing problem of generated keys and records significant results of factorization with algorithm for the greatest common divisor of real samples of public modules. In this work, we have successfully factorized 66 modules from 1024 to 2048 bit RSA cryptosystem, which still represents a big security risk RSA cryptosystem.

Key words: factorization, RSA, algorithm, complexity

Obsah

Zoznam skratiek

RSA Rivest–Shamir–Adleman

M milión

nsd najväčší spoločný deliteľ

BUNV bez ujmy nad všeobecnosťou

SSH secure shell[1]

SSL secure sockets layer[2]

PGP pretty good privacy[3]

[1]Pozri RFC4253
[2]Pozri RFC6101
[3]Pozri RFC4880

Úvod

Asymetrická kryptografia je základnou a neoddeliteľnou súčasťou kryptografie. Jej princíp spočíva v použití rôznych kľúčov a metód na šifrovanie a dešifrovanie správy. Jeden z najúspešnejších a najviac rozšírených šifrovacích algoritmov v súčasnosti je RSA, ktorého zakladateľmi sú Ron Riverst, Adi Shamir a Leonard Adleman. A práve v kontexte RSA sa najviac spomína slovo faktorizácia, pretože bezpečnosť celého kryptografického systému RSA stojí (a padá) na probléme rozkladu čísla na súčin prvočísiel, takzvanú faktorizáciu. V tejto práci sa budeme zaoberať faktorizáciou Fermatovým algoritmom, Pollard $p-1$ a Pollard ρ algoritmom. Vybrané algoritmy budeme skúmať z hľadiska časovej zložitosti a ukážeme si, aké sú reálne možnosti faktorizácie klasickými faktorizačnými algoritmami. Porovnáme ich časové zložitosti na reálnej množine dát, ktorá má špecifické vlastnosti vopred pripravené tak, aby sme vyzdvihli jednotlivé prvky faktorizačných algoritmov. V druhej časti práce sa budeme zaoberať hlavne faktorizáciou pomocou algoritmu najväčšieho spoločného deliteľa a budeme sa sústrediť na chyby v kryptografickom systéme RSA. Pozrieme sa, aké existujú a ako reálne fungujú spätné dvierka v týchto kryptografických systémoch, ktoré sa môžu zneužívať na dešifrovanie správ a následnú stratu súkromia. Taktiež si ukážeme, kde a na akých serveroch tieto „chyby“ existujú, vysvetlíme si základné princípy týchto chýb a taktiež sa pozrieme na bezpečnosť RSA v kontexte klasickej faktorizácie. Na malej vzorke verejných RSA modulov, sme získali prekvapivé výsledky podobne, ako na to poukázali Lenstra a kol. v článku Ron was wrong, White was right v roku 2012. Na margo tohto, sa autori článku Ron was wrong, White was right nezaoberali kde a ako sa tieto moduly vyskytujú. Nám sa podarilo poukázať na niečo, prečo práve takáto faktorizácia pomocou algoritmu najväčšieho spoločného deliteľa nemusí byť práve náhoda.

Kapitola 1

Úvod do štúdia asymetrickej kryptografie

V prvej kapitole sa zameriame na základné princípy asymetrickej kryptografie . Uvedieme si základné definície kryptografických systémov, ich vlastnosti a príklady využitia. Budeme sa venovať hlavným vlastnostiam týchto kryptografických systémoch a ich matematickému základu. Taktiež si popíšeme a definujeme základné matematické funkcie, ktoré budeme neskôr využívať.

1.1 Definícia základných pojmov

Kryptografický systém formálne definujeme ako usporiadanú päticu so šifrovacími a dešifrovacími funkciami, pravidlami, ktorých definičný obor a obor hodnôt nemusí mať stále rovnakú mohutnosť. Formálna definícia podľa [1] hovorí o podmienke konečnosti množiny otvorených a šifrovaných textov a inverzii šifrovacej a dešifrovacej funkcie. Inač povedané, kryptografický systém nazývame takú usporiadanú päticu, v ktorej existuje šifrovacie pravidlo pre kľúč K a dešifrovacie pravidlo pre kľúč K tak, aby platilo $d_K(e_K(x)) = x$.

Definícia 1.1 Kryptografický systém nazývame usporiadanú päticu $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ kde:

1. $\mathcal{P}$ je konečná množina otvorených textov;

2. $\mathcal{C}$ je konečná množina šifrovaných textov;

3. $\mathcal{K}$ je konečná množina kľúčov;

4. Pre každé $K \in \mathcal{P}$, existuje šifrovacie pravidlo $e_K \in \mathcal{E}$ a korešpondujúce dešifrovacie pravidlo $d_K \in \mathcal{D}$, kde $e_K : \mathcal{P} \to \mathcal{C}$ a $d_K : \mathcal{C} \to \mathcal{P}$ sú funkcie také, že platí

$d_K(e_K(x)) = x$ pre každý otvorený text $x \in \mathcal{P}$ [1].

Kryptografia sa delí na dve základne časti a to na symetrickú kryptografiu a asymetrickú kryptografiu. Základný rozdiel je v tom, že v symetrickej kryptografii sa používa rovnaký kľúč na šifrovanie aj dešifrovanie dát, zatiaľ čo v asymetrickej kryptografii sa používa iný kľúč na šifrovanie a iný kľúč na dešifrovanie dát. Myšlienka skrytá za asymetrickou kryptografiou spočíva v nemožnosti získať d_K zo znalosti e_K. Výhoda asymetrickej kryptografie spočíva práve v možnosti zverejniť svoj *verejný kľúč* e_K tak, aby ktokoľvek mohol využívať tento verejný kľúč na šifrovanie správy a utajením *súkromného kľúča* d_K tak, aby osoba, ktorá zverejní verejný kľúč, bola schopná dešifrovať správu bez straty dôvernosti. V našej práci sa budeme venovať asymetrickej kryptografii a to pre jej nadväznosť na problém rozkladu čísla na súčin prvočísel, faktorizáciu. Hlavným reprezentantom asymetrickej kryptografie v kontexte faktorizácie je práve kryptografický systém RSA. Na to, aby sme definovali RSA kryptografický systém (RSA problém), potrebujeme definovať funkciu najväčší spoločný deliteľ.

Definícia 1.2 Definujeme funkciu nsd (najväčší spoločný deliteľ) z množiny $\mathbb{N}^2$ do množiny $\mathbb{N}$ tak, že ak $\text{nsd}(a, b) = n$, tak n je najväčšie číslo také, že n delí čísla a a b bez zvyšku.

Definícia 1.3 Definujeme RSA problém podľa [2] nasledovne: Nech je dané kladné prirodzené číslo n, ktoré je produktom dvoch odlišných nepárnych prvočísel p a q, kladné prirodzené číslo e také, že $\text{nsd}(e, (p-1)(q-1)) = 1$ a číslo c, pre ktoré existuje číslo m, také, že:

$$m^e \equiv c \bmod n.$$

Číslo $n = pq$ nazývame modul a číslo e verejný exponent.

Z definície RSA problému vyplýva, že ak poznáme prvočísla p a q, dokážeme RSA problém vyriešiť. Číslo n je teda zložené číslo z dvoch prvočísel p a q. Rozklad čísla na prvočísla, v našom prípade čísla n, na prvočísla p a q sa nazýva faktorizácia.

Definícia 1.4 Nech $n \in \mathbb{N}, n \geq 2$. Prvočíselný rozklad (faktorizácia) označíme každý zápis $p_1^{m_1} \cdot p_2^{m_2} \cdot \ldots \cdot p_k^{m_k}$, ktorý spĺňuje nasledujúce podmienky:
1. $p_1^{m_1} \cdot p_2^{m_2} \cdot \ldots \cdot p_k^{m_k} = n$
2. $k, m_1, \ldots, m_k \in \mathbb{N}$

3. $p_1, \ldots, p_k$ sú rôzne prvočísla.

Napríklad faktorizácia, alebo rozklad čísla na súčin prvočísel čísla 6 je $2 \cdot 3$. Vidíme, že rozklad malých čísel je jednoduchý. Uvedieme ale jedno číslo, modul, ktorý sa používa práve v asymetrickej kryptografii:

135066410865995223349603216278805969938881475605667027524485143851526510604859533833940287150571909441798207282164471551373680419703964191743046496589274256239341020864383202110372958725762358509643110564073501508187510676594629205563685529475213500852879416377328533906109750544334999811150056977236890927563

Vyššie uvedené číslo je 1024 bitový modul. Najväčší problém pri faktorizácii reálne používaných modulov v asymetrickej kryptografii je výber prvočísel, z ktorých sa tieto moduly vytvárajú. Ako môžeme vidieť na príklade vyššie, toto číslo nemá v prvočíselnom rozklade žiadne „malé“ prvočísla (2,3,5,7,...), ale využívajú sa dve rozdielne, približne 512 bitové prvočísla (v 1024 bitovom RSA), ktoré sú od seba „dostatočne vzdialené“ na to, aby nebolo možné previesť útok takzvanou odmocninou čísla a následnou aproximáciou.

Výhodou využívania asymetrickej kryptografie, s rozdielnym kľúčom na šifrovanie a dešifrovanie, je vlastnosť obslúžiť viac ľudí s použitím jedného kľúča. Napríklad, ak máme 20 ľudí, ktorí chcú medzi sebou komunikovať šifrovanou formou, pri použití symetrickej kryptografie by sme musel vygenerovať kľúč pre každú dvojicu. V tomto prípade je to počet kombinácií z 20, vyberáme 2 ľudí a tieto kombinácie robíme bez opakovania, celkovo 190 rozdielnych kľúčov, aby sa zachovala dôvernosť dát. Ak použijeme asymetrickú kryptografiu, tak v tomto prípade je to iba 20 verejných kľúčov. Ako sa ale ukazuje, asymetrická kryptografia a práve RSA nie je veľmi bezpečný kryptografický systém práve kvôli faktorizácii algoritmom nsd, o ktorom budeme písať neskôr a práve preto, že RSA je jedným z najrozšírenejších asymetrických kryptografických systémov, v ktorých sa využíva problém faktorizácie, sme si ho vybrali ako hlavného reprezentanta v tejto práci. Ďalej budeme skúmať jeho vlastnosti a spôsoby prelomenia tohto kryptografického systému.

1.2 Základy algebry

Aby sme mohli bližšie popísať problém faktorizácie v asymetrickej kryptografii, potrebujeme definovať niekoľko funkcií a viet z algebry. Medzi základné vety patrí Eulerova veta a Eulerova funkcia, ktoré sa využívajú pri dôkaze správnosti kryptografického systém RSA. Potrebujeme si ale ujasniť pár základných tvrdení z kongruencie a jej aritmetiky:

Definícia 1.5 Nech $m \mid a - b$, potom budeme písať, že

$$a \equiv b \bmod m.$$

Čítame a je kongruentné s b modulo m [3].

Medzi základné vlastnosti operácií s modulom patrí súčet, súčin, rozdiel ale aj krátenie. Tieto vlastnosti zhrnieme v nasledujúcej vete, ktorej dôkaz uvádzať nebudeme. Bližšie informácie a dôkazy sú dostupné v [3].

Veta 1.6 *Nech* $a \equiv b \bmod m$ *a* $c \equiv d \bmod m$. *Potom platí*

$$a + c \equiv b + d \bmod m,\ a - c \equiv b - d \bmod m,\ a \cdot c \equiv b \cdot d \bmod m$$

Naviac, pre každé a, b, c, m *platí*
1. $a \equiv b \bmod m \Leftrightarrow ca \equiv cb \bmod cm$;
2. ak c, m *sú nesúdeliteľné, potom* $a \equiv b \bmod m \Leftrightarrow ca \equiv cb \bmod m$

Definícia 1.7 *Eulerova funkcia* $\varphi(n)$ určuje pre $n \in \mathbb{N}, n > 1$ počet čísel z intervalu $1, \ldots, n-1$ nesúdeliteľných s číslom n.

Eulerova funkcia vyjadruje počet čísel, ktoré sú nesúdeliteľne s $n \in \mathbb{N}$. Ak n je prvočíslo, potom $\varphi(n) = n-1$. Nasledujúcu vetu nazývame Eulerova veta, ktorej dôkaz aj definíciu sme čerpali z [3]. Pripomeňme si, že čísla a, b nazývame **nesúdeliteľné**, ak platí, že $\mathrm{nsd}(a, b) = 1$.

Veta 1.8 (Eulerova veta) Ak sú a, m nesúdeliteľné, potom platí

$$a^{\varphi(m)} \equiv 1 \pmod m$$

K dôkazu sa nám bude hodiť jedná pomocná veta, ktorú ale pre jednoduchosť dokazovať nebudeme. Označme

$$m^* = \{k \in \{1, \ldots, m-1\} : \mathrm{nsd}(k, m) = 1\}.$$

Eulerovu funkciu potom môžeme zapísať ako $\varphi(m) = |m^*|$

Veta 1.9 *Nech a, m sú nesúdeliteľné čísla, definujeme zobrazenie*

$$f_a : m^* \to m^*$$

$$x \mapsto ax \bmod m$$

Potom je zobrazenie f_a bijekcia.

Pristúpime k dôkazu Eulerovej vety.

Dôkaz.
Uvážme nasledujúci výpočet, kde f_a je zobrazenie definované v predchádzajúcej vete:

$$\prod_{b \in m^*} b = \prod_{b \in m^*} f_a(b) = \prod_{b \in m^*} ab \bmod m \equiv \prod_{b \in m^*} ab = a^{\varphi(m)} \cdot \prod_{b \in m^*} b \pmod{m}.$$

Prvá rovnosť platí vďaka tomu, že v oboch prípadoch násobíme cez všetky prvky množiny m^*, iba v rôznom poradí. Označíme

$$c = \prod_{b \in m^*} b,$$

práve sme ukázali, že

$$c = a^{\varphi(m)} \cdot c \pmod{m}.$$

Číslo c je nesúdeliteľné s m (pretože je súčinom čísel nesúdeliteľných s m), takže im môžeme podľa Vety 1.6 krátiť a dostávame $1 \equiv a^{\varphi(m)} \pmod{m}$ [3].

□

Nakoniec si uvedieme vetu, ktorá bude slúžiť na dôkaz korektnosti Euklidovho algoritmu na výpočet najväčšieho spoločného deliteľa a samotný rozšírený Euklidov algoritmus.

Veta 1.10 *Nech $a, b \in \mathbb{N}, a > b > 0$. Potom $nsd(a, b) = nsd(b, a \bmod b)$* [3].

Dôkaz.
Nech $a, b \in \mathbb{N}$, potom:

$$\mathrm{nsd}(a, b) \mid a \wedge \mathrm{nsd}(a, b) \mid b \ \text{(Definícia 1.2)}$$

$$\Rightarrow \mathrm{nsd}(a, b) \mid (a - qb), q \in \mathbb{Z} \ \text{(celočíselný násobok, dôsledok Vety 1.6)}$$

$$\Rightarrow \mathrm{nsd}(a, b) \mid r, r = a - qb$$

$$\text{nsd}(a,b) \mid \text{nsd}(b,r)$$

A súčasne platí:

$$\text{nsd}(b,r) \mid b \wedge \text{nsd}(b,r) \mid r$$

$$\Rightarrow \text{nsd}(b,r) \mid (qb+r), q \in \mathbb{Z} \text{ (celočíselný násobok, dôsledok Vety 1.6)}$$

$$\Rightarrow \text{nsd}(b,r) \mid a, a = qb + r$$

$$\Rightarrow \text{nsd}(b,r) \mid \text{nsd}(a,b)$$

Takže platí, že $\text{nsd}(a,b) = \text{nsd}(b,r)$

□

Existuje rozšírený Euklidov algoritmus a klasický Euklidov algoritmus. My si uvediem formálny dôkaz rozšíreného Euklidovho algoritmu, z ktorého odvodíme klasický Euklidov algoritmus. Dôkaz tohto algoritmu a samotný algoritmus sme čerpali z [3]. Obe tieto tvrdenia sa vzťahujú na Euklidovské obory, ktoré ale nebudeme v tejto práci bližšie rozoberať. Pre nás je podstatný fakt, že obor prirodzených čísel s operáciami sčítania, odčítania, násobenia a delenia sú taktiež Euklidovské obory. Euklidovské obory označujeme symbolom R. Euklidovské obory sú špecifické normou, ktorú označujeme $\nu(n)$. V obore prirodzených číslach označíme $\nu(n) = n$.

Algoritmus 1: Rozšírený Euklidov algoritmus

Vstup : $a, b \in R,\ \nu(a) \geq \nu(b)$.

Výstup: $\text{nsd}(a,b)$ a $u, v \in R$ splňujúce $\text{nsd}(a,b) = u \cdot a + v \cdot b$.

begin

- $a_0 \leftarrow a, u_0 \leftarrow 1, v_0 \leftarrow 0$
- $a_1 \leftarrow b, u_1 \leftarrow 0, v_1 \leftarrow 1$
- **while** $a_{i+1} \neq 0$ **do**
  - $a_{i+1} \leftarrow r,\ u_{i+1} \leftarrow (u_{i-1} - u_i q),\ v_{i+1} \leftarrow (v_{i-1} - v_i q)$,
  - kde q, r volíme tak, aby
  - $a_{i-1} = a_i q + r$ a $\nu(r) < \nu(a_i)$
- **return** a_i, u_i, v_i

Veta 1.11 *Euklidov algoritmus nájde v Euklidovskom obore R pre akýkoľvek vstup $a, b \in R$ hodnotu $nsd(a,b)$ a nejaké $u, v \in R$ splňujúce*

$$nsd(a,b) = u \cdot a + v \cdot b.$$

Dôkaz.

Vzhľadom k tomu, že $\nu(a_0) \geq \nu(a_1) > \nu(a_2) > \ldots > \nu(0) \geq 0$, algoritmus sa musí po konečnom počte krokov zastaviť. Označme K číslo kroku, v ktorom sa tak stane. Je potrebné dokázať, že

$$\text{nsd}(a, b) = u_K \cdot a + v_K \cdot b = a_K.$$

Vzhľadom k tomu, že $\text{nsd}(a_K, 0) = a_K$ stačí dokázať, že nsd dvoch po sebe idúcich prvkov postupnosti $a_0, a_1, \ldots, a_K$ sa nemení, tj. že

$$(1)\ \forall i = 1, \ldots, K \text{ platí } \text{nsd}(a_{i-1}, a_i) = \text{nsd}(a_i, a_{i+1})$$

$$(2)\ \forall i = 0, \ldots, K \text{ platí } a_i = u_i \cdot a + v_i \cdot b$$

Tieto tvrdenia vyplývajú z vyjadrenia

$$a_{i-1} = a_i q + a_{i+1}$$

Pre dôkaz (1) si stačí uvedomiť, že dvojica a_{i-1}, a_i majú spoločného deliteľa ako dvojica a_i, a_{i+1} podľa Vety 1.10. Indukciou overíme (2). Pre $i = 0, 1$ zrejme tvrdenie platí. Ďalej predpokladajme, že platí $a_{i-1} = u_{i-1}a + v_{i-1}b$ a $a_i = u_i a + v_i b$, potom

$$a_{i+1} = a_{i-1} - a_i q = (u_{i-1}a + v_{i-1}b) - (u_i a + v_i b) \cdot q =$$

$$= (u_{i-1} - u_i q) \cdot a + (v_{i-1} - v_i q) \cdot b = u_{i+1}a + v_{i+1}b.$$

□

Algoritmus 1 popisuje rozšírený Euklidov algoritmus. Klasický Euklidov algoritmus nepočíta hodnoty u, v.

Algoritmus 2: Euklidov algoritmus [2]

Vstup : $a, b \in R,\ \nu(a) \geq \nu(b)$.

Výstup: $\text{nsd}(a, b)$.

while $b \neq 0$ **do**
  $r \leftarrow a \bmod b, a \leftarrow b, b \leftarrow r$.
return a

	Binárny Goppa kód	RSA	GRS kód
n	2,960	3,072	546
k	2,288	3,072	396
veľkosť kľúča	1,537,536	6,144	540,267
Zložitosť šifrovania	72	5,406	1,679
Zložitosť dešifrovania	15,302	6,643,013	3,228,153

Tabuľka 1.1: Porovnanie pôvodného McEliece k.s., RSA a vylepšenia GRS kódmi [4].

1.3 Prehľad súčasných asymetrických kryptografických systémov

Existuje mnoho asymetrických kryptografických systémoch, niektoré sú založené na probléme faktorizácie popísanej v časti 1.1, iné sú založené napríklad na probléme diskrétneho logaritmu. Diskrétny logaritmus spočíva v riešení rovnice $a^x \equiv b \bmod m$, čo sa ukazuje ako veľmi náročný problém. Zástupcom diskrétneho logaritmu v kontexte asymetrickej kryptografie je napríklad **Diffie–Hellman** protokol na výmenu kľúča alebo **ElGamal** kryptografický systém [2]. Taktiež poznáme asymetrické kryptografické systémy založené na probléme redukcie mreží [2]. Problém spočíva v nájdení redukovanej bázy, ktorá splňuje určité vlastnosti. Týmto kryptografickým systémom sa bližšie venovať nebudeme, ale spomenieme si jeden kryptografický systém, ktorý je kandidátom za nástupcu kryptografického systému RSA po príchode kvantových počítačoch, nazývaný **McEliece** kryptografický systém. Výhoda **McEliece** kryptografického systému spočíva práve vo využívaný iných princípov ako je problém faktorizácie alebo problém diskrétneho logaritmu [4]. Je založený na probléme, náročnosti, dekódovania lineárneho bloku kódu bez akejkoľvek viditeľnej štruktúry.

Pôvodný **McEliece** kryptografický systém adoptoval generátor matíc binárneho Goppa kódu ako súkromný kľúč a využil transformáciu a permutáciu matíc na skrytie súkromného kľúča vo verejnom kľúči [4]. Tento kryptografický systém je odolný voči kryptoanalýze aj po viac ako 30 rokov a dodnes nebol objavený polynomiálny útok na tento kryptografický systém, no vzrastajúca sila výpočtového výkonu a schopnosť optimalizovať útoky vytvorili potrebu aktualizovať jeho pôvodné parametre [5]. Práve tieto vylepšenia popisuje autor článku [4], ktorý bol vydaný v časopise Journal of Cryptology začiatkom roka 2016. Autor spomínaného článku [4] popisuje nielen vylepšenia **McEliece** kryptografického systému, ale aj návrhy bezpečnejšej formy

Názov	Problém	Publikácia
RSA	problém faktorizácie	1977
Rabin	problém faktorizácie	1979
ElGamal	diskrétny logaritmus	1985
McEliece	dekódovanie lineárneho kódu	1978
Merkle–Hellman knapsack	problém súčtu podmnožiny	1978
Chor–Rivest knapsack	problém súčtu podmnožiny	1988
Goldwasser–Micali probabilistic	kvadratický zvyšok	1982
Blum–Goldwasser probabilistic	problém faktorizácie	1984
Benaloh cryptosystem	kvadratický zvyšok	1994
Cayley–Purser algorithm	kvadratický zvyšok	1999

Tabuľka 1.2: Základné asymetrické kryptografické systémy.

šifrovania, ktoré sú ešte odolnejšie útokom ako v prípade pôvodného **McEliece** kryptografického systému. Tabuľka 1.1 porovnáva časovú zložitosť šifrovania, dešifrovania a veľkosť kľúčov medzi kryptografickým systémom RSA, pôvodným **McEliece** kryptografickým systémom a vylepšeniami popísanými v článku [4], použitím GRS kódov. Viac informácii k tejto problematike popisuje autor [4].

V Tabuľke 1.2 sú základné asymetrické kryptografické systémy, ktoré boli publikované v súčasnosti. Ako môžeme vidieť, mnohé z nich sa opierajú o problém faktorizácie, kde modul je generovaný ako v prípade RSA problému popísaný v Definícií 1.3. Napríklad v **Blum–Goldwasser probabilistic** kryptografickom systéme sa využíva ako verejný kľúč práve modul a ako súkromný kľúč sa využíva prvočíselný rozklad, verejného modulu a ďalej sa využívajú vlastnosti prúdových šifier na šifrovanie otvoreného textu. Zo znalosti faktorizácie verejného modulu sa spočíta inicializačný vektor prúdovej šifry a tým sa dešifruje šifrovaná správa. Výhodou tohto kryptografického systému je schopnosť šifrovať dlhšiu správu v porovnaní s RSA. Naopak, nevýhodou tohto kryptografického systému je jeho závislosť na bezpečnosti použitej prúdovej šifry.

Rabinov kryptografický systém taktiež využíva verejný modul ako v prípade kryptografického systému RSA a ďalej sa opiera o vlastnosti odmocniny v zvyškových triedach. Jeho nevýhoda spočíva v komplikovanejšom a časovo zložitejšom spôsobe dešifrovania otvoreného textu. Výhodou **Rabinovho** kryptografického systému je jeho bezpečnosť. Iba znalosť faktorizácie, rozkladu, je potrebná na dešifrovanie

šifrovaného textu a bolo dokázané, že prelomenie **Rabinovho** kryptografického systému je ekvivalentne problému faktorizácie v prípade útoku choosen plaintext attack [2], čo v prípade RSA nie je až také jednoznačné. Naopak, Rabinov kryptografický systém nie je bezpečný voči choosen ciphertext attack a existuje algoritmus, ktorý dokáže Rabinov kryptografický systém prelomiť. V kryptografickom systéme RSA sa stretávame s množstvom útokov, ktoré sú za určitých podmienok schopné prelomiť kryptografický systém RSA aj bez riešenia problému faktorizácie. Existuje niekoľko útokov na kryptografický systém RSA, ktoré sú bližšie popísané v [1, 2, 6] a niektoré útoky sú aj predmetom Kapitoly 2.

Kapitola 2

RSA

Definícia RSA problému, uvedená v Kapitole 1 ako Definícia 1.3, popisuje veľmi jednoducho problém, s ktorým kryptografický systém RSA pracuje. V tejto kapitole sa ale pozrieme na formálne teoretické vlastnosti kryptografického systému RSA. Kryptografický systém RSA nie je len o RSA problému. RSA problém formálne vyjadruje problém šifrovania a dešifrovania, na druhej strane nerieši spôsoby generovania kľúčov a nepoukazuje na formálnu stránku šifrovania a dešifrovania a na teórii, ktorá v tomto kryptografickom systéme platí. Keďže kryptografický systém RSA patrí medzi jeden z najrozšírenejších kryptografických systémov a je hlavným zástupcom problému faktorizácie v kontexte asymetrickej kryptografie, bola pre nás jednoznačná voľba zaradiť tento kryptografický systém medzi predmet štúdia tejto práce.

2.1 Základy RSA

Kryptografický systém RSA publikovaný autormi Ron Riverst, Adi Shamir a Leonard Adleman v roku 1977 je v súčasnosti jeden z najpoužívanejších kryptografických systémov [2]. Opiera sa o myšlienku Eulerovej vety, popísanej v Kapitole 1 ako Vetu 1.8, ktorú využíva na princíp šifrovania a dešifrovania. Využíva Eulerovu funkciu a Euklidov algoritmus na výpočet parametrov verejného a súkromného kľúča. Nasledujúci algoritmus, označený ako Algoritmus 3, popisuje spôsob generovania verejného a súkromného kľúča. Algoritmus 4 popisuje princíp šifrovania a dešifrovania správy a nasledujúca veta dokazuje správnosť kryptografického systému RSA. Dôkaz správnosti kryptografického systému RSA je možné predviesť dvoma spôsobmi. Je možné tento dôkaz opierať o Malú Fermatovu vetu alebo o Eulerovu vetu. V našom prípade zvolíme Eulerovu vetu, ktorá je aj súčasťou pôvodnej publikácie, v ktorej bol po prvý krát

predstavený kryptografický systém RSA. Princípy kryptografického systému RSA sú veľmi jednoduché na popísanie a korektné dokázanie správnosti. Aj tento fakt prispel k rozšíreniu kryptografického systému RSA, pretože jeden zo základných princípov kryptografie spočíva v jej jednoduchosti.

Môžeme vidieť, že generovanie kľúča je v podstate veľmi jednoduché. Jediným zložitým problémom je generovanie náhodných prvočísel. Zistiť, či číslo p je prvočíslo, nie je predmetom tejto práce, ale existujú algoritmy, ktoré riešia tento problém v polynomiálnom čase. Generovanie náhodných prvočísel spočíva v tom, že sa vyberie náhodné číslo potrebnej dĺžky a otestuje sa, či je toto číslo prvočíslo. Ak nie, generuje sa náhodné číslo odznova. Ponúka sa otázka, či vygenerované číslo je naozaj náhodne. Tento problém ďalej rozoberáme v Kapitole 4.

Algoritmus 3: Generovanie kľúča kryptografického systému RSA [2]

Výstup: Verejný a súkromný kľúč kryptografického systému RSA

1. Vygeneruj dve, približne rovnako veľké, rozdielne prvočísla p a q.
2. Spočítaj $n = pq$ a $\phi = (p-1)(q-1)$.
3. Vyber náhodné prirodzené číslo $e, 1 < e < \phi$, také, že $\text{nsd}(e, \phi) = 1$.
4. Pomocou rozšíreného Euklidovho algoritmu spočítaj unikátne prirodzené číslo $d, 1 < d < \phi$, také, že $ed \equiv 1 \bmod \phi$.
5. Verejný kľúč je (n, e), súkromný kľúč je d.

Len na pripomenutie definície RSA problému, Definícia 1.3, n nazývame verejný modul, e nazývame verejný exponent a d nazývame súkromný exponent. Princíp šifrovania a dešifrovania spočíva v reprezentovaní otvoreného textu ako číslo $m \in \mathbb{N}, 1 < m < n$ a následným umocnením čísla m na verejný exponent e.

Algoritmus 4: Šifrovanie a dešifrovanie kryptografickým systémom RSA [2]

1. Šifrovanie

Reprezentuj otvorený text ako číslo m z intervalu $[1, n-1]$

Spočítaj $c = m^e \bmod n$

2. Dešifrovanie

Spočítaj $m = c^d \bmod n$

Dôkaz nasledujúcej vety sme čerpali z článku [7], publikovaný autormi kryptografického systému RSA.

Veta 2.1 *Algoritmus 4 funguje.*

Dôkaz.
Z Algoritmu 3 vieme, že $ed = 1 \bmod \phi(n)$, kde $\phi(n)$ je Eulerova funkcia, $\phi(n) = (p-1)(q-1)$, $n = pq$ je verejný modul. Potom z definície kongruencie platí $\phi(n) \mid ed-1$, respektíve $k\phi(n) + 1 = ed$ pre nejaké prirodzené číslo k. Platí taktiež

$$m^{ed} \equiv m^{k\phi(n)+1} \bmod n$$

Z Eulerovej vety platí, že ak p nedelí m

$$m^{p-1} \equiv 1 \bmod p$$

potom platí

$$m^{k\phi(n)+1} = (m^{(p-1)})^{(q-1)k} m \equiv m \bmod n$$

a taktiež, ak q nedelí m

$$m^{q-1} \equiv 1 \bmod q$$

potom platí

$$m^{k\phi(n)+1} = (m^{(q-1)})^{(p-1)k} m \equiv m \bmod n$$

Obe tieto rovnice platia pre každé m, aj $m = 0 = p \equiv 0 \bmod p$, respektíve $m = 0 = q \equiv 0 \bmod q$. Potom spolu dostávame

$$m^{k\phi(n)+1} = (m^{\phi(n)})^k m \equiv m \bmod n$$

□

Z dôkazu Vety 2.1 vyplýva, že RSA kryptografický systém funguje pre ľubovoľné $m \in \mathbb{N}, m < n$, aj v prípade, že $m = p$ alebo $m = q$. V súčasnej dobe sa odporúča používať modul o veľkosti 2048 bitov [8]. Bezpečnosť 2048 bitového modulu RSA kryptografického systému sa považuje za ekvivalentnú bezpečnosti 112 bitového kryptografického systému AES [9]. Aktuálne sa ale neodporúča používanie príliš veľkých modulov (viac ako 4096 bitov), kvôli následnej chybe v DNS protokole [8].

Na záver poznamenajme, že kryptografický systém RSA je s postupom času na ústupe. S príchodom kvantových počítačoch sa stáva problém faktorizácie polynomiálny problém, čo ma za následok prelomenie RSA v reálnom čase. V roku 2010 bol publikovaný článok, v ktorom bol opísaný spôsob faktorizácie 768 bitového kryptografického systému RSA [10] a tento 768 bitový modul bol aj skutočne faktorizovaný. Tento fakt mal za následok prerušenie podpory 1024 bitových modulov kryptografického systému RSA v dôležitých inštitúciách ako sú napríklad vládne agentúry alebo armáda.

2.2 Faktorizácia v kontexte RSA

V Sekcii 2.1 sme načrtli, že faktorizovanie verejného RSA modulu rieši RSA problém, čo dokazuje aj nasledujúca veta. Práve z tohto dôvodu sme sa rozhodli zaradiť RSA kryptografický systém medzi reprezentanta problému faktorizácie v asymetrickej kryptografii. Ak poznáme prvočíselný rozklad verejného modulu kryptografického systému RSA, vieme v polynomiálnom čase prelomiť kryptografický systém RSA. Verejný modulu n je tvorený dvoma, približne rovnako veľkými prvočíslami. Potom $\phi(n) = (p-1)(q-1)$, to znamená, že podľa Algoritmu 3 vieme zo znalosti e, p, q, kde $pq = n$ vypočítať súkromný exponent d kryptografického systému RSA a následne vieme dešifrovať ľubovoľnú šifrovanú správu. Takýmto spôsobom sme schopní prelomiť kryptografický systém RSA bez toho, aby cieľ útoku tušil, že jeho kryptografický systém je prelomený.

Veta 2.2 *Problém výpočtu súkromného exponentu d kryptografického systému RSA zo znalosti verejného kľúča (n, e) a problém faktorizácie n, sú výpočtovo ekvivalentné* [2].

Dôkaz.
Dôkaz tohto tvrdenia priamo vyplýva z Algoritmu 3. Totižto $ed \equiv 1 \bmod \phi(n)$. Potom platí, že

$$ed = 1 + \phi(n)k, \text{pre nejaké } k \in \mathbb{N}$$

a ak poznáme prvočíselný rozklad n, potom $\phi(n) = (p-1)(q-1)$. Dosadením dostávame polynóm, ktorého riešenie je triviálne. Opačný smer je predmetom Vety 2.4.

□

V súčasnej dobe, klasickými faktorizačnými metódami, popísanými v Kapitole 3, sme schopní faktorizovať najviac 768 bitový verejný modul kryptografického systému RSA, ktorého faktorizácia je s časovou zložitosťou ekvivaletná výpočtu trvajúcemu 2000 rokov na 2.2 GHz procesore [10]. Samozrejme, existujú aj iné metódy faktorizovania verejného modulu, bližšie popísané v Sekcii 3.3, s ktorými sa nám podarilo faktorizovať až 2048 bitový verejný modul kryptografického systému RSA, pričom v tomto prípade nejde o klasický spôsob faktorizácie ľubovoľného verejného modulu a vieme takto faktorizovať verejné moduly len za určitých špecifických podmienok.

V prípade, že súkromný exponent RSA splňuje podmienku $d < \frac{1}{3}n^{(\frac{1}{4})}$, útočník vie v reálnom čase faktorizovať verejný modul kryptografického systému RSA a získať

súkromný exponent [6]. Tento útok sa nazýva Wienerov útok a využíva reťazový zlomok. Wienerov útok sa opiera o myšlienku približnej aproximácie $\frac{k}{d}$ ako $\frac{e}{\phi(n)}$ [6]. Algoritmus 5 sme čerpali z [11] a je to generalizácia Wienerovho útoku pre ľubovoľné $ed \equiv x \bmod n$.

Algoritmus 5: Generalizovaný Wienerov útok

Vstup : (n, e), kde $n = pq$ a $ex + y = 0 \bmod \phi(n)$, pre $0 < x < \frac{1}{3}n^{\frac{1}{4}}$ a $y \leq cn^{-3/4}ex$.

Výstup: p, q.

1. Spočítajte reťazový zlomok expanzie $\frac{e}{n}$

Pre každé konvergentne $\frac{k}{x}$:

2. Spočítajte $s = n + 1 - \frac{ex}{k}, t = \sqrt{s^2 - 4n}$ a $p' = \frac{1}{2}(s + t)$
3. Aplikujte Coppersmithov algoritmus na kandidáta $p' + (2k + 1)n^{\frac{1}{4}}$, pre $k = -3, -2, \ldots, 2$. Ak výstup z Coppersmithovho algoritmu je faktorizácia n, zastav.

Bližšie postupy a vysvetlenia Algoritmu 5 sú k dispozícií v [6, 11]. Coppersmithov algoritmus je popísaný v [11], kde ide o algoritmus, v ktorom máme na vstupe aproximáciu p a na výstupe faktorizáciu verejného modulu n

Veta 2.3 *Nech $n = pq$ je verejný modul kryptografického systému RSA. Nech máme aproximáciu p s maximálnou chybou $n^{\frac{1}{4}}$. Potom vieme faktorizovať n s časovou zložitosťou $log(n)$* [11].

Dôkaz Vety 2.3 je obsiahnutý v článku [11]. Pre jednoduchosť tento komplikovaný dôkaz uvádzať nebudeme, keďže by to zabralo väčšinu tejto práce.

Zaujímavý je aj spôsob faktorizácie verejného modulu zo znalosti súkromného aj verejného kľúča. Aj keď faktorizácia zo znalosti súkromného aj verejného kľúča neprispieva k prelomeniu kryptografického systému RSA, keďže poznáme súkromný kľúč, je na mieste tento algoritmus popísať. Tento spôsob faktorizácie môžeme využiť na overenie správnosti generovania verejného modulu alebo na spätnú rekonštrukciu algoritmu generovania verejného modulu, využiteľnú na iné vedecké účely.

Veta 2.4 *Nech n je verejný modul kryptografického systému RSA, d je súkromný kľúč a e je verejný kľúč. Potom zo znalosti n, e, d vieme faktorizovať n.*

Dôkaz.

Nech $k = de - 1$. Vieme, že k je násobok $\phi(n)$ a keďže $\phi(n)$ je nepárne číslo, potom

$k = 2^t r$, pre r nepárne a $t \geq 1$. Dostávame $g^{k/2} = 1$ pre každý generátor $g \in \mathbb{Z}_n^*$ (pozri [2], generátor cyklickej grupy, resp. Veta 1.8 $g^{\phi(n)k} \equiv 1 \bmod n$). To znamená, že $g^{k/2}$ je druhá odmocnina jednotky modulo n. Čínskou vetou o zvyškoch vieme, pozri [3], že existujú 4 korene druhej odmocniny z jednotky v grupe $\mathbb{Z}_n^*$. Dva korene sú ± 1 a zvyšné dva sú $x = 1 \bmod p$ a $x = -1 \bmod q$. Nasledovne výpočtom $\text{nsd}(x-1, n)$ dostávame faktorizáciu n. Výpočet tohto algoritmu má časovú zložitosť $O(n^3)$, kde $n = \log_2 n$.

□

Z dôkazu Vety 2.4 dostávame algoritmus faktorizácie n zo znalosti n, d, e. V podstate nám stačí spočítať k ako je uvedené v dôkaze Vety 2.4, zvoliť náhodný prvok g z množiny $\{2, \ldots, n-1\}$ a vypočítať g^k. Ak g^k splňuje určité podmienky uvedené v dôkaze Vety 2.4, dostávame faktorizáciu verejného modulu.

Algoritmus 6: Faktorizácia verejného modulu zo znalosti verejného a súkromného exponentu RSA

Vstup : n, e, d.

Výstup: p, q.

$k \leftarrow de - 1$

Zvoľ náhodné $g \in \{2, \ldots, n-1\}, t \leftarrow k$

while $x \leq 1$ **or** $nsd(x-1, n) \leq 1$ **do**

  if $2 \mid t$ **then** $t \leftarrow t/2, x \leftarrow g^t \bmod n$

  else Zvoľ náhodné $g \in \{2, \ldots, n-1\}, t \leftarrow k$

return $nsd(x-1, n), \frac{n}{nsd(x-1,n)}$

Na záver si uvedieme jednu vetu, ktorá určuje vzťah medzi problémom faktorizácie a RSA problémom. Nasledujúca veta znamená, že problém faktorizácie je aspoň taký náročný, ako je RSA problém. Respektíve, že kryptografický systém RSA je najviac taký bezpečný, ako je náročné riešiť problém faktorizácie.

Veta 2.5 *RSA problém $\leq_P$ problém faktorizácie, kde znak $A \leq_P B$ znamená, že algoritmus A je polynomiálne redukovateľný na B* [2].

Dôkaz.

Je to priamy dôsledok viet 2.2,2.3,2.4.

□

Kapitola 3

Základné faktorizačné algoritmy

V súčasnej dobe poznáme niekoľko faktorizačných algoritmov, ktoré dokážu riešiť RSA problém. Medzi základné algoritmy patria napríklad Pollard $p-1$ algoritmus, Lenstrov algoritmus, Fermatova faktorizácia, Pollard ρ alebo tie zložitejšie algoritmy ako napríklad Number field sieve alebo jeho zovšeobecnenie General number field sieve [2, 12]. Základné algoritmy ako Pollard's rho majú časovú zložitosť $O(\sqrt{p})$ modulárnych súčinov , kde p je prvočíslo [2]. Všetky tieto algoritmy dokážu riešiť RSA problém, avšak na druhej strane nepoznáme faktorizačné algoritmy, ktoré by dokázali vyriešiť RSA problém v reálnom čase. Jeden z najrýchlejších faktorizačných algoritmov, ktorý existuje, je (General) Number field sieve s časovou zložitosťou v Ln notácií $\mathrm{Ln}[\frac{1}{3}, c]$, kde $c = (92 + 26\sqrt{13})^{(1/3)}/3 = 1,9019$, popísaný už v roku 1993 [12]. Tento algoritmus bol neskôr vylepšený zmenšením konštanty c, na hodnotu približne 1,526285 aj to len pre špeciálne prípady faktorizovaných čísel [12]. Práve tento algoritmus stojí za faktorizovaním 768 bitového modulu [10]. Autori článku [10] tvrdia, že faktorizácia 768 bitového modulu by trvala viac ako 2000 rokov na jednovláknovom 2.2 GHz procesore a faktorizácia 1024 bitového modulu pomocou algoritmu Number field sieve je približne 1000 krát náročnejšia, ako faktorizácia 768 bitového modulu a naviac, v súčasnosti je štandardom používanie až 2048 bitového modulu. Tieto fakty a ďalšie tvrdenia popísane v nasledujúcich sekciách len potvrdzujú fakt, že schopnosť faktorizovať jedno číslo pomocou základných algoritmov, bez databázy prvočísel a pri veľkosti takej, aké sa používajú v súčasnosti v asymetrickej kryptografii, konkrétne v RSA, je v reálnom čase nemožné.

3.1 Pollard $p-1$ algoritmus

Pollardov $p-1$ algoritmus, publikovaný v roku 1974 [1], sa opiera o myšlienku Fermatovej vety, ktorá je ekvivalentná Eulerovej vete (Veta 1.8), zovšeobecnenej pre prvočísla [13]. Hľadáme prvočíslo p také, ktoré delí číslo n. Ak B je nejaký násobok čísla $p-1$, potom $p \mid \text{nsd}(z^L - 1, n)$, pre nejaké $z \in \mathbb{N}$ [13], kde $z^{p-1} \equiv 1 \bmod p$. Najprv si ale musíme definovať B hladké a mocninovo hladké čísla.

Definícia 3.1 Prirodzené číslo n nazývame B hladké, ak sú všetky prvočíselným deliteľom čísla n menšie ako číslo B, kde B je prirodzené číslo. Prirodzené číslo n nazývame B mocninovo hladké, ak sú všetky mocniny prvočíselným deliteľom čísla n menšie alebo rovné B, kde B je prirodzené číslo.

Vrátime sa späť k myšlienke Pollardovho $p-1$ algoritmu. Hľadáme p také, ktoré delí n. Z Eulerovej vety vieme, že

$$a^{p-1} \equiv \bmod p.$$

Pretože predpoklady Eulerovej vety sú splnené a hodnota $\phi(p) = p - 1$, môžeme priamo využiť Eulerovú vetu. Takýto tvar Eulerovej vety sa nazýva Fermatova veta. Predpokladajme, že $p-1$ je B mocninovo hladké. Potom ale musí platiť, že $p-1 \mid B!$. Potom z Eulerovej vety (a jej ekvivalentu Fermatovej vety) platí, že

$$a^{B!} = a^{\phi(p)^k} \equiv 1 \bmod \ \ p, k \in \mathbb{N}$$

pre nejaké $a \in \mathbb{N}, a > 1$. Z definície kongruencia potom platí, že $p \mid (a^{B!} - 1)$a $p \mid n$. Vyžitím algoritmu najväčšieho spoločného deliteľa dostávame $\text{nsd}(a^{B!} - 1, n) > 1$. Ak zvolíme B vhodné číslo, respektíve ak budeme B stále zvyšovať, dostaneme prvočíselný rozklad čísla n.

Pollardov $p-1$ algoritmus ma nevýhodu vo voľbe hranice B. Ak je voľba B nevhodná, algoritmus skončí neúspešne, no potom môžeme zväčšiť konštantu B a ak položíme konštantu $B = \sqrt{n}$, tak časová zložitosť tohto algoritmus je horšia ako skusmé delenie prvočíslami [1]. Práve preto je tento algoritmus vhodný na výpočet prvočíselného rozkladu takých čísel $n = pq$, ktorých prvočíselný rozklad čísla $p-1$ alebo $q-1$ sú malé čísla. V kontexte asymetrickej kryptografii je možné tomuto útoku odolať tak, že položíme $p = 2p_1 + 1$ a $q = 2q_1 + 1$, kde q_1, p_1 sú náhodné prvočísla približne rovnakej veľkosti [1]. Dôkaz správnosti funkčnosti tohto algoritmu sme popísali v myšlienke tohto algoritmu vyššie, preto ho v presnejšej forme uvádzať nebudeme.

Algoritmus 7: Pollard $p-1$ algoritmus

Vstup : n, B.

Výstup: p, q.

$a \leftarrow 2$

for $j \leftarrow 2$ **to** B **do**
  $a \leftarrow a^j \bmod n$

$d \leftarrow \text{nsd}(a-1, n)$

if $1 < d < n$ **then**
  return d

else
  return fail

Dôkaz sa opiera o Eulerovu vetu a je len formálnym zhrnutím myšlienky vyššie. Je zjavné, že tento algoritmus funguje a je jednoznačná jeho konečnosť. Zaujímavá je aj časová zložitosť tohto algoritmu, ktorá závisí práve na vhodnej voľbe konštanty B.

Veta 3.2 *Časová zložitosť Algoritmu 7 je* $O(B \log B(\log n)^2 + \log(n)^2)$.

Dôkaz.
V Algoritme 7 je presne $B-1$ modulárnych mocnení, každá najviac potrebuje $2\log_2 B$ modulárnych súčinov, využitím algoritmu binárneho mocnenia. Algoritmus najväčšieho spoločného deliteľa má časovú zložitosť $O(\log^2(n))$. Potom časová zložitosť celého algoritmu je $O(B \log^2 B(\log n) + (\log^2 n))$.

□

Zodpovedať na otázku, aké B zvoliť pri použití Algoritmu 7 je veľmi ťažké. Autor [2] odporúča použiť B v rozmedzí 10^5až 10^6. Ak algoritmus skončí neúspešne, odporúča sa použiť prvočísla väčšie ako B, označíme ich $q_1, q_2, \ldots, q_l$ a nasledujúcim výpočtom $a^{q_i}, \forall i \in \{1, \ldots, l\}$. Iní autori tvrdia, že je vhodná voľba $B = n^{1/6}$ a je ukázané, že takouto voľbou máme pravdepodobnosť faktorizácie rovnú $1/27$. Okrem tohto existujú aj iné vylepšenia tohto algoritmu, napríklad autor [2] nevyužíva zvolenie premennej a ako číslo 2, ale vyberá náhodné číslo a využíva mocnenie nie pre každé $j < B$ ale len pre prvočíselný rozklad čísla B. Algoritmus 7 sme implementovali použitím knižne GMP[1], v programovacom jazyku C, ktorá je prispôsobená na prácu s veľkými číslami, rýchlou aritmetikou násobenia, delenia a mocnenia, aby

[1]Knižnica GMP, The GNU Multiple Precision Arithmetics Library, dostupná na https://gmplib.org/

sme dosiahli približne rovnakú časovú zložitosť operácií, ako pri faktorizácii verejných modulov kryptografického systému RSA,

3.2 Pollard ρ algoritmus

Druhý faktorizačný algoritmus je Pollard ρ. Tento algoritmus už nezávisí od voľby konštanty B, ako to bolo v prípade Pollard $p-1$ algoritmu, ale závisí na vhodnej voľbe polynómu, vo väčšine prípadov x^2+1. Nech p je menšie (najmenšie) prvočíslo z prvočíselného rozkladu čísla $n = pq$ a predpokladajme, že existujú dve čísla $x, x' \in \{1, \ldots, n-1\}$ také, pre ktoré platí $x \neq x'$ a $x \equiv x' \bmod p$. Potom $p \leq \text{nsd}(x-x', n) < n$ a tým získavame faktor čísla n, pretože $p \mid x-x'$ a $p \mid n$ [1]. Ako ale tento princíp využiť bez znalosti čísla p ? Najprv si uvedieme jednoduchú vetu, ktorú neskôr využijeme.

Veta 3.3 *Nech $n, a, b \in \mathbb{N}$. Potom $n \bmod a = (n \bmod ab) \bmod a$*

Dôkaz.
Z definície kongruencie vieme, že $n = ka + z$, kde $k, z \in \mathbb{N}, z < a$ a čísla k, z sú určené jednoznačne. Taktiež z definície kongruencie vieme, že $n = k_1 ab + m, k_1, m \in N$ a čísla k_1, m sú určené jednoznačne a $n \equiv m \bmod ab$. Ak $m < a$, potom $m \equiv m \bmod a$, z čoho potom vyplýva, že $k = k_1 b$. Ak $m \geq a$, potom $m \equiv m' \bmod a$. Potom ale platí, že $m = ak_3 + m'$, kde $k_3, m' \in \mathbb{N}$. Ak spojíme tieto rovnice dostávame, že $n = k_1 ab + ak_3 + m' = a(k_1 b + k_3) + m'$ a čísla k_1, b, k_3, m' sú určené jednoznačne. Potom platí, že $m' = z$.

□

Nasledujúcu myšlienku sme čerpali z [1, 2]. Predpokladajme, že funkcia f je polynóm s prirodzenými koeficientami, napríklad $f(x) = x^2 + a^2$, kde a je malá konštanta a nech $x \mapsto f(x) \bmod n$ je zobrazenie, ktoré vyzerá ako náhodné. Nech postupnosť prvkov $\{x_n\}_1^\infty$ je definovaná takto $x_i = f(x_{i-1}) \bmod n, \forall i \geq 2$. Položme pre nejaké m konečnú podmnožinu množiny postupnosti $\{x_n\}_1^\infty$, tj. $X = \{x_1, \ldots, x_m\}$ a pre jednoduchosť predpokladajme, že množinu X tvoria unikátne prvky. Teraz budeme hľadať dva prvky množiny X také, pre ktoré plati $\text{nsd}(x_j - x_i, n) > 1$. V každom kroku, kedy spočítame nové x_i, by sme mali spočítať $\text{nsd}(x_j - x_i, n), \forall j < i$. Tento prístup sa ale dá vylepšiť jednoduchým trikom.

[2]Polynóm $x^2 + a$ má najlepšie cyklické vlastnosti pre tento spôsob faktorizácie, ale môžu sa využívať aj iné polynómi, napríklad $x^4 + 1$, kedy už počítanie štvrtej mocniny spôsobuje zhoršenie časovej zložitosti faktorizácie týmto algoritmom

Predpokladajme, že sme našli také i, j, pre ktoré platí, že $x_i \equiv x_j \bmod p$. Potom ale platí, že $f(x_i) \equiv f(x_j) \bmod p$ podľa dôsledku Vety 1.6. Pripomeňme si, že $x_{i+1} = f(x_i) \bmod n$ a $x_{j+1} = f(x_j) \bmod n$. Potom

$$x_{i+1} \bmod p = (f(x_i) \bmod n) \bmod p = f(x_i) \bmod p,$$

podľa Vety 3.3. Obdobne platí

$$x_{j+1} \bmod p = f(x_j) \bmod p.$$

Práve z tohto dôvodu dostávame, že $x_{i+1} \equiv x_{j+1} \bmod p$. Opakovaním tohto argumentu dostávame nasledujúci dôležitý výsledok:

$$\text{Ak } x_i \equiv x_j \bmod p, \text{potom } x_{i+k} \equiv x_{j+k} \bmod p, \forall k \in \mathbb{N}.$$

Ak si označíme $l = j - i$, znamená to, že $x_{i'} \equiv x_{j'} \bmod p$ ak $j' > i' \geq i$ a $j' - i' \equiv 0 \bmod l$. Inač povedané, počnúc nejakým i, dostávame $x_i \equiv x_j \bmod p$ pre nejakú fixnú dĺžku l. Ako sme už spomínali, hľadáme také $x_i \equiv x_j \bmod p$, kde $i < j$, počítaním najväčšieho spoločného deliteľa. Nie je potrebné nájsť hneď prvý takýto výskyt a práve z tohto dôvodu budeme počítať $j = 2i$, čím využijeme Floydov cyklus hľadajúci algoritmus [2].

Algoritmus 8: Pollard ρ algoritmus

Vstup : n, x_1.

Výstup: p také, že $p \mid$.

$x \leftarrow x_1$
$x' \leftarrow f(x) \bmod n$
$p \leftarrow \text{nsd}(x - x', n)$
while $p = 1$ **do**
    $x \leftarrow f(x) \bmod n$
    $x' \leftarrow f(x') \bmod n$
    $x' \leftarrow f(x') \bmod n$
    $p \leftarrow \text{nsd}(x - x', n)$
if $p = n$ **then**
    return fail
else
    return p

Vstupom do Algoritmu 8 je faktorizované n a nejaké, počiatočne zvolené x_1, väčšinou vyberané náhodne. Môže sa stať, že tento algoritmus nenájde prvočíselný rozklad čísla n. To sa ale stane len a len vtedy ak $x = x'$. Potom stačí zvoliť iné, náhodne počiatočné x_1 [1]. Dôkaz správnosti kvôli rozsahu uvádzať nebudeme, ale je ho možné nájsť v [14]. Ide totiž o veľmi komplexný dôkaz postavený na rozsiahlej algebrickej a analytickej teórii.

Časová zložitosť tohto algoritmu sa pohybuje približne, v O anotácií $O(\sqrt{p})$, v kontexte kryptografického systému RSA máme $O(n^{1/4})$. Dôkaz časovej zložitosti sme čerpali z [1, 2, 13]. Časová zložitosť tohto algoritmu je ale heuristická, keďže z formálneho dôkazu vyplýva mierne lepšia časová zložitosť. V reálnom prípade je ale skutočná časová zložitosť veľmi blízka časovej zložitosti odhadnutej v nasledujúcej vete.

Veta 3.4 *Časová zložitosť Algoritmu 8 je* $O(\sqrt{p})$

Dôkaz.

Ak $x_i \equiv x_j \bmod p$, potom je to taktiež prípad $x_{i'} \equiv x_{2i'} \bmod p$ pre každé i' také, že $i' \equiv 0 \bmod l, i' \geq i$. Po nejakých i krokoch dostávame medzi l po sebe idúcich prirodzených čísel $i, \cdots, j-1$, jedno číslo deliteľné l. Preto najmenšia hodnota i' ktorá splňuje tieto dve podmienky je najviac $j-1$. Odtiaľto, počet iterácií, ktoré sú potrebné na nájdenie deliteľa p čísla n je najviac $\sqrt{p}$. Totižto, ak by sme mali polynóm $x^2 + a$, po $\sqrt{p}$ krokoch dostávame číslo, ktoré sme už určite použili.

□

Tento dôkaz nie je úplne formálne správny. Ale veta, ktorá hovorí o aproximácii krokov výpočtu Algoritmu 8 pokrýva rozsah jednej tretiny bakalárskej práce. Ak by sme to chceli formulovať trochu odlišne, tak potom môžeme povedať, že na to, aby sme vygenerovali opakujúcu sa postupnosť potrebujeme aspoň $\sqrt{p}$ prvkov grupy Z_p, na čo sa dá využiť aj narodeninový paradox. Samozrejme takáto časová zložitosť sa aproximuje pre polynóm $x^2 + 1$. Bližšie k časovej zložitosť tohto algoritmu je možné čerpať z [14].

3.3 Fermatov faktorizačný algoritmus

Posledný klasický faktorizačný algoritmus je Fermatov algoritmus. Jeho zaradenie do výberu je spojené s tým, že tento algoritmus tvorí základ pre najrýchlejšie faktorizačné algoritmy. Pracuje s myšlienkou približnej odmocniny a následnej približnej aproxi-

mácie čísla $\sqrt{n}$. Samozrejme, pri použití faktorizačného algoritmu GNFS (najrýchlejší súčasný faktorizačný algoritmus), je aproximácia komplexná a pracuje s množstvom teórie z pravdepodobnosti, štatistiky, algebry a analýzy. Preto si uvedieme len základný Frematov faktorizačný algoritmus na bližšie pochopenie myšlienky faktorizácie skrytou za najväčšími úspechmi v súčasnosti [12, 13]. Zaujímavý je aj poznatok, že aj keď tento algoritmus bol objavený už v roku 1600, dodnes je používaný v kontexte faktorizácie algoritmom GNFS [15].

Myšlienka Fermatovej faktorizácie je skrytá za riešením rovnice $n = x^2 - y^2$ [15]. Túto rovnicu vieme upraviť na štvorec základným vzorcom ako $n = (x - y)(x + y)$ a predpokladajme, keďže sme v kontexte RSA, že $n = pq$. Potom budeme hľadať také p a q, ktoré splňujú podmienku $p = (x - y)$ a $q = (x + y)$. Riešením týchto rovníc je zjavne $x = \frac{(p+q)}{2}$ a $y = \frac{(p-q)}{2}$ [15]. Ďalej pre partikulárne riešenie týchto rovníc máme, že $x = \sqrt{n + y^2}$ a $y = \sqrt{x^2 - n}$. Ak si položíme na začiatok tohto algoritmu $x_1 = \lceil \sqrt{n} \rceil$, kde znak $\lceil \cdot \rceil$ značí celú hornú časť čísla a položíme $x_{i+1} = x_i + 1$, vieme pre každé i spočítať, či $y_i = \sqrt{x_i^2 - n}$ je prirodzené číslo a či platí, že $(x_i + y_i), (x_i - y_i)$ je netriviálny faktor čísla n. Ak sú obe podmienky splnené, dostávame netriviálny prvočíselný rozklad čísla n.

Algoritmus 9: Fermatov faktorizačný algoritmus

Vstup : n

Výstup: p, q

```
for x from ceil(sqrt(n)) to n do
    y ← x^2 − n
    if jeStvorec(y) then
        y ← sqrt(y)
        p ← (x − y)
        q ← (x + y)
        if p ≠ 1 and p ≠ n(y) then
            return p,q
```

Funkcie $jeStvorec(y)$ v tomto prípade testuje, či platí, že $\exists m \in \mathbb{N}, m^2 = y$. Algoritmus 9 je ešte efektívnejší s použitím zoznamu prvočísel alebo s využitím takzvaným sít, kedy nie je potreba testovať všetky čísla, či sú odmocninou ako testuje funkcia $jeStvorec$. Taktiež sa využíva Frematova faktorizácia s pokusným delením. Je zrejme, že tento algoritmus je možné vylepšovať viacerými spôsobmi. Naopak, časová zložitosť

tohto algoritmu nie je veľmi odlišná od skusmého delenia.

Veta 3.5 *Časová zložitosť Algoritmu 9 je v najhoršom prípade $O(n)$. Ak p a q sú približne rovnako veľké prvočísla, potom časová zložitosť Algoritmu 9 je $O(\sqrt{n})$.*

Dôkaz.
Vieme, že $x = \frac{(p+q)}{2}$ a $y = \frac{(p-q)}{2}$. Ak sú p a q blízke prvočísla, y spočítame veľmi jednoducho a faktorizácia je triviálna. Ak sú ale p a q dve, približne rovnako veľké prvočísla, potom potrebujeme prejsť všetky čísla v rozsahu $\log_2 \sqrt{n}$ bitov, aby sme dostali požadovaný výsledok. Naopak, ak n je prvočíslo, tak časová zložitosť je $O(n)$. □

3.4 Faktorizácia algoritmom najväčšieho spoločného deliteľa

V tejto časti sa budeme zaoberať faktorizáciou algoritmom najväčšieho spoločného deliteľa, konkrétne Euklidovým algoritmom, ktorý poskytuje s miernymi úpravami dostatočne rýchly algoritmus na získanie potrebných výsledkov. Euklidov algoritmus, popísaný v Kapitole 1, má časovú zložitosť $O(\log^2(n))$ [2]. Dôkaz časovej zložitosti Euklidovho algoritmu sa prevedie pomocou Fibonacciho čísel, kedy výpočet Euklidovým algoritmom najväčšieho spoločného deliteľa je logaritmický v závislosti od dĺžky vstupu. Euklidov algoritmus, na rozdiel od faktorizačných algoritmov popísaných vyššie, patrí do triedy kvadratických algoritmov, čo je v praxi výhodnejšie a hlavne rýchlejšie. Nevýhodou tohto algoritmu je potrebná postačujúca množina verejných modulov. Euklidov algoritmus dokáže nájsť **prvočíselný rozklad** modulov RSA kryptografického systému práve vtedy, keď existujú dve rozdielne čísla, povedzme $n_1 = p_1q_1$ a $n_2 = p_2q_2$, pričom platí, že buď $p_1 \mid n_2$ alebo $q_1 \mid n_2$. Euklidov algoritmus navyše dokáže spočítať rozklad takýchto modulov v reálnom čase, čím za určitých podmienok rieši problém RSA.

Veta 3.6 *Nech $n_1, p_1, q_1, n_2, p_2, q_2 \in \mathbb{N}, p_1, q_1, p_2, q_2$ sú prvočísla. Nech $n_1 = p_1q_1$ a $n_2 = p_2q_2$, pričom $n_1 \neq n_2$ a nech platí, že buď $p_1 \mid n_2$ alebo $q_1 \mid n_2$, tak $nsd(n_1, n_2) = p_1$ alebo $nsd(n_1, n_2) = q_1$.*

Dôkaz.
Je priamym dôsledkom Vety 1.10.

□

Podľa Vety 3.6, najväčší spoločný deliteľ Euklidovým algoritmom dokáže faktorizovať verejný RSA modul práve vtedy, ak sa v dvoch verejných moduloch nachádza práve jedno rovnaké prvočíslo. Naskytuje sa otázka, aká je pravdepodobnosť, že v dvoch verejných moduloch bude existovať jedno rovnaké prvočíslo ? A je možné tento algoritmus reálne použiť v praxi ?

3.4.1 Pravdepodobnosť výsledku výpočtu najväčšieho spoločného deliteľa ako faktorizácie RSA modulov

Na to, aby sme vedeli určiť, na akej množine modulov je možné získať signifikantný výsledok výpočtu najväčšieho spoločného deliteľa, potrebujeme zistiť, koľko existuje prvočísel. Na to slúži veta prvočísel, ktorá aproximuje počet prvočísel menších ako nejaké číslo n.

Veta 3.7 *Počet prvočísel menších ako $n \in \mathbb{N}$ je približne $n/\ln(n)$ [16].*

Dôkaz Vety 3.7 uvádzať nebudeme, keďže by to tvorilo väčšinu tohto materiálu. Ak vieme, koľko približne existuje prvočísel menších ako N, je jednoduché spočítať, koľko existuje 512 bitových prvočísel. Hľadáme prvočísla, ktoré majú 512 bitov. Potom najväčšie možné 512 bitové číslo je $2^{512} - 1$. Podľa Vety 3.7 potrebujeme horné ohraničenie, takže naše horné ohraničenie je 2^{512} a najmenšie 512 bitové číslo je 2^{511}. Ak tieto hodnoty dosadíme do Vety 3.7, dostávame, že 512 bitových prvočísel je približne $1,88530 \times 10^{151}$. Nasledujúca veta určuje, koľko modulov potrebujeme na to, aby sme s pravdepodobnosťou 50 % dokázali faktorizovať verejný modul kryptografického systému RSA algoritmom najväčšieho spoločného deliteľa.

Veta 3.8 *Nech $n, k \in \mathbb{N}$, n označuje počet prvočísel, ktoré používame na generovanie verejného modulu kryptografického systému RSA, k označuje počet rôznych verejných modulov kryptografického systému RSA. Potom ak $k = \lceil \sqrt{n \ln(2)} \rceil$, tak máme 50 % pravdepodobnosť úspechu faktorizácie algoritmom najväčšieho spoločného deliteľa.*

Dôkaz.

BUNV predpokladajme, že získavame len unikátne verejné moduly. Nech p je pravdepodobnosť, že algoritmom najväčšieho spoločného deliteľa získame aspoň jeden fak-

tor verejného modulu. Označíme si p' pravdepodobnosť javu, že algoritmom nsd nezískame žiadny výsledok. Potom platí, že $p = 1 - p'$. Ako vypočítame p'? Nech máme práve jeden verejný modul. V tomto prípade je pravdepodobnosť $p' = 1$. Ak budeme mať práve dva verejné moduly, tak $p' = \frac{(n-2)}{n}$, kde n označuje počet prvočísel. Ak chceme, aby jav p nenastal a v každom verejnom module máme práve dve prvočísla, potom môžeme vyberať z množiny nepoužitých prvočísel, ktorých počet je $n-2$. Ak budeme mať práve tri verejné moduly, tak $p' = \frac{(n-2)}{n}\frac{(n-4)}{n}$, pretože v prvom kroku máme k dispozícií práve $n-2$ prvočísel a v druhom kroku práve $n-4$ prvočísel, ktoré využívame v moduloch a počítame pravdepodobnosť javu p', kde vyberáme také prvočísla, ktoré sme ešte nepoužili. Ak si pravdepodobnosť javu p' napíšeme pre nejaké $k-$ počet modul, dostávame, že $p' = \frac{(n(n-2)(n-4)...(n-2(k-1)))}{n^k}$, čo si môžeme prepísať ako $p' = (1-\frac{2}{n})(1-\frac{4}{n})\dots(1-\frac{2(k-1)}{n})$. Využijeme aproximáciu pomocou funkcie e^n kde platí, že $e^{(-1/n)} \approx 1-\frac{1}{n}, e^{(-2/n)} \approx 1-\frac{2}{n}, \dots, e^{(-k/n)} \approx 1-\frac{k}{n}$. Ak si p' aproximujeme funkciou e^n, potom dostávame vzťah $e^{(-2/n)}e^{(-4/n)}\dots e^{(-2(k-1))} = e^{([-2-4...-2(k-1)]/n)}$. Súčet v exponente vieme ľahko spočítať, pretože vieme, že platí vzťah: $1+2+\dots+n = \frac{n(n+1)}{2}$, potom ale platí, že $-2-4\dots-2(k-1) = -2(1+2+\dots+k-1) = \frac{-2(k-1)(k-1+1)}{2} = -(k-1)k$. Dosadením do aproximácie vyššie máme, že $e^{([-2-4...-2(k-1)]/n)} = e^{([-(k-1)k]/n)}$. Položme teraz $p = 0,5$. Pravdepodobnosť p' dostávame ako $p' = 0,5$. Ak teraz budeme riešiť rovnosť $0,5 = e^{([-(k-1)k]/n)}$, zlogaritmovaním oboch strán máme $\ln(1/2) = [-(k-1)k]/n$, následným prenásobením oboch strán n a vykrátením -1, dostávame, že $n\ln 2 = (k-1)k$. Nakoniec, pri dostatočne veľkom k môžeme približne aproximovať súčin $k(k-1)$ ako k^2, čím dostávame požadovanú rovnosť $k \approx \sqrt{(n\ln 2)} \approx 0,8325\sqrt{n}$. □

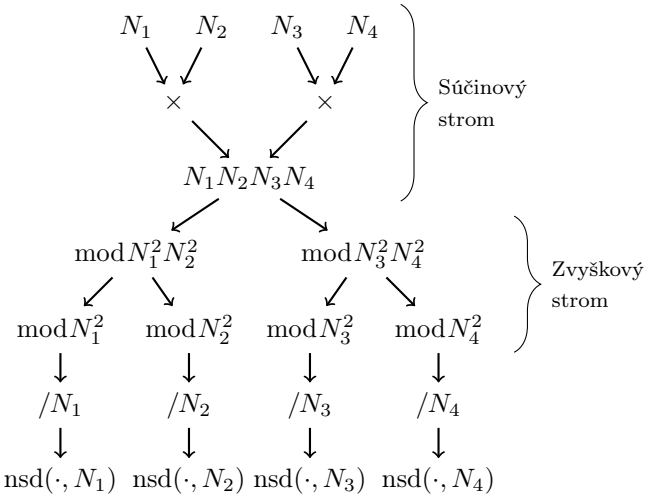
Ak do Vety 3.8 dosadíme za n počet 512 bitových prvočísel, ktorý sme vypočítali vyššie a vypočítame hodnotu k tak, aby sme mali približne 50 % pravdepodobnosť že nám algoritmus nsd dá ako výsledok prvočíselný rozklad, dostávame približne hodnotu $2,898881 \times 10^{150}$. To znamená, že by sme potrebovali približne $2,898881 \times 10^{150}$ verejných 1024 bitových modulov, aby sme mali približne 50 % pravdepodobnosť faktorizácie algoritmom nsd.

3.4.2 Vylepšenie algoritmu najväčšieho spoločného deliteľa

Samotný algoritmus na nájdenie najväčšieho spoločného deliteľa síce funguje v kvadratickom čase, ale merania ukázali, že na vzorke približne 700000 verejných modulov výpočet najväčšieho spoločného deliteľa pre každú dvojicu kľúčov by trval približne

20 dní na jednovláknovom 3,2 GHz procesore. Keďže potrebujeme spraviť výpočet pre každú dvojicu verejných modulov, tak časová zložitosť tohto algoritmu je $O(m^2n^2)$ pre m verejných modulov, každý dĺžky približne n bitov. Aj keď v našom prípade ide iba o zanedbateľný počet dní, napríklad podľa [17] výpočet 4,4 milióna rozdielnych RSA modulov by na 2,93 GHz procesore trval jeden rok.

Práve preto využijeme inú formu výpočtu najväčšieho spoločného deliteľa pre túto vzorku dát. Euklidov algoritmus môžeme vylepšiť podľa [18] pomocou rýchlej aritmetiky na časovú zložitosť $O(n \log^2 n \log\log n)$. Nech $m \in \mathbb{N}, N_1, \ldots, N_m$ sú rôzne moduly, potom označíme $N = N_1 \times N_2 \times \ldots \times N_m$. Budeme postupovať tak, že najprv vypočítame súčin všetkých modulov, tak vypočítame číslo N a potom budeme počítať takzvaný zvyškový strom ako je znázornené na obrázku 1. Nakoniec vypočítame m krát funkciu nsd pre N_i a $N \bmod N_i^2$ a ak sa výsledok z funkcie nsd nachádza v intervale $(0, \ldots, N_i)$, tak máme výsledok.



Obr. 3.1: Výpočet súčinového a zvyškového stromu

Počítanie súčinu ako je uvedené vyššie na obrázku 3.1 je efektívnejšie ako počítanie súčinu ΠN_i zaradom. Násobenie je asociatívne ale „časová zložitosť nie je.“ Vieme, že $(a \times b) \times (c \times d) = (((a \times b) \times c) \times d)$, potom ale platí, že výpočet $(a \times b) \times (c \times d)$ je mierne rýchlejší ako výpočet $(((a \times b) \times c) \times d)$ [19]. Pretože je lepšie a rýchlejšie násobiť najprv malé čísla a až nakoniec pár veľkých. Práve preto výsledný súčin N budeme počítať ako na obrázku 3.1. Takýto model výpočtu sa používa, aby vstupy do algoritmu súčinu boli vyvážené a maximalizovala sa efektívnosť rýchlosti výpočtu súčinu [19]. Tieto fakty bližšie rozoberá autor článku [18]. Tento výpočet je možné ešte zrýchliť rýchlou Furierovou transformáciou, čím sa dostávame na časovú zložitosť

$\Omega(n\log(n)\log(\log(n)))$ súčinu dvoch n bitových čísel podľa [17].

Ďalej využijeme podobný princíp ako sme použili pri výpočte čísla N aj na výpočet najväčšieho spoločného deliteľa použitím zvyškových stromov. Počítame funkciu modulo s umocnením číslom N_i na druhú. Ak by číslo N_i nebolo umocnené, dostávali by sme vo výsledkoch 0, keďže N je súčinom týchto čísel. Klasický Euklidov algoritmus má časovú zložitosť výpočtu dvoch $n-$bitových čísel $O(n^2)$, respektíve $O(\log(n)^2)$, ak n je prirodzené číslo (nie počet bitov). Tento algoritmus je možné ešte vylepšiť použitím rýchlej celočíselnej aritmetiky, vieme teda dosiahnuť časovú zložitosť $O(n\log(n)^2\log(\log(n)))$ pre n –bitové číslo [17]. Ak by sme sa nesnažili tieto algoritmy zefektívniť, tak výpočet najväčšieho spoločného deliteľa 1024 bitového čísla klasickým Euklidovým algoritmom podľa [20] na priemernom 32 bitovom procesore trvá približne 15μs, potom pri výpočte faktorizácie pomocou algoritmu najväčšieho spoločného deliteľa tak, že počítame nsd každej unikátnej dvojice, dostávame už pri vzorke (napríklad) 4,4 milióna rôznych modulov 6×10^{13} rozdielnych výpočtov, čo znamená približne 30 rokov počítania [17].

Ak to teda celé zhrnieme, využitím rýchlych aritmetických algoritmov, násobenie a modulárna aritmetika na n –bitových číslach môže byť uskutočnená s časovou zložitosťou $O(n\log(n)\log(\log(n)))$ a výpočet najväčšieho spoločného deliteľa s časovou zložitosťou $O(n\log(n)^2\log(\log(n)))$ [18, 17]. Potom výpočet súčinu mn –bitových čísel a modulárna aritmetika, resp. vytvorenie zvyškového stromu, bude mať časovú zložitosť $O(nm\log(m)\log(mn)\log(\log(nm)))$, pretože výška súčinového stromu je $\log(m)$ a veľkosť čísla je maximálne $m \times n$ bitov a časová zložitosť najväčšieho spoločného deliteľa je $O(nm\log(n)^2\log(\log(n)))$ [17]. To nám dáva značne lepšiu časovú zložitosť ako v prípade klasického násobenia, ktorého časová zložitosť je $O((mn)^2)$ a výpočtu najväčšieho spoločného deliteľa [17, 19]. Taktiež, ak by sme nepočítali modulárny strom, tak algoritmus najväčšieho spoločného deliteľa čísel N_i a N by mal časovú zložitosť $O(mn^2)$, čo je značne pomalšie ako tieto výhodnejšie algoritmy.

Kapitola 4

Výsledky výskumu

4.1 Klasické faktorizačné algoritmy

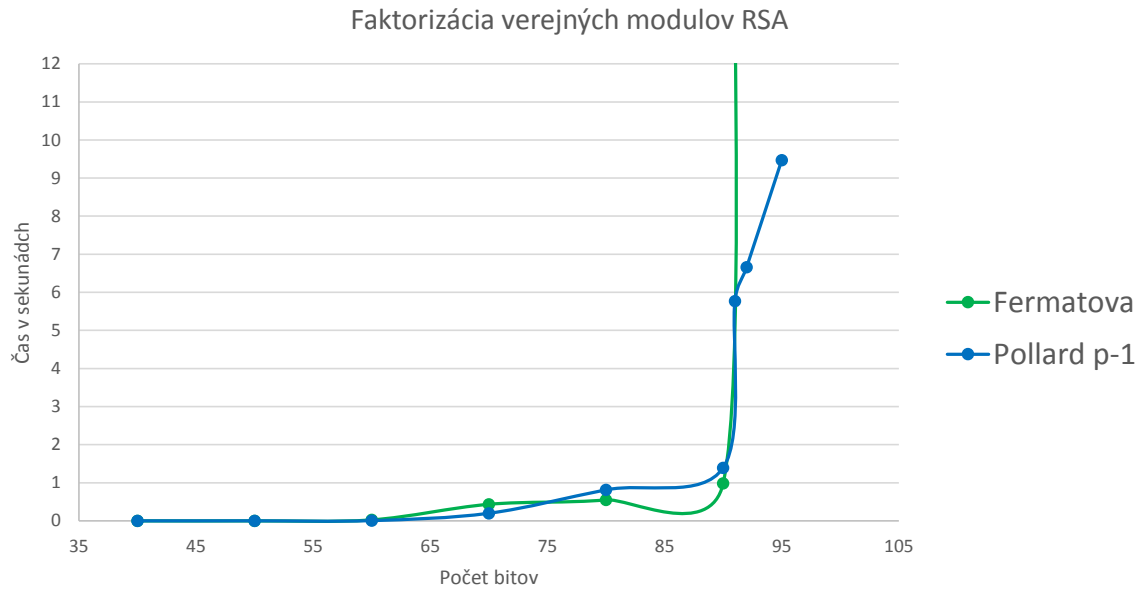
Testovanie a porovnávanie časovej zložitosti jednotlivých faktorizačných algoritmov sme vykonávali na niekoľkých vopred pripravených dátach. Medzi prvú testovaciu sadu sme zaradili skutočné verejné moduly kryptografického systému RSA, samozrejme, s malým počtom bitov, respektíve znakov. Začali sme faktorizovať 40 bitový verejný modul až po 95 bitový verejný modul kryptografického systému RSA. Tieto verejné moduly boli generované operačným systémom CentOS. Tabuľka 4.1 vyjadruje počet sekúnd potrebných na faktorizáciu verejného modulu RSA uvedeného v prvom stĺpci. Hodnoty, ktoré nie sú vyplnené, znamenajú, že faktorizácia nebola úspešná v reálnom čase (v našom prípade je to viac ako 1 minúta). Ako si môžeme všimnúť, Pollard $p-1$ nedokázal faktorizovať verejné moduly od 40 bitov vyššie. Je to z dôvodu, ktorý sme popísali v Sekcii 3.1, totižto RSA moduly sú generované myšlienkou, ktorá je popísaná v Sekcii 3.1. tak, aby boli odolné voči faktorizácii Pollard $p-1$ algoritmom. Časová zložitosť Fermatovej faktorizácie je individuálna v jednotlivých prípadoch vzhľadom na vzdialenosť prvočísel p a q, ktorá je premenlivá v závislosti na vygenerované moduly. Pollard ρ algoritmus má časovú zložitosť exponenciálnu ako sme predpokladali v Sekcii 3.2.

Nasledujúci test sme previedli na zložených číslach, ktoré boli tvorené dvoma veľmi blízkymi prvočíslami. Tento test by mal ukázať efektivitu Fermatovho algoritmu. Číslo n, ktoré sme faktorizovali, bolo tvorené ako súčin dvoch po sebe idúcich prvočísel, napríklad $n = p(5000000)p(5000001)$, kde $p(n), n \in \mathbb{N}$ je $n-$ té prvočíslo v jedinej rastúcej bijekcií množiny prvočísel do množiny prirodzených čisel. Z Grafu 4.3 môžeme vidieť, ako sa správa Fermatova faktorizácia, práve vtedy, keď prvočísla sú blízko seba,

Fak. číslo	Počet bitov	Fer. [s]	Poll. ρ [s]	Poll. $p-1$ [s]
640452262147	40	0,000021	0,000297	2,268636
698859554554279	50	0,000046	0,000929	
1031503157807531911	60	0,026032	0,0066	
84.....06567	70	0,435088	0,199696	
72.....86443	80	0,550785	0,814837	
78.....17139	90	0,983017	1,391858	
16.....41239	91	15,864214	5,77218	
41.....56403	92		6,65647	
37.....16001	95		9,470728	

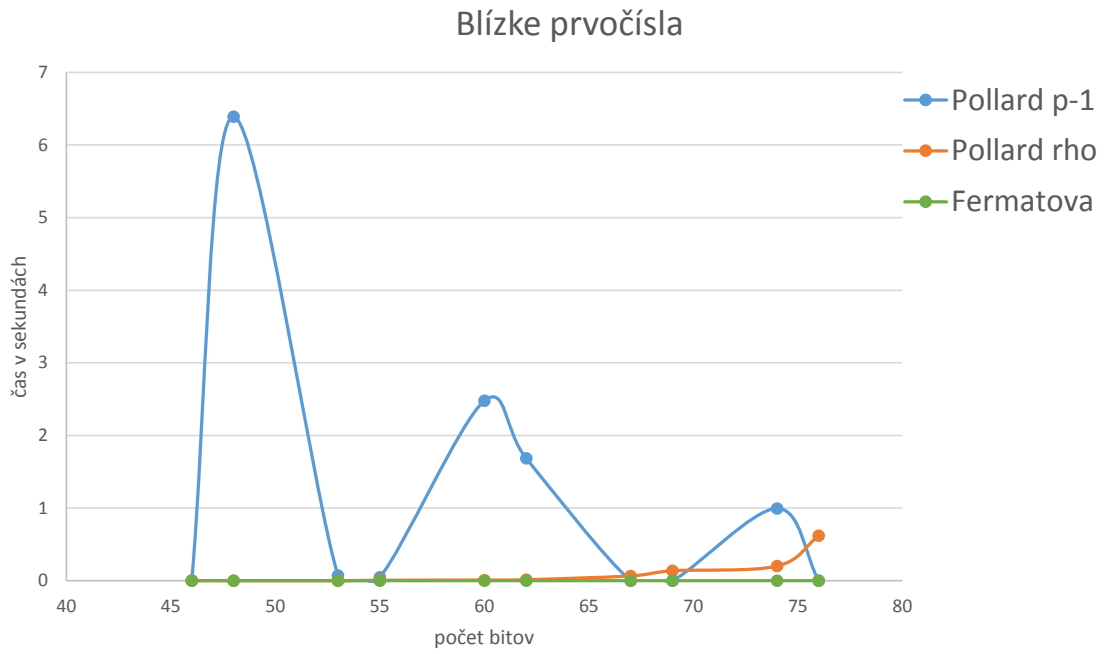
Tabuľka 4.1: Faktorizácia modulov kryptografického systému RSA.

respektíve dve, po sebe nasledujúce prvočísla, používané ako modul n, ktorý chceme faktorizovať. Jeho časová zložitosť je konštantná v porovnaní s napríklad Pollard $p-1$ alebo Pollard ρ algoritmom. V Grafe 4.1 môžeme vidieť aj správanie Pollardovej $p-1$, ktorá závisí od prvočíselného rozkladu čísla $p-1$ ak $n=pq$.



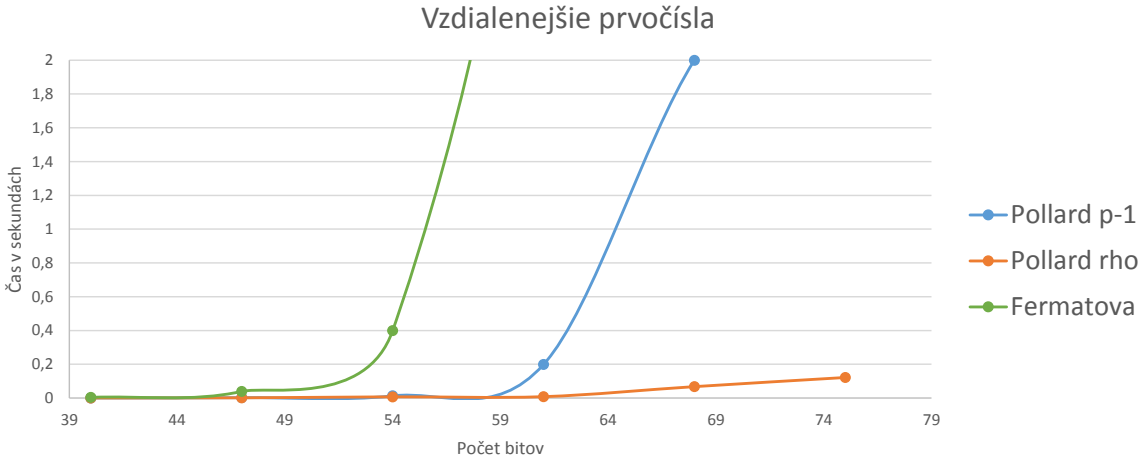
Graf 4.2: Porovnanie času faktorizácie modulov tvorene dvoma blízkymi prvočíslami

Naopak, ak budeme uvažovať číslo n tvorené dvoma vzdialenejšími prvočíslami (s rastúcim počtom bitov je vzdialenosť prvočísel desaťkrát väčšia), môžeme z Grafu 4.4 pozorovať, aká je časová zložitosť Fermatovej faktorizácie. Pollard ρ algoritmus je



Graf 4.3: Porovnanie času faktorizácie modulov tvorene dvoma blízkymi prvočíslami

v tomto prípade efektívny, pretože časová zložitosť tohto algoritmu sa opiera o veľkosť najmenšieho prvočísla, bližšie popísane v Sekcii 4.2. Pollardov $p-1$ algoritmus komentovať nebudeme, keďže jeho časová zložitosť závisí od prvočíselného rozkladu $p-1$.



Graf 4.4: Faktorizácia modulov tvorených dvoma vzdialenými prvočíslami

Na záver by sme chceli poznamenať, že faktorizácie čísla $n = p(5) * p(100000000)$ trvala Pollar ρ algoritmom približne $0,000033$ sekúnd, Fermatov faktorizačný algoritmus nebol schopný v reálnom čase faktorizovať tento verejný modul.

4.2 Faktorizácia algoritmom najväčšieho spoločného deliteľa

Ak by sme počítali nsd ako bolo uvedené na začiatku tak, že spočítame funkciu nsd pomocou Euklidovho algoritmu pre každú dvojicu najväčšieho spoločného deliteľa, tak odhadovaný čas do ukončenia programu na našej vzorke približne 1.1M kľúčov by bol približne 40 dní. Využitím rýchlejšieho algoritmu, ktorý sme z časti opisovali vyššie, sme dostali časovú zložitosť pri výpočte nsd na vzorke 1.1M kľúčov približne 2350 sekúnd. Vytvorenie súčinu, respektíve prvého produktového stromu, trvalo približne 207 sekúnd, vypočítanie zvyškového stromu približne 2009 sekúnd a spočítanie nsd 1.1M krát trvalo zvyšných 134 sekúnd a s pamäťovou zložitosťou približne 16GB. Môžeme vidieť, že časová úspora je už len pri vzorke 1.1M kľúčov veľmi výrazná oproti klasickému algoritmu. Využívali sme priamo zdrojový kód zverejnený autormi článku [17]. Autori článku [17] tvrdia, že na výpočet týmto algoritmom na vzorke 11M rozdielnych RSA kľúčov potrebovali približne 5.5 hodín na jednojadrovom 3.30 GHz procesore a pamäťová zložitosť bola okolo 32 GB. Paralerizáciou bolo možné tento algoritmus ešte zrýchliť na dobu trvania 1.3 hodín a s pamäťovou zložitosťou približne 60GB, čo činilo celkové náklady na ich výpočet iba 5$ [17]. V našom prípade náklady na službu google computing, ktorú sme využívali činili okolo 15 $. Vidíme, že časová zložitosť je naozaj značne vylepšená. Poďme sa pozrieť, ako sme získavali potrebné dáta a nakoniec na výsledky samotnej faktorizácie.

4.2.1 Získavanie dát

Veľký problém nastáva už len pri zberaní verejných kľúčov. Nakoľko skenovanie celej internetovej siete realizovanej pomocou programu zmap viedlo k zablokovaniu internetového pripojenia, museli sme sa zamerať aj na iný princíp zbierania dát.

Pomocou zmap sa nám podarilo dostať 1.558.520 verejných IP adries, ktoré mali otvorený port 22 (ide o štandardný SSH port) a približne 209.499 IP adries, ktoré mali otvorený port 443 (ide o štandardný HTTPS/SSL port). Na tieto IP adresy sme aplikovali jednoduchý skript, ktorý z týchto IP adries získal verejný modul, ak to je možné. Najväčší problém pri získavaní RSA kľúčov bola neprístupnosť protokolov SSL, kedy mnohokrát nebolo možné ani len nadviazať spojenie. Takýto algoritmus získavanie RSA kľúčov nebol ale príliš efektívny. Na prejdenie všetkých 1558520 verejných IP adries z protokolu SSH sme potrebovali približne 20 dní. Skenovanie SSL

protokolu a získanie RSA kľúčov z tohto protokolu bolo ešte pomalšie, preto sme sa rozhodli, že takéto kľúče získavať vo veľkom množstve nebudeme.

Po preskenovaní všetkých IP adries sme získali 1363129 RSA modulov z protokolu SSH a 93505 RSA modulov z protokolu SSL. Eliminovaním duplicitných RSA kľúčov sme dostali už len 591864 unikátnych RSA modulov z protokolu SSH. Množina získaných výsledkov nebola dostatočná a preto sme sa rozhodli, že získame ďalšie verejné kľúče z PGP serverov. Práve tieto servery slúžia užívateľom na ukladanie svojich verejných kľúčov a vytvárajú takzvané dumpy, výpisy, v pravidelných intervaloch. Využili sme jednoduchý shell skript, ktorý zo serveroch, kde sa nachádzajú PGP výpisy dokáže exportovať verejné kľúče RSA. Takýmto spôsobom sa nám podarilo získať 418190 verejných kľúčov zo servera `pgp.key-server.io` a 822648 kľúčov zo servera `keyserver.mattrude.com/dump/`. Tieto verejné RSA kľúče boli vo väčšine prípadov unikátne (približne 100 kľúčov bolo duplicitných).

Nasledovala požiadavka, aby sa do výskumu zapojili aj kľúče, ktoré sa používajú na Univerzite Pavla Jozefa Šafárika. Tu sme využili program nmap, ktorým sme skenovali celú sieť IP adries UPJŠ a získali sme 80 SSH RSA kľúčov, z toho 67 unikátnych a 68 SSL RSA kľúčov, z toho 62 unikátnych. Celkovo to činilo 129 unikátnych RSA kľúčov. Využili sme rovnaké skripty ako v prípade získavania SSH a SSL kľúčov z iných zdrojov.

Samozrejme že aj pri tejto metóde je možné využiť multivláknové algoritmy a takto zefektívniť algoritmus, ale v našom prípade úplne postačuje aj takýto model zberu dát. Najväčšie problémy pri tvorbe týchto skriptov nastávali pri konverzii a extrakcii verejných modulov a exponentov. Existuje mnoho štandardov, ako sa majú takéto moduly a exponenty distribuovať, čím sa spôsobuje nejednoznačnosť medzi jednotlivými štandardmi a problémy pri extrakcii dát. V prípade SSH protokolu sme využívali PUB súbory, ktoré vytvára linuxový program SSH–keyscan, určený na extrakciu kľúčov. SSH–keyscan vytvára výstupný formát popísaný v štandarde RFC 4716, následne sme PUB súbory formátovali na súbory typu PEM (Privacy Enhanced Email), ktoré sú popísane v štandardoch RFC 1421 až RFC 1424 a ktoré sme následne dekódovali programom openssl z kódovania asn1 na čitateľnú formu, z ktorej bolo možné už priamo extrahovať verejný modul a exponent v podobe čísla. Extrahovať dáta priamo zo súborov PUB je príliš komplikované vzhľadom na formát týchto súboroch popísaný v štandardoch.

V prípade SSL protokolu sme využívali openssl program, priamo na sťahovanie dát, ktorý už je schopný generovať RSA verejný kľúč vo formáte X.509. Formát X.509 je

kódovaný v base64 kódovaniu. Takýto formát stačilo následne odkódovať pomocou base64 a využiť prečítanie štruktúry ans1, čím sme priamo získali RSA verejný modul a exponent. Najjednoduchšie bolo exportovanie verejných kľúčov z PGP serveroch, kde sa využíval jednoduchý pgpdump softvér.

4.2.2 Analýza výsledku faktorizácie

V našom prípade sa nám podarilo faktorizovať 66 verejných modulov, z toho 3 z protokolu SSL, 11 z protokolu SSH a z PGP výpisu zvyšných 52 verejných modulov. Všetky verejné moduly, ktoré sa nám podarilo faktorizovať z protokolu SSH mali správne generované prvočíslo po formálne stránke. Myslíme tým veľkosť prvočíselného rozkladu. To znamená, že moduly sú tvorené dvoma približne rovnako veľkými prvočíslami. Rovnaký výsledok sme mohli pozorovať aj pri faktorizácii kľúčov z protokolu SSL. Pri bližšom skúmaní vlastností jednotlivých serveroch so spoločným faktorom sme si všimli niekoľko zaujímavostí. Pomocou programu nmap sme hľadali súvis s operačným systémom a už faktorizovanými kľúčmi. Vo väčšine prípadov ide o operačný systém Linux s jadrom 2.6.x alebo o zariadenia, v ktorých sa nám nepodarilo jednoznačne určiť operačný systém. Medzi faktorizovanými zariadeniami boli napríklad routre, switche, tlačiarne a iné sieťové zariadenia. Konkrétne ide o zariadenia s operačným systémom DD–WRT v24–SP2, Cisco RV042 WAP, Huawei S9300 switch, Cisco ASR 9010 router alebo aj napríklad tlačiareň C2380. Autori článku [17] hovoria o problémoch v starších operačných systémoch, kde entropia generovania náhodných čísel bola veľmi malá. Inač povedané hovoria o tom, že operačný systém Linux sa stretával s veľmi nevhodným generátorom náhodných čísel a že je možné, že práve tento generátor spôsobil to, že operačné systémy pri generovaní RSA kľúčov vygenerovali rovnaké prvočísla, ktoré použili vo verejnom RSA kľúči. Rovnaké dôvody objavili aj autori článku [21]. Práve článok [21] je jadro celej tejto myšlienky a bola to prvá faktorizácia RSA verejných modul touto metódou úspešná. Podarilo sa im faktorizovať 21419 rôznych certifikátov, kde ale vo výskume boli zahrnuté aj menšie ako 512 bitové moduly [21]. My sme faktorizovali len moduly väčšie ako 1024 bitov. Naopak v inom výskume sa podarilo faktorizovať z 12M približne 64081 RSA modulov. Tabuľka 4.2 znázorňuje výsledky výskumu bezpečnosti RSA modulov, kde sa využíval rovnaký algoritmus ako sme využívali my [17].

Naopak, faktorizácia z PGP priniesla zaujímavé výsledky. Väčšina z faktorov verejných modulov RSA z PGP výpisu boli veľmi malé prvočísla ako napríklad:

	Náš výskum		Výskum z roku 2012	
	SSH	SSL	SSH	SSL
Počet IP adries	1 558 520	209 499		
Počet verejných modulov	1 363 129	93 505		
Počet unikátnych modulov	436 644	55 845	10 216 363	12 828 613
Počet fakt. RSA kľúčov	11	3	2 459	64 081
... v percentách	0,0026%	0,0054%	0,0241%	0,4995%

Tabuľka 4.2: Porovnanie výsledkov s výskumom z roku 2012.

4294967297, 12884901891, 6242474487359, 357, 2, 5485. Dokonca niektoré z nich nie sú ani prvočísla. Ako môžeme vidieť, verejný RSA modul v PGP výpise mal faktor číslo 2. Potom faktorizácia v takomto prípade je triviálnou záležitosťou. Iba dva verejné moduly z PGP výpisu sa nám podarilo faktorizovať tak, žeby obsahovali správne generovaný RSA modul po formálne stránke, čím myslíme dĺžku prvočíselného rozkladu. Autor [22], ktorý robil podobný výskum, ale len na PGP serveroch, kontaktoval jednotlivé osoby, ktoré mali takéto zlé vygenerované verejné moduly uložené na PGP serveroch. Zväčša išlo o RSA moduly, ktoré boli automaticky nahrané operačným systémom na PGP server. Iní tvrdili, že ich kľúče boli uložené na servery pomocou programu gnupg alebo iným komerčným programom.

Zaujímavým pozorovaním bolo taktiež skúmanie serveroch, na ktorých sme faktorizovali verejný modul získaný pomocou protokolu SSL. Ide konkrétne o tieto IP adresy: 115.183.28.97, 124.193.190.21 a 124.205.10.1. Všetky IP adresy patria Čínskej telekomunikačnej spoločnosti a pri skúmaní okolia jednotlivých IP adries (napríklad IP adresy 124.205.10.0–124.205.10.255 a pod.) sme narazili na rôzne webové stránky, ktoré ponúkajú prihlasovacie formuláre k službám ako sú HUAWEI server a iné služby, servery, známych spoločností. Operačný systém v tomto prípade nebol jednoznačne určený a každá z týchto IP adries obsahovala iný operačný systém (vyplýva to z tvaru odoslaných a prijatých požiadaviek). Tento fakt je odlišný od skutočností zistených na iných faktorizovaných zariadeniach. Na rozdiel od toho prípadu, vo faktorizovaných moduloch bola približná zhoda medzi operačným systémom. To poukazuje na skutočnosť, či v tomto prípade nie je možná práve zámerná zhoda faktorizácie verejných modulov. K tomuto pozorovaniu nás vedie aj skutočnosť, že kým v iných prípadoch sme faktorizovali dve zariadenia, v tomto prípade Čínskej telekomunikačnej spoločnosti došlo k faktorizovaniu až troch verejných moduloch. Pre hlbšiu analýzu sme skú-

mali okolie IP adries (rozumieme celú masku podsiete) 115.183.28.97, 124.193.190.21 a 124.205.10.1. Tento experiment priniesol zaujímavé výsledky. Na okolí týchto IP adries sme boli schopný faktorizovať 23 rôznych verejných modulov s rôznymi operačnými systémami. Z 23 rôznych faktorizovaných verejných modulov sme získali iba 3 rozdielne prvočísla vypočítané algoritmom nsd. V tomto prípade ide o príliš veľkú náhodu. Z tohto dôvodu sa naskytuje otázka, či za týmto faktom nemáme hľadať inú súvislosť.

Medzi faktorizovanými modulmi boli napríklad aj moduly patriace spoločnosti Frontier Communications Solutions sídliacej v New Yorku, ktorá odmietla s nami komunikovať o tom, že sme faktorizovali ich verejný modul nejakého servera, pokiaľ nie sme ich klientmi. Taktiež sme narazili na rozsahy IP adries napríklad Kórejskej telekomunikačnej spoločnosti alebo aj Brazílskeho poskytovateľa internetového pripojenia.

Ďalším zaujímavým pozorovaním je, že verejné moduly, ktoré zdieľali jedno prvočíslo, boli použité na serveroch, ktoré patrili jednej spoločnosti. Napríklad IP adresy 74.45.0.36 a 74.45.228.134 zdieľali jeden spoločný modul a obe patrili spoločnosti Frontier Communications Solutions. V našom prípade nenastala situácia, aby sme faktorizovali dva rozdielne moduly, ktoré sú použité na serveroch, ktoré majú IP adresy patriace rozdielnym inštitúciám. Tento fakt poukazuje na to, že telekomunikačné spoločnosti používajú rovnaké, zastarané zariadenia, ktoré majú chybnú implementáciu generovania náhodných čísel alebo takýto prístup sa využíva na iné, nešpecifikované záujmy. Na Univerzite Pavla Jozefa Šafárika nebola zaznamenaná žiadna chyba vo verejných modulov, no počet verejných modulov z Univerzity Pavla Jozefa Šafárika je k počtu všetkých verejných modulov, využívaných v tomto experimente zanedbateľný. Ide len o približne 100 rôznych verejných modulov. Posledný test sme vykonali na IP adresách 74.45.0.0–255. Keďže IP adresy 74.45.0.36 a 74.45.228.134 obsahovali rovnaký verejný modul a obe patria spoločnosti Frontier Communications Solutions, rozhodli sme sa vyskúšať faktorizovať dáta z okolia IP adresy 74.45.0.36. Na rozsahu IP adries 74.45.0.0 až 74.45.0.255 (všetky tieto IP adresy patria Frontier Communications) sme získali až 101 kľúčov z portu 22, z toho unikátnych 44 a podarilo nám faktorizovať 3 verejné moduly. Vo všetkých prípadoch boli použité „približne“ rovnaké operačné systémy, respektíve ich Linuxové jadrá.

Záver

Faktorizácia jedného RSA modulu spôsobuje vážne bezpečnostné riziko a je neprijateľná v dnešnej informačnej spoločnosti, kedy mnoho osobných dát je posielaných pomocou internetových sietí a únik týchto dát môže mať veľmi veľké následky. Ľudia dlhodobo považovali kryptografický systém RSA za jeden z najbezpečnejších kryptografických systémoch, ale dnes vidíme, že tomu tak nie je. Nakoniec, v časoch, keď sa myšlienka kvantových počítačoch stáva stále reálnejšie a faktorizácia na kvantových počítačoch je v súčasnej dobe polynomiálny problém, je kryptografických systém RSA na ústupe, kde už v súčasnosti môžeme pozorovať rôzne chyby, ktoré tento kryptografický systém obsahuje.

Faktorizovať jeden 1024 bitový modul je veľký objav, faktorizovať 21419 takýchto verejných modulov je už štatistika. Aj takýmito slovami zhrnuli svoje poznatky Lenstra a kol. v článku [21]. Nám sa podarilo z 1,2M unikátnych verejných modulov faktorizovať 66, čo sa dá pokladať za stále veľké bezpečnostné riziko. Napriek faktu, že tento problém bol objavený už v roku 2012, stále sa s týmto problémom stretávame. Poukázali sme na fakt, že faktorizácia takouto metódou nie je zapríčinená náhodou, ale existuje vážne podozrenie, že za tým stoji implementačná chyba linuxového jadra 2.6.x, či dokonca aj zámer poskytovateľov internetových služieb. Bližšie sme skúmali faktorizované servery a objavili sme niekoľko zaujímavosti vo vzťahu k faktorizácii a vlastníctve jednotlivých modulov. Ukazuje sa, že problém s faktorizáciou nastáva práve na zariadeniach patriacich jednej spoločnosti, to znamená, že ak sa nám podarilo faktorizovať verejné moduly, tak všetky tieto moduly patrili jednej spoločnosti. Tento fakt poukazuje na to, že práve faktorizované zariadenia majú spoločné vlastnosti, ako napríklad značka, typ, rok výroby a aktualizáciu software.

Taktiež sme poukázali na fakt, aká veľká je potrebná množina verejných modulov, aby sme dokázali metódou nsd faktorizovať jeden verejný modul, samozrejme, správne generovaný a že so vzorkou 1.2M unikátnych verejných modulov je tento spôsob nemožný. Ak by sme mali zodpovedať otázku, či sa naozaj Ron so svojím RSA

mýlil, je možné tvrdiť, že RSA je zatiaľ stále bezpečný kryptografický systém, ak by sme k generovaniu kľúčov a testovaniu ich bezpečnostných vlastností pristupovali obozretnejšie. Nakoniec, je väčšia šanca vyhrať v lotérií ako vygenerovať dve rovnaké prvočísla pri vhodnom a sofistikovanom generátore náhodných čísel.

V druhej časti práce sme skúmali jednotlivé faktorizačné algoritmy z hľadiska ich časovej zložitosti a výskumom sme overili platnosť našich tvrdení. Vyzdvihli sme vlastnosti jednotlivých faktorizačných algoritmov a ukázali sme, kedy tieto algoritmy sú účinné a kedy nie. Komparáciou sme analyzovali ich správanie na vopred pripravených vstupných dátach a porovnali sme čas faktorizácie jednotlivých algoritmov. Nakoniec sme vyvodili záver efektívnosti jednotlivých faktorizačných algoritmov a poukázali sme aj na problém faktorizácie v kontexte asymetrickej kryptografie, kde sme venovali pozornosť kryptografickému systému RSA a jeho vzťahu k problému faktorizácie, a ukázali sme, že prelomiť kryptografický systém RSA je najviac také náročné, ako je riešenie problému faktorizácie.

Zoznam použitej literatúry

[1] Stinson, D. R.: *Cryptography: theory and practice*. Ontario: CRC press, 2006, ISBN 1–58488–508–4.

[2] Menezes, A. J.; Van Oorschot, P. C.; Vanstone, S. A.: *Handbook of applied cryptography*. Boca Raton: CRC press, 1996, ISBN 0–8493–8523–7.

[3] Stanovský, D.: *Základy algebry*. Praha: Matfyzpress, 2010, ISBN 80–7378–105–0.

[4] Baldi, M.; Bianchi, M.; Chiaraluce, F.; a kol.: Enhanced public key security for the McEliece cryptosystem. *Journal of Cryptology*, ročník 29, č. 1, 2016, str. 1–27.

[5] Bernstein, D. J.; Lange, T.; Peters, C.: Attacking and defending the McEliece cryptosystem. In *Post − Quantum Cryptography*, Springer, 2008, ISBN 978–3–540–88702–7, str. 31–46.

[6] Boneh, D.; a kol.: Twenty years of attacks on the RSA cryptosystem. *Notices of the AMS*, ročník 46, č. 2, 1999, str. 203–213.

[7] Rivest, R. L.; Shamir, A.; Adleman, L.: A method for obtaining digital signatures and public − key cryptosystems. *Communications of the ACM*, ročník 21, č. 2, 1978, str. 120–126.

[8] Barker, E.; Dang, Q.: NIST special publication 800 − 57 part 3: Application − specific key management guidance. *NIST Special Publication*, 2015, str. 57.

[9] Barker, E.; Barker, W.; Burr, W.; a kol.: NIST Special Publication 800 − 57 Recommendation for Key Management− Part 1: General. *NIST Special Publication*, 2012.

[10] Kleinjung, T.; Aoki, K.; Franke, J.; a kol.: Factorization of a 768 − bit RSA modulus. In *Advances in Cryptology − CRYPTO 2010*, Springer, 2010, ISBN 978–3–540–16076–2, str. 333–350.

[11] Blomer, J.; May, A.: A generalized Wiener attack on RSA. In *Public Key Cryptography – PKC 2004*, Springer, 2004, ISBN 978–3–540–21018–4, str. 1–13.

[12] Lenstra, A. K.; Lenstra Jr, H. W.; Manasse, M. S.; a kol.: The number field sieve. In *The development of the number field sieve*, Springer, 1993, ISBN 978–3–540–57013–4, str. 11–42.

[13] Swenson, C.: *Modern cryptanalysis: techniques for advanced code breaking*. Indiana: John Wiley & Sons, 2008, ISBN 978–0–470–13593–8.

[14] Cohen, H.: *A course in computational algebraic number theory*. New York: Springer Science & Business Media, 2013, ISBN 978–3–662–02945–9.

[15] Barnes, C.: Integer Factorization Algorithms. *Oregon State University*, 2004, [online], 7.12.2004, [cit. 2016−03−28], Dostupné online: `http://connellybarnes.com/documents/factoring.pdf`.

[16] Liu, R.; Ye, Y.: Prime number theorem. 2012, [online], 6.10.2012, [cit. 2016−04−1], Dostupné online: `http://math.uchicago.edu/~may/REU2012/REUPapers/LiuR.pdf`.

[17] Heninger, N.; Durumeric, Z.; Wustrow, E.; a kol.: Mining your Ps and Qs: Detection of widespread weak keys in network devices. In *USENIX Security 12*, 2012, str. 205 –220.

[18] Bernstein, D. J.: Fast multiplication and its applications. In *Algorithmic number theory: lattices, number fields, curves and cryptography*, ročník 44, Cambridge University Press, 2008, ISBN 0–521–80854–5, str. 325 –384.

[19] MIRONOV, I.: Factoring RSA Moduli. Part I. 2012. In: windowsontheory.org, [online], 15.5.2012, [cit. 2016−03−31], Dostupné online: `https://windowsontheory.org/2012/05/15/979`.

[20] Granlund, T.; GNU, M.: The GNU multiple precision arithmetic library. [online]. 2016 [cit. 2.4.2016]. Dostupné online: `http://gmplib.org/`.

[21] Lenstra, A. K.; Hughes, J. P.; Augier, M.; a kol.: Ron was wrong, Whit is right. Cryptology ePrint Archive, Report 2012/064, 2012, `http://eprint.iacr.org/`.

[22] SRIRAM, K.: Analyzing RSA OpenPGP keys in the sks–keyserver pool. [online]. 7.10.2014 [cit. 3.4.2016]. Dostupné online:

http://kbsriram.com/2014/10/analyzing-rsa-openpgp-keys-in
-the-skskeyserver-pool.html.