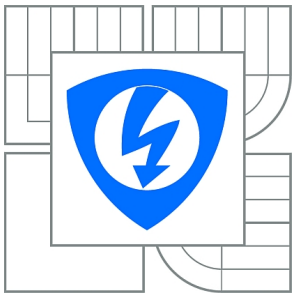
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

MODERNÍ ASYMETRICKÉ KRYPTOSYSTÉMY

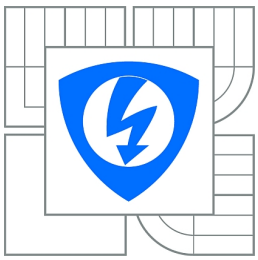
MODERN ASYMMETRIC CRYPTOSYSTEMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. VLADISLAV WALEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. LUKÁŠ MALINA
SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Vladislav Walek	***ID:***	98425
Ročník:	2	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Moderní asymetrické kryptosystémy

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte a popište moderní asymetrické kryptosystémy. Zaměřte se nejen na současně používané metody, ale také na nové koncepty a návrhy asymetrických kryptosystémů. Srovnejte jejich vlastnosti především z hlediska bezpečnosti a efektivity. Dále odhadněte a zdůvodněte jejich budoucí perspektivu. Vybrané asymetrické kryptosystémy prakticky implementujte, a poté podrobně analyzujte a porovnejte jejich efektivitu, výkonnostní charakteristiky jednotlivých funkcí (šifrování, dešifrování, ustanovení parametrů atd.) a bezpečnostní vlastnosti implementace.

DOPORUČENÁ LITERATURA:

[1] STALLINGS, William. Cryptography and Network Security. 4th edition. [s.l.] : [s.n.], 2006. 592 s. ISBN 0131873164.

[2] MENEZES, Alfred, VAN OORSCHOT, Paul, VANSTONE, Scott. Handbook of applied cryptography. Boca Raton : CRC Press, 1997. 780 s. ISBN 0849385237.

[3] FREEMAN, Adam, JONES, Allen. Programming .NET Security. [s.l.] : [s.n.], 2003. 704 s. ISBN 0596004427.

Termín zadání:	7.2.2011	***Termín odevzdání:***	26.5.2011

Vedoucí práce: Ing. Lukáš Malina

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

Abstrakt

Asymetrická kryptografie používá dvojici klíčů k šifrování veřejný klíč a k dešifrování soukromý klíč. Mezi asymetrické kryptosystémy patří RSA, ElGamal, eliptické křivky a jiné. Obecně je asymetrická kryptografie používaná hlavně pro utajování krátkých zpráv pro přenos šifrovacího klíče pro symetrickou kryptografii.

Práce pojednává o těchto systémech a implementuje vybrané systémy (RSA, ElGamal, McEliece, eliptické křivky a NTRU) do programu. Pomocí programu lze testovat vlastnosti vybraných kryptosystémů. Díky naměřeným hodnotám jsou porovnány tyto systémy a lze vyhodnotit jejich časovou a paměťovou náročnost. Z výsledků lze předpovědět jejich budoucí použití v moderních informačních systémech.

Klíčová slova

Kryptografie, Asymetrická kryptografie, faktorizace, problém diskretního logaritmu, RSA, ElGamal, McEliece, eliptické křivky, NTRU encrypt

Abstract

Asymmetric cryptography uses two keys for encryption public key and for decryption private key. The asymmetric cryptosystems include RSA, ElGamal, Elliptic Curves and others. Generally, asymmetric cryptography is mainly used for secure short messages and transmission encryption key for symmetric cryptography.

The thesis deals with these systems and implements selected systems (RSA, ElGamal, McEliece, elliptic curves and NTRU) into the application. The application can test the features of chosen cryptosystems. These systems and their performance are compared and evaluated with the measured values. These results can predict the future usage of these systems in modern informatics systems.

Key words

Cryptography, Asymmetric cryptography, factorization, discrete logarith problem, RSA, ElGamal, McEliece, Elliptic Curves, NTRU Encrypt

Bibliografická citace:

WALEK, V. Moderní asymetrické kryptosystémy. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 63 s. Vedoucí diplomové práce Ing. Lukáš Malina.

Prohlášení

Prohlašuji, že svoji diplomovou práci na téma „Moderní Asymetrické Kryptosystémy“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č.1/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne ....

podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Ing. Lukášovi Malinovi za cenné rady a konzultace, které mi pomohly při vypracování práce.

V Brně dne ....

podpis autora

Obsah

Seznam ilustrací

Seznam tabulek

ÚVOD

Zabezpečování dat proti zneužití se používá ve světě dennodenně při přenosu různých informací přes Internet. Komunikace na Internetu není vždy zcela bezpečná a důvěryhodná, a proto je třeba účinně zabezpečit důležité informace před jejích zneužitím.

Cílem diplomové práce je seznámení s moderními asymetrickými kryptosystémy a se základními vlastnostmi tohoto odvětví. Práce je zaměřená na porovnání moderních kryptografických systémů z hlediska jejich efektivity, náročnosti na výpočetní stroj a bezpečnosti. V první a druhé části jsou vysvětleny základní a nejčastější matematické problémy, na kterých jsou různé systémy založeny. Každý systém je postaven na jednom z těchto problémů a snaží se jej využít ve vlastní prospěch. Na neřešitelnosti daného problému stojí bezpečnost klíčů a tím bezpečnost celého systému. Ve třetí části jsou popsány algoritmy různých systémů pro generování klíčů a algoritmy pro samotné šifrování zpráv určené pro přenos. Je popsána náročnost výpočtu algoritmů i jejich vytížení na výpočetní stroj. Ve čtvrté části je pak určena jejich bezpečnost z hlediska prolomení i jejich efektivita v nasazení pro běžné použití. V páté kapitole jsou zobrazeny tabulky všech moderních asymetrických metod z hlediska jejich bezpečnosti, efektivity a výpočetní náročnosti na hardware. Další dvě částí diplomové práce obsahují popis jazyku Java a popis implementace vybraných kryptografických systémů v tomto prostředí. Poslední část obsahuje výsledky naměřených hodnot, které jsou vyneseny do porovnávacích grafů. Poslední kapitola obsahuje autorovu úvahu na téma budoucího využití implementovaných kryptosystémů z hlediska jejich efektivity a bezpečnosti. V závěru budou všechny dosažené znalosti z praktické části shrnuty a zhodnoceny.

1 KRYPTOGRAFIE

1.1 Základní pojmy

Kryptografie je věda zabývající se problematikou utajování dat, autentizací dat a důvěryhodností všech zúčastněných entit. **Kryptografickou ochranu** lze definovat jako speciální ochranu založenou na nemožnosti vyřešení určitého matematického problému v reálném čase. Zabývá se hlavně utajením smyslu informace pomocí matematických operací – šifrování. Věda zabývající se překonáváním kryptografických ochran se nazývá **Kryptoanalýza**. Přístup k utajeným datům má pouze entita, která vlastní znalosti k vyřešení matematického problému s pomocí vstupní tajné informace. Při šifrování se vždy používá algoritmus, který převede tajnou informaci na kryptogram $C = E(Z, K)$, kde K je šifrovací klíč. Předpokládaje, že není možno získat původní zprávu z kryptogramu i se znalostí algoritmu, proto není třeba algoritmus utajovat. Stačí pouze utajit klíč pro distribuci šifrovacího systému. Tato skutečnost platí zejména pro asymetrické kryptosystémy. Více informací v [1], [2], [3].

1.2 Symetrická kryptografie

Symetrická kryptografie (angl. *Symmetric-key encryption*) je třída kryptografických systémů používající jeden klíč – **Tajný klíč** K. Tajný klíč slouží jak pro šifrování tak i pro dešifrování $K_{š} = K_{d}$, a proto musí každá z komunikujících stran klíč znát a musí jej držet v tajnosti. V některých případech se klíče pro šifrování a dešifrování nemusí být stejné, ale stačí když jsou od sebe lehce odvoditelné, viz [3]. Hlavní nevýhodou těchto systémů je, že při odcizení klíče je schopen útočník jednoduše dešifrovat zprávu. Dále je útočník schopen vypočítat dešifrovací klíč z šifrovacího jednoduchým způsobem. Není založen na žádném matematickém problému jak je to v asymetrické kryptografii. Z toho důvodu je třeba při šifrování dodržet všechna doporučení co se týče volby hodnot parametrů potřebných pro šifrování. Je třeba používat velké klíče, které poskytují větší bezpečnost. Tyto klíče jsou veliké řádově 80 bitů a větší. Pořád jsou ale menší než klíče u asymetrické kryptografie. Symetrická kryptografie je díky své rychlosti při výpočtu klíče a rychlosti při šifrování zprávy nejrozšířenější šifrovací metodou a používá se hlavně pro šifrování v běžném použití, tj. pro přenos tajných zpráv na nešifrovaném kanále. Mezi hlavní metody patří systém AES (Advanced

Encryption Standard), která je nástupcem systému DES, který byl určen jako nebezpečný. Je to zcela nový algoritmus a nevychází z DES. Používá délky klíče 128 až 256 bitů a je dnes nejpoužívanějším systémem z třídy symetrické kryptografie. Další symetrické systémy jsou 3DES (využívající modifikaci algoritmu DES), Blowfish, RC5, FEAL, IDEA, SAFER, KHUFU a jiné, viz [1]. Některé systémy pracují jako proudové šifry (zpracují data po jednotlivých bitech) nebo jako blokové šifry (zdrojové data rozdělí na bloky dat a pak je šifruje). Symetrická kryptografie se v praxi používá ve vrstvě SSL – „*Secure Sockets Layer*“ jako nástroj pro šifrování samotné přenášené informace, která má být utajena. Jednou z nevýhod symetrické kryptografie je možnost rozluštění tajného klíče z několika již přenesených zpráv. Více pak v [1], [3], [6].

1.3 Asymetrická kryptografie

Asymetrická kryptografie (angl. *Public Key Cryptosystem*) je skupina založena na matematickém pojmu **Jednocestné funkce** $y = f(x)$. Tato funkce má základní vlastnost v tom, že výpočet lze provést pouze v jednom směru. Výpočet výstupní hodnoty y ze vstupní proměnné x je poměrně jednoduchý, ale výpočet vstupní hodnoty x z výstupní y je velmi náročný, tj. neřešitelný v reálném čase. Asymetrická kryptografie používající pro šifrování princip dvou odlišných klíčů – **Veřejný klíč** K_{pub}, jehož parametry jsou veřejné známé a **Soukromý klíč** K_{priv}, kterého hodnota je známá pouze generující entitě nebo její majitelovi. Veřejný klíč se používá ve dvou případech, buď jak šifrovací klíč při utajování zprávy nebo jako dešifrovací klíč pro ověření digitálního podpisu. Při generování klíčů se používá jednocestné funkce, protože u asymetrické kryptografie musí být zaručena nemožnost výpočtu klíče soukromého z klíče veřejného nebo naopak. Pro znázornění je zde příklad: Uživatel A vygeneruje dva klíče, veřejný klíč pak umístí na veřejném místě kde uživatel B je schopen si jej stáhnout. Uživatel B pak zprávu určenou pro uživatele A zašifruje veřejným klíčem $C = f(K_{pub}, Z)$. Tuto zprávu C pošle nezabezpečeným kanálem, a protože jenom uživatel A zná soukromý klíč je schopen ji dešifrovat $Z = f(K_{priv}, C)$. V tomto případě uživatel A používá systém dvou klíčů jenom pro příchozí šifrovaný přenos. Pokud chce poslat šifrovanou zprávu uživateli B musí použít pro zašifrování veřejný klíč uživatele B. Pokud by svým soukromým klíčem zašifroval zprávu, pak by si ji mohl dešifrovat každý, kdo zná veřejný klíč a pak by zpráva nebyla chráněna. Charakteristickou vlastností asymetrické kryptografie je skutečnost, že šifrovat lze pouze jedním klíčem a druhým dešifrovat. V jiném případě se

nejedná o asymetrickou kryptografii. Dalším hlavním využitím asymetrické kryptografie je ověřování autentičnosti odesílatele a autentičnosti zprávy pomocí digitálního podpisu a digitálních certifikátů.

Asymetrická kryptografie byla vyvinutá z důvodů problémů vyskytujících se při symetrické kryptografii, hlavním problémem je distribuce klíče mezi dvěma komunikujícími entitami. Tuto skutečnost eliminuje asymetrická kryptografie, jelikož při použití dvou klíčů, lze jenom veřejným klíčem šifrovat a soukromým dešifrovat. Eliminuje také distribuci klíčů, protože jedna entita používá pro šifrování pouze jeden klíč. Asymetrická kryptografie byla vyvinutá v polovině 70. let minulého století a vyskytla se v protokolu Diffie-Hellman pro ustanovení klíčů mezi dvěma entitami pro šifrovací přenos na nezabezpečeném kanále. Dnes se používá hlavně pro bezpečný přenos klíče pro symetrickou kryptografii, při digitálním podpisu nebo při systému certifikátu pro ověření autentičnosti druhé strany. Výpočet klíčů i samotné šifrování zprávy je náročné na výpočetní stroj, a proto je rychlost celého procesu vysoká. Proto se asymetrická kryptografie používá hlavně pro výměnu klíčů pro šifrování symetrickou kryptografii. Mezi výhody asymetrické kryptografie patří nemožnost výpočtu klíčů, a proto lze uchovávat dvojicí klíčů i pro mnoho let, viz [1], [3], [5].

1.4 Hashovací funkce

Hashovací funkce je další skupina z oboru kryptografie. Je to speciální jednocestná kompresní funkce, která přiřazuje vstupním datům s libovolnou délkou posloupnost znaků s pevnou délkou (obvykle 128, 256 bitů) tzv. **Hash** *h*. Od hashovací funkce se požadují tyto vlastnosti:

- nemožnost výpočtu původní zprávy z hashe,
- hash vytvořen ze zprávy s libovolnou délkou by měl mít konstantní délku,
- hash z každé zprávy by měl být unikátní tzn. aby nedošlo ke kolizi dvou hashů.

Hashovací funkce se zpravidla používají hlavně pro autentizaci šifrované nebo nešifrované zprávy. Potom se jeví hash jako autentizační kód, který je buď přenesen se

zprávou, nebo integrován v kryptogramu, nebo poslán příjemci jinou formou. Postup ověřování pomocí hashe se pak provádí takto: Příjemce příjme zašifrovanou zprávu a k ní odpovídající hash h. Zprávu dešifruje a pak stejným systémem vypočte ze zprávy hash h'. Pokud se $h = h'$ pak je zpráva autentická a nebyla změněna. V dnešní době se hlavně používají systémy MD5, SHA-1 a SHA-2, přesto, že MD5 byla prolomena a prokázala se jako nebezpečná hashovací funkce. Více v [3], [1], [2].

1.5 Digitální podpis

Digitální podpis (angl. *Digital Signature*) je autentizační systém pro ověřování pravosti elektronických dat. Využívá přitom algoritmy asymetrické kryptografie, ale v opačném případě jako šifrování pomocí asymetrických systémů, viz [1], [2], [3]. Podepisuje se soukromým klíčem a veřejným se ověřuje autentičnost. Při digitálním podpisu se využívá i zmíněné hashovací funkce. Přenášený dokument lze podepsat digitálně několika způsoby. Odesílatel používá pro podpis svůj soukromý klíč, tj. zašifruje jím nějakou zprávu Z (domluvenou s příjemcem) a zprávu pak připojí k dokumentu jako podpis a odešle podepsaný dokument příjemci. Příjemce pomocí veřejného klíče odesílatele dešifruje podpis a pokud zpráva je neporušená dokument je pravý. V jiném případě může odesílatel vytvořit hash h z posílaného dokumentu, ten pak zašifrovat (podepsat) a připojit jej k odesílanému dokumentu. Příjemce pak ze zprávy vytvoří hash h' a dešifruje přiložený podpis. Pokud $h = h'$ je zpráva autentická. V podstatě digitální podpis umožňuje příjemci ověřit autentičnost odesílatele a pravost přijaté zprávy.

1.6 Infrastruktura správy a distribuce klíčů

Při podepisování dokumentů se ověřuje pravost zprávy, ale nastává problém jak ověřit totožnost odesílatele. Velké instituce, firmy, banky požadují vysokou bezpečnost co se týče distribuce klíčů a jiných informací pro šifrovanou komunikaci. Tuto skutečnost řeší **Infrastruktura správy a distribuce klíčů** tzv. **PKI** (angl. *Public Key Infrastrukture*), která se zabývá autentičností entit při výměně komunikací pomocí tzv. **Certifikátů**. Certifikát je speciální dokument elektronicky podepsaný určitou důvěryhodnou stranou, určitou institucí nazývanou **Certifikační autorita**, dále jenom **CA,** který obsahuje informace pravost entity a jeho veřejného klíče. **CA** vlastní svůj veřejný klíč K_{pubCA} pro ověření certifikátu a obsahuje všechny certifikáty zaregistrovaných uživatelů, kteří chtějí možnost ověřování jejich totožnosti. Celá infrastruktura se skládá z **CA**, **RA** což

je **Registrační autorita** zabývající se registrací žadatelů o certifikaci veřejného klíče, adresářové služby určující hierarchii celého systému PKI a **CLR** (angl. *Certificate Revocations List*) neboli seznamem již nedůvěryhodných certifikátů např. z důvodu odtajnění uživatelova veřejného klíče. Ověřování totožnosti komunikujících stran se provádí v pár krocích. Příjemce si nejdřív vyžádá certifikát odesílatele od **CA**. Pomocí veřejného klíče K_{pubCA} si dešifruje podepsaný certifikát a ověří zda odesílatel je ten za koho se vydává a hlavně ověří jeho veřejný klíč. Registrace uživatelů se provádí následujícím způsobem, žadatel nejdřív vygeneruje soukromý K_{privA} a veřejný klíč K_{pubA} a se svými údaji požádá registrační autoritu o vydání certifikátu. **RA** zpracuje jeho údaje **ID,** další data **DD** a veřejný klíč K_{pubA} do jednoho rámce $F_A = (ID, DD, K_{\text{pubA}})$ a bezpečně jej doručí certifikační autoritě. Ta danou informaci F_A podepíše svým tajným klíče K_{privCA} $\rightarrow P_{\text{A}} = E(F_{\text{A}}, K_{\text{privCA}})$ a předá vygenerovaný certifikát $CRT_{\text{A}}(F_{\text{A}}, P_{\text{A}})$ žadateli. PKI využívá systém symetrické kryptografie a asymetrické kryptografie. Obě třídy šifrování mají své přednosti, u symetrického systému je to je rychlost, možnost ověřování offline. U asymetrického systému veřejný klíč poskytuje lokalitu důvěry a znemožňuje vydávání kohokoliv za někoho jiného, viz [5], [3]. Nasazení PKI je v dnešní době běžné, lze ji najít ve většině moderních bezpečnostních systémech.

2 MATEMATICKÉ PROBLÉMY V ASYMETRICKÝCH KRYPTOSYSTÉMECH

Většina asymetrických systému je založena na různých matematických problémech. Bezpečnost těchto systémů spočívá v nerozluštitelnosti v reálném čase při použití problémů popsaných v této kapitole. Moderní kryptosystémy vždy využívají jednu z těchto technik. Všechny níže popsané problémy jsou ze skupiny NP, to znamená, že výpočet by neměl být proveditelný v reálném čase, pokud tomu tak není daný kryptosystém nelze považovat za bezpečný. Pokud ale při zvolení správné délky klíče a vhodném zvolení parametrů, pravděpodobnost prolomení šifry konverguje k nule a tudíž lze ji považovat za bezpečnou. Více informací lze pak najít v [1], [2], [10].

2.1 Problém faktorizace čísla

Problém faktorizace čísel je problém rozložení libovolného velkého celého čísla na součin prvočíselných mocnin

$$n = p_1^{e^1} p_2^{e^2} p_3^{e^3} p_4^{e^4} \dots p_k^{e^k} \tag{2.1}$$

Prvočísla mají důležitý význam jak v matematice tak v kryptografii. Díky své vlastnosti dělitelnosti pouze jedničkou nebo sebe samou tvoří bezpečnostní základ mnoha asymetrických kryptosystémů. Všechny popsané asymetrické metody v kapitole 3 používají ve svých algoritmech prvočísla z důvodu jejich bezpečnosti. V podstatě lze říct, že každé neprvočíslo je složeno z dvou nebo více prvočísel a postupným rozkladem složitého čísla lze určit všechny prvočísla. Více pak o prvočíslech v [8]. Asymetrické systémy založeny na problému faktorizace jsou např. RSA a další. Problém faktorizace čísla patří do skupiny NP a co – NP, tzn. spadá do skupiny problémů neřešitelných v reálném čase. Problematika zjištění zda celé číslo je složené nebo prvočíslo, je v podstatě méně náročná operace než rozklad daného čísla na prvočísla. V kryptografii existuje několik metod pro výpočet faktorizace čísel, a každá z nich je efektivní pro určitou délku složitého čísla n a jeho vlastností. Proto je třeba zvážit, jak vypočítat složité číslo, jinak by algoritmus mohl pracovat do nekonečna nebo se zacyklit. Často jsou vytvořeny ochrany proti zacyklení, které ale závisí na vlastnostech faktorizovaného čísla. Entita při rozkladu složeného čísla by měla nejdřív použít metody pro nalezení

malých prvočísel a pak aplikovat složité algoritmy na počítání všech obsažených prvočísel. Je jasné, že složité metody budou požadovat větší výpočetní výkon nebo více času na rozklad. Metody pro faktorizaci čísel a jejich principy jsou popsaný níže, více pak v kapitole 3 v [1] nebo v [25], [7].

2.1.1 Zkušební dělení

Jakmile je určeno, že faktorizované číslo n je číslo složené je doporučeno použít nejdřív algoritmus Zkušebního dělení. Tato metoda je efektivní pro celá čísla, pokud se n skládá z malých prvočísel p. Při hledání větších prvočísel pak doba operací vyžaduje velké množství času. Zkušební dělení je blíže popsáno v [1].

2.1.2 Pollardův ró algoritmus

Pollardův ró algoritmus je speciální metoda pro hledání malých prvočísel ze složeného čísla n. Pokud určíme, že S je konečná skupina všech prvočísel, ze kterých se skládá číslo n, a funkce $x_{i+1} = f(x_i)$ pro $i > 0$ kde x_0 je náhodně zvolené číslo ze skupiny S, pak se muže algoritmus zacyklit. Proto je vhodné v případě kolize $x_i = x_j$ kontrolovat všechny duplikátní hodnoty a tím zabránit spadnutí systému. Ochranu proti zacyklení obstarává *„Floyd's cycle-finding algorithm“*. Pollardův ró algoritmus je popsaný níže: Nejdříve se zvolí a,b a vypočte se $a \leftarrow a^2 + 1 \bmod n$, $b \leftarrow b^2 + 1 \bmod n$. Dále se vypočte největší společný dělitel $d = \gcd(a - b, n)$. Pokud společný dělitel je $1 < d < n$ pak se algoritmus ukončí a vrátí d, pokud $d = n$ zakončí se algoritmus neúspěchem. (více v kapitole 3.2.2 v [1]).

2.1.3 Pollardův $p - 1$ algoritmus

Pollardův $p - 1$ algoritmus je speciální faktorizační algoritmus pro nalezení součin jakýchkoliv velkých prvočísel ze složitého čísla n. Hlavní myšlenka je, že zvolí se číslo B, které souvisí s číslem n tak, že všechny prvočísla čísla n jsou menší nebo rovno B. Pak je třeba náhodně zvolit celé číslo a dle (2.2)

$$2 \leq a \leq n-1 \tag{2.2}$$

a vypočítat $d = \gcd(a,n)$. Pokud je $d \geq 2$ pak vrací hodnotu d jako výsledek, pokud ne, počítá se pro každé prvočíslo $q \leq B$ nové a dle (2.4)

$$l = \lfloor \frac{\ln n}{\ln q} \rfloor \tag{2.3}$$

$$a \leftarrow a^{q^l} \bmod n \tag{2.4}$$

Pak se vypočítá $d = \gcd(a - 1, n)$. Pokud platí, že $d = 1$ nebo $d = n$, pak se algoritmus ukončí s výsledkem neúspěchu, pokud ne algoritmus vrátí číslo d. Náročnost výpočtu Pollardova $p - 1$ algoritmu pro nalezení prvočísla p ze složeného čísla je $O(B \ln n / \ln B)$. Více informací o tomto algoritmu v [1].

2.1.4 Faktorizace pomocí eliptických křivek

Systém eliptických křivek pro výpočet faktorizace složitých čísel je označován jako algoritmus pro speciální použití, je vlastně zobecněním Pollardova $p - 1$ algoritmu. Algoritmus eliptických křivek má tendenci nejdřív vyhledat malá prvočísla p, čímž pracuje rychleji, protože čas potřebný pro výpočet závisí na velikosti prvočísel p, ze kterých je složeno n. Nicméně, algoritmus není účinný pro výpočet velkých prvočísel p, a proto se používá jiné metody, které jsou rychlejší a mnohem efektivnější než systém eliptických křivek pro faktorizaci. Více v [7].

2.1.5 Faktorizace pomocí náhodných mocnin

Metoda pomocí náhodných mocnin (angl. *Random square factoring*) je faktorizační algoritmus, pro který platí závislosti

$$x^2 \equiv y^2 \pmod{n} \qquad \text{tak aby} \qquad x \equiv \pm y \pmod{n} \tag{2.5}$$

Tento algoritmus je nad rámec této práce, je popsán v kapitole 3.2.6 v [1]. Doba potřebná k výpočtu je stejná jako u eliptických křivek, ale náročnost na výpočetní stroj je podstatně menší, protože nepracuje se složitými algoritmy jak je to u eliptických křivek.

2.1.6 Faktorizace pomocí „síta číselného tělesa“

Algoritmus „síta číselného tělesa“ (angl. *Number field sieve factoring*) také patří do skupiny „náhodných mocnin“. V roce 1996 po určení rychlosti výpočtu algoritmu pomocí experimentů, je to dnes nejlepší algoritmus pro výpočet prvočísel s různými délkami ze složeného čísla n. Více v [1].

2.2 Problém výpočtu diskrétního logaritmu

Problém výpočtu diskrétního logaritmu DL je další z problému, se kterými počítají různé systémy asymetrické kryptografie. Spočívá na principu rozložení složeného čísla na mocninu jiného čísla $a = g^x$. Nalezená mocnina x, kterou lze zapsat jako $x = \log_g a$, se pak jmenuje diskrétní logaritmus. Na tomto problému pracují systémy ElGamal, Diffie-Helmanův protokol a další. Obecně lze napsat, že pro g z G je třeba najít číslo x takové, že $g^x \equiv a \bmod n$ kde n je řád skupiny G a a je známý parametr. Při počítání diskrétního logaritmu je třeba použít vhodnou metodu, protože každá metoda počítá hodnotu x jinak. Při náročném problému může nevhodná metoda celý proces výpočtu zacyklit nebo vypočítat nesprávný výsledek. Problém diskrétního logaritmu je považován za neřešitelný problém v reálném čase pro velká celá čísla. Kryptosystémy založeny na diskrétním logaritmu jsou považovány za velmi bezpečné při správné délce klíče. Další informace jsou pak v [3] a [10].

2.2.1 Algoritmus „Baby-step Giant-step“

Algoritmus pracuje na principu násobení pomocí zkoušek. Hodnoty pak uchovává v tabulce pro porovnání, čímž je jasné, že potřebuje velkou paměť. Při použití malého objemu paměti se zvyšuje čas výpočtu, a proto je třeba najít kompromis mezi pamětí a časem výpočtu. Při složitějších metodách je lépe použít jiný algoritmus pro výpočet diskrétního logaritmu, viz [1].

2.2.2 Polladrův „ró“ algoritmus pro diskrétní logaritmus

Pollardův „ró“ algoritmus pracuje se stejnou náročností jak předchozí algoritmus, ale nepotřebuje ke svému výpočtu velký objem paměti. Proto je tento algoritmus více používán. Nepracuje na porovnávání všech hodnot pouze na porovnávání hodnot z podskupin. Tyto podskupiny rozdělují skupinu G na 3 části kde v každé podskupině se provede daný algoritmus, ale s jinými parametry, proto je potřeba 3–krát méně paměti oproti předchozímu algoritmu.

2.2.3 Pohlig–Hellmanův algoritmus

Pohlig–Hellmanův algoritmus je další ze skupiny metod pro výpočet diskrétního logaritmu. Jeho hlavní vlastností je faktorizace řádu n skupiny G. Výsledkem bude

diskrétní algoritmus vypočtený dle (2.6).

$$x \equiv x_i \pmod{p_i^{e_i}} \tag{2.6}$$

pro vypočtený pomocí $1 \leq i \leq r$ Gaussova algoritmu. Pohlig–Hellmanův algoritmus je efektivní pouze pro malá prvočísla p_i.

2.2.4 „Index–calculus“ algoritmus

„Index–calculus“ algoritmus je nejefektivnější metodou pro výpočet diskrétního logaritmu v dnešní době. Princip spočívá na určení velmi malé podmnožiny *S* ze skupiny *G*, tak aby všechny významné položky ze skupiny *G* byly i položkami podmnožiny *S*, tzn. všechny významné prvky jsou ve podmnožině *S*. Díky tomu je výpočetní čas velmi nízký. Bohužel zde nastává problém, protože není žádný efektivní způsob na určení podmnožiny *S*. Více pak v [1].

2.3 Problém „lattice“

Problém „lattice“ neboli problém „mřížky“ je jedna z moderních metod pro asymetrickou kryptografii. Tento problém spadá do skupiny post-kvantových algoritmů a je velmi odolný proti útokům hrubou silou a jiné. Teoreticky je odolný vůči kvantovým počítačům, ale pouze proto, že dnes neexistuje žádný účinný kvantový algoritmus, který by uměl počítat problémy typu mřížky. Algoritmy, které tento problém řeší jsou normální algoritmy pracující na konvenčních výpočetních strojích a rychlost kvantových algoritmů se rovná rychlosti normálních. Jediný kvantový algoritmus schopný řešit tento problém je Shorův[1] algoritmus, ale nikdy nebyla prokázaná jeho úspěšnost nebo účinnost. Nové testy a experimenty prokazují, že tento algoritmus není vůbec schopen řešit problémy typu mřížky. Další kvantové algoritmy nejsou dosud známy, a proto se považuje lattice za jeden z neřešitelných problémů v dnešním světě. Více v [15], [16], [17].

„Lattice“ je podstatě problém založen na speciální matematická mřížce se skupinou bodů v *n*-rozměrovém prostoru. Je definovaná jako skupina vektorů $(\boldsymbol{b}_1,\ldots,\boldsymbol{b}_n)$ dle rovnice (2.7)

1 Peter Shor je americký matematik. Svůj algoritmus vymyslel v roce 1994 pro řešení problému faktorizace.

$$L(b_1, \ldots, b_n) = \{\sum_{i=1}^{n} x_i b_i : x_i \in \mathbb{Z}\} \tag{2.7}$$

kde (b_1,...,b_n) jsou tzv. báze mřížky. V kryptosystémech se nejčastěji používá q-násobná mřížka kde q je prvočíslo. Lze i říci, že každá mřížka ze skupiny celých čísel je q-násobná mřížka pro nějaké q. Existuje několik nekvantových výpočetních metod na lattice.

2.3.1 Shortest Vector Problem

Nejčastější je metoda „*Shortest Vector Problem*“ neboli **problém nejkratšího vektoru**, která hledá s pomocí znalosti bází, jak je již z názvu zřejmé, nejkratší nenulový vektor v mřížce. Nejznámějším algoritmem pro výpočet tohoto problému je algoritmus *LLL*[2], který počítá s časem $2^{n(\log \log n)\hat{}2 \,/\, \log n}$, další z algoritmů je pro určení nejkratší vektoru je Kannanův algoritmus, který počítá s časem $2^{O(n \log n)}$. Na principu tohoto problému závisí soukromý klíč systému NTRUEncrypt.

2.3.2 Closest Vector Problem

Další metodou je „*Closest Vector Problem*“ neboli **problém nejbližšího vektoru**, který hledá v mřížce L se znalostí bází a vektoru **t** (nemusí být součástí mřížky) vektor **v**, který leží nejblíže vektoru **t**.

Problém lattice a algoritmy pomocí kterých lze počítat lattice jsou velmi složité a nad rámec této práce. Více informací v [15], [16], [27].

2.4 Problém „Batohu“

Problém „Batohu“ (angl. *Knapsack Problem*) je další z problémů, na kterých jsou postaveny některé kryptosystémy. Tento problém vychází z kombinační matematiky a pomocí něj se řeší i jiné problémy v běžném životě (např. při logistice, při investičním podnikání). Tvoří základy pro kombinatoriku, komplexní teorii, kryptografii a aplikovanou matematiku. Jde v podstatě o otázku: Jak nejlépe vložit věci do batohu a nepřekročit jeho kapacitu?

Problém batohu se opírá o kapacitu c nějakého tělesa tzv. batohu. Položky, které chceme

2 Název pochází od vynálezců algoritmu Lenstra, Lenstra, Lovász. Někdy označován jako L^3.

vložit do batohu mají určitou hodnotu *v* (u některých případů se nepočítá V jako hodnotu nýbrž jako objem, v kryptografii se problém batohu vztahuje pouze na čísla a proto je třeba znát pouze jejich hodnoty). Také lze říci, že některé položky se vůbec nevejdou do batohu. Lze pak napsat, že:

$$\sum_{j=1}^{n} x_j v_j \leq c \tag{2.8}$$

kde x_j nabývá hodnoty 1 pokud je položka vložena do batohu, 0 pokud není, a n je počet položek. Dále je třeba upozornit, že kapacita batohu je konstantní a nemůže se měnit. Metodou pro řešení problému batohu je **dynamické programování**, kdy si program pamatuje vypočtené metody a porovnává je, aby nedošlo k duplicitním výpočtům. Nevýhodou je požadavek na velkou paměť pro uložené hodnoty. Algoritmy pro dynamické programování jsou např. Horowitzův-Sahni algoritmus, Tothův algoritmus a jiné. Další metodou je metoda **Větve a hranice**, která prohledává batoh do hloubky použitím stromového stavového prostoru. Jiné metody jsou např. Redukční metoda, Aproximační metoda a další. Více o těchto metodách v [19], [22]. Jakožto u předchozích problémů, i problém batohu se dělí na podproblémy.

2.4.1 „0–1 Knapsack“ problém

Binární problém batohu (angl. *0–1 Knapsack Problem*) je nejdůležitější problém batohu, protože se vyskytuje v hodně jiných situacích jiných od kryptografie. Je to jediná metoda, která je schopna řešit problémy s velkým počtem proměnných a proto je výskyt nejčastější problém batohu vůbec.

2.4.2 „Subset–Sum“ problém

Bezpečnost prvních asymetrických kryptosystémů byla založena právě na problému „Součtu podmnožin“ neboli angl. *„Subset–Sum problem“*. Princip tohoto problému, spočívá v daném množině kladných celých čísel $\{a_1,a_2,a_3, \ldots a_n\}$, který představuje batoh systému. V tomto batohu nesmí být žádné duplicitní hodnoty. Otázka je zda je v batohu taková podmnožina, které suma je rovna číslu *s*. Existují dva způsoby jak tento problém řešit. První metoda je počítání pomocí větvení, pak náročnost výpočtu je 2^N a druhá, která dělí batoh na dvě množiny a počítá všechny sumy obou skupin. Náročnost druhé metody je $2^{N/2}$, bohužel potřebuje dvakrát více paměti než první metoda. Více v [22].

3 ANALÝZA ASYMETRICKÝCH KRYPTOSYSTÉMŮ

3.1 Systém RSA

Systém RSA je nejrozšířenější asymetrický kryptosystém v dnešní době. Název systému pochází od návrhářů systému Rivest, Shamir, Adelman. Systém RSA je používán pro šifrování zpráv a poskytuje také autentizaci dokumentů pomocí digitálního podpisu. RSA kryptosystém je postaven na problému faktorizace čísla na prvočísla. Více informací v [4], [3], [1].

Algoritmus pro generování soukromého a veřejného klíče provádí tyto kroky: entita zvolí dvě velmi velké prvočísla p a q, které neleží vedle sebe a jsou té samé velikosti. Nejdřív se vypočítá $n = pq$, a $r = (p - 1)(q - 1)$. Je zvolen vlastní veřejný klíč K_{pub} tak, aby $\gcd(K_{pub},r) = 1$ (pomocí Euklidova algoritmu), tzn. klíč musí být nesoudělný s r. Soukromý klíč pak algoritmus vypočte jako $K_{priv} = K_{pub}^{-1} \bmod r$. Pak veřejný klíč entity je K_{pub} a číslo n, které je pak potřeba k dešifrování a soukromý klíč je K_{priv}.

Postup šifrování je jednoduchý a rychlý. Odesílatel si zjistí veřejný klíč příjemce a zprávu Z nejdřív převede na číselný formát (pomocí ASCII) a pak rozdělí na bloky o stejné délce. Délka každého bloku z_i musí být menší než n, tzn. $z_i < n$. Pak se každý i-tý blok zašifruje

$$c_i = z_i^{K_{pub}} \bmod n \tag{3.9}$$

a bloky c_i se pak spojí do zašifrované zprávy C. Příjemce jej pak dešifruje s pomocí svého soukromého klíče na původní zprávu tak:

$$z_i = c_i^{K_{priv}} \bmod n \tag{3.10}$$

Bezpečnost RSA závisí na neschopnosti vypočítat faktorizaci čísla n, které je součinem čísel p a q. Právě pomocí těchto čísel se počítá proměnná r, která definuje vlastnost soukromého klíče. Při nesprávné volbě p a q není systém bezpečný a útočník může komprimovat celý systém. Šifrovací algoritmus pak pracuje na problému výpočtu diskrétního logaritmu. Vhodná délka modulu n je dnes až 1024, tzn. číslo velké 2^{1024} s

tím, že velikost soukromého klíče by měla být stejné délky. Více o doporučených délkách parametrů v kapitole 4.4.

3.2 Systém ElGamal

Systém ElGamal je asymetrický kryptosystém založený na problému výpočtu diskrétního logaritmu, viz [1], [21]. Vychází z principu Diffie-Hellmanova algoritmu pro generování dvojici klíčů. Jeho hlavní nevýhodou je dvojnásobná délka šifrované zprávy oproti nezašifrované zprávy. Jeho nasazení je hlavně v PGP – „*Pretty Good Privacy*“ systémech a v GPG – „*GNU Privacy Guard*“ systémech.

Výpočet dvojice klíčů se provádí následujícím způsobem nad multiplikativní cyklickou skupinou *G*. Entita nejdřív zvolí multiplikativní skupinu *G* s řádem *n* s generátorem α. Pak zvolí náhodné číslo *a* aby:

$$1 \leq a \leq n-1 \tag{3.11}$$

a vypočte α^a. Entita pak dostane veřejný klíč $K_{\text{pub}}(\alpha,\alpha^a,G)$ a soukromý klíč $K_{\text{priv}}(a)$.

Postup šifrování se je pak: odesílatel pak pomocí multiplikativní skupiny *G* zvolí zprávu *m* jako element této skupiny a zvolí číslo *k* ze stejného intervalu jako číslo *a* a vypočte:

$$\gamma = \alpha^k \qquad \delta = m \cdot (\alpha^a)^k \tag{3.12}$$

pak zašifrovanou zprávu $c = (\gamma,\delta)$ pošle příjemci. Příjemce vypočte pomocí soukromého klíče a γ^a, γ^{-a} a pak vypočte zprávu *m* :

$$m = (\gamma^{-a}) \cdot \delta \tag{3.13}$$

3.3 Systém McEliece

Systém McEliece byl vyvinut Robertem McEliecem v roce 1978. Tento systém nezapadá do žádné skupiny problémů popsaných v kapitole 2. Je imunní vůči Shorovu algoritmu a tím pádem je nejlepším kandidátem na post-kvantový kryptosystém, viz [1]. McEliece používá tzv. Goppa kódy, pomocí kterých tento systém odolává útokům. Při šifrování používá také náhodné generování čísel. Hlavní výhodou oproti RSA je rychlost výpočtu, bohužel na úkor náročnosti, protože soukromý a veřejný klíč je

reprezentován jako matice s velmi velkým rozměrem. Tento systém je pouze bezpečný pro určité parametry, pro všechny jiné lze jej považovat za neefektivní a nepoužitelný. Tyto parametry je třeba dodržovat a každá bezpečná kombinace je používaná pro jiné účely.

Generování klíčů se vypočte následovně: k, n a t jsou pevné parametry dané systémem. Entita vypočte $k \times n$ generační matici $\boldsymbol{G}$, zvolí náhodně $k \times k$ nesingulární binární matici S a $n \times n$ permutační matici $\boldsymbol{P}$. Vypočte matici $\hat{\boldsymbol{G}} = \boldsymbol{SGP}$. Veřejný klíč je pak $K_{\text{pub}}(\hat{\boldsymbol{G}},t)$ a soukromý klíč je $K_{\text{priv}}(\boldsymbol{S},\boldsymbol{G},\boldsymbol{P})$.

Odesílatel nejdřív převede zprávu na binární číslo m délky k, zvolí binární chybový vektor z délky n. Vypočte šifrovanou zprávu jako binární vektor $\mathbf{c} = m\hat{\boldsymbol{G}} + \mathbf{z}$. Příjemce pak zprávu dešifruje jako $\hat{c} = c\boldsymbol{P}^{-1}$ kde $\boldsymbol{P}^{-1}$ je inverzní matice k $\boldsymbol{P}$. Pak se pomocí $\hat{c}$ vypočte $\hat{w}$, a pak se vypočte m jako $m = \hat{w}\boldsymbol{S}^{-1}$.

3.4 Systém Merkle-Hellman

Merkle-Hellmanův systém je založen na problému „batohu“, konkrétně na problému „Subnet – Sum“, viz [1]. Je jedním z jednodušších kryptosystémů, bohužel je považován za nebezpečný z důvodu prolomení ochrany a prolomení celého systému. Nelze jej použít pro autentizaci dokumentů, protože veřejným klíčem lze pouze šifrovat a soukromým pouze dešifrovat.

Princip generování klíčů je ve zvolení náhodného vektoru $\mathbf{w} = (w_1,w_2,...w_N)$, kde N je bitová délka zprávy. Vektor $\mathbf{w}$ by měl být zvolen tak aby aktuální prvek byl větší než suma všech předchozích prvků. Dále se zvolí parametry q tak aby $q > w_N$ a r tak aby $\gcd(r,q) = 1$. Vypočte se vektor $\boldsymbol{\beta} = (\beta_1,\beta_2,...,\beta_N)$ kde $\beta_i = w_i r \pmod q$. Pak veřejný klíč je reprezentován jako $K_{\text{pub}}(\beta)$ a soukromý klíč $K_{\text{priv}}(\mathbf{w},q,r)$.

Šifrování se provádí s vektorem $\boldsymbol{\alpha} = (\alpha_1,\alpha_2,...,\alpha_N)$, který představuje bity zprávy m. Pro každý i-tý vzorek se počítá rovnice

$$C = \sum_{i=1}^{N} \alpha_i \cdot \beta_i \tag{3.14}$$

A výsledný kryptogram C pošle odesílatel příjemci.

Dešifrování je jednoduché. Příjemce vypočítá $s = r^{-1} \pmod{q}$ a provede operaci

$$c' = c * s \pmod{q} \tag{3.15}$$

Pak převede c' na součet prvků v batohu pomocí „Greedyho“ algoritmu. Příjemce zvolí největší číslo w_k z vektoru **w** a porovná jej s c'. Pokud je $w_k > c'$ pak je $\alpha_k = 0$, jinak je $\alpha_k = 1$. Tuto činnost opakuje až do vytvoření vektoru **α,** více ve zdroji [2].

3.5 Systém Rabin

Systém Rabin, který také jako RSA, je založen na problému faktorizace čísla. Je to první systém, u kterého bylo prokázáno, že výpočet původní zprávy ze zašifrované je stejně náročný jako výpočet faktorizace. Šifrování Rabin je velmi rychlá operace vůči RSA, rychlost dešifrování je poměrně stejná jako u RSA, viz [1], [26].

Výpočet klíčů je následující: entita nejdřív zvolí vysoké prvočísla p a q stejné velikosti a vypočte $n = pq$. Pak veřejný klíč je $K_{pub}(n)$ a soukromý je dvojice $K_{priv}(p,q)$.

Šifrovaní se provádí tak, že odesílatel převede zprávu na číslo m z rozsahu $\{0,1,...,n-1\}$ a vypočte:

$$c = m^2 \bmod n \tag{3.16}$$

A pošle jej příjemci. Příjemce vypočte druhé mocniny c mod n a dostane výsledek zprávy m_1, m_2, m_3 a m_4. Příjemce se pak rozhodne, která z těchto zpráv je pravá zpráva m. Dešifrování vytváří další nepravé výsledky a správný výsledek je vždy třeba odhadnout. Právě tato vlastnost zabránila systému Rabin rozšíření v praktickém využití ve světě.

3.6 Systém Probabilistic

Systém Probabilistic využívá při šifrování náhodná čísla a tím prokazatelně dosahuje velmi vysoké úrovně zabezpečení. Dokáže stejnou zprávu zašifrovat pokaždé jinak, tzn. že výsledkem je vždy jiný kryptogram. Této vlastnosti naznačuje vysokou bezpečnost, protože útočník nikdy nedostane dva stejné kryptogramy a není schopen zjistit princip systému. Nevýhodou je, že útočník je schopen při použití speciálních pravděpodobnostních tabulek vyřešit tento problém v reálném čase, viz [1]. Z toho

důvodu není nasazení systému na principu probabilistic tak veliké. Mezi dva nejznámější systémy patří **Goldwasser–Micaliův** kryptografický systém a **Blum–Goldwasserův** kryptografický systém.

Goldwasser–Micaliův systém generuje tak, že entita zvolí dva velmi velké prvočísla p a q a vypočítá jejich součin $n = pq$. Pak zvolí celé číslo y tak aby $y < n$ a $y \bmod n = 1$. Pak veřejný klíč je $K_{pub}(n,y)$ a soukromý klíč je $K_{priv}(p,q)$. Vypočítání soukromého klíče z veřejného je problém faktorizace, který není řešitelný v reálném čase.

Odesílatel převede zprávu na jednotlivé bity m_i kde $i = \{1, 2,...,t\}$ a t je délka zprávy, a pro každý bit zvolí celé číslo x_i. Pokud je $m_i = 1$ vypočítá $c_i \leftarrow yx^2 \bmod n$, pokud $m_i \neq 1$ pak $c_i \leftarrow x^2 \bmod n$. Pak všechny bity c_i spojí v kryptogram C a pošle jej příjemci. Příjemce pak rozdělí kryptogram C na na bity m_i kde $i = \{1,2,...,t\}$ a vypočítá

$$e_i = \left(\frac{c_i}{p} \right) \tag{3.17}$$

pomocí Jacobiho algoritmu. Pokud je $e_i = 1$ nastaví se $m_i \leftarrow 0$, jinak je $m_i \leftarrow 1$. Příjemce pak sestaví z bitů m_i celou zprávu m.

Hlavní nevýhodou **Goldwasser–Micaliova** algoritmu je zvětšení kryptogramu oproti původní zprávě. Bohužel tuto vlastnost nelze potlačit, jelikož je dána právě systémem probabilistic. **Blum–Goldwasserův** algoritmus pracuje na stejném principu, pouze šifrovací mechanismus je modifikovaný. Jeho nevýhodou je možnost zjištění klíče z více kryptogramů, a proto se moc nepoužívá v praxi. Při všech algoritmech systému probabilistic by měl mít modul n délku nejméně 1024 bitů, při menší délce ho lze považovat za nepřijatelný z hlediska bezpečnosti. [1]

3.7 Systémy s eliptickými křivkami

Systémy s eliptickými křivkami (angl. *Elliptic curves*) jsou dnes považovány za moderní kryptografické systémy. V praxi se více a více rozšiřuje, protože přináší mnohem lepší výsledky než jiné metody. Vědomosti o eliptických křivkách jsou známy již delší dobu, bohužel jejich nasazení je realizováno pomalu. Malé nasazení je zapříčiněno tím, že jiné asymetrické systémy jako je např. RSA, ElGamal, jsou již zaběhlé a ověřené. Princip eliptických křivek se používá při šifrování a digitálním

podpisu. Nejsou příliš náročné na výpočetní výkon, a proto lze schémata na bázi eliptických křivek používat i na strojích s nižším výkonem. Tento systém poskytuje stejnou bezpečnost jako systém RSA, ale s kratší délkou klíče a proto vede k nižším nárokům na paměť. Pro tu samou bezpečnost potřebuje systém RSA klíč s délkou 1024 bitů, systém eliptických křivek pouze 160 bitů. Rychlost výpočtu je mnohem vyšší než u jiných systémů hlavně při podepisování dokumentů, bohužel nižší při samotném šifrování a ověřování. Více pak v [14]. Systém eliptických křivek je zranitelný vůči kvantovým algoritmům, konkrétně Shorovu algoritmu pro počítání diskrétního logaritmu.

Eliptická křivka je matematický objekt, který je v algebře popsán rovnicí $y^2 = x^3 + ax + b$, kde a,b jsou reálné konstanty a označuje se E. Pomocí této rovnice lze dělat běžné operace jako sčítaní a odečítání. Pokud se sečtením dvou bodů P a Q na rovinné křivce vytvoří přímka, která se již neprotne s křivkou, tj. protne se v nekonečnu. Lze napsat, že má přímka s křivkou společný bod O, neboli „nulový bod“. Pro nulový bod je třeba definovat operace, tzn. že $P + O = P$, $O + O = O$ a také $-O = O$. Soubor všech bodů ležících na křivce E lze nazvat jako grupa E. Body x,y,a,b jsou reálná čísla, pro kryptografii jsou tyto body ze souboru celých čísel, nazývaných jako „těleso“ F. Nejčastěji se v kryptografii používá dvě tělesa F(P) – těleso pouze prvočísel a operace probíhají mod p a těleso F(2^m) – což je těleso binární, operace probíhají mod 2^m, kde m je přirozené číslo. Lze napsat, že eliptická křivka E nad tělesem F(P) s množinou $P(x,y)$ kde x,y jsou prvky z tělesa F(P) má rovnici $y^2 \equiv x^3 + ax + b \pmod{p}$, kde a,b jsou také prvky z tělesa F(P).

Pro výpočet soukromého a veřejného klíče entita použije bod P, který lze určit z eliptické křivky E nad konečným tělesem. Zvolí se K_{priv} a entita vypočítá $Q = K_{priv}P$ a pak je veřejný klíč $K_{pub}(Q, P)$. Bezpečnost soukromého klíče spočívá ve vyřešení problému diskrétního logaritmu (2.2). Lze počítat bod P takto:

$$P + P = 2P + P = 3P + P = 4P + P = 5P \ldots = rP = O \qquad (3.18)$$

kde číslo r lze označit jako řád bodu P. Úloha určení koeficientu r, tzn. v tomto případě generace soukromého klíče, spadá do problému diskrétního logaritmu.

Existují různé metody šifrování pomocí eliptických křivek. Při šifrování nepostačí

jenom znalost veřejného klíče a hodnotu bodu *P*, je třeba znát i rovnici křivky *E,* na které systém pracuje. Jedna z nich je modifikovaná metoda **ElGamal pro eliptické křivky**. Metoda šifruje takovým způsobem, že odesílatel interpretuje zprávu *m* jako bod na křivce P_m a obě strany musí znát bod *G* na křivce *E*. Nejdříve zvolí parametr *k* a pomocí něj zašifruje zprávu a odešle kryptogram $C = (kG, P_m + kK_{pub})$ kde $K_{pub} = K_{priv}G$. Příjemce vynásobí první část kryptogramu svým soukromým klíčem:

$$kG \cdot K_{priv} = K_{priv} kG = kK_{pub} \tag{3.19}$$

a odejme ji od kryptogramu *C* a dostane bod P_m, který reprezentuje zprávu *m*. U většiny metod se pouze systém modifikuje pomocí eliptické křivky, která představuje skupinu bodů, tedy konečné těleso a tím nahrazuje aktuální množinu, nad kterou metoda pracuje. Většinou se systém eliptických křivek používá pro digitální podpis nebo pro ustanovení symetrického klíče pro šifrování. Eliptické křivky jsou používány v systémech OpenSSL, NSS, GnuPG a jiné. Všechny platformy již tento kryptosystém hojně implementují do svých technologií. Více informací v [13], [14] a [24].

3.8 Systém NTRUEncrypt

Systém NTRU je mladý kryptosystém postavený na problému „lattice“. NTRU je kruhový kryptosystém a je to další ze systému, který používá náhodná čísla při šifrování. Šifrování a dešifrování je velmi rychlé s nízkým požadavkem na paměť. Lze jej využívat pro šifrovací kryptosystémy pracující s minimálním hardwarem např. čipové karty, klíče pro odemykání aut, domů, atd. Jeho generace klíčů je jednoduchá také velmi rychlá a přitom velmi efektivní, klíče jsou distribuovaná jako matice. Tento systém byl vytvořen v roce 1996 a je to jedna z nejmladších používaných asymetrických metod pro šifrování. Ze začátku měl algoritmus problémy s dešifrováním, ale tyto problémy byly vyřešeny v roce 2005 a dnes neexistuje algoritmus na prolomení jeho bezpečnosti. Používá se nejenom pro asymetrickou kryptografii, ale i pro digitální podpis. Metoda pro digitální podpis se jmenuje NTRUSign. Více pak v [15], [17] a [18].

Generování klíčů závisí na třech parametrech *N*, *p* a *q*. Parametry *p* a *q* se volí tak, že pouze jeden musí být prvočíslo tedy musí platit $\gcd(p,q) = 1$, a *q* je velmi velké číslo větší než *p*. Entita zvolí dva polynomy **f**,**g** ze skupiny L_g stupně $N - 1$ s koeficienty

$\{-1,0,1\}$. Pak entita vypočítá pomocí Euklidova algoritmu rovnici (3.20):

$$F_q * \mathbf{f} \equiv 1 \pmod{q} \quad \text{a} \quad F_p * \mathbf{f} \equiv 1 \pmod{p} \tag{3.20}$$

kde F je $F = [F_0,F_1,...,F_{N-1}]$ kde F_i je element kruhu. Použitá operace v rovnici (3.20) je operace konvoluce. Entita vypočítá polynom **h**, který je zároveň veřejným klíčem K_{pub}:

$$\mathbf{h} \equiv \mathbf{F_q} * \mathbf{g} \pmod{q} \tag{3.21}$$

Pak je veřejný klíč reprezentován jako polynom $K_{pub}(h)$ a soukromý klíč jako polynom $K_{priv}(f, F_p)$.

Proces šifrování taky není náročný. Odesílatel převede zprávu na m bloků, pak zvolí náhodný polynom **Φ**, použije veřejný klíč a vypočítá rovnici:

$$e \equiv p\Phi * h + m \pmod{q} \tag{3.22}$$

a odešle zprávu příjemci. Náročnost šifrování je v tomto případě $O(N^2)$, při použití FFT (Rychlé Fourierovy Transformace) lze snížit náročnost na $O(N \log N)$. FFT lze použít, protože prováděná operace je konvoluce. Příjemce obdrží kryptogram e. Soukromým klíčem K_{priv} reprezentovaným jako polynom **f** a s pomocí polynomu $\mathbf{F_p}$ dešifruje zprávu jako rovnici (3.23):

$$a \equiv \mathbf{f} * e \pmod{q} \tag{3.23}$$

hodnotu a z rovnice (3.23) použije a vypočte zprávu m z rovnice (3.24):

$$m \equiv \mathbf{F_p} * a \pmod{p} \tag{3.24}$$

Hlavní bezpečnostní vlastností systému NTRU je správná volba parametru N, p a q. Celý systém lze rozdělit na tři skupiny podle bezpečnosti. Na **Mírnou bezpečnost**, **Velkou bezpečnost** a na **Vysokou bezpečnost**. Mírná bezpečnost je skupina, která má délky parametru takové $(N,p,q) = (251; 3; 128)$ a s délkami klíčů $(K_{pub}, K_{priv}) =$ (642 b; 340 b). Mírná bezpečnost se hodí pro šifrování krátkých zpráv. Velká bezpečnost závisí na délkách $(N,p,q) = (347; 3; 128)$ a délkách klíčů $(K_{pub}, K_{priv}) = (1169$ b; 530 b). Vysoká bezpečnost pak na délkách parametrů $(N,p,q) = (503; 3; 256)$ a $(K_{pub}, K_{priv}) =$ (4024 b; 1595 b). Vysoká bezpečnost poskytuje nejlepší zabezpečení z uvedených, ale pro vysokou velikost klíčů je velmi náročná na výpočetní stroj.

4 ZHODNOCENÍ A PERSPEKTIVA MODERNÍCH ASYMETRICKÝCH KRYPTOSYSTÉMŮ

Většina z uvedených asymetrických kryptosystémů v kapitole 3 jsou studovány již delší dobu. Existuje řada jiných kryptosystémů, které ale z hlediska bezpečnosti nejsou používány a již se od nich opouští. V praxi pro asymetrické šifrování se nejvíce používá systém RSA. Je také nejdéle studován a nejvíce modifikován. Některé uvedené systémy byly nejdříve vyvinuty pro asymetrickou kryptografii, ale v praxi se neosvědčily. Proto byly nasazeny v jiných odvětvích kryptografie jako například při digitálních podpisech. V dnešní době se zvyšuje výkon výpočetních strojů a zdokonalují se algoritmy používané pro prolomení šifer, neboli prolomení daného problému, na kterém je systém založen. Zvyšováním výkonu výpočetních strojů se zvyšuje i pravděpodobnost prolomení bezpečnostních ochran kryptosystémů v reálném čase a tak je třeba reagovat na tyto změny. Např. u RSA se bezpečnost systémů projevuje zvyšováním délky klíčů. Jedná se o snadné opatření, ale tím se zvyšuje i čas potřebný ke generaci klíčů a pro samotné šifrování a dešifrování. Z toho důvodu se kryptografici snaží vymyslet složitější kryptosystémy a složitější matematické problémy, na kterých jsou kryptosystémy založeny. Právě proto se více nasazuje systémy s větší bezpečností bez nutnosti zvyšování velikosti klíčů. Nejčastěji jsou to systémy pracující na problému lattice nebo na problému eliptických křivek. Tyto systémy jsou budoucností asymetrické kryptografie. S nástupem vyšších výkonů výpočetních strojů se budou systémy NTRU a eliptické křivky používat častěji.

4.1 Porovnání dle náročnosti systému

Každý systém je nějak náročný na výpočetní přístroj. Podle náročnosti pak lze jednoduše určit zda systém lze použít na daném hardwaru. Následující tabulka (Tab. 4.1) zobrazuje kryptosystémy dle náročnosti výpočtu klíčů a šifrování na výpočetní systém. Náročnost je uvedena jako nejhorší možná složitost – notace Omikron, kde N je délka vstupní informace kryptosystému, nejčastěji je to délka šifrovaného bloku.

Tab. 4.1: Porovnání kryptosystémů dle náročnosti.

	RSA	NTRU	ElGamal	McEliece	M-H	Rabin
Náročnost výp. klíčů	$O(N^2)$	$O(N)$	$O(N)$	$O(N^2)$	$O(N)$	$O(N)$
Náročnost šifrování	$O(N^3)$	$O(N^2)$	$O(N^2)$	$O(N^2)$	$O(N)$	$O(N^2)$
Náročnost dešifrování	$O(N^3)$	$O(N^2)$	$O(N^2)$	$O(N^2)$	$O(N)$	$O(N^3)$

Z tabulky (Tab. 4.1) lze vidět, že systémy NTRU, ElGamal a Merkle-Hellman jsou nejméně náročné na výpočetní výkon. Náročnost závisí na správné implementaci. Pak lze efektivně využít celý výpočetní výkon stroje a náročnost zmenšit. [9] [20] [24]

4.2 Porovnání dle délky klíče

U každého systému závisí bezpečnost na délce klíčů, délce zvoleného modula a na velikosti grupy, nad kterou je kryptosystém vytvořen. Čím větší klíč je použit, tím menší je pravděpodobnost vypočítání soukromého klíče z klíče veřejného. Některé systémy jako RSA již neposkytují přijatelné zabezpečení s délkou soukromého klíče 512 bitů, proto se zvýšení bezpečnosti projeví použitím většího klíče. Z toho důvodu je rychlost šifrování dost vysoká než u jiných systému. U systému RSA délky klíčů již sahají do velmi velkých čísel a samotný systém požaduje velkou paměť a vysoký výpočetní výkon. Proto se hledají jiné alternativy, které poskytují tu samou bezpečnost s menšími klíči. Tabulka Tab. 4.1 zobrazuje náročnost kryptosystémů. Tyto údaje jsou relativní, protože náročnost je závislá na délce vstupního bloku zprávy a ta je závislá na délce klíče, proto nelze počítat rychlost nebo efektivitu z tohoto údaje. Následující tabulka (Tab. 4.2) zobrazuje bezpečnost kryptosystémů porovnáním podle délky klíčů. Jednotky jsou uvedeny v bitech.

Tab. 4.2: Porovnání bezpečnosti dle délky klíčů. Zdroj [9].

Blokové šifry	RSA	Eliptické křivky
[b]	[b]	[b]
80	1024	163
128	3072	283
192	7680	409
256	15360	571

Z tabulky Tab. 4.2 lze vidět, že kryptosystém eliptických křivek poskytuje tu samou bezpečnost s menší délkou klíče než u systému RSA. Právě proto je nasazení eliptických křivek častější. Díky menším nárokům na paměť je systém eliptických křivek více používán v malých přístrojích s menší pamětí, např mobilní telefony, smart karty a jiné (zdroj [9], [20]).

4.3 Kryptosystémy dle data vytvoření

Následující tabulka (Tab. 4.3) zobrazuje datum vytvoření, popř. datum vytvoření patentu na daný kryptosystém.

Tab. 4.3: Datum vytvoření kryptosystému.

RSA	1978
McEliece	1978
Merkle-Hellman	1978
Rabin	1979
ElGamal	1984
Probabilistic	1984
Eliptické křivky	1985
NTRU	1996

Z tabulky lze vidět, že systém RSA je nejstarší. Znalostí o tomto systému je za svojí životnost velmi hodně, proto je to nejpoužívanější systém v dnešní době. Kryptosystémy eliptických křivek a NTRU jsou nejmladší ze všech. Teprve teď se začínají používat, díky většímu výpočetnímu výkonu a znalostem, pomocí kterých lze ověřit jejich bezpečnost. Jejich nasazení není tak velké, protože kryptoanalitických znalostí o starších systémech je více a bylo provedeno více pokusů o útoky, které systémy přečkaly.

4.4 Doporučené délky klíče pro přijatelnou bezpečnost

Každý kryptosystém je bezpečný pouze tehdy když jsou dodrženy všechny doporučené parametry, délky klíčů apod. Následující tabulka (Tab. 4.4) zobrazuje délky klíčů a délku modulu pro přijatelnou bezpečnost, ale i pro efektivitu systému a rychlost. U systému Rabin nebyly tyto parametry přímo zjištěny, ale z praxe lze jednoduše určit, že velikost p a q musí být nejméně 512 bitů a musí to být prvočísla jako je to u systému

RSA.

Tab. 4.4: Doporučené délky parametrů v bitech pro přijatelnou bezpečnost (zdroj [18], [9], [20]).

	K_{pub}	K_{priv}	*modul*	*p*	*q*	Délka *N*	*t*	*k*
	[b]	[b]	[b]	[b]	[b]	[b]	[b]	[b]
RSA	3*	2048	2048	1024	1024	–	–	–
NTRU	1169	530	–	3	128	251	–	–
ElGamal	512	512	1024	–	–	–	–	–
McEliece	512	512	1024	–	–	–	38	644
M-H	–	–	512	–	–	–	–	–
Rabin	1024	1024	1024	512	512	–	–	–
Probabilistic	–	–	1024	–	–	–	–	–
Elip. Křivky	160	160	512	–	–	–	–	–

U jiných kryptosystému údaj není uveden z důvodu, protože nebyl nalezen. U všech systému pracujících s operaci modulo je doporučená délka modulu od 1024 až 2048 bitů. Délka veřejného klíče je pouze ilustrativní a může se pohybovat i v menších číslech. Snížení veřejného klíče u RSA až na velikost 3 bitů vede k vyšší rychlosti šifrování. Na druhé straně musí mít soukromý klíč délku podstatně větší z důvodu bezpečnosti. Tuto volbu délky veřejného klíče lze provést pouze pokud systém není určen k dlouhému používání nebo nemusí poskytovat maximální bezpečnost. Takový případ systému je na příklad při posílání zpráv s krátkou platností (posílání bankovních příkazů). Jedině systém NTRU má menší hodnotu modulů, ale operaci modulo provádí dvakrát s jinými hodnotami. U systému eliptických křivek je doporučeno používat 160 bitové klíče. S volbou menších velikostí klíčů a modulu lze systém prolomit v reálném čase. U systému McEliece nemusí být klíče velké, protože klíče jsou definovány jako matice a počítají se z parametrů *t* a *k*.

4.5 Doporučení NIST pro budoucí použití

Organizace NIST je Národní Institut Standardů a Technologií založen americkou federální vládou už v roce 1901. Zabývá se vědeckým měřením a novými technologickými inovacemi v průmyslu a jiných oblastech vědy. Tato organizace se mimo jiné zabývá bezpečnostními doporučeními pro asymetrickou kryptografii. Tabulka Tab. 4.5 zobrazuje navyšování délek klíčů v závislosti na jejich matematickém

problému do příštích let.

Tab. 4.5: Doporučení NIST z roku 2007 pro délky klíčů pro budoucí použití.

Rok	Symetrická kryptografie	Symetrická kryptografie	Asymetrická kryptografie	DLP	Eliptické křivky
[-]	[b]	[-]	[b]	[b]	[b]
2007-2010	80	2TDEA[3]	1024	1024	160
2011-2030	112	3TDEA[4]	2048	2048	224
2030-2050	128	AES-128	3072	3072	256
2050-2070	192	AES-192	7680	7680	384
2070-více	256	AES-256	15360	15360	512

Modře označené řádky představují dle doporučení Národní Bezpečnostní Služby NSA v USA za vysoce bezpečné, označované „Top Secret“ (zdroj [37]). Více informací lze získat na stránkách viz [37] .

4.6 Shrnutí

Systém eliptických křivek dnes přináší vysokou bezpečnost při šifrování. Díky délce klíčů 160 bitů (Tab. 4.4) je náročnost na systém nízká a tím je šifrování rychlejší. Používá menší hodnotu modulu než RSA. Při šifrování neumocňuje, pouze násobí vstupní blok zprávy s klíčem. Vyžaduje menší paměť, a proto je více a více častější využití tohoto systému. Jeho nasazení je hlavně v malých výpočetních systémech jako jsou mobilní telefony, smart karty a jiné. Poskytuje lepší bezpečnost než RSA s menší délkou šifrovacího klíče. Je vhodným nástupcem systému RSA.
NTRU kryptosystém je dalším z perspektivních systémů pro budoucí využití. Díky tomu, že stojí na problému mřížky je jeho prolomení takřka nemožné i v blízké budoucnosti. Také jako eliptické křivky používá menší délky parametrů než RSA. Díky menším klíčům a menší délce modula lze ho uplatňovat na malých zařízeních s nízkým výkonem výpočtu. Systém NTRU je rychlý při výpočtu klíčů. Většina hodnot z tabulek v kapitole 4 jsou teoretické hodnoty čerpány z dokumentů uvedených v literatuře, viz [1], [9], [11], [18].

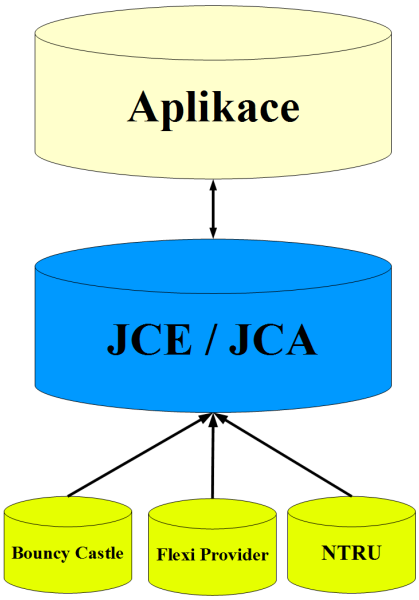
3 2TDEA znamená provedení cyklu šifrování 2-krát za sebou.
4 3TDEA je provedení cyklu šifrování 3-krát, jako je to u 3DES.

5 IMPLEMENTACE KRYPTOGRAFICKÝCH METOD DO PROGRAMU

V rámci diplomové práce byl vytvořen program, který měří časy generování klíčů, šifrování a dešifrování implementovaných kryptografických systémů. Program dokáže zašifrovat a dešifrovat textovou zprávu. Výsledky se ukládají do zvolené složky v podobě textových souborů. Složka obsahuje i soubor se změřenými časy vynesených do grafů. Program byl vytvořen v programovacím jazyku Java pro jeho jednoduchost a širokou podporu kryptografických balíčků. Popis Javy a její práci s kryptografii je popsán v kapitole 5. Program byl vytvořen ve vývojovém programu NetBeans, z důvodu snadného vytváření grafického rozhraní. Zapouzdření potřebných kryptografických balíčku se provádí jednoduše přes vkládací funkci v programu NetBeans.

5.1 Kryptografie v Javě

Základní kryptografické funkce v Javě jsou realizovány přes JCA – Java Cryptography Architecture a přes JCE – Java Cryptography Extension. Použitím těchto dvou architektur umožňuje Java implementovat kryptografické knihovny a „frameworky“ jiných skupin vývojářů, tzv. „*Providerů*“. Tímto způsobem lze jednoduše zahrnout nové kryptografické systémy do platformy Javy. Implementace těchto kryptografických systémů do programů je pro vývojáře pak snazší. Architektury JCA a JCE mají na starosti celý průběh šifrovacího a dešifrovacího cyklu. Při šifrování se používá třída „Cipher“, která je součástí JCA a JCE. Pro každý kryptografický systém se tato metoda liší. Vývojář má volné ruce při implementování kryptografické metody, protože nemusí měnit celý zdrojový kód. Stačí pouze, aby změnil poskytovatele balíčku. Například při implementaci RSA systému lze použít dva balíčky – Bouncy Castle nebo Flexi Provider. Vývojář až po napsání kódu může zvolit, ze kterého balíčku se systém RSA bude volat a to jediným řádkem - *Security.addProvider(new org.bouncycastle.jce….* Hlavní blokové schéma architektur JCA/JCE zobrazuje Obr. 5.1.



Obr. 5.1: Vrstvy kryptografie v Javě.

Pro práci s kryptografickými metodami má Java speciální soubory, které slouží k omezování práv programu – „local_policy.jar“ a „US_export_policy.jar“. Tyto soubory definují politiku Javy pro vývojové prostředí a pro správnou funkci programu. Soubory politiky se nacházejí v adresáři C:\Program Files\Java\jre\lib\security\. Pro uvolnění práv je třeba přepsat soubory, které lze získat z adresy viz [33]. Při práci se simulačním programem se doporučuje tyto soubory nainstalovat. Více informací uvedeno v [28], [33], [34].

Velkým balíčkem, který obsahuje většinu starších kryptografických systémů, je balíček společnosti Bouncy Castle, viz [29]. V programu byly použity systémy RSA, ElGamal a Eliptické křivky. Další systémy obsažené v balíčku, jsou popsány v [29] na webových stránkách společnosti. Dalším použitým balíčkem je balíček Flexi Provider, který obsahuje nástroje kryptografického systému McEliece a systém eliptických křivek. Při implementaci systému eliptických křivek z balíčku Flexi Provider nastala chyba při šifrování, a proto byl použit systém z knihovny Bouncy Castle. V programu jsou implementovány následující kryptografické metody s názvem použitého balíčku:

- **RSA** – balíček Bouncy Castle

- **ElGamal** – balíček Bouncy Castle
- **McEliece** – balíček Flexi Provider
- **Eliptické Křivky** – balíček Bouncy Castle
- **NTRU** – autor Tim Buktu (zdroj [35])

Bouncy Castle je jedním z největších poskytovatelů kryptografických systémů. Byl vytvořen skupinou australských vývojářů. Je dostupný pod MIT licencí a lze jej použít i pro komerční účely. Obsahuje asymetrické a symetrické kryptografické systémy, digitální podpis, hashovací funkce a jiné. Balíček Bouncy Castle je dostupný i pro programovací jazyk C#. Více o balíčku ve zdrojích [28], [29].

Flexi Provider je další knihovnou poskytující kryptografické systémy. Byla vytvořena německými vývojáři a je dále vyvíjena Technickou Univerzitou v Darmstadtu, viz [32].

Balíček poskytující kryptografický systém NTRU není uznávaným balíčkem v Javě, a proto se jeho implementace liší od výše uvedených knihoven. Balíček je stále ve vývoji a jeho implementace ještě není tak efektivní jak u starších systémů. Existuje další společnost poskytující systém NTRU pro Javu, ale její politika zakazuje delší užívání systému bez zaplacení licence a zakazuje zveřejnění zdrojových kódů programu. Z toho důvodu byl zvolen balíček autora Tima Buktu.

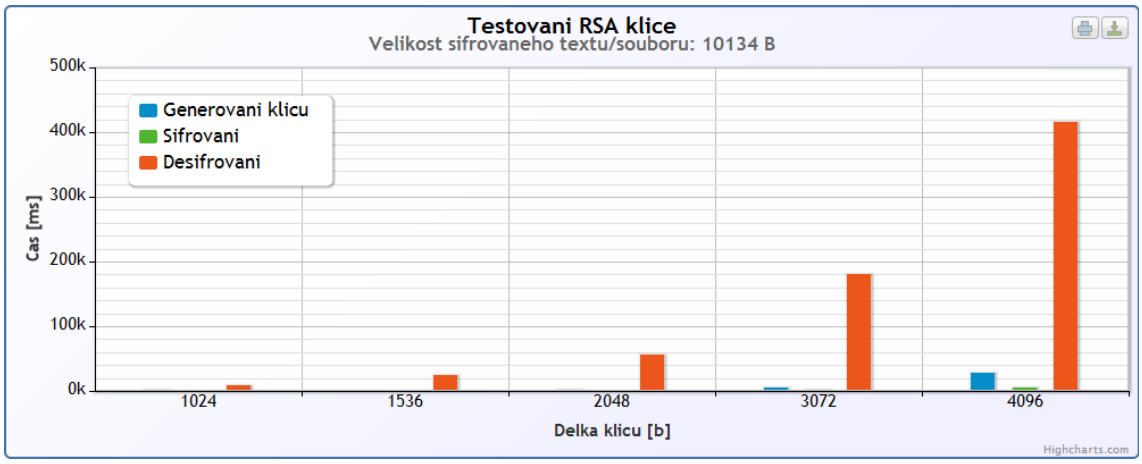
5.2 Popis zdrojového kódu programu

Každý implementovaný kryptografický systém má v programu vlastní třídu. Třída obsahuje metody pro generování klíčů, šifrování a dešifrování. Tato třída dědí všechny proměnné a metody rodičovské třídy *„CryptAbstract“*. Pro všechny systémy je zdrojový kód třídy stejný. Pouze třídy Eliptických křivek a NTRU mají složitější implementaci. Balíček NTRU poskytuje 3 možnosti generování šifrovacích klíčů:

- menší délka klíčů a menší bezpečnost
- vetší velikost klíčů a vetší bezpečnost
- poměr mezi velikosti klíčů a bezpečnosti

Program poskytuje 2 testy. Pro každý test je vytvořena třída, ve které se implementují metody generování klíčů, šifrování a dešifrování každého kryptografického systému. Obě třídy obsahují metody pro globální nastavení testu (nastavuje uživatel pomocí uživatelského rozhraní, viz kapitola 5.4). Třída uživatelského rozhraní navíc obsahuje metody pro nastavení délky klíčů a délku šifrovaného bloku. Dále jsou v programu třídy, sloužící pro převod textového řetězce na bloky, které se pak šifrují. Všechny třídy a jejich metody jsou popsány v příloze A.

Program porovnává změřené časy a vykresluje je pomocí grafů do HTML stránky. Výsledek lze zobrazit ve webovém prohlížeči. O vykreslení se stará knihovna Highcharts (odkaz na stránky viz [36]), která je naprogramovaná v jazyku JavaScript. Tato knihovna se jednoduše implementuje a nabízí velkou škálu typů grafů. Tím, že Highcharts pracují pod JavaScriptem, není třeba vytvářet žádný webový server. JavaScript sám o sobě pracuje na klientském počítači, tj. zdrojový kód se kompiluje ve webovém prohlížeči uživatele. Výsledná stránka se zobrazí i bez použití webového serveru. Je třeba ještě v prohlížeči povolit spouštění JavaScriptu, jinak se zmiňovaná stránka nezobrazí. Výsledky se zapisují přímo do zdrojového kódu JavaScriptu v hlavičce HTML souboru. Tímto je ulehčeno implementování složitých systémů a nadstaveb pro Javu. Proces vykreslování není náročný na výpočetní stroj a tím zbytečně nezatěžuje celý program. Obr. 5.2 zobrazuje vykreslování grafů.

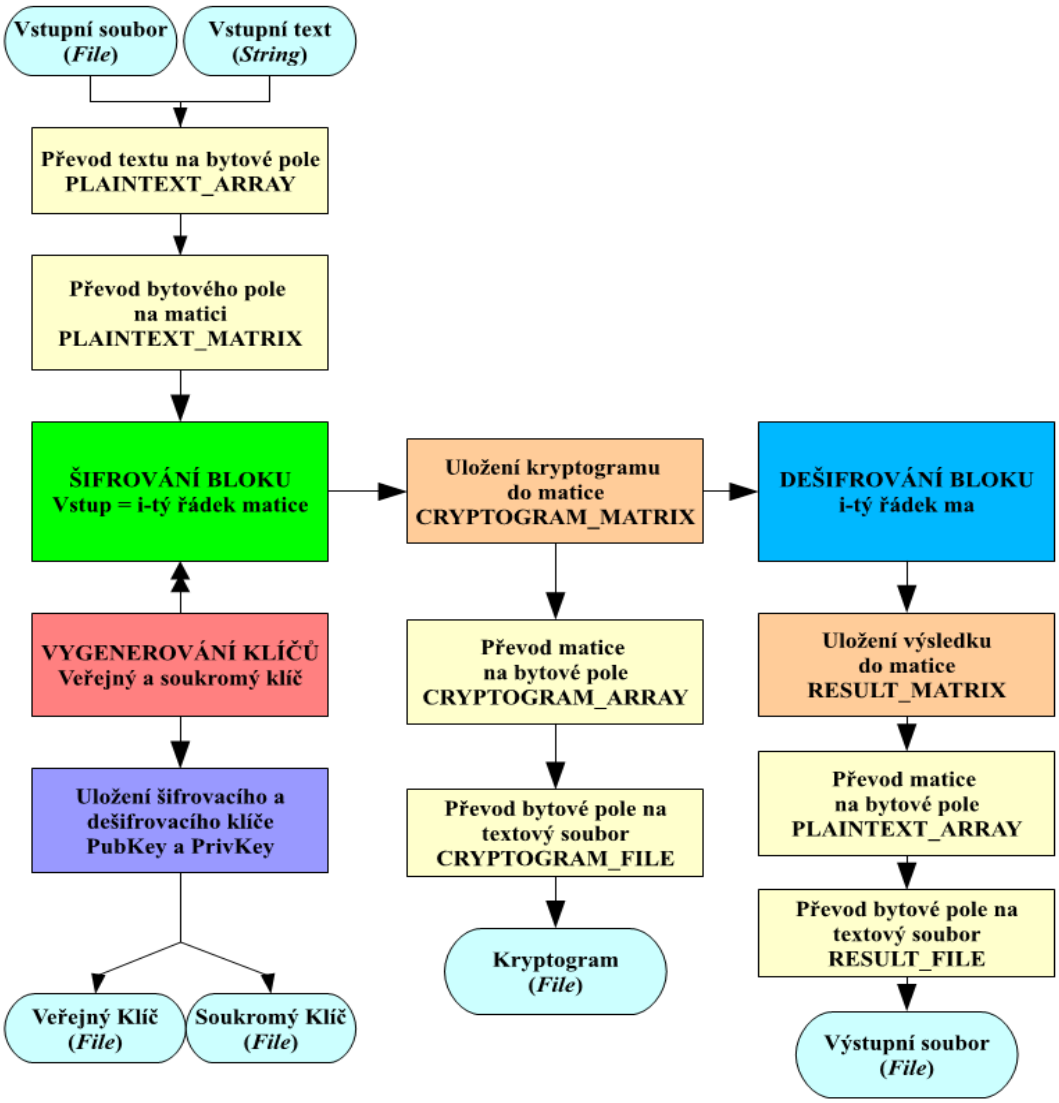


Obr. 5.2: Vykreslování změřených hodnot do grafu.

Výsledný graf je interaktivní. Najetím myší na sloupec se zobrazí naměřený čas. Časy jsou měřeny v nanosekundách, ale pro lepší porovnání jsou zobrazeny v milisekundách. V levém horním rohu je zobrazena legenda. Kliknutím na parametr v legendě daný sloupec času zmizí a zvětší se popř. zmenší další naměřené časy. Systém Highcharts disponuje i ukládáním výsledných grafů do formátů JPEG, PNG a SVG. Graf lze uložit i do formátů PDF. U formátů PDF není grafika ideální, proto je lepší importovat grafy do zmíněných formátů.

5.3 Průběh šifrování

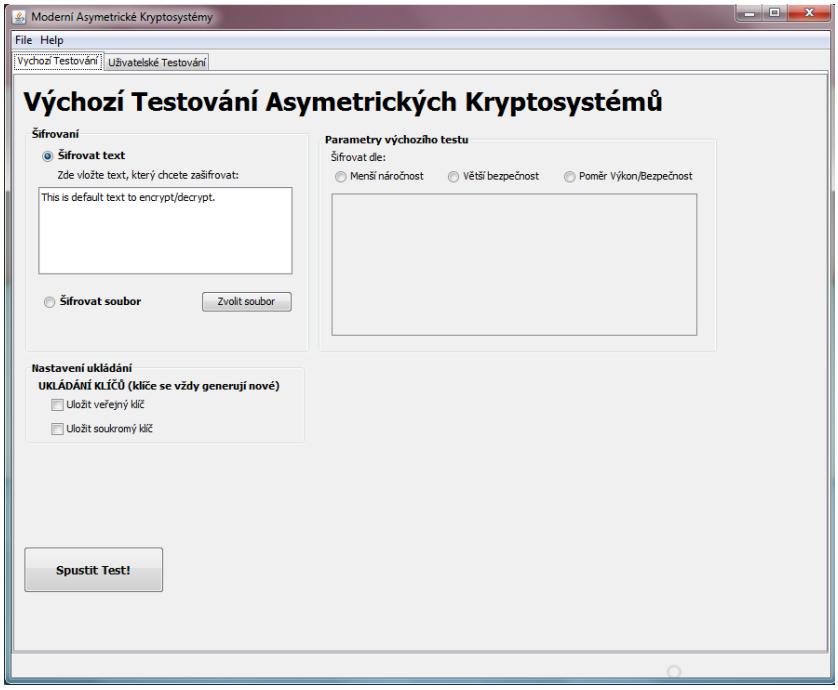
Při spuštění zvoleného testu se provádí nejdřív převod textu na pole obsahující bytovou reprezentaci znaků abecedy. Vytvořené bytové pole se převede na bloky, které se uloží jako matice. Každý řádek matice je šifrovaným blokem. Po převodu se vygenerují potřebné klíče určené velikostí a typem instance. Tato instance je přednastavená a nelze ji změnit. Techniku k ochraně před determismem lze použít pouze v některých systémech. Při rozložení větší zprávy se celý proces šifrování a dešifrování provádí v cyklech a při vstupním textu menším než nastavená délka bloku se provádí v jednom cyklu. Obr. 5.3 zobrazuje zjednodušený model průběhu obou testů.



Obr. 5.3: Zjednodušený model průběhu obou testů.

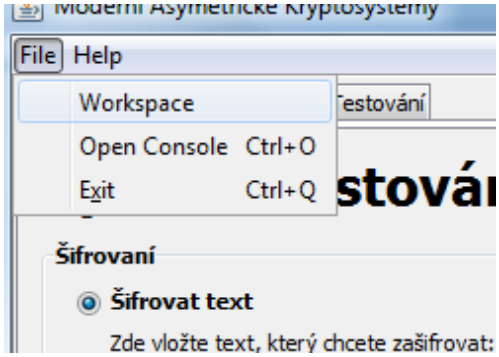
5.4 Popis uživatelského rozhraní programu

Grafické rozhraní tvoří základ přístupu uživatele k nastavením programu a jeho používání. Výchozím zobrazením je obrazovka pro výchozí test, jak zobrazuje Obr. 5.4.

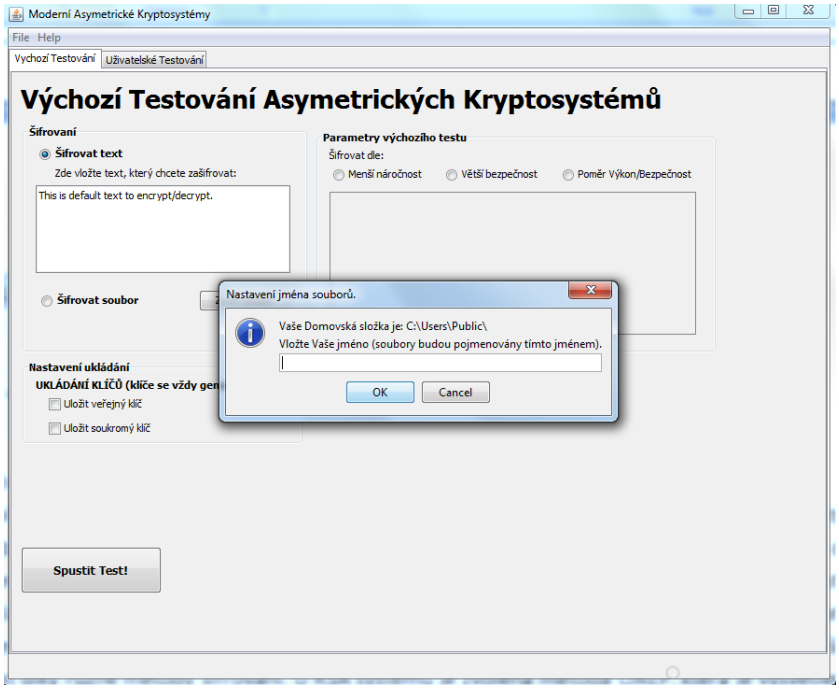


Obr. 5.4: Výchozí obrazovka uživatelského rozhraní programu.

V panelu hlavního menu lze nastavit domovskou složku, kde se mají ukládat všechny výsledky. Po zvolení složky si pak uživatel zvolí jméno, které se pak vyskytuje v názvech výsledku. Pokud uživatel nezvolí jméno (stiskne tlačítko „*Cancel*“) program vytvoří složky s výchozím jménem MAK (zkratka od *Moderní Asymetrické Kryptosystémy*). Jméno MAK se pak vyskytuje i ve jménech výsledků. Pokud uživatel nezvolí domovskou složku, vytvoří se složka MAK v tom samém adresáři, kde se nachází program. Uživatel musí mít práva zapisování do daného adresáře, jinak program vyhodí chybu. Nastavení domovské složky zobrazují Obr. 5.5 a Obr. 5.6.

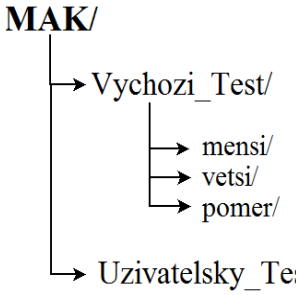


Obr. 5.5: Nastavení domovské složky.



Obr. 5.6: Nastavení jména výchozí složky a jména pro test.

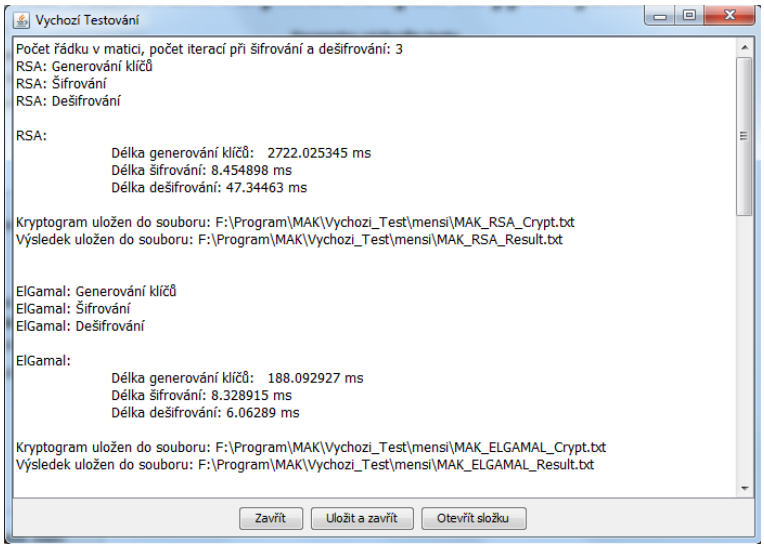
Ve vytvořené složce (MAK) se vytvoří další složky, které obsahují výsledky provedených testů. Hierarchii domovské složky zobrazuje Obr. 5.7. Složka „*Vychozi_Test*“ obsahuje podsložky pro každý typ testování. Složka „*mensi*“ je pro testování s menší náročnosti, „*vetsi*“ pro větší bezpečnost a „*pomer*“ pro testování poměru mezi bezpečností a náročností. V každé podsložce se po testu objeví textové soubory s výslednými kryptogramy a dešifrovanými textovými řetězci s názvy testovaných systémů. Zvolený strom složek byl navržen tak, aby uživatel měl lepší přehled po ukončení testu. Podsložky obsahují i HTML soubor s výsledným grafem.



Obr. 5.7: Složky pro uložení výsledků.

5.5 Testování

Po nastavení domovské složky lze již přejít k samotnému testování implementovaných systémů. Pokud uživatel nenastaví domovskou složku, v programu vyskočí okno pro zadání domovské složky, jak je zobrazeno na Obr. 5.6. Uživatel má k dispozici testování kryptografických metod dva způsoby. Výchozí testování s přednastavenými hodnotami nebo uživatelské testování, kde lze zvolit délku klíčů. Při testování lze vybrat šifrování vlastního vloženého textu, nebo textového souboru. Při šifrování se vstupní data převedou na bytové pole, které se pak šifruje. Systémy šifrují pouze určitou velikost vstupního bloku, proto se nejdřív vstupní bytové pole převede na bytovou matici. Výsledná matice se pak šifruje do kryptogramu, který je také reprezentován maticí. Při převodu se automaticky na konec matice vloží nuly, aby matice byla celá vyplněná určitou hodnotou. Kdyby se ta nestalo, program by vyhodil chybu o šifrování neexitující hodnoty. Po dešifrování se znovu matice převede na bytové pole a to pak na textový řetězec typu „*String*“. Výstup se pak ukládá do domovské složky do textového souboru, pro ověření správného šifrování a dešifrování. Výstup se ukládá pro každý kryptografický systém. S výstupem se ukládá i zašifrovaný vstupní blok dat, pomocí kterého lze porovnat různorodost šifrování pro každý systém. U každého systému se liší zašifrovaný text díky různé metody šifrování. V testu lze zvolit i ukládání veřejného a soukromého klíče každého systému. Tyto soubory slouží pouze pro znázornění klíčů, nikoliv pro další použití. U testování se šifrovací a dešifrovací klíč volí znovu, zcela náhodně. Průběh testu zobrazuje konzole Obr. 5.8.

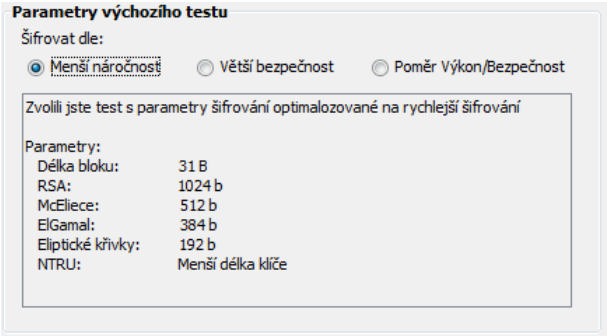


Obr. 5.8: Konzole zobrazující průběh testování.

Konzole zobrazuje aktuální stav testu a výsledné časy kryptografických systémů. Tento výstup lze uložit do textového souboru, který se taktéž nachází v domovské složce v podsložce dokončeného výchozího testu nebo ve složce uživatelského testu. Tlačítko „*Otevřít složku*“ slouží pro otevření složky, kde se nachází aktuální výsledky.

5.5.1 Výchozí testování

Záložka „*Výchozí testování*“ slouží k přednastavenému testu, kde lze porovnat systémy dle přednastavených parametrů. Délky klíčů jednotlivých kryptografických systémů jsou zvoleny dle tabulky Tab. 4.4 a zdrojů [9], [20], [37]. Ve výchozím testu lze testovat kryptografické systémy s menší náročnosti, větší bezpečnosti a poměrem mezi bezpečnosti a náročnosti. Nastavené testování s menší náročnosti je zvoleno tak, aby velmi nezatěžovalo výpočetní stroj. Při této volbě parametrů systémy poskytují minimální bezpečnost. Délky byly zvoleny dle Obr. 5.9.



Obr. 5.9: Možnosti přednastaveného testu.

Parametr délka bloku určuje velikost šifrovaného bloku a je pro všechny systémy stejná. Je zvolena dle systému McEliece, protože tento systémy podporuje pouze délku šifrovaného bloku vypočtenou dle následující rovnice (5.25)

$$B = \frac{K}{16} \tag{5.25}$$

kde K je velikost klíčů. Popis vypočtení délky bloku pro jednotlivé kryptografické systémy zobrazuje Tab. 5.1.

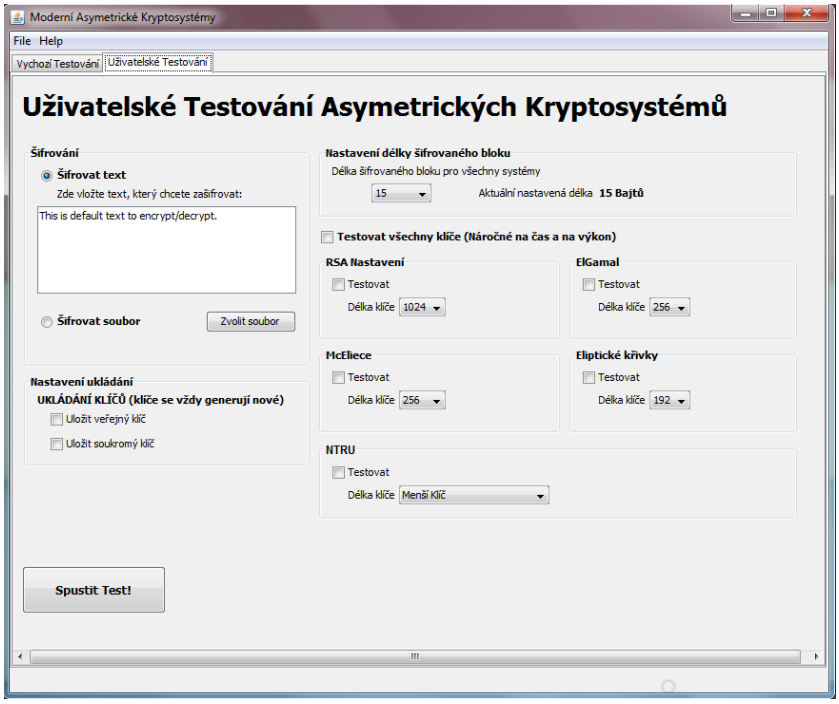
Tab. 5.1: Rovnice pro výpočet délky bloku pro šifrování.

	Rovnice pro vypočtení délky bloku [B]
RSA	$B = \frac{K}{8} - 11$
ElGamal	$B = \frac{K}{8} - 1$
McEliece	$B = \frac{K}{16}$
Eliptické křivky	Nespecifikuje délku bloku
NTRU	Pro menší klíč = 170 B Pro poměr = 186 B pro větší klíč = 247 B

Kde *K* je délka soukromého klíče. Systém McEliece se již nedoporučuje používat kvůli jeho prolomení, zde je jenom pro porovnání s jinými systémy. Test se spouští tlačítkem „*Spustit Test*“, po kterém se otevře okno konzole.

5.5.2 Uživatelské testování

Volitelný test lze spustit v záložce uživatelské testování. Jak u výchozího testu je na uživateli zda šifrovat vlastní text nebo textový soubor. Znovu si uživatel může vybrat, jestli ukládat šifrovací a dešifrovací klíč. Test se liší možností, že si uživatel zvolí potřebnou délku klíče. U všech systémů jsou stejné klíče jako u výchozího testu. Jsou přidané délky, které se zpravidla nepoužívají, neboť jsou příliš malé a nebezpečné nebo příliš velké. Velký klíč poskytuje až nadměrnou bezpečnost. Pokud uživatel zvolí velký klíč, test bude samozřejmě trvat delší dobu. Uživatelský test je zobrazen na Obr. 5.10.



Obr. 5.10: Grafické rozhraní uživatelského testu.

Uživatel pomocí zaškrtávacího políčka volí, který systém se má otestovat. Délka šifrovaného bloku je vždy vypočítá dle Tab. 5.1 a bude stejná pro všechny systémy. Program vždy nastaví délku šifrovaného bloku na největší možnou. Uživatel si pak může zvolit vlastní délku šifrovaného bloku ze seznamu. V uživatelském testu lze zvolit testování všech délek klíčů pro daný kryptografický systém. Jak je uvedeno v programu, tato možnost je náročná na čas. Například u systému ElGamal generování dlouhých klíčů je až řádově v minutách. Znovu se test spustí tlačítkem „*Spustit Test!*“, po kterém následuje otevření okna konzole. Po dokončení testu se výsledky uloží do složky „*Uzivatelsky_Test*“. Jednotlivé kryptogramy a dešifrované texty obsahují ve svém názvu i délku klíče, se kterým byl test spuštěn. Opět se změřené časy exportují do html souboru. Ve výsledné webové stránce jsou grafy zobrazeny podle kryptografických systémů.

Všechny naměřené výsledky se smažou a nahradí novými při opětovném testu. Uživateli je doporučeno, aby si výsledky zálohoval před spuštěním nového testu.

6 VYHODNOCENÍ NAMĚŘENÝCH HODNOT

6.1 Parametry výpočetního stroje

Pomocí testovacího programu vytvořeného v rámci diplomové práce se změřily následující charakteristiky. Byl změřen čas generování klíčů, čas šifrování a čas dešifrovaní pro zvolený vstupní text nebo soubor. Měření probíhalo ve vývojovém prostředí NetBeans. Test byl spuštěn na operačním systému **Windows 7**. Počítač, na kterém byl spuštěn test, měl následující parametry:

- Dvoujádrový procesor Intel Core Duo
- 1,86 GHz taktovací frekvence procesoru
- 2 GB operační paměť.

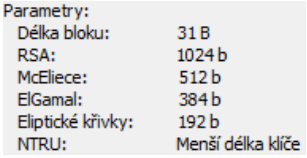
Jako vstupní data test šifroval textový soubor o velikosti **10134 B**, náhodně stažený z internetu. Tento soubor je veřejně přístupný a jeho obsah není v důležitý pro testování. Výsledky všech měření jsou uloženy na přiloženém paměťovém disku v příloze A.

6.2 Měření výchozího testu

Měření výchozího testu mělo 3 fáze. Měření s menší náročnosti, měření s větší bezpečnosti a měření poměr bezpečnost a náročnost.

6.2.1 Menší náročnost

Pro menší náročnost byly zvoleny parametry dle Obr. 6.1



Obr. 6.1: Parametry testu s menší náročností.

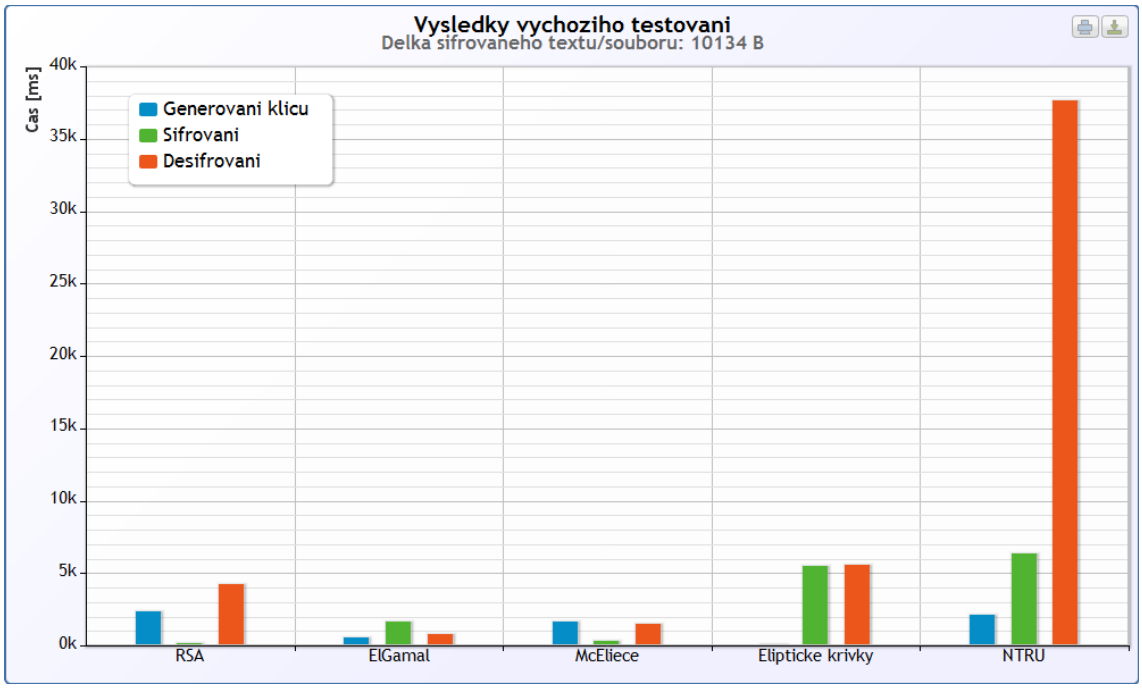
Pro RSA systém se doporučuje používat klíče 1024 a větší. Systém ElGamal šifruje velmi malým klíčem, neboť generování většího klíče způsobuje vetší zatížení. U

systému ElGamal a McEliece není bezpečnost zaručena, délky klíčů jsou zvoleny pouze pro porovnání. Pro eliptické křivky se doporučuje používat již klíč 192 b kvůli minimální bezpečnosti. Délka vstupního bloku je zvolena dle (5.25) (délka klíče se bere dle systému McEliece). Následující tabulka Tab. 6.1 zobrazuje naměřené hodnoty:

Tab. 6.1: Naměřené hodnoty pro test s menší náročností.

	Čas generování klíčů[ms]	Čas šifrování [ms]	Čas dešifrování[ms]
RSA	2422,87	207,96	4260,07
ElGamal	549,71	1701,21	860,59
McEliece	1695,60	353,71	1494,55
Eliptické Křivky	141,35	5540,1	5643,28
NTRU	2123,19	6389,06	37703,89

Všechny časy jsou vyneseny do grafu Obr. 6.2.



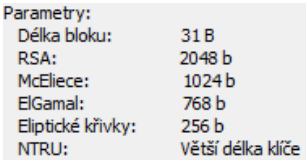
Obr. 6.2: Naměřené časy pro test s menší náročností.

Z Tab. 6.1 a Obr. 6.2 lze konstatovat, že pro menší náročnost nejlepších výsledků v generování klíčů dosahuje systém eliptických křivek. Tuto hodnotu lze očekávat, protože systém eliptických křivek používá velmi malé délky klíčů, a proto je čas nejmenší. Nejlepších výsledků v šifrování z naimplementovaných systémů dosahuje systém RSA, ale má nejdelší čas generování klíčů. U dešifrování je nejrychlejší

ElGamal. Naopak nejhorších výsledků v dešifrování dosahuje systém NTRU, který dešifruje víc jak 2 minuty. Jeho vysoké zpoždění je dáno iteracemi procesů šifrování a dešifrování. Systém podporuje větší šifrované bloky (viz Tab. 5.1) tak se jeho délka dešifrování může zmenšit.

6.2.2 Větší bezpečnost

Pro větší bezpečnost byly zvoleny délky klíčů zobrazeny na Obr. 6.3.



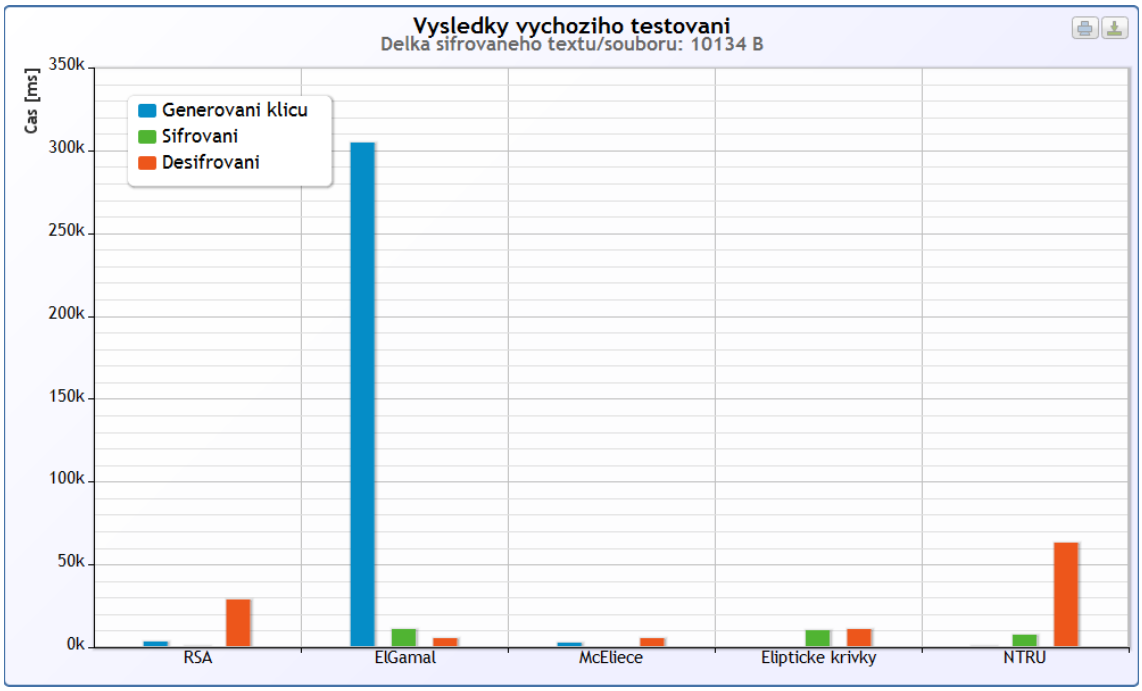
Obr. 6.3: Parametry testu s větší bezpečností.

Pro systém ElGamal byla zvolena menší délka klíčů. Při větší délce klíčů program má tendenci se zacyklit při tomto testu. U eliptických křivek je délka klíčů velká, za to systém poskytuje vysokou bezpečnost. Následující tabulka Tab. 6.2 zobrazuje naměřené hodnoty.

Tab. 6.2: Naměřené hodnoty pro test s menší náročností.

	Čas generování klíčů[ms]	Čas šifrování [ms]	Čas dešifrování [ms]
RSA	3538,22	770,26	29255,40
ElGamal	305293,44	11404,88	5744,24
McEliece	2944,65	330,94	5499,38
Eliptické Křivky	166,29	10947,59	11025,29
NTRU	1232,56	7992,06	63452,55

Naměřené časy vyneseny do grafu zobrazuje Obr. 6.4. Nejdelší čas pro generování klíčů potřebuje ElGamal. Pro vygenerování s délkou klíče 768 b potřebuje ElGamal přes 5 minut. Ovšem tento čas se může lišit v závislosti na výkonu výpočetního stroje. Tento systém se již moc nepoužívá, ale pořád je nejrychlejší z implementovaných systémů v dešifrování. Znovu jsou Eliptické Křivky nejrychlejší ve vygenerování klíčů. Opět se používá délka šifrovaného bloku menší, a proto u některých novějších systémů je čas šifrování a dešifrování vyšší než u starších systémů.



Obr. 6.4: Naměřené časy pro test s vetší bezpečností.

6.2.3 Poměr mezi bezpečnosti a náročnosti

Pro poslední přednastavený test byly délky klíčů zvoleny s optimální délkou klíčů a se silnější bezpečnosti. Délky klíčů při tomto testu určují minimální parametry pro použití v blízké budoucnosti. Parametry byly zvoleny dle Obr. 6.5.

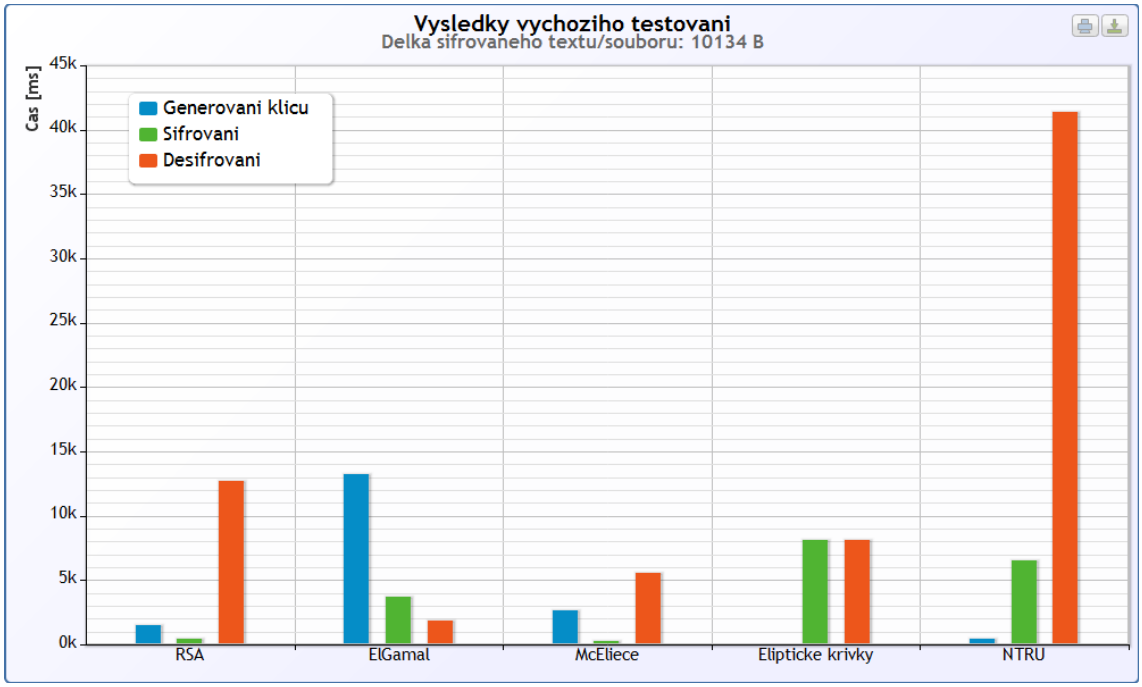
Obr. 6.5: Parametry testu s poměrem mezi bezpečností a náročností.

V tomto testu u systému McEliece jsou parametry zvoleny tak aby systém sloužil jenom pro porovnání s jinými implementovanými systémy. Následující Tab. 6.3 zobrazuje naměřené časy aktuálního testu.

Tab. 6.3: Naměřené hodnoty pro test s menší náročností.

	Čas generování klíčů [ms]	Čas šifrování [ms]	Čas dešifrování [ms]
RSA	1500,97	484,16	12717,46
ElGamal	1331,78	3740,03	1880,97
McEliece	2669,78	331,9	5568,77
Eliptické Křivky	73,52	8152,35	8124,33
NTRU	510,9	6547,76	41406,82

Naměřené časy jsou vyneseny do grafu Obr. 6.6



Obr. 6.6: Naměřené časy pro test s poměrem mezi bezpečností a náročností.

Při poměrovém testu jsou RSA a McEliece nejrychlejší v šifrování. Rychlost šifrování a dešifrování u Eliptických křivek je čas srovnatelný. Natož NTRU u dešifrování je nejpomalejší. Ve výchozím testu jsou všechny parametry přednastaveny. Tím, že se vstupní zpráva dělí na *31 B* bloky, jsou některé systémy pomalé v procesu šifrování a dešifrování. Uživatelský test umožňuje nastavení vlastní délky bloku.

6.3 Měření uživatelského testu

Jak již bylo řečeno, tento test umožňuje nastavit různé délky klíčů pro každý implementovaný kryptografický systém a délku šifrovaného bloku. Grafické rozhraní je

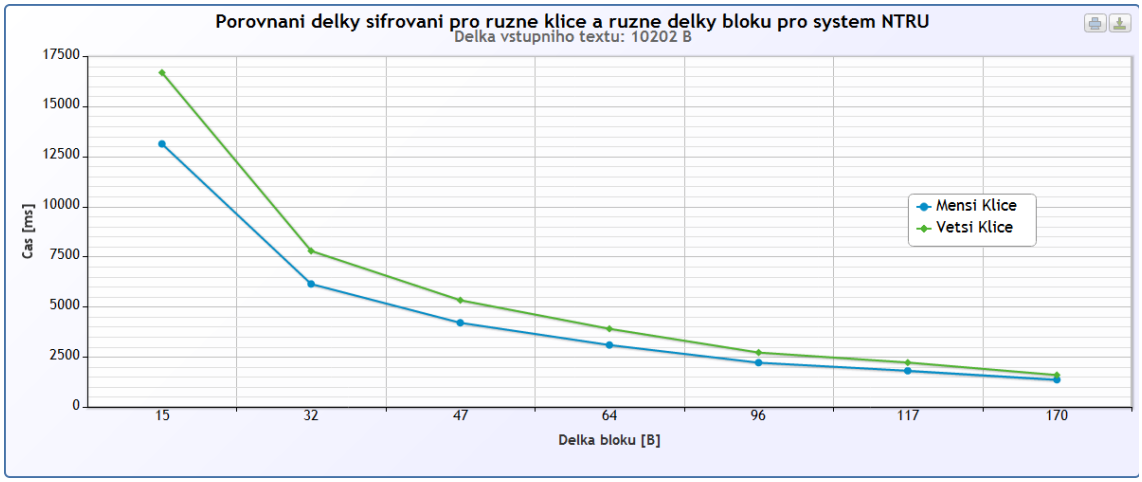
zobrazeno na Obr. 5.10. Uživatelský test byl spuštěn pro všechny klíče metody a pro každý kryptografický systém. Délka bloku byla stejná pro všechny, tj. *15 B*. Naměřené hodnoty zobrazuje tabulka Tab. 6.4. Z tabulky lze vidět, že se zvyšující délkou klíče se zvyšuje doba šifrování a dešifrování lineárně. Nejrychlejší systémy v generování klíčů jsou systém eliptických křivek a systém NTRU.

Tab. 6.4: Tabulka s naměřenými hodnotami pro všechny kryptosystémy a jejich klíče.

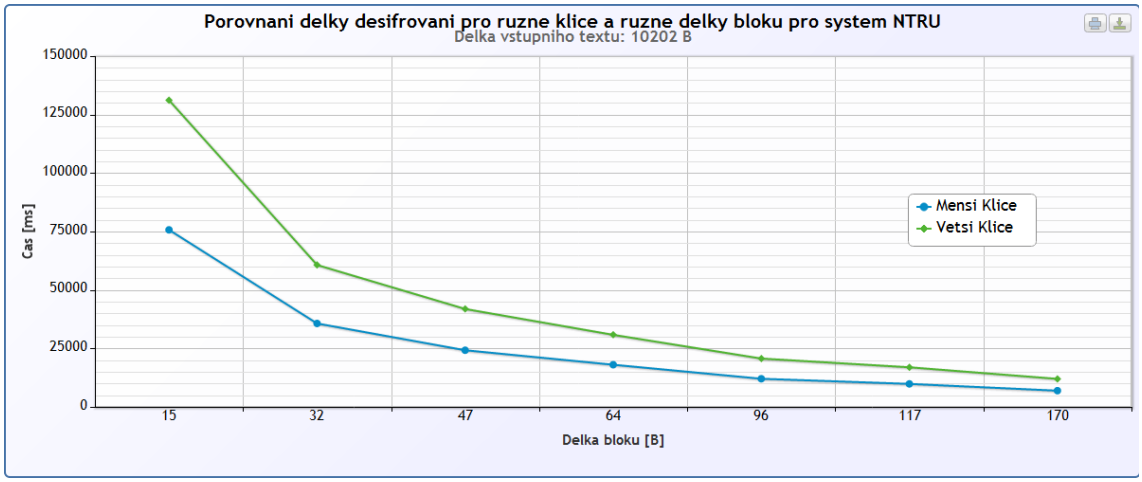
	Délky klíčů [b]	Čas generování klíčů [ms]	Čas šifrování [ms]	Čas dešifrování [ms]
RSA	1024	2177,98	427,37	9020,06
	1536	614,48	899,60	26282,61
	2048	3403,51	1486,76	57838,74
	3072	6510,88	3251,94	182212,23
	4096	28617,86	6130,27	416988,66
ElGamal	256	582,13	1199,22	601,56
	512	95999,84	7682,73	3924,00
	768	48142,97	23930,50	11805,93
McEliece	256	1690,56	65,79	1033,32
	512	363,35	711,41	3137,82
	768	1510,34	650,17	11557,01
	1024	1314,73	652,09	11545,37
Eliptické Křivky	192	150,86	11455,79	11771,55
	224	58,21	16575,70	16556,84
	256	41,63	22715,33	22844,10
	384	96,87	60228,25	60098,92
NTRU	Menší klíč	853,05	12954,80	74931,98
	Poměr	625,82	13258,91	84730,53
	Větší klíč	799,81	16620,76	130779,47

6.4 Měření NTRU pro různé délky bloků

V následujícím testu bylo porovnání menšího a většího klíče systému NTRU pro různé délky šifrovaného bloku. Bylo změřeno, že se zvětšující se délkou bloku čas šifrování a dešifrování klesá pro oba klíče shodně. Toto tvrzení zobrazuje Obr. 6.8 a Obr. 6.7. Obrázek Obr. 6.9 zobrazuje porovnání menšího klíče a jeho největší možnou délku bloku, tj. 170 B a většího klíče s jeho největší možnou délkou, tj. 247 B.

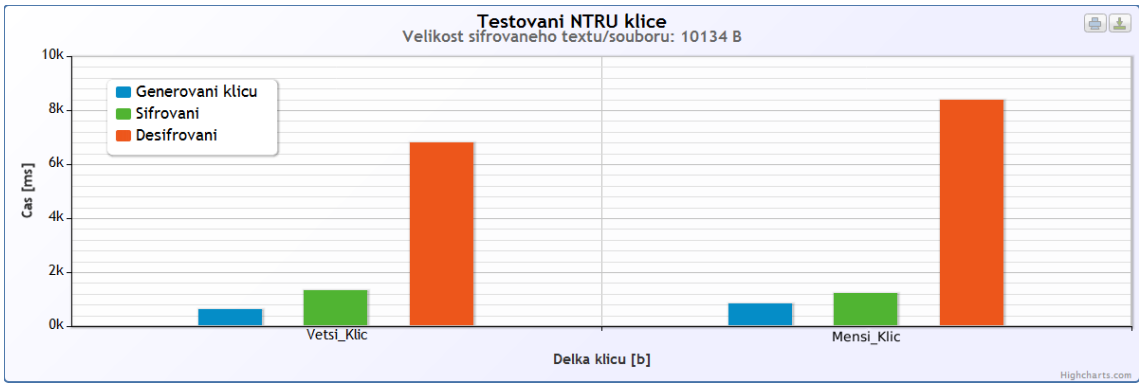


Obr. 6.7: Závislost času šifrování na délce bloku pro systém NTRU.



Obr. 6.8: Závislost času šifrování na délce bloku pro systém NTRU.

Lze vidět, že systém je s větším klíčem v dešifrování rychlejší než s menším klíčem. Díky menšímu počtu iterací je rychlejší. Doba šifrování se liší o 300 milisekund.



Obr. 6.9: Porovnání NTRU s menším klíčem (blok 170 B) a větším (blok 247 B).

6.5 Shrnutí

Z výchozího testu vyplývá, že systém RSA i přes jeho velkou délku klíčů dosahuje přijatelné výsledky při šifrování a dešifrování. Z naměřených hodnot lze vidět , že se zvětšující délkou klíčů se čas zvyšuje lineárně oproti jiným systémům (ElGamal). Z uživatelského testu lze rovněž vidět, že se čas zvyšuje a tím se i zvyšují nároky na výkonnost výpočetního stroje. Z Tab. 6.4 lze zjistit, že šifrování u RSA je velmi rychlé, ale dešifrování je pomalé. Díky tomuto zjištění, lze říci, že systém RSA by šlo nasadit pro šifrování na méně výkonném výpočetním stroji (např. mobilní telefon) a dešifrování by se provádělo třeba na serveru. Ovšem vygenerování klíčů by bylo prováděno na serveru. Naopak by šlo použít systém Eliptických křivek. Díky rychlému vygenerování klíčů může se systém uplatnit v méně výkonných přístrojích. Časy šifrování a dešifrování jsou srovnatelné, a proto lze použít systém eliptických křivek na méně výkonných přístrojích pro šifrování i dešifrování. Systém NTRU (tj. implementace Tima Buktu) dosahuje u všech délek klíčů stejné hodnoty. Změna délky klíčů nemá velký vliv na čas vygenerování. Totožné hodnoty jsou i u šifrování a dešifrování. Systém dosahuje při dešifrování časy větší jak 1 minuta, proto jej nelze nasadit jako rychlejší kryptografický systém. Balíček Tima Buktu je ještě ve vývoji. Zdokonalováním pak lze obdržet lepší výsledky než ty, které jsou uvedeny v práci. Starší systémy ElGamal a McEliece, slouží v programu pouze pro porovnání a jejich praktické využití v moderních systémech je málo pravděpodobné. V programu jsou sice tyto systémy z nejlepšími výsledky, ale nelze je již považovat za bezpečné. Systém ElGamal poskytuje s malou délkou klíčů takřka žádnou bezpečnost. Zvýšení bezpečnosti tohoto systému by vedlo k vysokému času generování větších klíčů a tím

k vysokému požadavku na výkon výpočetního stroje. Vygenerování klíčů s délkou 2048 bitů by obyčejný domácí počítač zvládl za několik desítek minut. Odhadem asi 30-50 minut. Systém McEliece se ani nedoporučuje používat, byl prolomen v minulém století a je považován za nebezpečný.
U testování velmi záleží na délce zprávy. Zde se testovaly systémy s délkou textového souboru nad 10 kB, to znamená, že při zvolené malé délce šifrovaného bloku je větší počet iterací.

Systémy RSA a ElGamal i přesto, že se hojně používají jsou staré a v některých případech velmi neefektivní. Zvětšování bezpečnosti je na úkor zvětšování klíčů a tím i zvětšování větších nároků na počítač. Podle Tab. 4.5 se předpokládá, že v polovině aktuálního století budou potřeba klíče o délce až 10000 bitů. Toto je zcela nepřípustné, protože již 2048 bitový soukromý klíč u RSA již velmi zatěžuje výpočetní systém, viz Tab. 6.2 a Tab. 6.4. Vyvíjejí se nové přístroje s vyšší výpočetní silou, viz dnešní mobilní přístroje, ale i na těchto přístrojích je systém RSA náročný. V současnosti se systémy z eliptickými křivkami staly vhodnou alternativou k systému RSA. Díky svým výhodám v menší náročnosti na paměť se zdá, že v příštích letech zcela nahradí staré asymetrické kryptosystémy. Poskytují stejnou bezpečnost s menšími klíči jak RSA s 1024 bitovým klíčem, ale zvyšování velikosti klíčů není až tak strmé. Jejich nasazení je dnes pomalé a není zcela jisté jestli systém eliptických křivek nevymění nějaký jiný kryptosystém. Systém NTRU je mladší systém než eliptické křivky a není až tak známý a studovaný. Z Tab. 4.4 lze vidět, že systém pro dostačující bezpečnost potřebuje větší veřejný a soukromý klíč než systém eliptických křivek. Tabulka Tab. 6.4 zobrazuje časy šifrování a dešifrování, které jsou jedny z nejvyšších naměřených časů v uživatelském testu. Je to ale systém NTRU, který jako jediný poskytuje ochranu před kvantovým počítačem, viz [15], [16].

ZÁVĚR

V rámci diplomové práce byly teoreticky porovnány moderní asymetrické kryptosystémy z hlediska jejich praktičnosti, náročnosti a bezpečnosti. Každý systém je unikátní díky svým vlastnostem a má jiné využití. Systémy RSA a ElGamal vzhledem ke svým vlastnostem dnes začínají být již neefektivní nicméně se stále používají. Pro přijatelnou bezpečnost musí pracovat s velkými klíči. Velké klíče prodlužují čas potřebný k šifrování a dešifrování. Systémy NTRU a eliptické křivky uvedeny v kapitole jsou považovány za jedny z modernějších metod. Hlavní výhodou je jejich velká kryptografická bezpečnost vzhledem k malé velikosti klíčů. S menší velikosti klíčů vede k menší náročnosti a větší výpočetní efektivnosti systému. Nové kryptosystémy, zejména eliptické křivky, by mohly v budoucnu nahradit stávající systémy jako RSA a ElGamal. V kapitole 4 jsou porovnány zmíněné kryptosystémy.

Byl vytvořen program, který implementuje systémy RSA, ElGamal, McEliece, systém Eliptických křivek a NTRU. Tento program efektivně porovnává změřené hodnoty všech implementovaných kryptografických metod. V kapitole 5 je popsán celý program. S programem lze pracovat přes jeho grafické rozhraní. V kapitole 6 jsou všechny provedené testy podrobně popsány. Výsledky jsou vyneseny do grafů. Pomocí programu lze porovnat aktuální moderní systémy a určit jejich použití v budoucnu.

V dnešní době je systém RSA nejvíce nasazen v praxi a lze říct, že ještě pár let bude. Z naměřených hodnot lze zjistit, že systém RSA dosahuje nejlepších výsledků v průměru ze všech změřených systémů. Druhým systémem, který dosahoval lepší výsledky než jiné měřené systémy je systém eliptických křivek, a proto je vhodnou alternativou pro RSA.

LITERATURA

[1] MENEZES, Alfred, VAN OORSCHOT, Paul, VANSTONE, Scott. *Handbook of applied cryptography*. Boca Raton : CRC Press, 1997. 780 s. ISBN 0849385237.

[2] STALLINGS, William. *Cryptography and Network Security*. 4th edition. [s.l.] : [s.n.], 2006. 592 s. ISBN 0131873164.

[3] BURDA, Karel. *Bezpečnost informačních systémů*. Brno : FEKT Vysokého učení technického v Brně., 2005. 104 s.

[4] VYCHODIL, Vilém. *Algoritmus RSA* [online]. 2002-03-04 [cit. 2010-11-20].

[5] WEISE, Joel. *Public Key Infrastructure Overview* [online]. California : Sun Microsystems, Inc. 2001-09-01 [cit. 2010-11-22].

[6] MASAŘÍK, Karel, CVRČEK, Daniel. *Symetric Key's Infrastructure* [online]. 2003-04-03 [cit. 2010-11-22].

[7] ŠOKA, Michal. *Paralelní faktorizace celých čísel*. PRAHA: FEL Českého vysokého učení technické v Praze., 2008. 46 s.

[8] AGRAWAL, Manindra, KAYAL, Neeraj, SAXENA, Nitin. *Primes is in P* [online]. 2005-09-01 [cit. 2010-11-23].

[9] LASOŇ, Martin. *Porovnání bezpečnosti kryptosystému RSA a Eliptických křivek* [online]. 2005-03-01 [cit. 2010-11-23].

[10] McCURLEY, Kevin, S.. *The Discrete Logarithm Problem* [online]. 1990 [cit. 2010-11-27].

[11] PINKAVA, Jaroslav. *Eliptické křivky v kryptografii a mezinárodní normy* [online]. 2003-05-23 [cit. 2010-11-27].

[12] SUJATHA, R. *Arithmetic of elliptic curves through the ages* [online]. 2009-08-05 [cit. 2010-11-27].

[13] KOBLITZ, Neal. *Elliptic Curve Cryptosystems* [online]. 1987 [cit. 2010-11-27].

[14] MENEZES, Alfred J., OKAMOTO, Tatsuaki, VANSTONE, Scott A. *Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field* [online]. 1993-09-05 [cit. 2010-11-27].

[15] MICCIANCIO, Daniele, REGEV, Oded. *Lattice-based Cryptography* [online] 2008-07-22 [cit. 2010-11-27].

[16] AJTAI, Miklós, KUMAR, Ravi, SIVAKUMAR, D. *A Sieve Algorithm for the Shortest Lattice Vector Problem* [online]. 2001-07-08 [cit. 2010-11-27].

[17] ATİCİ, Ali Can, BATINA, Lejla, FAN, Junfeng, VERBAUWHEDE, Ingrid, YALÇİN, S.Berna Örs. *Low-cost Implementation of NTRU for pervasive security* [online]. 2008-05-08 [cit. 2010-11-27].

[18] HOFFSTEIN, Jeffrey, PIPHER, Jill, SILVERMAN, Joseph H. *NTRU: A Ring-based Public Key Cryptography* [online]. 1998-11-24 [cit. 2010-11-27].

[19] MARTELLO, Silvano, TOTH, Paolo. *Knapsack Problem*. Chichester : John Wiley & Sons, 1990. 296 s. ISBN 0471924202.

[20] CHALLA, Narasimham, PRADHAN, Jayaram. *Performance Analysis of Public key Cryptographic Systems RSA and NTRU* [online]. 2007-09-01 [cit. 2010-12-03].

[21] ELGAMAL, Taher. *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms* [online]. 1985-07-01 [cit. 2010-12-03].

[22] IBARRA, Oscar H., KIM, Chul E. *Fast Approximation Algorithms for the Knapsack and Sum of Subnet Problems* [online]. 1975-10-01 [cit. 2010-12-04].

[23] HOFFSTEIN, Jeff, LIEMAN, Daniel, PIPHER, Jill, SILVERMAN, Joseph H. *NTRU: A Public key Cryptosystem* [online]. 1999-08-11 [cit. 2010-12-08].

[24] OCHODKOVÁ, Eliška. *Přínos teorie eliptických křivek k řešení moderních kryptografických systémů* [online]. 2003-06-05 [cit. 2010-12-08].

[25] OKAMOTO, Tatsuaki, UCHIYAMA, Shigenori. *A New Public-Key Cryptosystem as Secure as Factoring* [online]. 2001-06-30 [cit. 2010-12-08].

[26] SCHIMDT-SAMOA, Kajta. *A New Rabin-type Trapdoor Permutation Equivalent to Factoring and Its Applications* [online]. 2005-08-05 [cit. 2010-12-10].

[27] AJTAI, Miklós. *The Shortest Vector Problem in L_2 is NP-hard for Randomized Reductions, Extended Abstract* [online]. 1998-07-05 [cit. 2010-12-10].

[28] HOOK, David. *Beginning Cryptography with Java*. Wrox Press, 2005. 480 s. ISBN 0764596330.

[29] Bouncycastle.org [online]. 2006 [cit. 2011-05-22]. The Legion of the Bouncy Castle. Dostupné z WWW: <www.bouncycastle.org>.

[30] Oracle.com [online]. 1993 [cit. 2011-05-22]. Java ™ Cryptography

Architecture (JCA) Reference Guide. Dostupné z WWW: <http://download.oracle.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html>.

[31] ČEČÁK, Ondřej. Linuxsoft.cz [online]. 2004 [cit. 2011-05-22]. Internetové bankovnictví v Linuxu. Dostupné z WWW: <http://www.linuxsoft.cz/article_list.php?id_kategory=173>.

[32] FlexiProvider [online]. 2001 [cit. 2011-05-22]. Dostupné z WWW: <http://www.flexiprovider.de/>.

[33] Oracle.com [online]. 2011 [cit. 2011-05-22]. Java SE Downloads. Dostupné z WWW: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>.

[34] Attachmate.com [online]. 2010 [cit. 2011-05-22]. Reflection for the Web and Java Virtual Machines. Dostupné z WWW: <http://support.attachmate.com/techdocs/1674.html>.

[35] BUKTU, Tim. Http://sourceforge.net [online]. 2010 [cit. 2011-05-22]. NTRU. Dostupné z WWW: <http://sourceforge.net/projects/ntru/>.

[36] HØNSI, Torstein. Highcharts.com [online]. 2011 [cit. 2011-05-22]. Highcharts JS. Dostupné z WWW: <http://www.highcharts.com/>.

[37] GIRY, Damien. Keylength.com [online]. 2011 [cit. 2011-05-22]. BlueKrypt | Cryptographic Key Length Recommendation. Dostupné z WWW: <http://www.keylength.com/en/>.

SEZNAM POUŽITÝCH ZKRATEK

3DES	Triple Data Encryption Starndard
AES	Advanced Encryption System
CA	Certifikační Autorita
CLR	Certificate Revocations List
DES	Data Encryption Standard
DLP	Discrete Logarithm Problem
FEAL	Fast data Encipherment Algorithm
FFT	Fast Fourier Transformation
GPG	GNU Privacy Guard
HTML	HyperText Markup Language
IDEA	International Data Encryption Algorithm
JCA	Java Cryptography Architecture
JCE	Java Cryptography Extensions
JPEG	Joint Photographic Experts Group
JVM	Java Virtual Machine
M-H	Merrkle-Hellman
MAK	Moderní Asymetrické Kryptosystémy
MD5	Message-Digest Algorithm
NIST	National Institute of Standards and Technology
NSA	National Security Agency
NSS	Network Security Services
PDF	Portable Document Format
PGP	Pretty Good Privacy
PKI	Public Key Infrastructure
PNG	Portable Network Graphics
RA	Registrační Autorita
RC5	Rivest Cipher
RSA	Rivest,Shamir,Adleman
SAFER	Secure and Fast Encryption Routine
SHA	Secure Hash Algorithm
SSL	Secure Sockets Layer
SVG	Scalable Vector Graphics

SEZNAM PŘÍLOH

PŘÍLOHA A: OBSAH PŘILOŽENÉHO DVD

Přiložený DVD disk obsahuje vytvořený testovací program, návod na jeho instalaci, návod na jeho používání a zpracování výstupních hodnot. Disk obsahuje výsledky testů prováděných v rámci diplomové práce.

- **JavaDoc/ -** obsahuje popis zdrojového kódu programu v HMTL souborech.
- **Návody/** – obsahuje návody v HTML k používání programu.
- **Policy_Soubory_Java/ –** soubory pro kryptografii v Javě.
  - local_policy.jar
  - US_export_policy.jar
- **Program/** – spouštěcí soubory programu.
  - Charts/ – složka s nástroji pro vykreslování grafů (NEMĚNIT).
  - lib/ – knihovny potřebné pro program.
  - CryptProgram.jar – spouštěcí soubor programu.
  - Instalace.txt – návod k instalaci programu.
  - README.txt
- **TestSoubor/** – výsledky testu provedeno v rámci diplomové práce.
- **ZdrojovéKódy/** – zdrojové kódy pro program NetBeans