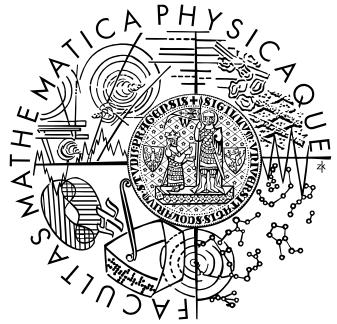
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Jiří Mareš

Srovnání algoritmů pro kryptografii s veřejným klíčem

Katedra algebry

Vedoucí bakalářské práce: RNDr. David Stanovský, Ph.D.

Studijní program: Matematika, Obecná matematika

2010

Rád bych poděkoval doktoru Davidu Stanovskému, svému vedoucímu, za cenné podněty a rady, bez kterých by tato práce nevznikla. Za cenné rady a zvláště trpělivost bych rád poděkoval i své manželce. V neposlední řadě patří můj dík PhDr. Ivě Daňkové za jazykovou úpravu.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 13.12.2010 Jiří Mareš

OBSAH

Název práce: Srovnání algoritmů pro kryptografii s veřejným klíčem
Autor: Jiří Mareš
Katedra (ústav): Katedra algebry
Vedoucí bakalářské práce: RNDr. David Stanovský, Ph.D.
e-mail vedoucího: David.Stanovsky@mff.cuni.cz

Abstrakt: V předložené práci se zabýváme srovnáním základních algoritmů pro šifrování s veřejným klíčem – algoritmy RSA, Rabinovou a ElGamalovou metodou. Odvozujeme teoretickou složitost šifrování a dešifrování jednoho bloku a odvozujeme předpokládaný model chování při zdvojnásobení velikosti klíče. Rovněž provádíme praktická měření rychlosti jednotlivých metod na klíčích velikosti 64 – 4096 bitů a statisticky je vyhodnocujeme. U některých algoritmů uvádíme speciální případy a diskutujeme výhody a nevýhody a jejich praktické použití. Na závěr srovnáváme rychlosti jednotlivých algoritmů a porovnáváme naměřené výsledky s teoretickými předpoklady.

Klíčová slova: RSA, Rabin, ElGamal, šifrování, dešifrování

Title: Comparison of public key cryptography algorithms
Author: Jiří Mareš
Department: The Department of Algebra
Supervisor: RNDr. David Stanovský, Ph.D.
Supervisor's email address: David.Stanovsky@mff.cuni.cz

Abstract: In the present work we study comparison of basic public key encryption algorithms – RSA, Rabin and ElGamel method. We derive theoretic complexity of encrypting / decrypting of one block and we derive an expected model of its behavior with the key of double size. We also take practical measurements of speed of each algorithm using keys sized 64 – 4096 bits and we statistically analyze the results. We also mention special cases of some algorithms and discuss the advantages and disadvantages of their practical usage. At the end of this thesis we make a comparison of the speed of algorithms and we also compare the measured data with theoretical hypothesis.

Keywords: RSA, Rabin, ElGamal, encryption, decryption

1 ÚVOD

1.1 CÍL PRÁCE

Cílem práce je implementace a srovnání nejběžnějších algoritmů pro šifrování s veřejným klíčem – RSA, Rabin a ElGamal. Porovnání provedeme na náhodných datech a na klíčích odpovídající délky, resp. odpovídající bezpečnosti, což je v případě těchto tří algoritmů – dle bezpečnostního doporučení pro velikost klíčů – ekvivalentní. Předmětem práce je rovněž teoretické odvození asymptotické časové složitosti jednotlivých algoritmů a závislosti rychlosti šifrování / dešifrování na velikosti klíče při vstupu stejné délky.

Práce se naopak nezabývá teorií generování prvočísel (ať už obecných či ve speciálním tvaru) ani dokazováním či odvozováním teoretické bezpečnosti jednotlivých algoritmů.

1.2 POUŽITÉ METODY

Pro implementaci a srovnání všech algoritmů byl použit jazyk C společně s knihovnou pro práci s velkými čísly – GMP. Veškerá měření byla provedena na počítači s procesorem AMD Athlon(tm) X2 Dual-Core QL-64 2.10 GHz, pamětí 4 GB RAM a 64-bitovým operačním systém Ubuntu. Pro šifrování a dešifrování byly použity náhodně vygenerované klíče binární velikosti 64 bitů až 4096 bitů; za zprávy byly použity soubory typu xml s velikostí od 50 kB do 1,2 MB. Jelikož při šifrování není podstatné, jestli jsou zdrojová data textem či jen náhodnými byty, byly zvoleny zprávy tak, aby bylo možné verifikaci dešifrování provést „na první pohled“.

Na rozdíl od například kompresních algoritmů, které využívají četnosti jednotlivých znaků, toto u zkoumaných tří algoritmů s veřejným klíčem nehraje žádnou roli. Veškerá vstupní data jsou transformována do binárního řetězce, se kterým se, dle daného algoritmu specificky, pracuje – tedy číslo se násobí, mocní apod.

Karacubovo násobení je v programu vynuceno pro násobení všech čísel. Pro klíče nižších velikostí pak sice násobení nemusí být zcela optimální, ale pomůže nám při odvozování závislosti rychlosti šifrování na velikosti klíče. Můžeme snadno použít asymptotické složitosti Karacubovy metody a nemusíme se zabývat případy, kdy došlo k přelomu, tj. kdy se přestalo používat „školské“ násobení a kdy bylo nahrazeno Karacubovou metodou.

Měření samotné probíhalo tak, že 50 různých zpráv bylo šifrováno / dešifrováno 50 klíči; celkem tedy proběhlo 2500 měření. Od klíče velikosti 2048 byl počet kvůli veliké časové náročnosti redukován na 250; 25 různých zpráv bylo šifrováno 10 klíči. Každá zpráva byla (v různou dobu, tedy nikoliv postupně za sebou) stejným klíčem šifrována / dešifrována pětkrát. Nejpomalejší a nejrychlejší čas byl z měření odstraněn, ze zbylých tří časů byl do výsledku započítán jejich průměr. Takto došlo k eliminaci možného zkreslení, které bylo způsobeno vlivem hardwaru či softwaru (větší zahřátí procesoru, démoni na pozadí OS apod.).

Všechny tři algoritmy jsou implementovány „blokově“ tak, že jednotlivé bloky textu jsou na sobě zcela nezávislé. Jsou tedy vždy šifrovány celé byty. V každém kroku by bylo možné šifrovat o 7 bitů více, ale došlo by ke ztrátě autonomie. V případě ztráty či přehození bloků by mohlo dojít k chybnému dešifrování bloků následujících. Jelikož je šifrování po celých bytech (tedy znacích) aplikováno na všechny tři algoritmy, při jejich vzájemném srovnávání nehraje tento fakt žádnou roli.

Podkapitoly teoretického rázu, tedy části věnující se představení jednotlivých algoritmů a popisu šifrování a dešifrování, byly zpracovány podle [1] a [2].

2 RSA

2.1 POPIS ALGORITMU

Algoritmus RSA, který je pojmenován po svých autorech – Ronu Rivestovi, Adi Shamirovi a Lenu Adlemanovi – je nejstarším kryptografickým systémem s veřejným klíčem (zveřejněn byl v roce 1978) a dodnes patří mezi nejrozšířenější a nejpoužívanější na světě. Jeho bezpečnost je úzce spjata s problémem hledání faktorizace velkých čísel, resp. s jeho obtížností. Dosud není znám algoritmus, který by dokázal problém faktorizace řešit efektivně (tedy v polynomiálním čase), a jeho existence se nepředpokládá. Ještě než si představíme, jak šifrování a dešifrování funguje, je nutné si ukázat, jakým způsobem se klíče generují a co je vlastně veřejným a soukromým klíčem pro algoritmus RSA myšleno.

Generování klíčů:
Nejprve je nutné zvolit, resp. náhodně vygenerovat, dvě lichá prvočísla p a q a spočítat jejich součin $n = pq$. Je vhodné, aby čísla p a q byla (zhruba) stejné binární velikosti. Dále je nutné zvolit celé kladné číslo (šifrovací exponent) $1 < e < (p-1)(q-1)$ takové, že $NSD\big(e, (p-1)(q-1)\big) = 1$, tj. e musí být nesoudělné s $(p-1)(q-1)$.

Jelikož jsou čísla p a q lichá, $(p-1)$ i $(q-1)$ jsou sudá, a tedy e musí být nutně číslo liché.

Následně je nutné spočítat přirozené číslo d takové, že $1 < d < (p-1)(q-1)$ a $ed \equiv 1\ (mod(p-1)(q-1))$, které nutně existuje. Číslo d (dešifrovací exponent) je možné najít například pomocí rozšířeného Euklidova algoritmu[1].

Dvojice čísel (n, e) se nazývá veřejný klíč a číslo d pak klíč soukromý. Je vidět, že soukromý klíč d není možné bez znalosti p a q (resp. $(p-1)(q-1)$) dopočítat.

[1] Viz například [2], kapitola 2, strana 67, algoritmus 2.107

Bezpečnostní doporučení udává, že pro dlouhodobější šifrování (tedy klíč není jednorázový či jen na krátkou dobu) by měl být veřejný klíč, resp. číslo n, velikosti alespoň 768 – 1024 bitů.

Velikost bloku a reprezentace textu jako čísla:

Aby bylo možné použít RSA jako blokovou šifru, je nutné zdrojový, tedy šifrovaný, text nejprve rozdělit na menší části – bloky. Každý blok se poté převede na přirozené číslo. Velikost bloku je určena tak, aby každé takto převedené číslo bylo vždy menší nežli n.

Pro obecnou abecedu $A = \{a_1, \dots, a_x\}$ je velikost bloku $b = \lfloor \log_x(n-1) \rfloor$ a šifrovaný text $\overline{m} = m_1 \dots m_k$. Text $\overline{m}$ odpovídá přirozenému číslu $m = \sum_{j=1}^{k} m_j x^{k-j}$.

V implementaci použité v této práci se za abecedu bere rozšířená znaková sada ASCII o $x = 2^8$ znacích. Velikost klíče n je ve všech případech tvaru 2^{8y}, a velikost šifrovaného bloku je tak $(y-1)$.

Popis šifrování a dešifrování:

Šifrování pomocí algoritmu RSA probíhá tak, že se šifrovaný text M nejprve rozdělí na $\left\lceil \frac{M}{b} \right\rceil$ bloků velikosti b a každý z těchto bloků reprezentovaných přirozeným číslem m je šifrován pomocí funkce $c = m^e \bmod n$. Takto vznikne zašifrovaný text c. Je zřejmé, že pokud je prováděna redukce modulo n, pak vzniklý text $c < n$, avšak obě čísla c a n mohou mít stejný počet bitů, tedy stejnou binární velikost. Velikost výsledného bloku je tedy $b + 1$.

Dešifrování probíhá obdobným způsobem, avšak namísto šifrovacího exponentu e je použit dešifrovací exponent d. Z každého zašifrovaného bloku c je původní text (resp. jeho celočíselná reprezentace) m rekonstruován

pomocí funkce[2] $m = c^d \ mod \ n$. Číslo m se poté snadno převede na text $\overline{m}$ a tyto bloky se na závěr zřetězí do výsledného textu M.

Dešifrování však lze urychlit pomocí čínské věty o zbytcích, a to následovně: Nejprve se spočítají čísla $m_p = c^{d \ mod \ (p-1)} \ mod \ p$ a $m_q = c^{d \ mod \ (q-1)} \ mod \ q$. Dále se pomocí rozšířeného Euklidova algoritmu spočtou čísla y_p a y_q taková, že $y_p p + y_q q = 1$. Jelikož jsou čísla y_p a y_q zcela nezávislá na šifrovaném textu, je možné je předpočítat. Původní reprezentace m je zrekonstruována jako $m = \left(m_p y_q q + m_q y_p p\right) \ mod \ n$.

V této práci budou představeny a srovnány výsledky z obou těchto dešifrovacích metod.

2.2 ŠIFROVÁNÍ V PRAXI

Ještě než se pustíme do představení implementace šifrování RSA, která byla v této práci použita, podívejme se, jakou asymptotickou složitost šifrování očekáváme.

Je vidět, že jedinou operací, kterou při šifrování provádíme, je modulární mocnění. To má v implementaci, která je použita v knihovně GMP (s upravenými parametry pro meze Karacubova násobení), složitost $\mathcal{O}\left(k^{(\log_2 3)} \log_2 e\right)$. Kromě k-bitového parametru n, resp. počtu bitů k, vystupuje v odhadu složitosti i binární délka šifrovacího exponentu e. Pro náhodně zvolený exponent $e < n$ vychází asymptotická složitost šifrování jednoho bloku $\mathcal{O}\left(k^{(\log_2 3)} k\right) = \mathcal{O}\left(k^{(\log_2 3)+1}\right)$, naopak pro pevně zvolený (malý) exponent je asymptotická složitost rovna složitosti Karacubova násobení, tedy $\mathcal{O}\left(k^{(\log_2 3)}\right)$.

Ostatní operace, jako například převedení bloku $\overline{m}$ na odpovídající reprezentaci m, mají složitost ve srovnání s násobením zanedbatelnou.

[2] Důkaz, že dešifrování funguje, je mimo rámec této práce. Je možné jej najít například v [2], kapitola 8, strana 286

Nyní se podívejme, jak vzroste náročnost výpočtu při přechodu od klíče velikosti k bitů k velikosti $2k$-bitů.

Rozlišme zároveň dva případy – šifrovací koeficient e je konstantní a pro klíče obou velikosti stejný, nebo je e voleno náhodně, tedy jeho velikost se blíží k-bitům, resp. $2k$-bitům.

Uvažujme tutéž zprávu velikosti M. K šifrování jednoho bloku potřebujeme jedno modulární násobení dvou čísel binární délky $k = 2^s$, exponent je konstanta e (a to taková, která má v binárním zápise pouze dvě jedničky). To obnáší $\log_2 e$ násobení čísel velikosti k. Složitost výpočtu je $(\log_2 e)\, M(k)$, pro celou zprávu M pak $\left\lceil \frac{M}{\frac{k}{8}-1} \right\rceil (\log_2 e) M(k) = T(M,k)$. $M(k)$ značí čas potřebný pro vynásobení dvou čísel velikosti k-bitů.

Zdvojnásobíme-li počet bitů, dostáváme

$$T(M,2k) = \left\lceil \frac{M}{\frac{k}{4}-1} \right\rceil (\log_2 e) M(2k).$$

Použijeme-li nyní „hrubý“ odhad $\left\lceil \frac{M}{\frac{k}{8}-1} \right\rceil \geq 2 \left\lceil \frac{M}{\frac{k}{4}-1} \right\rceil$ a $M(2k) \leq 3M(k) + ck$ dle Karacubova násobení, dostaneme $T(M,2k) \leq \frac{3}{2} T(M,k) + ck$, pro nějakou konstantu c.

Můžeme postupovat jako v důkaze složitosti Karacubova násobení[3] a obdržíme postupně

[3] Viz například [3], stránka 17, tvrzení 3.2

$$T(M,2^s) \leq \frac{3}{2}T(M,2^{s-1}) + c2^{s-1} \leq \frac{3}{2}\left(\frac{3}{2}T(M,2^{s-2}) + c2^{s-2}\right) + c2^{s-1} \leq$$

$$\cdots \leq \left(\frac{3}{2}\right)^s T(M,1) + c2^{s-1}\left(\left(\frac{3}{4}\right)^{s-1} + \cdots + \left(\frac{3}{4}\right) + 1\right) \leq \left(\frac{3}{2}\right)^s T(M,1) +$$

$$c2^{s-1}\frac{\left(\frac{3}{4}\right)^s - 1}{\left(\frac{3}{4}\right) - 1} \leq \left(\frac{3}{2}\right)^s T(M,1) + 4c2^{s-1} - 2c\left(\frac{3}{2}\right)^s \leq C_1 k^{(\log_2 3)-1} + C_2 k \leq Ck$$

V druhém případě, tedy kdy e není konstantou, ale má rovněž binární délku k, obdržíme pro

$T(M,k) = \left\lceil \frac{M}{\frac{k}{8}-1} \right\rceil (2k)M(k)$ a $T(M,2k) = \left\lceil \frac{M}{\frac{k}{4}-1} \right\rceil (4k)M(2k)$.

Nyní postupujme podobně jako v předchozím případě. Použijme odhady $\left\lceil \frac{M}{\frac{k}{8}-1} \right\rceil \geq 2\left\lceil \frac{M}{\frac{k}{4}-1} \right\rceil$ a $M(2k) \leq 3M(k) + ck$ a dle Karacubova násobení dostaneme $T(M,2k) \leq 3T(M,k) + ck$ pro nějakou konstantu c.

Analogickým postupem obdržíme horní odhad $C_1 k^{(\log_2 3)} + C_2 k \leq Ck^{(\log_2 3)}$.

Oba předpokládané modely je nutné rozšířit o konstantu C_3, která symbolizuje „konstantní režii programu“, tedy například čtení z disku.

A nyní se podívejme na implementaci[4], která byla pro tuto práci zvolena. Jak již bylo popsáno v úvodu, šifrovány byly zprávy v rozmezí velikosti 50 kB a 1,2 MB; klíče byly velikosti 64 bitů až 4096 bitů. Klíče nižších velikostí by jistě nebyly vhodné pro praktické použití, avšak pomůžou nám nahlédnout, jak se skutečná rychlost šifrování (nikoliv teoretická, jak jsme se snažili nahlédnout na začátku této podkapitoly) s narůstajícím klíčem mění.

Jak již bylo naznačeno výše, rychlost šifrování nezávisí pouze na velikosti klíče n, ale i na šifrovacím koeficientu e. Pro porovnání rychlosti byly v této práci zvoleny tři nejběžnější koeficienty – 3, 65537 a koeficient náhodný, tedy velikosti $< n$. První dva koeficienty jsou v praxi využívané kvůli vysoké

[4] Implementaci je možné nahlédnout v souboru rsa.c; generování klíčů pak v common_functions.c

rychlosti šifrování, která je srovnatelná s metodou Rabinovou. Avšak proti těmto nízkým koeficientům existují speciální metody, resp. útoky, které snižují bezpečnost šifry. Tyto útoky jsou mimo rámec této práce. Proto konstantní šifrovací koeficienty jsou představeny spíše ilustrativně a pro další srovnání algoritmů bude použit koeficient náhodný.

Následující tabulka udává rychlost šifrování (počet kilobytů za sekundu) v závislosti na velikosti klíče n a již zmíněných třech šifrovacích koeficientech. Pro ilustraci je zde kromě průměrné hodnoty, se kterou budeme nadále pracovat, i minimální a maximální naměřená rychlost.

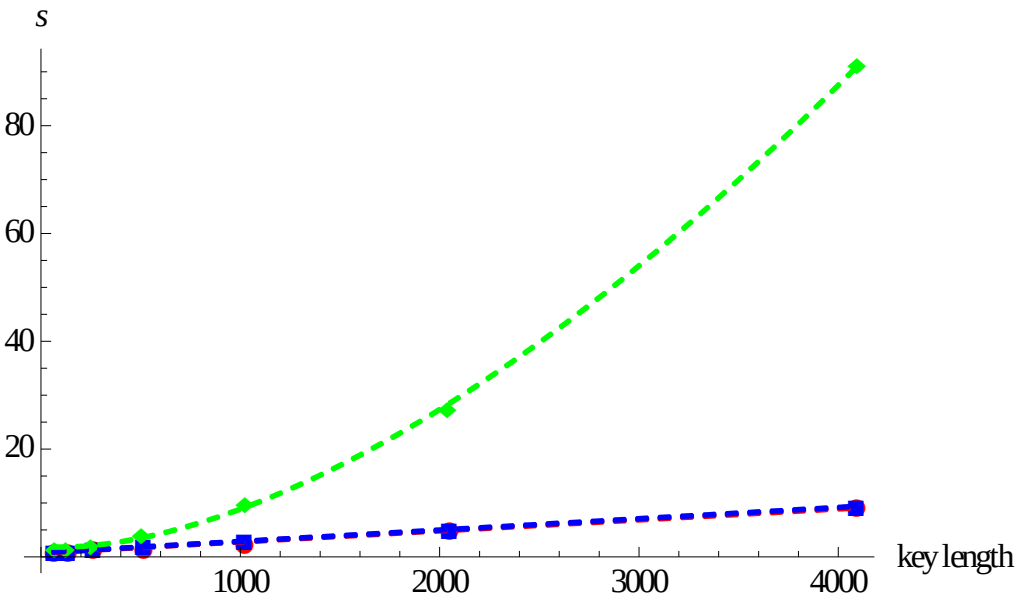
Počet bitů n	Max. počet bitů e	Min(kB/s)	Avg(kB/s)	Max(kB/s)	V toleranci
64	2	1099,3	1167,9	1252,0	99,40 %
64	17	958,0	1071,6	1201,8	97,64 %
64	64	686,8	790,0	846,7	99,40 %
128	2	958,0	1107,8	1201,8	97,16 %
128	17	958,0	1041,2	1201,8	97,04 %
128	128	622,1	680,0	710,5	99,92 %
256	2	760,4	794,4	834,6	99,96 %
256	17	684,3	767,2	801,2	99,48 %
256	256	423,9	451,6	480,7	99,64 %
512	2	532,2	572,4	600,9	99,44 %
512	17	532,2	551,0	600,9	99,76 %
512	512	228,1	236,2	240,8	100,00 %
1024	2	343,4	363,8	369,8	99,88 %
1024	17	342,2	351,6	369,8	99,88 %
1024	1024	103,5	104,7	105,7	100,00 %
2048	2	206,1	207,6	208,7	100,00 %
2048	17	199,7	201,4	202,8	100,00 %
2048	2048	37,0	37,1	37,2	100,00 %
4096	2	112,2	112,6	113,0	100,00 %
4096	17	108,8	109,2	109,8	100,00 %
4096	4096	11,2	11,2	11,2	100,00 %

Z tabulky je zřejmé, že čím menší je velikost klíče, tím větší je rozdíl minimální a maximální hodnoty od průměru.

Poslední sloupec v tabulce značí, kolik procent zpráv „padne“ do pětiprocentní odchylky od průměru. Je tedy vidět, že „výkyvy“ nejsou časté. Při bližší analýze dat nebylo odhaleno, že by existovala zpráva, která by byla šifrována

výrazně rychleji nežli ostatní. Totéž platí i o klíči n; nebyl zjištěn klíč, který by šifroval výrazně rychleji než ostatní. Jak již bylo řečeno, pracujeme s omezenou přesností na 10 ms. To může být zásadní důvod (a to zejména pro krátké zprávy a krátké klíče), proč tyto výkyvy vznikají. Teoreticky může být při šifrování téže zprávy dvěma různými klíči rozdíl 1 ms, přesto se do výsledku započítá při zaokrouhlení rozdíl 10 ms. Dalším významným faktorem, který se neprojeví na hodnotě průměrné, ale na minimu a maximu ano, je velikost šifrovacího exponentu (toto samozřejmě neplatí u koeficientů konstantních).

Níže uvedený graf zobrazuje, jak dlouho trvá šifrování 1 MB (funkce je rekonstruována na základě naměřených hodnot a předpokládaného modelu).



Obr. 1 Rychlost šifrování 1 MB v závislosti na délce klíče. Červená barva značí šifrovací koeficient 3, modrá koeficient 65537 a zelená obecný koeficient.

2.3 DEŠIFROVÁNÍ V PRAXI

Podobně jako při šifrování, i zde se nejprve podívejme na teoretický odhad složitosti. Standardní dešifrování, které nevyužívá čínskou větu o zbytcích, bude mít zjevně složitost obdobnou jako šifrování. Tedy $\mathcal{O}\left(k^{(\log_2 3)+1}\right)$.

Podívejme se tedy na rychlejší dešifrování. Nejprve musíme spočítat čísla $m_p = c^{d \bmod (p-1)} \bmod p$ a $m_q = c^{d \bmod (q-1)} \bmod q$. Jelikož je $c > p$ (resp. $c > q$), budeme počítat ve skutečnosti $(c \bmod p)^{d \bmod (p-1)} \bmod p$ (analogicky pro q). Složitost výpočtu m_p a m_q je tak $\mathcal{O}\left(\left(\frac{k}{2}\right)^{(\log_2 3)+1}\right) = \mathcal{O}\left(k^{(\log_2 3)+1}\right)$.

Následně potřebujeme spočítat $m = \left(m_p y_q q + m_q y_p p\right) \bmod n$. Součiny $y_q q$ a $y_p p$ máme již předpočítány a jsou menší nežli n. Zbývá nám tedy spočítat dva součiny čísel velikosti k a $\frac{k}{2}$. Z definice Karacubova násobení je jasné, že nám stačí jen 2 násobení (sčítání nepočítaje, to má asymptotickou složitost nižší) čísel velikosti $\frac{k}{2}$. Tato složitost je tedy $\mathcal{O}\left(k^{(\log_2 3)}\right)$. Celková složitost je tak $\mathcal{O}\left(k^{(\log_2 3)+1}\right) + \mathcal{O}\left(k^{(\log_2 3)}\right) = \mathcal{O}\left(k^{(\log_2 3)+1}\right)$. To je asymptotická složitost shodná s tou, která je u šifrování standardního. Pokud se však podíváme na počet operací, které jednotlivé algoritmy pro dešifrování provádějí, zjistíme, že standardní verze vyžaduje $2(k-1)M(k)$, kdežto dešifrování založené na čínské větě o zbytcích potřebuje $4\left(\frac{k}{2} - 1\right) M\left(\frac{k}{2}\right) + 4M\left(\frac{k}{2}\right) = 2kM\left(\frac{k}{2}\right)$. Při použití odhadu z Karacubovy metody vychází druhá metoda přibližně třikrát rychlejší.

Nyní zkusme opět odhadnout chování při zdvojnásobení délky klíče. Nebudeme se zabývat standardním algoritmem, protože odhad je totožný s tím, který jsme vyvodili pro šifrování s náhodnou délkou šifrovacího klíče.

$T(M,k) = \left\lceil \frac{M}{\frac{k}{8}-1} \right\rceil 2kM\left(\frac{k}{2}\right)$ a $T(M,2k) = \left\lceil \frac{M}{\frac{k}{4}-1} \right\rceil 4kM(k)$.

S využitím odhadů $\left\lceil \frac{M}{\frac{k}{8}-1} \right\rceil \geq 2\left\lceil \frac{M}{\frac{k}{4}-1} \right\rceil$ a $M(k) \leq 3M\left(\frac{k}{2}\right) + ck$ obdržíme $T(M,2k) \leq 3T(M,k) + ck$. S tímto vzorcem jsme se již setkali a odhadovaný model je $Ck^{(\log_2 3)}$. Tedy opět shodný jako dešifrování standardní metodou. Ale to není překvapení, očekávané konstantní zrychlení je skryto právě

v konstantách. Výsledný model je opět nutné rozšířit o konstantu C_3 symbolizující režii.

Podívejme se nyní na to, jak probíhalo dešifrování v praxi. Vstupem pro dešifrování byly samozřejmě zprávy zašifrované v předchozím kroku. Podobně jako jsme rychlost v případě šifrování měřili na základě velikosti vstupní zprávy, budeme v případě dešifrování vycházet z velikosti výstupní zprávy. Rychlost dešifrování je podobně jako šifrování závislá na dvou parametrech – na velikosti klíče a velikosti dešifrovacího koeficientu. Avšak oproti šifrování si nemůžeme dešifrovací koeficient zvolit, je plně závislý na e. Proto při dešifrování budeme zkoumat pouze chování při náhodné velikosti koeficientu, a nikoliv při velikosti konstantní.

Následující tabulka udává rychlost standardního dešifrování a dešifrování za použití čínské věty o zbytcích (počet kilobytů za sekundu) v závislosti na velikosti klíče n. Standardní metoda je v tabulce označena jako S, metoda za použití čínské věty pak ČV. Pro ilustraci je zde kromě průměrné hodnoty, se kterou budeme nadále pracovat, i minimální a maximální naměřená rychlost.

Počet bitů n	Použitá metoda	Min(kB/s)	Avg(kB/s)	Max(kB/s)	V toleranci
64	S	684,3	816,6	961,5	96,92 %
64	ČV	656,2	766,4	818,3	98,23 %
128	S	598,8	692,5	760,4	98,87 %
128	ČV	622,1	749,5	801,2	97,32 %
256	S	399,2	454,4	480,7	99,03 %
256	ČV	479,0	551,3	598,8	99,13 %
512	S	225,7	237,1	241,9	100,00 %
512	ČV	342,2	367,4	377,2	99,97 %
1024	S	101,3	104,3	105,7	100,00 %
1024	ČV	178,0	189,8	193,1	99,40 %
2048	S	36,9	37,1	37,3	100,00 %
2048	ČV	85,7	86,1	86,5	100,00 %
4096	S	11,2	11,2	11,3	100,00 %
4096	ČV	31,9	32,0	32,1	100,00 %

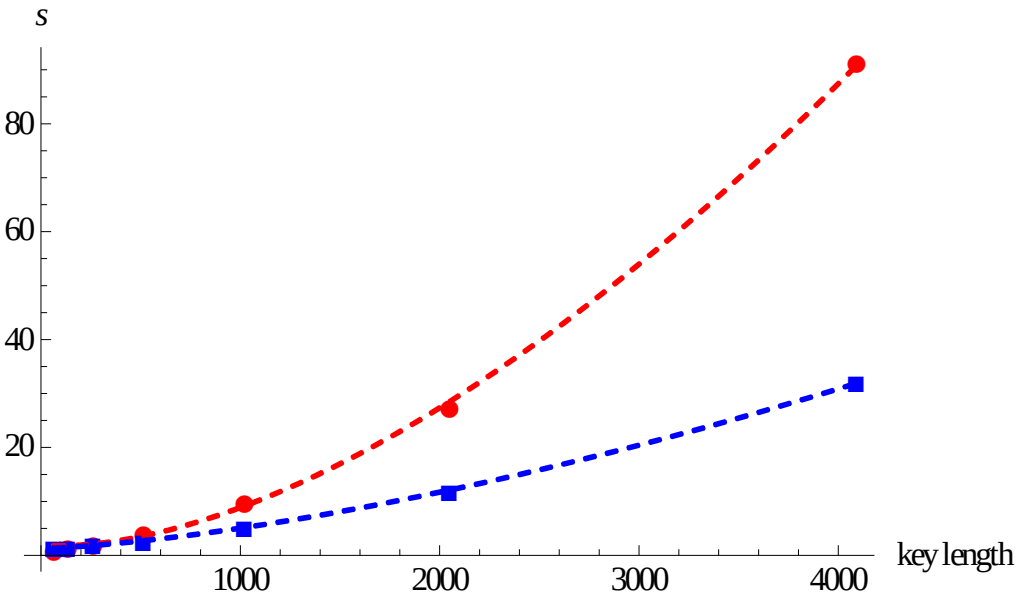
Z tabulky je zřejmé, že s výjimkou klíče o velikosti 64-bitů je dešifrování pomocí čínské věty o zbytcích rychlejší. Pro vyšší hodnoty klíčů se dokonce

rychlost liší až trojnásobně. Což je i hodnota teoreticky vypočtená. Důvod, proč pro klíče nižních velikostí neposkytuje ČV výrazné zrychlení, tkví v reprezentaci čísel, kterou knihovna GMP používá (v našem případě jde o bázi 2^{64}). Proto pro menší klíče nemusí metoda práce s polovičními čísly přinést žádnou úsporu a může být i pomalejší nežli standardní metoda.

Z tabulky je rovněž vidět, kolik procent zpráv se svou rychlostí dešifrování výrazněji (o více než 5 %) liší od průměru. Podobně jako v případě šifrování je takových zpráv pouze nepatrný zlomek. Opět nebyla odhalena zpráva, jejíž dešifrování by bylo výrazně rychlejší než u jiných zpráv. Totéž platí i o klíči n. Důvod, proč tyto odchylky vznikají, je totožný jako u šifrování.

Pokud porovnáme standardní metodu pro dešifrování s šifrováním s náhodným šifrovacím exponentem, zjistíme, že rychlost je prakticky tatáž. To by nemělo překvapit; je aplikována stejná transformace – modulární mocnění.

Níže uvedený graf zobrazuje, jak dlouho trvá dešifrování 1 MB (funkce je rekonstruována na základě naměřených hodnot a předpokládaného modelu).



Obr. 2 Rychlost dešifrování 1 MB v závislosti na délce klíče. Červená barva značí standardní dešifrovací metodu, modrá pak metodu za použití čínské věty o zbytcích.

3 RABIN

3.1 POPIS ALGORITMU

Rabinův kryptografický systém byl zveřejněn v lednu roku 1979. Jeho autorem je Michael Oser Rabin. Podobně jako RSA, i bezpečnost Rabinovy metody je založena na problému hledání faktorizace velkých čísel. Avšak na rozdíl od RSA, u Rabinovy metody bylo dokázáno, že pokud dokáže útočník efektivně prolomit Rabinův kryptografický systém, pak dokáže i najít faktorizaci čísel. Opět se nejprve podíváme, jak vypadají klíče, a poté si ukážeme, jak probíhá šifrování a dešifrování. Rozdělení šifrovaného textu do bloků, velikost bloku a reprezentace textu jakožto přirozeného čísla, to vše je totožné s RSA[5].

Generování klíčů:

Generování klíčů probíhá obdobně, jako je tomu u RSA. Nejprve se vygenerují dvě prvočísla p a q a spočte se jejich součin $n = pq$. Na tato prvočísla však klademe ještě jednu podmínku: $p \equiv 3\ (mod\ 4)$ a $q \equiv 3\ (mod\ 4)$. Existuje i varianta algoritmu, která toto omezení nemá, avšak dešifrování je výrazně pomalejší. Proto v této práci budeme předpokládat, že obě čísla podmínku na kongruenci splňují. Číslo n se nazývá veřejným klíčem a dvojice (p, q) klíčem soukromým.

Bezpečnostní doporučení udává velikost n alespoň 768 – 1024 bitů.

Popis šifrování a dešifrování:

Při šifrování zdrojový (šifrovaný) text M opět rozdělíme do k bloků $\overline{m}_1, \dots, \overline{m}_k$ velikosti b. Každý blok, reprezentovaný číslem, je poté transformován pomocí funkce $c = m^2\ mod\ n$.

Dešifrování probíhá tak, že se spočtou 4 odmocniny $c\ mod\ n$. K tomu je potřeba nejprve pomocí rozšířeného Euklidova algoritmu dopočítat celá čísla

[5] Viz podkapitola 2.1, sekce Velikost bloku a reprezentace textu jako čísla

y_p a y_q taková, že $y_p p + y_q q = 1$. Opět si můžeme všimnout (jako je tomu u RSA), že čísla y_p a y_q jsou zcela nezávislá na šifrovaném textu a je možné je předpočítat. Potom se spočtou hodnoty
$m_p = c^{\frac{p+1}{4}} mod\ p$ a $m_q = c^{\frac{q+1}{4}} mod\ q$ a následně se spočítají čísla
$r = \left(y_p p m_q + y_q q m_p\right) mod\ n$ a $s = \left(y_p p m_q - y_q q m_p\right) mod\ n.$

Výsledné a hledané odmocniny jsou pak čísla $\pm r\ mod\ n$ a $\pm s\ mod\ n$. Jedna z těchto čtyř hodnot je pak hledanou reprezentací šifrované zprávy $\overline{m}$.

Zbývá však ukázat, jak lze poznat, která ze čtyř zpráv je ta pravá. Pokud je šifrován text, je možné správnou zprávu „uhodnout“ – buď tak, že verifikaci provádí člověk, nebo za použití pravděpodobnosti na základě například slovníku. Ani jedna z těchto metod však není vhodná, není-li šifrován text, ale bity. Pro tyto účely se provede modifikace algoritmu tak, aby šifroval pouze zprávy ve speciálním tvaru. Například na konec zprávy jsou doplněny kontrolní bity, některé, předem specifikované, části zprávy jsou duplikovány apod. Při dešifrování se poté postupuje tak, že vybereme ze čtyř možných kandidátů na původní zprávu ten, který odpovídá dohodnuté speciální formě. Tyto kontrolní znaky jsou pak ze zprávy odstraněny.

Je zřejmé, že čím větší chceme mít pravděpodobnost, že nedojde k chybě, tím více redundantních informací musíme k šifrovanému textu připojit a tím déle poběží šifrování i dešifrování, neboť blok velikosti b bude obsahovat pouze $b_1 < b$ bytů textu a zbylých b_2 bytů „padne“ na kontrolu, kde $b = b_1 + b_2$.

Konkrétní implementace, která byla zvolena v této práci, je popsána v následující podkapitole.

3.2 ŠIFROVÁNÍ V PRAXI

Nejprve se již tradičně podíváme, jakou očekáváme asymptotickou složitost šifrování jednoho bloku. Jedinou operací je druhá mocnina modulo n, tedy jedno násobení. Očekáváme tudíž složitost $\mathcal{O}\left(k^{(\log_2 3)}\right)$. Ještě než se

podíváme, jaký očekáváme scénář při zdvojnásobení klíče, zastavme se u otázky, jak vybrat správnou zprávu. V implementaci[6], která byla použita v této práci, byly zvoleny dvě metody. V první byly na konec každého bloku přidány dva byty 255 (tedy dvakrát binárních osm jedniček), a pravděpodobnost chybného dešifrování je tak $\frac{3}{65536}$. Dva byty se ukázaly v praxi poměrně efektivní. Šifrování je rychlé a k chybnému přijetí zprávy nedošlo za celý průběh měření ani jednou. V druhé metodě se předpokládalo, že zprávy budou vyhodnocovány nikoli automaticky, ale manuálně, a proto nebyl použit jediný kontrolní byte (ani bit). Tento speciální případ samozřejmě není pro praktické využití vhodný, lépe však popisuje chování funkce.

Nyní zkusme odhadnout očekávané chování šifrovací funkce při zdvojnásobení klíče.

Uvažujme zprávu velikosti M a velikost n je k-bitů. Šifrování jednoho bloku má složitost $M(k)$, pro celou zprávu M pak $\left\lceil \frac{M}{\frac{k}{8}-b-1} \right\rceil M(k) = T(M,k)$. $M(k)$ značí čas potřebný pro vynásobení dvou čísel binární velikosti k, b je pak počet kontrolních bytů.

Pokud použijeme stejné odhady $\left\lceil \frac{M}{\frac{k}{8}-\mathrm{b}-1} \right\rceil \geq 2 \left\lceil \frac{M}{\frac{k}{4}-b-1} \right\rceil$ a $M(2k) \leq 3M(k) + ck$, obdržíme $T(M,2k) \leq \frac{3}{2}T(M,k) + ck$. S tímto vzorcem jsme se také již setkali a odhadovaný model je $Ck + C_3$.

Následující tabulka udává rychlost šifrování kilobytů za sekundu v závislosti na velikosti klíče a na použité metodě – tedy na počtu kontrolních bytů. Z výsledků je zřejmé, že čím menší je velikost klíče, tím více ovlivňuje počet kontrolních bytů rychlost. Například je-li velikost klíče 64 bitů, pak se v případě dvou kontrolních bytů v každém kroku šifry, tedy v jednom bloku, šifruje 5 bytů originální zprávy. Pokud není použit žádný kontrolní byte, pak

[6] Implementaci je možné nahlédnout v souboru rabin.c; generování klíčů pak v common_functions.c

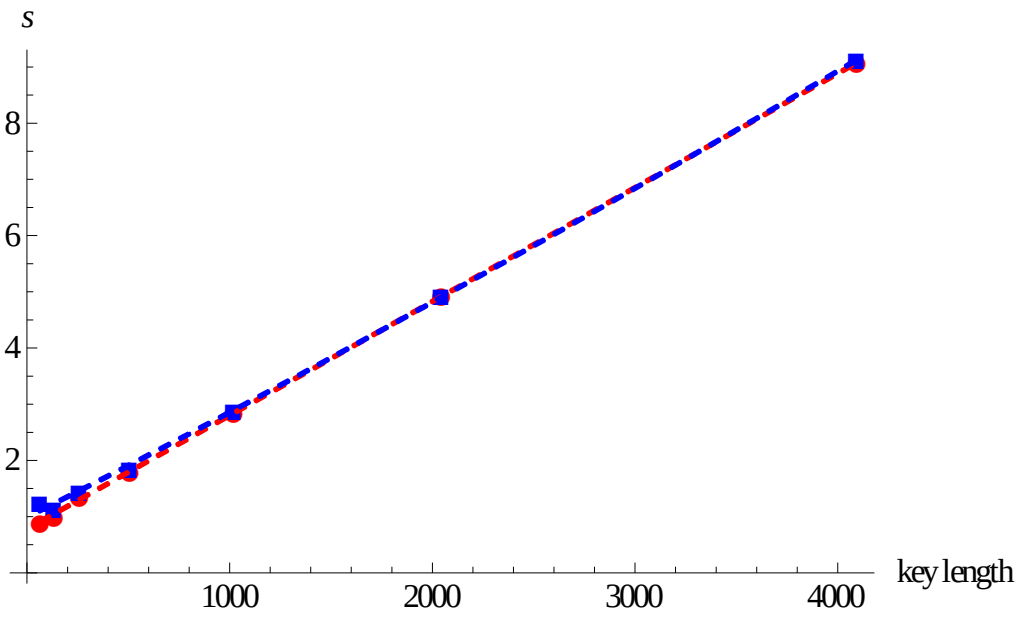
je šifrováno bytů 7. Tedy o 40 % více. Avšak v případě klíče velikosti 4096 bitů je v jednom případě šifrováno 509 a v druhém 511 znaků. Rozdíl nečiní ani 1 %. Tyto poměry jsou vidět i v níže uvedené tabulce, srovnáme-li průměrné rychlosti šifrování.

Počet bitů n	Počet kont. znaků	Min(kB/s)	Avg(kB /s)	Max(kB /s)	V toleranci
64	0	958,0	1119,2	1201,8	96,96 %
64	2	782,5	821,7	961,5	99,84 %
128	0	958,0	1034,6	1201,8	97,40 %
128	2	798,4	899,9	961,5	97,36 %
256	0	684,3	755,9	801,2	97,28 %
256	2	684,3	708,2	763,1	99,72 %
512	0	532,2	568,1	600,9	97,88 %
512	2	532,2	550,1	600,9	99,84 %
1024	0	342,2	359,6	369,8	100,00 %
1024	2	342,2	354,2	369,8	100,00 %
2048	0	206,8	208,1	209,9	100,00 %
2048	2	205,2	206,4	207,7	100,00 %
4096	0	112,5	112,8	113,2	100,00 %
4096	2	111,8	112,3	112,8	100,00 %

Podobně jako v předchozí kapitole, i zde sloupec v toleranci značí, kolik procent zpráv má rychlost počtu šifrovaných bytů „v toleranci“, tj. neliší se od průměru o více než pět procent.

Velmi pozoruhodné je, že při dvou kontrolních bytech je rychlost šifrování pro klíč velikosti 128 bitů vyšší než při klíči s 64 bity. To je způsobeno tím, že při 64-bitovém klíči je šifrováno pouze 5 bytů nesoucích informaci, kdežto pro klíč 128-bitový jich je 13. Tedy 2,6krát více. Režie spojená s vykonáním jednoho cyklu pak v tomto případě převýšila větší náročnost násobení čísel dvojité délky.

Níže uvedený graf zobrazuje, jak dlouho trvá šifrování 1MB (funkce je rekonstruována na základě naměřených hodnot a předpokládaného modelu).



Obr. 3 Rychlost šifrování 1 MB v závislosti na délce klíče. Červená barva značí šifrovací metodu bez kontrolních bytů, modrá pak metodu za použití dvou kontrolních bytů.

3.3 DEŠIFROVÁNÍ V PRAXI

Podívejme se nejprve na asymptotickou složitost. Dešifrování jednoho bloku zahrnuje spočtení m_p a m_q, což obnáší dvě modulární mocnění čísel velikosti $\frac{k}{2}$-bitů. Celkově tedy $\mathcal{O}\left(k^{(\log_2 3)+1}\right)$.

Následně potřebujeme spočítat r a s. Součiny $y_q q$ a $y_p p$ máme již předpočítány a jsou menší nežli n. Zbývá nám tedy spočítat dva součiny čísel velikost k a $\frac{k}{2}$. Z definice Karacubova násobení je jasné, že nám stačí jen 2 násobení (sčítání nepočítaje, to má asymptotickou složitost nižší) čísel velikosti $\frac{k}{2}$. Tato složitost je tedy $\mathcal{O}\left(k^{(\log_2 3)}\right)$. Celková asymptotická složitost dešifrování jednoho bloku je $\mathcal{O}\left(k^{(\log_2 3)+1}\right)$. Vybrání, která zpráva je „tou pravou“, má asymptotickou složitost nižší – v případě konstantního počtu kontrolních bytů je složitost konstantní, v případě duplikace části zprávy pak lineární. Tento odhad jistě nepřekvapí. Dešifrování je obdobné tomu, které je prováděno v algoritmu RSA.

Chová se dešifrovací funkce podobně i při zdvojnásobení klíče? Na to se nyní podíváme.

$$T(M,k) = \left\lceil \frac{M}{\frac{k}{8}-b-1} \right\rceil \left(4\left(\frac{k}{2}-1\right)M\left(\frac{k}{2}\right) + 4M\left(\frac{k}{2}\right)\right) = \left\lceil \frac{M}{\frac{k}{8}-b-1} \right\rceil \left(2kM\left(\frac{k}{2}\right)\right)$$

$$T(M,2k) = \left\lceil \frac{M}{\frac{k}{4}-b-1} \right\rceil \left(4kM(k)\right)$$

Opět využijeme odhadů $\left\lceil \frac{M}{\frac{k}{8}-\mathrm{b}-1} \right\rceil \geq 2\left\lceil \frac{M}{\frac{k}{4}-b-1} \right\rceil$ a $M(k) \leq 3M\left(\frac{k}{2}\right) + ck$ a získáme $T(M,2k) \leq 3T(M,k) + ck$. To nám dává již dobře známý model $Ck^{(\log_2 3)} + C_3$.

Následující tabulka opět udává rychlost dešifrování v závislosti na velikosti klíče a počtu kontrolních bytů. Podobně jako v ostatních případech i zde uvádíme minimální a maximální rychlost šifrování i počet zpráv, které se „vejdou“ do tolerance.

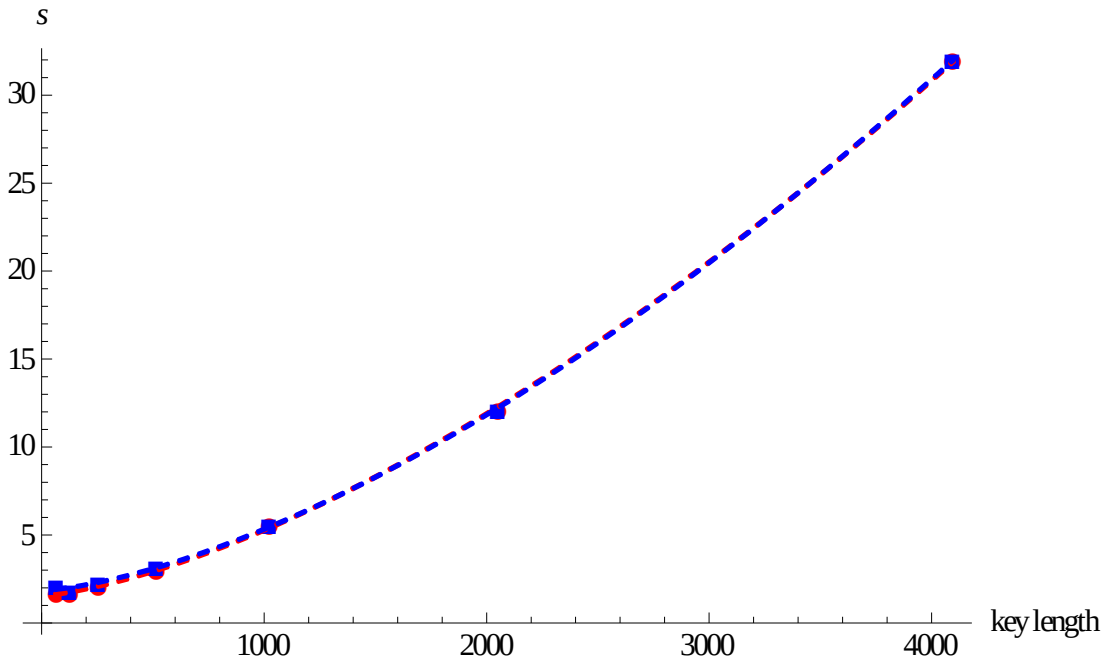
Počet bitů n	Počet kont. znaků	Min(kB/s)	Avg(kB/s)	Max(kB/s)	V toleranci
64	0	532,2	635,2	686,8	97,60 %
64	2	435,5	479,9	515,1	98,96 %
128	0	534,1	631,7	686,8	98,28 %
128	2	480,7	564,4	600,9	97,60 %
256	0	409,4	476,4	488,6	99,04 %
256	2	399,2	459,6	480,7	98,68 %
512	0	313,1	334,0	343,4	99,92 %
512	2	300,5	329,2	343,4	99,88 %
1024	0	176,1	183,7	185,8	100,00 %
1024	2	175,5	182,6	184,9	100,00 %
2048	0	84,4	84,7	85,0	100,00 %
2048	2	84,0	84,4	84,7	100,00 %
4096	0	32,0	32,1	32,1	100,00 %
4096	2	31,9	32,0	32,1	100,00 %

Z tabulky jsou ihned vidět dvě důležité věci. První je už podle algoritmu zřejmý fakt; dešifrování je výrazně pomalejší nežli šifrování. Ale to vzhledem k asymptotickým složitostem obou operací není překvapením.

Druhým pozorováním je fakt, že s rostoucím klíčem se rychlost dešifrování blíží rychlosti dešifrování u RSA (samozřejmě u varianty používající čínskou větu o zbytcích). Ale jak již bylo několikrát řečeno, pravděpodobnost úspěšného dešifrování (resp. počet kontrolních bytů, které v této práci používáme) by v praxi nemusela být dostatečná. Je zřejmé, že kdybychom použili například 10 kontrolních bytů namísto dvou, rychlost se sníží o cca 300 bytů za sekundu, avšak pravděpodobnost neúspěšného dešifrování se sníží na $\frac{3}{2^{80}} \approx 2{,}5.\,10^{-24}$. To je již pravděpodobnost akceptovatelná ve většině případů a rychlost je stále srovnatelná s RSA.

Jinak se opět potvrdilo to, co se ukázalo již v přecházejících algoritmech, resp. operacích. Počet zpráv, které mají větší (resp. menší) rychlost dešifrování nežli průměr s tolerancí pět procent, je relativně malý – 2,5 %.

Níže uvedený graf zobrazuje, jak dlouho trvá dešifrování 1 MB (funkce je rekonstruována na základě naměřených hodnot a předpokládaného modelu).



Obr. 4 Rychlost dešifrování 1 MB v závislosti na délce klíče. Červená barva značí dešifrovací metodu bez kontrolních bytů, modrá pak metodu za použití dvou kontrolních bytů.

4 ELGAMAL

4.1 POPIS ALGORITMU

Kryptografický systém ElGamal byl poprvé popsán Taherem ElGamalem v roce 1984. Na rozdíl od předešlých dvou algoritmů je bezpečnost této metody spojena s Diffie-Hellmanovým problémem, resp. s problémem diskrétního logaritmu[7].

Generování klíčů:
Nejprve se vygeneruje prvočíslo n a spočte se primitivní odmocnina $g \ mod \ n$. Poté se vygeneruje náhodné celé číslo $a, 0 \leq a \leq n-2$. Následně se spočte $A = g^a \ mod \ n$.

Veřejným klíčem je trojice (n, g, A), klíčem soukromým je pak číslo a.

Bezpečnostní doporučení udává velikost n alespoň 768 – 1024 bitů.

Popis šifrování a dešifrování:
Opět předpokládejme, že text M je rozdělen do k bloků $\overline{m}_1, \dots, \overline{m}_k$ a m číselná reprezentace šifrovaného bloku.

Pro šifrování je nutné nejprve vygenerovat náhodné celé číslo $b, 0 \leq b \leq n-2$ a spočítat hodnotu $B = g^b \ mod \ n$. Dále se spočte $c = A^b m \ mod \ n$. Výsledný šifrovaný text je pak dvojice (B, c). Číslo b je nutné generovat pro každý blok znovu. Pokud by bylo použito stejné b pro všechny bloky, pak na základě znalosti jediného bloku původního textu a odpovídajícího textu zašifrovaného by bylo možné dešifrovat libovolný jiný blok.

Dešifrování probíhá tak, že se spočte $m = B^{n-1-a} c \ mod \ n$. Tím je získán původní text.[8]

[7] Viz například [1], kapitola 7, strana 159, odstavec 7.4.1
[8] Snadný důkaz, že dešifrování funguje, je možné nahlédnout např. v [2], kapitola 8, strana 294, sekce 8.4.1

Je vidět, že na rozdíl od předešlých dvou algoritmů má ElGamalova metoda jednu hlavní nevýhodu – při šifrování vzniká text dvojnásobné velikosti oproti textu zdrojovému. To je také hlavní důvod, proč není algoritmus tolik rozšířený.

Šifrování je možné výrazně urychlit tak, že se hodnoty b, B a A^b, resp. sady hodnot b_i, B_i a A^{b_i}, předpočítají. Jsou totiž zcela nezávislé na šifrovaném textu.

4.2 ŠIFROVÁNÍ V PRAXI

Jak je již napsáno v předcházející podkapitole, rychlost šifrování je kromě velikosti klíče n závislá i na velikosti čísla b. V implementaci[9], která je použita v této práci, předpokládáme dvě věci. První je fakt, že hodnoty B a A^b máme předpočítané. To znamená, že v rámci šifrování negenerujeme náhodné koeficienty ani neprovádíme dvě modulární násobení. Je zřejmé, že tento předpoklad je omezující, avšak bez něho by měření rychlosti a srovnávání s ostatními algoritmy nemělo valný smysl. Bylo by totiž potřeba dvou modulárních násobení, což je oproti ostatním algoritmům minimálně dvojnásobná složitost.

Druhý předpoklad se může na první pohled zdát nesmyslný – šifrovací koeficient b je pro každý blok stejný. To je samozřejmě v praxi nemyslitelné. Pro naše účely, tedy pro účely měření rychlosti, je to však vhodné. Není třeba pro každou zprávu a klíč evidovat velké množství šifrovacích koeficientů a počítat průměrnou délku; pro každou zprávu nám stačí jediný koeficient. Vzhledem k tomu, že používáme několik desítek klíčů, a tedy i několik desítek náhodných koeficientů b, blíží se jejich průměr hodnotě, která by vznikla tehdy, kdybychom pro každou zprávu počítali c hodnot šifrovacího exponentu namísto hodnoty jediné.

[9] Implementaci je možné nahlédnout v souboru elgamal.c; generování klíčů pak v common_functions.c

Klíč n, který byl použit v této práci, má jedno specifikum. Pro klíče menší či rovné 2048 bitům je použito tzv. bezpečné prvočíslo, tedy prvočíslo n takové, že $\frac{n}{2} - 1$ je rovněž prvočíslem. Číslo v tomto formátu poskytuje vcelku jednoduchou metodu, jak hledat primitivní odmocninu modulu n. Pro klíč velikosti 4096 bitů se však tato metoda v praxi neosvědčila. Vzhledem k tomu, jak se vzrůstajícím klíčem klesá hustota prvočísel, nepodařilo se nalézt jediné bezpečné prvočíslo. Proto byl zvolen jiný postup. Bylo vygenerováno prvočíslo n a generátor g byl vygenerován na základě pravděpodobnostního algoritmu[10]. Pro účely šifrování jsou však obě metody ekvivalentní, nebylo zjištěno, že by n ve speciálním tvaru mělo „lepší vlastnosti".

Nyní se podívejme na asymptotickou složitost. Opět si připomeňme důležitý fakt – hodnoty B a A^b jsou předpočítané. Proto šifrování vyžaduje jediné násobení dvou čísel modulu n. Složitost je tak zřejmě rovna složitosti Karacubova násobení, tedy $\mathcal{O}\left(k^{(\log_2 3)}\right)$.

Teď zkusme zdvojnásobit velikost klíče a odhadnout chování šifrovací funkce. Vzhledem k tomu, že je použito jen jediné násobení, můžeme použít analogický důkaz z šifrování pomocí Rabinovy metody a dostaneme $Ck + C_3$.

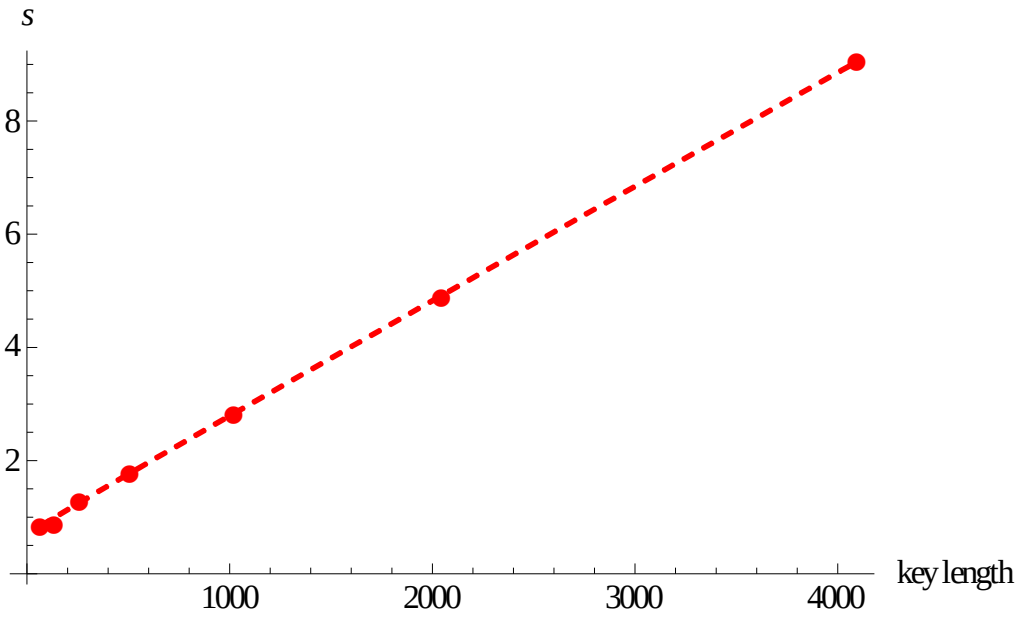
Níže uvedená tabulka ukazuje rychlost šifrování v závislosti na velikosti klíče n. Sloupce opět ukazují minimální, průměrnou a střední rychlost šifrování a také informaci, kolik procent zpráv je liší od průměru o více než 5 %.

Počet bitů n	Min(kB/s)	Avg(kB/s)	Max(kB/s)	V toleranci
64	1100,7	1192,8	1252	99,84 %
128	1019,2	1134,5	1222,9	97,16 %
256	686,8	782,3	809,3	99,92 %
512	532,2	568,9	600,9	98,28 %
1024	342,2	362,5	369,8	99,80 %
2048	205,2	208,4	209,9	100,00 %
4096	112,4	113,1	113,5	100,00 %

[10] Ten je možné vidět v elgamal.c, funkce lga_generate_keys

Šlo by dosáhnout i optimalizace tak, že by se při generování koeficientů (v „předpočítavací“ fázi) preferovaly ty koeficienty b, pro které by $A^b \bmod n$ bylo nejmenší. Ale zvláště pro větší klíče by se zrychlení takřka neprojevilo. Tuto variantu však v této práci nepředpokládáme.

Níže uvedený graf zobrazuje, jak dlouho trvá šifrování 1 MB (funkce je zrekonstruována na základě naměřených hodnot a předpokládaného modelu).



Obr. 5 Rychlost šifrování 1 MB v závislosti na délce klíče.

4.3 DEŠIFROVÁNÍ V PRAXI

Podobně jako v případě šifrování, i zde pro účely měření rychlosti předpokládáme, že každý blok je šifrován stejným koeficientem b. Avšak na rozdíl od šifrování nyní nemůžeme předpokládat, že B^{n-1-a} je hodnota předpočítaná. Sice je v našem případě B konstantní, ale jak již bylo řečeno v části o šifrování, to je pouze z důvodu jednodušší implementace. Při dešifrování musíme předpokládat, že je každé B jiné, jinak bychom předpokládali znalost b a nejednalo by se o šifrování s veřejným klíčem. Tedy i když je hodnota de facto konstantní, je počítána pro každý blok znovu, čímž se supluje změna b a složitost dešifrování zůstane zachována.

Jaká je vlastně složitost? Pro dešifrování je nutné provést nejprve $B^{n-1-a} \bmod n$, což odpovídá složitosti $\mathcal{O}\left(k^{(\log_2 3)+1}\right)$. Poté je potřeba jedno násobení dvou čísel velikosti k, což již asymptotickou složitost nezmění.

Nyní se zkusme opět podívat, jaké očekáváme chování dešifrovací funkce při zdvojnásobení velikosti klíče.

$T(M,k) = \left\lceil \frac{M}{\frac{k}{8}-1} \right\rceil (2k)M(k)$ a $T(M,2k) = \left\lceil \frac{M}{\frac{k}{4}-1} \right\rceil (4k)M(2k)$.

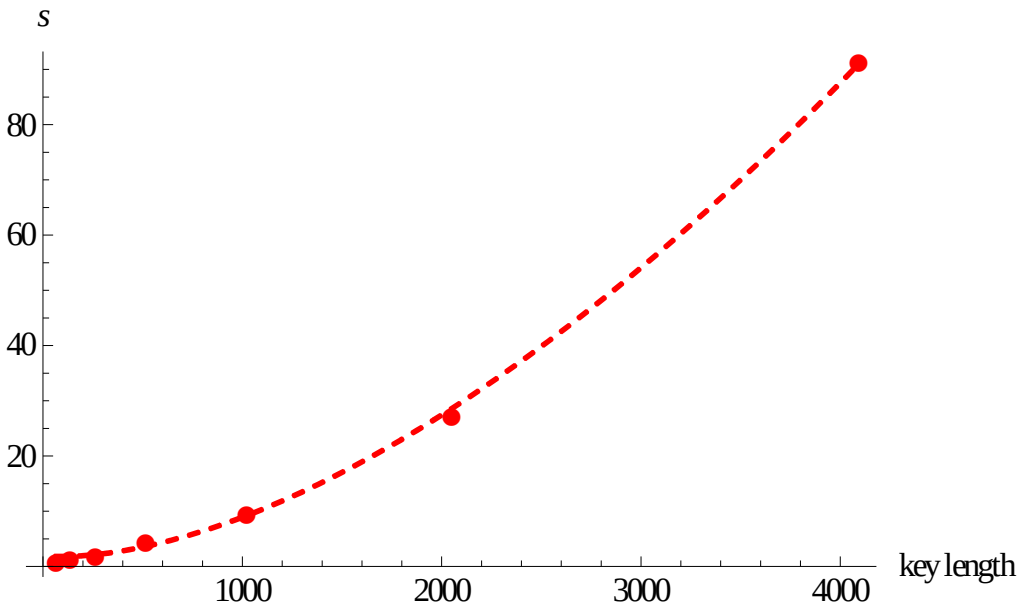
Znovu použijme odhady $\left\lceil \frac{M}{\frac{k}{8}-1} \right\rceil \geq 2 \left\lceil \frac{M}{\frac{k}{4}-1} \right\rceil$ a $M(2k) \leq 3M(k) + ck$ a dle Karacubova násobení dostaneme $T(M,2k) \leq 3T(M,k) + ck$ pro nějakou konstantu c.

Tedy odhad složitosti je $Ck^{(\log_2 3)} + C_3$.

Následuje tabulka rychlosti dešifrování, kde si opět ukážeme nejmenší, průměrnou a nejvyšší rychlost dešifrování v závislosti na klíči n, resp. na jeho velikosti k. Nechybí samozřejmě ani sloupec, který ukazuje, kolik zpráv se od průměru výrazněji odchyluje.

Počet bitů n	Min(kB/s)	Avg(kB/s)	Max(kB/s)	V toleranci
64	684,3	813,2	958,0	98,00 %
128	577,1	689,0	719,4	99,60 %
256	410,9	450,6	480,7	99,16 %
512	217,7	231,1	240,4	99,92 %
1024	101,9	103,5	104,7	100,00 %
2048	36,9	37,1	37,3	100,00 %
4096	11,2	11,2	11,2	100,00 %

Níže uvedený graf zobrazuje, jak dlouho trvá dešifrování 1 MB (funkce je zrekonstruována na základě naměřených hodnot a předpokládaného modelu).



Obr. 6 Rychlost dešifrování 1 MB v závislosti na délce klíče.

5 SROVNÁNÍ ALGORITMŮ

5.1 ŠIFROVÁNÍ

V této podkapitole se pokusíme srovnat rychlost šifrování všech tří algoritmů. U RSA zvolíme variantu náhodného šifrovacího exponentu e (metodu označíme RSA n) a opět z ilustrativních důvodů připojíme i konstantní šifrovací exponent 3 (RSA 3). U Rabinovy metody (RAB) použijeme variantu bez kontrolních bytů – sice není v praxi běžná, ale není zatížena „chybou“ diskutovanou v kapitole 3, kde větší klíč nemusí znamenat pomalejší šifrování. U ElGamalova šifrování (ELG) použijeme potom již představenou „předpočítanou“ variantu.

Varianta RSA s náhodným šifrovacím koeficientem má asymptotickou časovou složitost $\mathcal{O}\left(k^{(\log_2 3)+1}\right)$, ostatní tři metody, tedy RSA s konstantním šifrovacím koeficientem, Rabinova i ElGamalova metoda, mají pak složitost $\mathcal{O}\left(k^{(\log_2 3)+1}\right)$.

Následující tabulka shrnuje průměrné rychlosti šifrování v kilobytech za sekundu pro výše popsané čtyři metody v závislosti na velikosti klíče n.

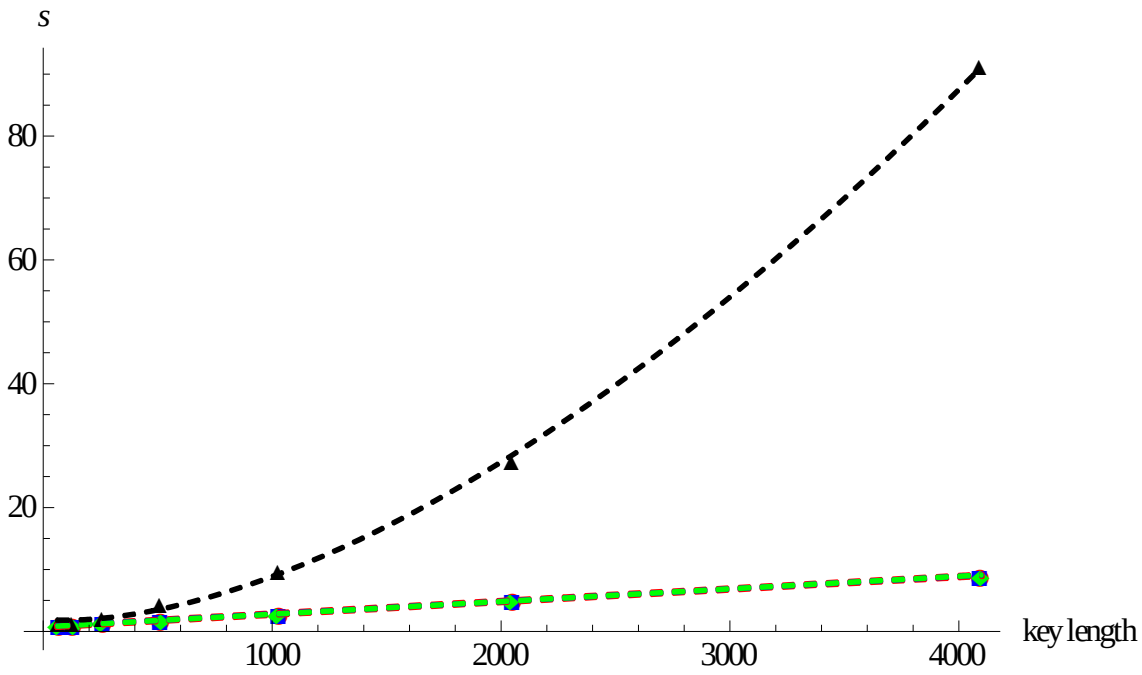
Počet bitů n	RSA n	RSA 3	RAB	ELG
64	790,0	1167,9	1119,2	1192,8
128	680,0	1107,8	1034,6	1134,5
256	451,6	794,4	755,9	782,3
512	236,2	572,4	568,1	568,9
1024	104,7	363,8	359,6	362,5
2048	37,1	207,6	208,1	208,4
4096	11,2	112,6	112,8	113,1

V případě RSA 3 se provádějí dvě násobení čísel velikosti přibližně k. V případě RAB je to pouze jedno násobení čísel velikosti k, kdežto v případě ELG metody se provádí násobení čísel n a m, kde $m = A^b \ mod \ n$. První číslo má velikost k, ale druhé na rozdíl od RAB může mít binární velikost menší. Pokud tedy porovnáváme RSA 3, RAB a ELG, měla by být metoda RSA 3

nejpomalejší a RAB s ELG podobně rychlé, resp. ELG velmi mírně rychlejší. Avšak v praxi se toto pozorování potvrdilo až pro klíče nejvyšších velikostí. Důvod, proč byla metoda RAB v praxi nepatrně pomalejší, se nepodařilo přesně zjistit. Pravděpodobně souvisí s neoptimální prací s pamětí a vyšší režií programu. Nicméně rozdíly jsou zcela minimální, a výsledky lze považovat za totožné. Vzhledem k tomu, jak podobné operace se ve všech třech metodách provádějí, je tento výsledek potvrzením očekávaného.

Jediná metoda, která je výrazněji pomalejší, je RSA n. To však vzhledem k vyšší asymptotické složitosti nepřekvapí.

Níže uvedený graf zobrazuje, jak dlouho trvá šifrování 1 MB (funkce je zrekonstruována na základě naměřených hodnot a předpokládaného modelu).



Obr. 7 Rychlost šifrování 1 MB v závislosti na délce klíče. Červená barva značí šifrovací metodu ELG, modrá RAB, zelená RSA 3 a černá RSA n.

Z obrázku je patrné, že LGA, RAB i RSA 3 na grafu téměř splývají.

5.2 DEŠIFROVÁNÍ

Podobně jako v předchozí podkapitole se zde pokusíme srovnat rychlost dešifrování u všech tří algoritmů. Opět zvolíme u Rabinovy metody případ s nulovým počtem kontrolních bytů. U RSA zvolíme rychlejší algoritmus pro dešifrování, založený na čínské větě o zbytcích. U ElGamalovy metody nemáme na výběr.

Asymptotická složitost všech tří metod je $\mathcal{O}\left(k^{(\log_2 3)+1}\right)$. Nicméně, jak již bylo naznačeno v dílčích kapitolách, RSA a RAB jsou metody, zvláště pro vyšší klíče, rychlejší.

Následující tabulka shrnuje průměrné rychlosti dešifrování pro tři výše popsané metody v závislosti na velikosti klíče n.

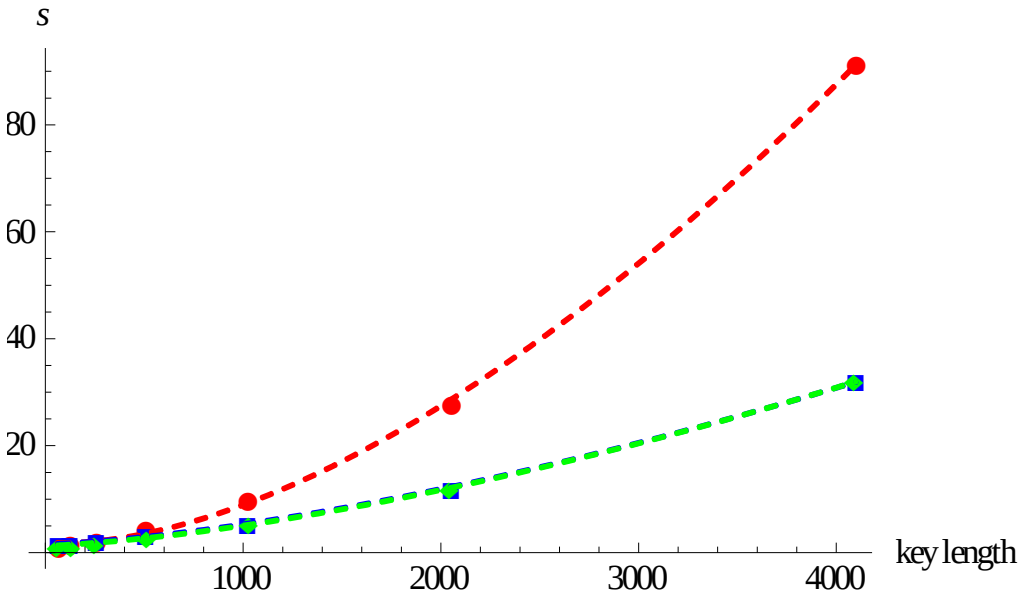
Počet bitů n	RSA	RAB	ELG
64	766,4	635,2	813,2
128	749,5	631,7	689,0
256	551,3	476,4	450,6
512	367,4	334,0	231,1
1024	189,8	183,7	103,5
2048	86,1	84,7	37,1
4096	32,0	32,1	11,2

Dešifrování RSA provádí méně operací sčítání nežli Rabinova metoda, proto očekáváme, že bude dešifrování u RSA nepatrně rychlejší. Toto pozorování se v praxi prokázalo pro klíče velikosti 128 – 2048 bitů. U klíče velikosti 4096 bitů je sice rychlejší metoda Rabinova, ale o pouze 60 bytů, což je hodnota odpovídající 2 ms. To je pod hranicí přesnosti našeho měření. Proto můžeme oba výsledky považovat za totožné.

Také si můžeme všimnout, že pro nejnižší sledovaný klíč, tedy pro klíč velikosti 64 bitů, je nejvýhodnější metodou ELG. Ale důvod je shodný s tím, který je uveden v podkapitole RSA – dešifrování[11].

[11] Viz kapitola 2.3, odstavec pod tabulkou rychlosti šifrování

Níže uvedený graf zobrazuje, jak dlouho trvá dešifrování 1 MB (funkce je zrekonstruována na základě naměřených hodnot a předpokládaného modelu).



Obr. 8 Rychlost dešifrování 1 MB v závislosti na délce klíče. Červená barva značí šifrovací metodu ELG, modrá RAB a zelená RSA.

Z obrázku je patrné, že RAB i RSA na grafu téměř splývají.

6 ZÁVĚR

V průběhu práce se podařilo implementovat tři základní algoritmy pro šifrování s veřejným klíčem, metody RSA, Rabin a ElGamal. Podařilo se odvodit asymptotické složitosti šifrování jednoho bloku a odvodit chování při zdvojnásobení klíče.

V praktické části byly představeny průměrné rychlosti šifrování a dešifrování jednotlivých algoritmů a byly diskutovány možné příčiny vzniku statistických odchylek a režie programu.

V závěrečné kapitole byly jednotlivé algoritmy srovnány a v praxi byly ověřeny informace o srovnání rychlosti jednotlivých algoritmů, které jsou uváděny v literatuře.

7 LITERATURA

[1] Buchman J.: *Introduction to cryptography*, Springer, 2001.

[2] Menezes A., van Oorschot P., Vanstone S.: *Handbook of Applied Cryptography*, CRC Press, 1996.

[3] Stanovský D.: *Počítačová algebra*, http://www.karlin.mff.cuni.cz/~stanovsk/vyuka/skripta_palg.pdf.

8 DODATKY

8.1 OBSAH CD

Adresář keys – obsahuje textové soubory s použitými klíči

Adresář library – obsahuje zdrojové soubory main.c (řídicí část programu), const.c (seznam konstant), common_functions.c (obecné procedury), rsa.c (funkce zajišťující šifrování / dešifrování metodou RSA), rabin.c (funkce zajišťující šifrování / dešifrování metodou Rabin) a elgamal.c (funkce zajišťující šifrování / dešifrování metodou ElGamal).

Adresář messages – obsahuje zdrojové xml zprávy, které byly šifrovány

Adresář results – obsahuje výsledky měření