PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

BAKALÁŘSKÁ PRÁCE

Asymetrické šifrování

2014 Pavel Urbášek

Anotace

Text popisuje principy asymetrické kryptografie, vlastnosti a rozdíly symetrických a asymetrických systémů. Podrobně rozebírá šifrovací funkce nejpoužívanějších asymetrických systémů, především systému RSA. Dále se zaměřuje na bezpečnost jednotlivých kryptosystémů, jejich možnosti a techniky k prolomení. Součástí práce je demonstrační software pro pochopení šifrovacích funkcí vybraných asymetrických kryptosystémů.

Děkuji vedoucímu bakalářské práce RNDr. Eduardu Bartlovi, Ph.D., za trpělivost a porozumění a za poskytnuté materiály, cenné rady i připomínky k tématu. Dále děkuji rodině za morální podporu při studiu. Především svému malému synovi Matějovi za dodanou energii potřebnou k dokončení studia.

Obsah

Seznam obrázků

Seznam tabulek

1. Úvod

Kryptografie má v dnešní moderních době informačních technologií nezastupitelné místo. Bez ní by nebyly možné online bankovní transakce a platby platebními kartami na internetu, přenosy citlivých osobních údajů v informačních systémech, provoz mobilních sítí GSM, síťová připojení prostřednictvím bezdrátových technologií, dále zabezpečená soukromá elektronická komunikace i zcela formální, s úřady a institucemi, využívající elektronický podpis. Obklopuje nás téměř na každém kroku aniž bychom si toho všimli. S její pomocí lze předávat zprávy elektronicky na dálku, bez obav ze zneužití obsahu zpráv a ztráty soukromí, aniž bychom museli udělat jediný krok z domova či kanceláře.

Kryptografie, ač na první pohled se zdá vědou složitou, je jednoduchá. Využívá sice složité matematické zákony, ale princip je přímočarý. Šifrovací systémy jsou z principu distribuce klíčů děleny pouze dva hlavní, symetrický a asymetrický. V praxi se dnes stále setkáváme s oběma systémy. Symetrické, navržené dříve, byly původně nástrojem jen pro vládní instituce a velké obchodní korporace. Až s příchodem asymetrických systémů se zabezpečení zpráv šifrováním přiblížilo i veřejnosti. Vzhledem ke svým rozdílným vlastnostem mají oba systémy v praxi stále své nezastupitelné místo. Velmi často se setkáváme s jejich vzájemnou kombinací.

V úvodních kapitolách textu jsou vyloženy kryptografické pojmy, popisovány základní principy šifrování a charakteristiky obou systémů. Následují matematické zákony na kterých je kryptografie založena a specifické aritmetické operace prováděné šifrovacími funkcemi a problematika velkých prvočísel pro použití v šifrování. Z asymetrických systémů jsou zde podrobně popsány šifry RSA, Merkle-Hellman a Diffie –Hellman, jejich výhody a bezpečnost. Další část je věnována kryptoanalýze. Zahrnuje základní způsoby prolamování šifer a možnosti prolomení u systémů RSA a Merkle-Hellman hrubou silou a faktorizací.

V poslední části práce je představen demonstrační program AsymmCrypt, který je součástí práce a je přiložen na disku CD-ROM. Tento software byl vytvořen pro účely studia tématu, zejména ověření správnosti matematických zákonů využitých v asymetrické kryptografii. Implementace šifrovacích algoritmů měla potvrdit otázku jednoduchosti principu šifrovaní pro nasazení v praxi. Pokusy s délkou klíče zase výpočetní možnosti k prolomení šifry RSA faktorizací modulu z veřejného klíče.

2. Kryptografie v dějinách

"Šifrování je metoda ochrany informací, které je datováno tisíce let zpět, je to umění antiky, které v dnešní době informační společnosti dostalo nový význam. V dějinách šifrování chránilo spojení, které bylo přenášeno přes nepřátelské území – obvykle za války nebo v rámci diplomatických služeb. Nejstarší šifry se objevují v prvních letech egyptské říše, okolo roku 2000 před Kristem, kdy byly pohřební zprávy vytesány v hieroglyfech do kamene.

V moderní době zvýšily potřebu šifrování vynález telegrafu a objev rádiových vln. Telegraf a rádio umožnily prakticky okamžitou komunikaci mezi různými druhy vojsk a nebo mezi armádou a jejím velením. Ale bez kryptografie mohou být všechna tato spojení snadno špióny monitorována, a to pouze se základní znalostí odposlechu. Bez toho, aby šifrování učinilo zprávy nečitelnými je potom žádná rádiová nebo telegrafická vojenská komunikace lepší než vysílání v otevřené řeči. Kódové knihy, eventuálně kryptografie samotná, umožnily využívání elektronické komunikace ve světových armádách. Kryptografie za svůj obrovský rozmach vděčí rozvoji vědy průběhu druhé světové války. První digitální počítače na světě byly vyrobeny pro dešifrování kódů. Spojenci v Británii pod vedením Alana Turinga vynaložili mimořádné úsilí při hledání kódu Enigma, který používali Němci. Mnoho lidí uznává, že toto úsilí vedlo ke zkrácení války."[7]

"Od dob druhé světové války se kryptografie a kryptoanalýza staly především matematickými úlohami. Díky široké dostupnosti dostatečně výkonných počítačů a celosvětové síti Internet coby média, se především kryptografie stala běžně využívaným nástrojem a není již nadále spolu s kryptoanalýzou výsadou jednotlivých vlád, nebo podobně velkých společností. Éru moderní kryptografie započal Claude Shannon, který je pravděpodobně otcem matematické kryptografie, díky práci, kterou vykonal v průběhu druhé světové války v oblasti zabezpečení komunikace.

V polovině 70. let se udály dva hlavní vývojové pokroky. Prvním bylo zveřejnění návrhu symetrické šifry DES (Data Encryption Standard). Navrhovaná šifra byla předložena výzkumné skupině v IBM pozvané Národním institutem standardů a technologií ve snaze vyvinout zabezpečené elektronické komunikační zařízení pro podniky, jako jsou banky a jiné velké finanční organizace. Po poradenství a modifikacích od NSA, která pracovala v zákulisí, byla šifra v roce 1977 přijata a zveřejněna. DES byla první veřejně přístupná šifra „požehnaná" vnitrostátním subjektem, jakým je NSA. Následující vývoj v roce 1976 byl ještě důležitější, jelikož změnil způsob, jakým kryptografické systémy fungovaly. Byla představena zcela nová metoda distribuce šifrovacích klíčů. To vedlo k takřka okamžité reakci v podobě vývoje nových algoritmů pro šifrování, které využívají asymetrickou kryptografii. Tento vývoj zlomil takřka monopolní postavení

vládních organizací na poli velmi kvalitního šifrování. Bylo to vůbec poprvé, kdy měly mimovládní organizace přístup k šifrování, které nemohl nikdo žádným jednoduchým způsobem prolomit, a to ani vláda, což takřka okamžitě vyvolalo značnou kontroverzi, která trvá dodnes. Až do roku 1996 byl export takových zařízení s klíčem delším než 40 bitů (příliš krátký, než aby představoval výraznou překážku pro zkušeného útočníka) z území Spojených států amerických striktně omezován. V roce 2004 volal bývalý ředitel FBI Louis Freeh, který svědčil před Národní komisí pro teroristické útoky na Spojené státy, po nových zákonech proti veřejnému užívání šifrování.

Jeden z nejvýznamnějších zastánců silného šifrování pro veřejnost byl Phil Zimmermann. Napsal a v roce 1991 vydal vysoce kvalitní šifrovací program PGP (Pretty Good Privacy, přeloženo jako Celkem slušné soukromí). Ve chvíli, kdy se cítil ohrožen legislativou, která zvažovala nutnost zavedení zadních vrátek ve všech kryptografických produktech vyvíjených ve Spojených státech, vydal freeware verzi PGP. Krátce po vydání PGP ve Spojených státech bylo vydáno také pro zbytek světa a Ministerstvo spravedlnosti Spojených států amerických odstartovalo dlouhé vyšetřování Zimmermanna pro údajné porušení restrikcí pro export kryptografických systémů. Případ byl nakonec uzavřen a freeware distribuce PGP se tak šířila po celém světě, až se PGP nakonec stalo otevřeným internetovým standardem.

Zatímco jsou moderní šifry, jako například AES a další, kvalitnější asymetrické šifry, považovány za nepřekonatelné, objevují se i nadále kryptografické systémy, které trpí špatným návrhem, popřípadě implementací. V posledních letech tak došlo k několika důležitým kryptoanalytickým průlomům do nasazených systémů, například prolomení prvního algoritmu pro šifrování sítí Wi-Fi, ochrany Content Scrambling System pro DVD a šifer A5/1 a A5/2 pro mobilní sítě GSM. Všechny tyto šifry byly symetrické. Zatím nebylo dokázáno, že by matematický základ kryptografie s veřejným klíčem byl neprolomitelný, v budoucnu by tak mohla nějaká matematická analýza vést k označení systémů založených na principu veřejného klíče za nebezpečné. Zatím však stále dochází k navyšování doporučované délky klíče a tím stoupá výpočetní výkon nutný k prolomení šifry."[12]

3. Jak pracuje šifrování

Šifrování využívá oblast matematiky, nazývanou kryptografie. Slouží k zabezpečení soukromí zpráv nebo dat. Cílem kryptografie je znemožnit, aby bylo možné získat ze šifrového textu původní otevřený text bez znalosti šifrovacího klíče. Používá komplikovanou matematiku s jednoduchými zásadami.

3.1. Terminologie

- **Kryptografie** - věda o utajování zpráv do podoby, která je čitelná jen se speciální znalostí.

  Mimo utajení zpráv zajišťuje také:

  - autentičnost - ověření původu zprávy, pravost
  - integritu - nezměnitelnost zprávy během přenosu
  - neodmítnutelnost - nelze popřít odeslání zprávy

- **Kryptoanalýza** - věda zabývající se metodami získávání obsahu šifrovaných zpráv bez možnosti použít tajné informace, s kterými byla zpráva zašifrována.

- **Kryptologie** - zahrnuje kryptografii a kryptoanalýzu.

další kryptografické pojmy:

otevřený text - zpráva určená k odeslání

šifrování - proces úpravy otevřeného textu na nesrozumitelný

šifrový text, kryptogram - aplikace šifrování na otevřený text

dešifrování - zpětný proces převodu kryptogramu na otevřený text

šifrovací algoritmus - matematická funkce provádějící šifrování

dešifrovací algoritmus - matematická funkce provádějící dešifrování

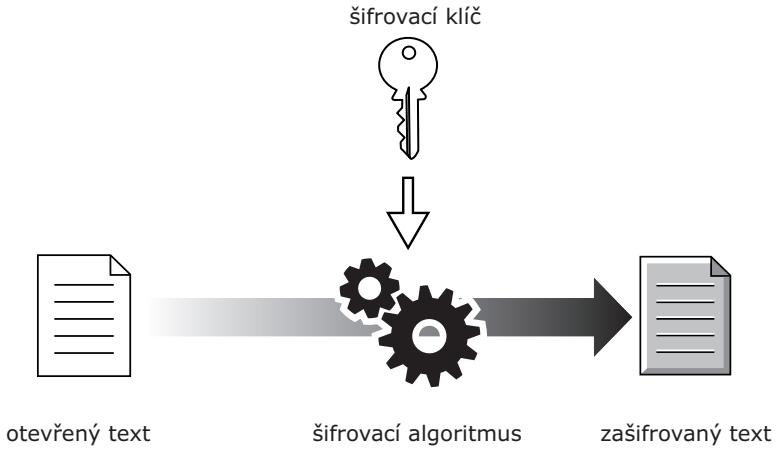
šifra - společné označení pro šifrovací a dešifrovací funkci

komunikační kanál - telefon, internet, LAN apod.

3.2. Hlavní části šifrovacího systému

Každý šifrovací systém obsahuje čtyři základní části:

1. zpráva (otevřený text)
2. šifrovací algoritmus
3. šifrovací klíč
4. šifrový (zašifrovaný) text



Obrázek 1. Jednoduchý šifrovací systém

3.3. Klíč a délka klíče

Z uživatelského pohledu jsou kryptografické klíče velmi podobné heslům používaných k přístupu k počítačovým systémům. Pouze správným klíčem se zpřístupní obsah zprávy. Stejně jako počítačové systémy používají hesla různých délek, také šifrovací systémy používají různé délky klíčů. Stejně jako u hesel, čím je klíč delší, tím je šifrovací systém obecně bezpečnější.

Délka hesla se délka obvykle měří ve znacích. U kryptografického klíče se většinou udává v bitech (binárních číslicích). Čím vyšší je počet bitů, který šifrovací algoritmus pro své klíče umožňuje, tím vyšší je počet všech možných klíčů a tím více se algoritmus stává bezpečnější. Toto tvrzení však nevede vždy k závěru, že například algoritmus se 112-ti bitovým klíčem je bezpečnější, než algoritmus s 56-ti bitovým klíčem. Mohou se vyskytnou ještě další ovlivňující faktory pro konečné určení vlastní bezpečnosti šifrování.

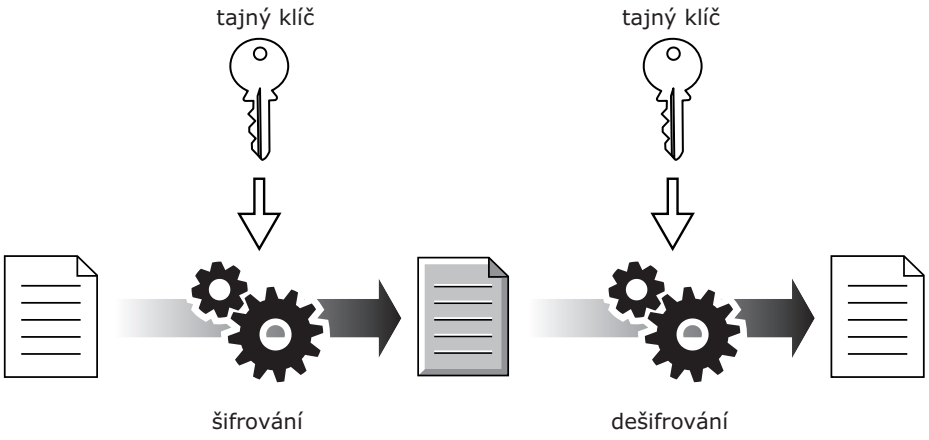
Třída algoritmu	Šifra	Bezpečnostní úroveň klíče | délka klíče			
		80 bit	128 bit	192 bit	256 bit
Faktorizace čísel	RSA	1024 bit	3072 bit	7680 bit	15360 bit
Diskrétní log.	Diffie-Hellman	1024 bit	3072 bit	7680 bit	15360 bit
Symetrický klíč	AES, 3DES	80 bit	128 bit	192 bit	256 bit

Tabulka 1. Délka klíče a jeho bezpečnost.[2]

4. Kryptografické systémy

4.1. Symetrický systém – šifrování se soukromým klíčem

Název vychází ze skutečnosti, že odesílatel i příjemce sdílejí stejný klíč, který musí být udržen v tajnosti. Pokud chceme s někým tajně komunikovat, musíme nejprve této osobě bezpečně doručit šifrovací klíč, který hodláme použít. Tento postup se nazývá distribuce klíče a jeho provedení je obtížné. Protože je tento klíč doslova klíčem k bezpečnosti, musíme najít správný postup, jak klíč doručit bez toho, aby se dostal do špatných rukou. Samotné šifrování a dešifrování zpráv probíhá pak pomocí tohoto klíče.



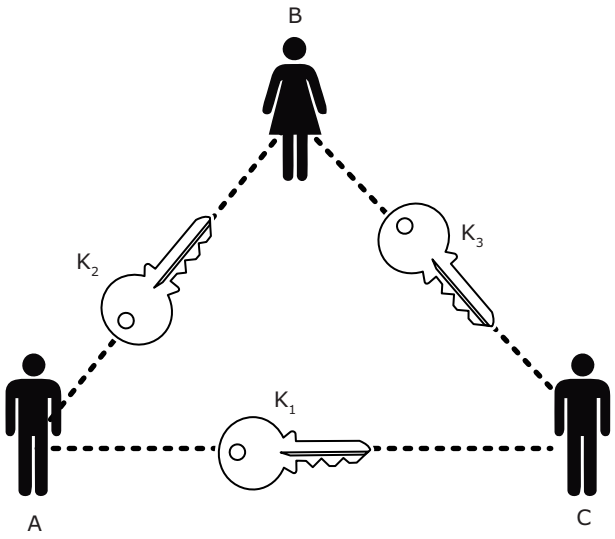
Obrázek 2. Symetrické šifrování

Mezi symetrické šifry patří například DES, 3DES, RC2, RC4 a IDEA.

Nevýhody symetrického šifrování

Systém se soukromým klíčem byl široce využíván v armádě, výzvědných službách a v oblasti financí a obchodu. Vždy však trpěl některými problémy. Většina těchto problémů souvisí s klíčem.

1. **Celkový počet klíčů** - předpokládejme že 3 osoby budou chtít spolu komunikovat pomocí systému soukromého klíče. Budou tedy potřebovat 3 klíče. Pokud se do systému zapojí další 2 osoby, situace se zkomplikuje, bude potřeba již 10 klíčů.



Obrázek 3. Bezpečný komunikační systém pro 3 osoby

Obecně lze říci, že počet lidí n, kteří chtějí bezpečně komunikovat pomocí tohoto systému bude potřebovat

$$\frac{n * (n-1)}{2} \ \textit{klíčů}.$$

2. **Distribuce klíče** - dalším problémem systému se soukromými klíči je to, že nelze někomu odeslat tajnou zprávu aniž bychom již předtím této osobě předali tajný klíč. Tudíž nelze vzájemně tajně komunikovat bez předchozí dohody. Jedním způsobem, který tento problém řeší je centrum distribuce klíčů (KDC – Key Distribution Center)

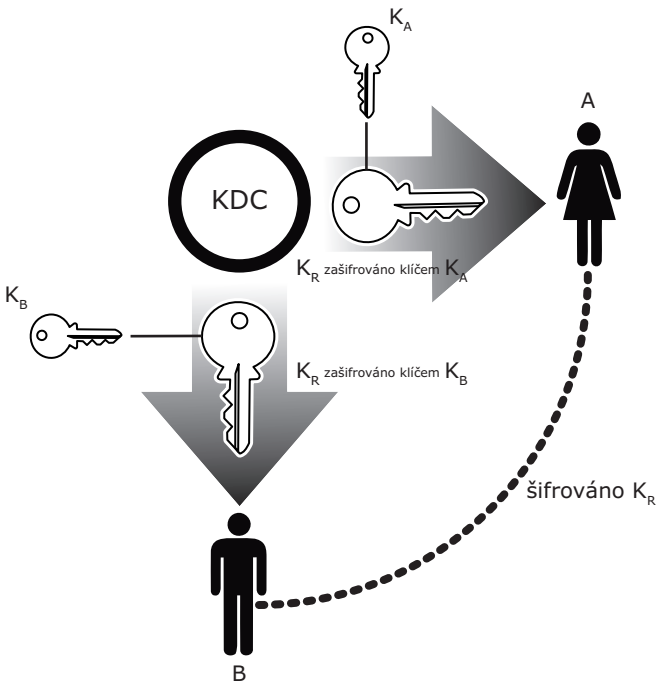
”Důsledkem těchto problémů je, že soukromé klíče jsou pro obyčejné lidi v každodenní práci nepraktické. Pokud by existovalo jen symetrické šifrování, byla by kryptografie nástrojem vlád a velkých společností a soukromé osoby by zůstaly odkázání na nechráněnou komunikaci.”[7]

4.1.1. Správa a distribuce klíčů (KDC)

Instituce nebo systémová služba, která připravuje šifrovací klíče a dodává je kterékoliv individuální dvojici osob, které spolu chtějí takto komunikovat. Pro správnou funkci musí mít uživatelé služby v KDC svůj vlastní soukromý klíč.

Postup získání klíče

Když chce *A* komunikovat s *B*, požádá KDC o klíč. Tam se připraví klíč relace pro *A* - *B*. Vygenerovaný klíč relace se zašifruje soukromým klíčem konkrétního příjemce klíče a odešle. [1]

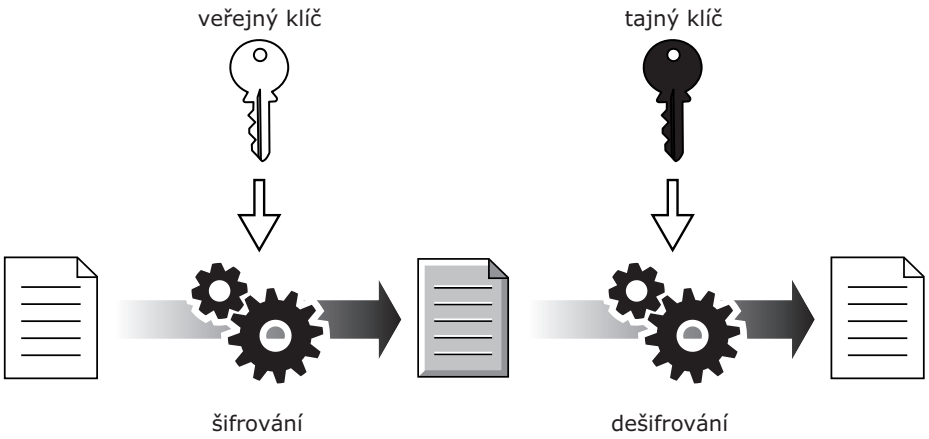


Obrázek 4. Klíč relace

[1] *Tento systém roky používala vláda Spojených států pro distribuci klíčů vojenským a výzvědným službám. Klíče připravoval Úřad pro národní bezpečnost (NSA), který je také distribuoval.*

4.2. Asymetrický systém - šifrování s veřejným klíčem

V symetrických systémech je jeden klíč sdílen dvěma osobami. Obě používají stejný klíč šifrování i dešifrování. Místo toho se v asymetrických systémech matematicky generuje dvojice klíčů, veřejný a soukromý. Veřejný klíč může znát kdokoliv, soukromý zůstane utajen. Pro zašifrování zprávy se použije veřejný klíč příjemce. Ten pak zprávu rozšifruje svým soukromým (tajným) klíčem.



Obrázek 5. Asymetrické šifrování

4.2.1. Systémy veřejných klíčů

První systém veřejného klíče popsali v roce 1976 Whitfield Diffie a Martin Hellman. Byly navrženy odlišné systémy. Diffieho a Hellmana a Ronalda Rivesta, Adi Shamira a Lena Alemana (RSA). Další systém vyvinuli Ralph Merkle a Martin Hellman.

- Merkle-Hellman[31] - systém zavazadlového algoritmu (batohu). Byl postaven podle matematické hry „problém batohu“. Z určitého souboru položek o různé váze je třeba naplnit batoh tak, aby celková hmotnost dosáhla určité hodnoty. Pro slabiny v algoritmu je dnes pro praktické účely nepoužitelný.

- RSA[34] - systém založený na obtížnosti rozkladu velkých čísel na prvočinitele. Využívá modulární umocňování. Může být používán i pro systém digitálního podpisu. RSA je jednou z nejsilnějších asymetrických šifer, které jsou známy.

- Diffie-Hellman[40] - exponciální výměna klíče. Systém pro používání dvěma aktivními účastníky konverzace, kdy oba účastníci začnou svým tajným klíčem a potom po výměně informací založených na těchto klíčích se derivuje třetí klíč, nazývaný klíč relace.

Další asymetrické systémy jsou například ElGamal, Rabin a LUC.

4.2.2. Digitální podpis

Digitální podpis lze chápat jako razítko na konci elektronického dokumentu a slouží k ověření pravosti zprávy. Bez digitálního podpisu neexistuje způsob, kterým by to bylo možno udělat. Velmi snadno lze totiž padělat jméno odesílatele a elektronickou hlavičku elektronického dopisu. Je tedy třeba upravit zprávu tak, aby nemohla být změněna. Systém používá asymetrické šifrování. Zabrání tím změnit kritické informace ve zprávě a vydávat se tak za autora zprávy nebo padělání zprávy. Přítomnost digitálního podpisu nestačí, je potřeba jeho matematické ověření, které potvrdí pravost zprávy.

Podepsání a ověření pravosti zprávy:

1. Odesílatel zprávy vytvoří matematickou funkcí její otisk (hash), který pak zašifruje svým soukromým klíčem a výsledek připojí na konec zprávy jako podpis.

2. Příjemce vytvoří ze zprávy také otisk a připojený podpis dešifruje veřejným klíčem[1] odesílatele. Výsledek dešifrování porovná s otiskem zprávy. Shodují-li se, je zpráva pravá.

4.2.3. Srovnání asymetrických a symetrických systémů

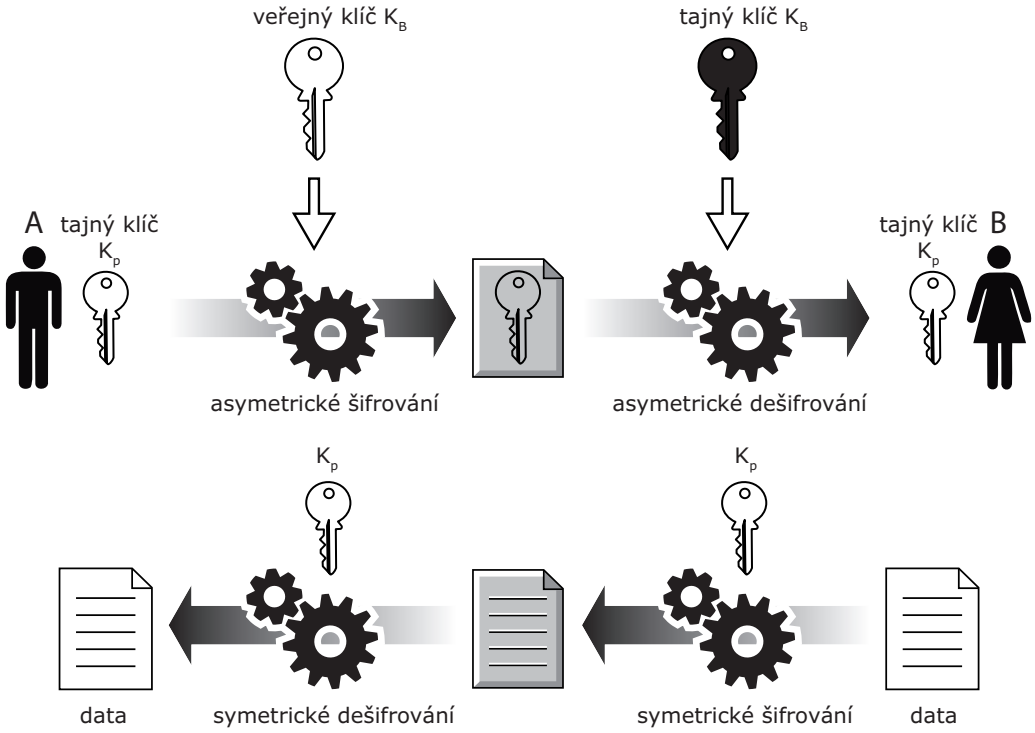
Asymetrické systémy mají před systémy se soukromými klíči několik výhod. Nejdůležitější z nich, je zveřejnění šifrovacího klíče bez obav z jeho zneužití. Kdokoliv může s jeho pomocí poslat zašifrovanou zprávu aniž by ji mohl někdo jiný dešifrovat, mimo vlastníka tohoto klíče, protože ten jediný má k dispozici klíč soukromý. Pravost veřejných klíčů řeší tzv. certifikáty a certifikační autority (CA). Další výhodou je, že není třeba tento klíč s postupem času měnit, jako se to doporučuje u systémů se soukromými klíči. Některé asymetrické systémy lze využít i k digitálnímu podpisu zpráv. Tato možnost u symetrických systémů není. Nevýhodou je užití několinásobně delších klíčů než u symetrických systémů a rychlost šifrování, která je u RSA řádově 1000-krát pomaleší než u DES.

[1] Užití klíčů je v tomto případě opačné než u klasického šifrování.

4.3. Hybridní systém

Výhody obou předchozích systémů lze sloučit a vytvořit kombinovaný systém. Pro šifrování velkých objemů dat, kdy by šifrování veřejným klíčem bylo pomalé, lze využít rychlost kódování symetrického systému a snadnou dostupnost veřejného klíče asymetrického systému.

Systémem s veřejným klíčem (výrazně pomalejší) se zašifruje jen soukromý klíč K_p k symetrickému systému (rychlý), kterým jsou šifrovány samotná data. Tímto způsobem se zajistí bezpečnost a zvýší se rychlost výpočtu šifrového textu, neboť velké objemy dat (řádově MB-GB) jsou zpracovány mnohonásobně rychlejším systémem a pomalejším je šifrován jen samotný tajný klíč, jehož velikost (řádově desítky B) je v tomto případě zanedbatelná.



Obrázek 6. Hybridní šifrování

Mezi hybridní šifry patří například SSL/TSL nebo PGP.

5. Transformace textu

Velká část zpráv určených k zašifrování je v textovém formátu. Pro aplikaci matematické šifrovací funkce je třeba vhodně upravit a převést jednotlivé znaky nebo celá slova na číslice. [1]

Nejjednodušší způsob jak to provést, je nahradit znaky odpovídající hodnotou dle ASCII. U velmi dlouhých zpráv je dobré celý text předtím zkomprimovat slovníkovým algoritmem. Pro některé šifrovací algoritmy je výhodnější dekadická číselná hodnota, například RSA[34], pro jiné binární reprezentace, například Merkle-Hellman[31]. Výsledky transformace znaků seskupíme do bloků. Pokud je poslední blok neúplný, doplníme jej nulami na potřebnou délku. Velikost bloku určuje maximální hodnota, kterou lze daným šifrovacím systémem zakódovat. Menší počet delších bloků výrazně urychlí proces šifrovací funkce.

V textu se často opakují některé znaky nebo celá slova. Abychom ztížili použít narušiteli k rozluštění kryptogramu frekvenční analýzu[43] provedeme další úpravy reprezentace znaků. Ke každé dekadické hodnotě znaku přidáme zprava jednu číslici. Začneme nulou, pro každý další stejný znak zvýšíme hodnotu o 1. Takto budou stejné znaky zakódovány pokaždé jinak. Při zpětné transformaci po dešifrování stačí jen první číslici zprava oříznout.

Příklad transformace slova ”matematika” s doplněním číslicí zprava:

znak	**m**	**a**	**t**	**e**	**m**	**a**	**t**	**i**	**k**	**a**
ASCII	109	97	116	101	109	97	116	105	107	97
doplněno	1090	970	1160	1010	1091	971	1161	1050	1070	972

Tabulka 2. Transformace s doplněním

Obdobnou funkci zajistí dělení do bloků délky větších než jsou násobky reprezentace znaků. Lze použít i kombinaci komprimace, doplnění a bloků.

Příklad transformacez předchozího příkladu s doplněním číslicí zprava a rozdělení do bloků délky 8, poslední blok doplněn nulami.

bloky	1090 970 1	160 1010 1	091 971 11	61 1050 10	70 1092 *00*

Tabulka 3. Transformace s doplněním a bloky

[1] Pravidla transformací textu pro asymetrické šifrovací systémy jsou specifikovány standardem PKCS (Public Key Cryptographic Standards). Pro RSA je to PKCS 1.

6. Modulární aritmetika

Modulární aritmetika provádí operace modulo s kladnými celými čísly. Jestliže a je celé číslo a n je kladné celé číslo, pak definujeme $a\ (mod\ n)$ jako zbytek po dělení čísla a číslem n, n se nazývá modul.

Pro každé a píšeme: $a = [a/n] * n + a\ (mod\ n)$

6.1. Kongruence

Celé číslo a je *kongruentní modulo* n s celým číslem b tehdy, když je rozdíl $a - b$ dělitelný číslem n.

$$a \equiv b\ (mod\ n)$$

Relace kongruence je ekvivalencí, protože je reflexivní, symetrická a tranzitivní. Třída ekvivalence celých čísel a je množina všech celých čísel kongruentní s $a\ mod\ n$. Pro dané n relace kongruence $a\ mod\ n$ dělí množinu $\mathbb{Z}$ na třídy ekvivalence, tzv. zbytkové třídy modulo n.

Je-li $a = qn + r$, kde $0 \leq r < n$, pak $a \equiv r\ (mod\ n)$.

Každé celé číslo a je jednoznačně kongruentní $mod\ n$ s kladným celým číslem z intervalu 0 až $n - 1$. Čísla a a r patří do téže třídy ekvivalence a r můžeme použít jako reprezentanta této třídy ekvivalence. Celá čísla modulo n jsou množinou tříd ekvivalence celých čísel $\{0, 1, \ldots, n - 1\}$, označovanou $\mathbb{Z}_n$. Operace sčítání a násobení v $\mathbb{Z}_n$ jsou prováděny modulo n.

Pro $n > 0$ a každé celé číslo a definujeme $[a]_n = \{x \mid x \equiv a\ mod\ n\}$ jako množinu celých čísel kongruentních a modulo n a nazýváme ji množinou zbytkových tříd modulo n.

Číslo $a \in \mathbb{Z}_n$. Multiplikativní inverzní prvek[1]. a^{-1} k prvku a modulo n je číslo $x \in \mathbb{Z}_n$ takové, že platí:

$$a + x \equiv 1\ (mod\ n)$$

Inverzní prvek a^{-1} existuje právě tehdy, když $gcd(a, n) = 1$.

[1]Pro nalezení multiplikativního inverzního prvku použijeme rozšířený Euklidův algoritmus (EEA)[24]

6.1.1. Čínská věta o zbytcích

(Chinese Reminder Theorem - CRT)

Nechť $m_1, \ldots, m_k$ jsou po dvou nesoudělná celá čísla, pro všechna i, j, platí:

$$(1 \leq i < j < k) \mid gcd(m_i, m_j) = 1, a_1, \ldots, \ a_k \in \mathbb{Z}$$

Potom soustava kongruencí

$$\begin{aligned} x &\equiv a_1 \ (mod \ m_1) \\ x &\equiv a_2 \ (mod \ m_2) \\ &\ldots\ldots\ldots \\ x &\equiv a_k \ (mod \ m_k) \end{aligned}$$

má řešení.

Všechna řešení této soustavy jsou navzájem kongruentní modulo:

$$M = m_1 * m_2 * \ldots * m_k$$

Čínská věta říká, že libovolné číslo z množiny $\{0, 1, \ldots, M\}$, tj, zbytkovou třídu modulo M lze jednoznačně reprezentovat pomocí zbytkových tříd modulo $m_1, m_2, \ldots, m_k$. Z toho vyplývá:

Nechť p, q jsou vzájemně nesoudělná celá čísla, pak pro libovolné $x \in \mathbb{Z}_{p*q}$

$$x \equiv a \ (mod \ p * q) \Leftrightarrow x \equiv a \ (mod \ p), \ x \equiv a \ (mod \ q)$$

6.2. Fermatova a Eulerova věta

6.2.1. Malá Fermatova věta

Pro libovolné prvočíslo p a každé celé číslo a platí:

$$a^p \equiv a(\ mod \ p)$$

Pro nesoudělná a a p, $(a, \ p) = 1$, dále platí:

$$a^{p-1} \equiv 1(\ mod \ p)$$

6.2.2. Eulerova funkce $\varphi(m)$

Pro přirozené číslo m udává počet čísel menších než m a nesoudělných m. Pro p a q různá prvočísla pro Eulerovu fukci φ platí:

$$\begin{aligned} \varphi(p) &= p-1 \\ \varphi(p^i) &= (p-1)p^{i-1} \\ \varphi(p*q) &= (p-1)(q-1) \end{aligned}$$

Jsou-li $p_1, \ldots, p_k$ různá prvočísla, platí:

$$\varphi(p_1^{i_1} \ \ldots \ p_k^{i_k}) = (p_1-1)p_1^{i_1-1} \ \ldots \ (p-1)p_k^{i_k-1}$$

Jsou-li $m,\ n$ nesoudělná přirozená čísla, platí:

$$\varphi(m*n) = \varphi(m)*\varphi(n)$$

6.2.3. Eulerova věta

Pro a a m nesoudělná přirozená čísla platí:

$$a^{\varphi(m)} \equiv 1 \ (mod\ m)$$[1]

Jestliže p je prvočíslo, pak pro každé přirozené číslo b nesoudělné s p platí:

$$b^{p-1}\ mod\ p = 1$$

Pro i, j, m přirozená čísla, kdy $(i,\ \varphi(m)) = 1$ a $(i*j)\ mod\ \varphi(m) = 1$, platí:

$$(x^i)^j\ mod\ m = x\ mod\ m$$

[1] Speciálním případem Eulerovy věty je dříve Fermatem dokázaná malá Fermatova věta.

6.3. Problém diskrétního logaritmu

Jsou-li dána čísla i, a a prvočíslo p, pak není obtížné vypočítat $b \equiv a^i \ (mod\ p)$. Je-li však dáno prvočíslo p, generátor a cyklické grupy $\mathbb{Z}_p^*$ a $b \in \mathbb{Z}_p^*$, je výpočetně obtížné nálézt i tak, že $b \equiv a^i \ (mod\ p), i \in \langle 1, p-1 \rangle$, tedy vyřešit *problém diskrétního logaritmu.*

Koncept diskrétnío logaritmu nebo-li indexu celého čísla modulo n poprvé představil Gauss. Má-li celé číslo n primitivní odmocninu a *modulo* n, pak podmnožina $\{a, a^2, \ldots, a^{\phi(n)}\}$ tvoří redukovaný systém zbytků modulo n. Přitom a je generátor cyklické grupy redukovaných zbytků modulo n. Je-li $gcd(b, n) = 1$, pak b můžeme vyjádřit ve tvaru $b \equiv a^i \ (mod\ n)$ pro $1 \leq i \leq \phi(n)$.

Nechť a je primitivní odmocnina celého čísla $n > 1$. Jestliže $gcd(b, n) = 1$, pak nejmenší kladné celé číslo i pro které platí $b \equiv a^i \ (mod\ p)$, kde $1 \leq i \leq \phi(n)$, se nazývá *diskrétní logaritmus* čísla b o základu $a \ (mod\ n)$.

Jestliže a je primitivní odmocnina modulo n, pak pro celá čísla s a t platí

$$a^s \equiv a^t \ (mod\ n)$$

právě tehdy když

$$s \equiv t \ (mod\ \phi(n))$$

Vlastnosti diskrétního logaritmu jsou podobné vlastnostem logaritmu reálného čísla o daném základu. Vyjádříme-li li disktrétní logartmus i jako $log_a\ b$, dostaneme $b \equiv a^{log_a\ b} \ (mod\ n)$ pro $1 \leq i \leq \phi(n)$.

Jeden z nejrychlejších algoritmů pro nalezení diskrétního logaritmu je obecné síto číselného tělesa (General Number Field Sieve). Má subexponeciální asymptotickou složitost $\Theta(e^{((ln\ p)^{1/3} ln(ln\ p))^{2/3}})$ a není pro velká čísla řešitelný. Na problému je založena celá řada kryptografických algoritmů, například Diffie-Hellman[40], ElGamal nebo metoda eliptických křivek.

7. Algoritmy pro počítání s *(mod n)*

7.1. Rozšířený Euklidův algoritmus

Euklidův algoritmus pro hledání největšího společného dělitele dvou čísel (gcd) vytváří ze dvou zadaných čísel $a_0 > a_1$ ostře klesající posloupnost $(a_i, i = 0, 1, \ldots, n+1)$, jejíž poslední člen $a_{n+1} = 0$ a předposlední $a_n = gcd(a_0, a_1)$. Přitom $(i+2)$ člen je právě zbytek po dělení i-tého členu členem $(i+1)$. Postupným dosazováním předchozích dvou členů posloupnosti za $(i+2)$ člen zároveň umožňuje vyjádřit předposlední člen jako celočíselnou lineární kombinaci prvních dvou členů.

Bezoutova rovnost

Pro dvě celá čísla a, b existují celá čísla x, y tak, že platí:

$$x * a + y * b = gcd(a, b)$$

Pokud $n = b$ je prvočíslo, $0 < a < n$, pak $gcd(a, n) = 1$.

Potom existuje celé číslo x tak, že:

$$(x * a)\ (mod\ n) = gcd(a, n) = 1, \quad x \in \langle 0, 1, \ldots, n-1 \rangle$$

Pokud platí, že $gcd(a, n) = 1$, můžeme Euklidův algoritmus použít pro nalezení inverzního prvku. Tato varianta se nazývá rozšířený Euklidův algoritmus (Extended Euclidean algorithm - EEA). Složitost algoritmu je $\Theta((log_2 n)^2)$ bitových operací. [9]

7.2. Gaussův algoritmus

Pro řešení x soustavy kongruencí dle CRT[21] můžeme využít Gaussův algoritmus a soustavu počítat jako:

$$x \equiv \sum_{i=1}^{k} a_i N_i L_i\ (mod\ M)$$

kde $M = m_1 * m_2 * \ldots * m_k, N_i = M/m_i$ a $L_i = N_i^{-1}\ (mod\ n)$.

Složitost algoritmu je $\Theta((log\ n)^2)$.

7.3. Modulární umocňování

je umocňování v rámci modulární aritmetiky. Výsledkem je hodnota, která vznikne umocněním základu a na exponent k modulo n. Přímočarý postup je nejprve spočítat mocninu a^k a pak dělit n. Nevýhodou však je, že mezivýsledky umocňování exponenciálně rostou a jednotlivá násobení jsou i s použitím algoritmů pro počítání s dlouhými čísly časově a paměťově náročné.

Řešením je provádět dělení se zbytkem průběžně během umocňování. Využijeme k tomu algoritmus pro rychlé modulární umocňování (Repeat Square and Multiply Algoritm - RSMA). Využívá binární umocňování zprava doleva, které výrazně snižuje počet operací násobení a nemá velké paměťové nároky.

Pro modulární násobení platí:

$$[(a(mod\ n)) * (b(mod\ n)]\ (mod\ n) = (a * b)\ (mod\ n)$$

Toho využijeme a například $88^7\ (mod\ 187)$ spočítáme efektivněji jako:

$$[(88^4\ (mod\ 187)) * (88^2\ (mod\ 187)) * (88^1)\ (mod\ 187))]\ (mod\ 187)$$

To znamená, že můžeme redukovat modulo n jednotlivé mezivýsledky.

Princip RSMA

je založen na tom, že pro binární reprezentaci čísla k:

$$\sum_{i=0}^{t} k_i 2^i,\ \ k_i \in \{0, 1\}$$

platí:

$$\Pi_{i=0}^{t}\ a_i^{k_i 2^i} = (a^{2^0})^{k_0} * (a^{2^1})^{k_1} * \ldots * (a^{2^t})^{k_t}$$

Vstup algoritmu RSMA: $a \in \mathbb{Z}_n,\ k \in \mathbb{Z}, 0 \leq k < n,\ k = \sum_{i=0}^{t} k_i 2^i$.

Výstup algoritmu RSMA: $a^k\ (mod\ n)$.

Složitost algortitmu je $\Theta((log_2\ n)^2)$ bitových operací.[9]

8. Prvočísla

8.1. Hledání velkých prvočísel

Prvočísla mají v asymetrických šifrovacích systémem klíčovou roli. Jsou hlavní součástí výpočtu šifrovacího klíče. K zajištění bezpečnosti systému je třeba vytvořit dostatečně dlouhé klíče. Z tohoto vyplývá i potřeba hledat prvočísla odpovídající velikosti. V současnosti jsou vyžadována přibližně více jak 100-ciferná prvočísla. Najít rychle takto velká prvočísla není snadné a klasické metody hledání prvočísel jako je například *Eratosthenovo síto* jsou nepoužitelné.

V teorii čísel byla již před staletími zkoumána funkce:

$$\pi(n) = počet\ prvočísel\ menších\ než\ n$$

Základní odhad této funkce podal Gauss:

$$\pi(n) \sim \frac{n}{ln\ n}$$

Z tohoto odhadu vyplývá, že relativní frekvence výskytu prvočísla v okolí čísla n je přibližně $\frac{1}{ln\ n}$ a průměrná vzdálenost $A(n)$ mezi dvěma prvočísly v okolí čísla n je $A(n) \sim ln\ n$.

$$A(10^{100}) = ln\ 10^{100}\ \sim\ 230$$

100-ciferná prvočísla za sebou tedy následují průměrně po 230 číslech. Je jich $\pi(10^{100}) - \pi(10^{99})$, tj. přibližně $3,9 * 10^{97}$. Z tohoto vyplývá, že lze velká prvočísla efektivně hledat i náhodně. V průměru bude stačit projít po sobě 115 čísel počínaje náhodným.[10]

8.2. Generování prvočísel

Pro získání velkého prvočísla vytvoříme číslo s požadovaným počtem cifer. Číslo generujeme náhodně po jednotlivých cifrách. Sudá čísla a čísla končící cifrou 5 zahodíme. Dělitelnost 3 ověříme ciferným součtem, zda je dělitelný 3. Dále vyzkoušíme zda není dělitelné malými prvočísly. Nakonec číslo ověříme testem na prvočíselnost. Pokud test označí číslo za složené, postup opakujeme.

8.3. Testy prvočíselnosti

Kategorie testů prvočíselnosti

- Pravděpodobnostní testy (testy složenosti) určí číslo které není prvočíslem spolehlivě jako číslo složené. Nedokážou však, že číslo označené jako prvočíslo je skutečně prvočíslem. Test se proto provádí opakovaně. Při proběhnutí testu $t - krát$ pro složené číslo n bude pravděpodobnost, že číslo n bude označeno jako prvočíslo ve všech t pokusech nejvýše $(1/2)^t$.

- Neomylné (deterministické) testy určí prokazatelně n jako prvočíslo.Tyto testy jsou ale výpočtově náročnější než pravděpodobnostní testy.

8.3.1. Zkušební dělení

Nejednodušším testem na prvočíselnost je algoritmus zkušebního dělení (Trial division). Je to neomylný test prvočíselnosti. Je však efektivní pouze pro malé hodnoty testovaných čísel, maximálně 24 cifer.

Algoritmus je založen na pokusném dělení testovaného čísla n postupně všemi čísly m tak, že $1 < n < m$. Pokud je pro některé m výsledek dělení beze zbytku, číslo není prvočíslem.

Vzhledem k velkému počtu zbytečných dělení lze algoritmus vylepšit tak, že dělíme pouze čísly $m \leq \sqrt{n}$ a vynecháním sudých čísel. Lze-li n vyjádřit jako součin dvou nebo více přirozených čísel, pak jedno z nich musí být menší nebo rovno $\sqrt{n}$. V případě, že máme seznam prvočísel v rozsahu $\langle 1, \sqrt{n} \rangle$ snížíme počet operací na minimum. Složitost algoritmu je řádově $\Theta(2^b)$ bitových operací, kde b je počet bitů čísla n.

8.3.2. Fermatův

Pravděpodobnostní test. Jak již z názvu vyplývá, je test odvozen od Malé Fermatovy věty[21]. Není ale typickým pravděpodobnostním testem prvočíselnosti, protože nedokáže odhalit speciální složená čísla - Carmichaelova[28].

Pro číslo n, o kterém předpokládáme, že je prvočíslem, kontrolujeme pro všechna přirozená čísla $a \in \langle 1, n-1 \rangle$, zda věta platí. V praxi by však otestování všech čísel v tomto intervalu trvalo příliš dlouho. Proto testujeme pouze určitý počet t. Pokud nalezneme jakékoliv číslo a pro které kongruence neplatí, je a důkaz složenosti. Pokud kongruence platí, ještě není dokázáno, že n je prvočíslo.

Test provedeme znovu s jiným a $t-krát$ pro t různých čísel a.

- Nechť n je liché složené číslo, a je libovolné celé číslo z intervalu $\langle 1, n-1\rangle$ a je splněna podmínka $a^{p-1} \not\equiv 1\ (mod\ n)$. Potom a se nazývá *Fermatův svědek* složenosti p.

- Nechť n je prvočíslo, a je libovolné celé číslo z intervalu $\langle 1, n-1\rangle$ a je splněna podmínka $a^{p-1} \equiv 1\ (mod\ n)$. Potom a se nazývá *Fermatův lhář* prvočíselnosti p.

Složitost algoritmu je $\Theta(t * log\ n)$.

Carmichaelovo číslo

je složené číslo n právě tehdy, když pro všechna celá čísla a nesoudělná s n, $gcd(a, n) = 1$, je splněna podmínka $a^{n-1} \equiv 1\ (mod\ n)$. Existuje nekonečně mnoho Carmichaelových čísel. Tyto čísla jsou testem správně rozeznána jako složená jen pokud platí $gcd(a, p) > 1$.[9]

Zesílením testu může být použití *Eulerova kritéria*, kdy pro prvočíslo p platí $a^{\frac{p-1}{2}} \equiv (\frac{a}{p})(modq\ p)$, pro všechna celá čísla a nesoudělná s p. $(\frac{a}{p})$ je Jacobiho symbol. Toto kritérium využívá *Solovay-Strassenův test*, který odstraňuje některé nedokonalosti Fermatova testu, ale pravděpodobnost chyby je stále vysoká.[4]

8.3.3. Miller-Rabin

Velmi často používaný pravděpodobnostní test. Je založen na skutečnosti, že je-li celé číslo n prvočíslo, má právě dvě odmocniny, $1\ (mod\ n)$ a $-1\ (mod\ n)$.

Nechť $a \in \mathbb{Z}_n^*$, číslo a je *kvadratický zbytek* modulo n, jestliže existuje $x \in \mathbb{Z}_n^*$ takové, že $x^2 \equiv a\ (mod\ n)$, pak Q_n je množina kvadratických zbytků modulo n.

Nechť $a \in Q_n$, jestliže $x \in \mathbb{Z}_n^*$ splňuje $x^2 \equiv a\ (mod\ n)$ pak se x nazývá *odmocnina* čísla a modulo n.

1. Jestliže je p liché prvočíslo, $a \in Q_n$, pak ma přesně dvě druhé odmocniny modulo p.

2. Obecně, nechť $n = p_1^{e_1} * p_2^{e_2} * \ldots * p_k^{e_k}$, kde $p_1, p_2, \ldots, p_k$ jsou navzájem různá prvočísla a $e_i \geq 1$. Pokud $e \in Q_n$, pak má přesně 2^k různých odmocnin modulo n.

Efektivnějšího testování docílíme doplněním Malou Fermatovou větou[21].

Pro libovolné liché prvočíslo p a libovolné celé číslo a, $1 \leq a \leq p-1$, $gcd(a, p) = 1$, platí $a^{\frac{p-1}{2}} = \pm 1 \ (mod\ p)$.

Pro vyloučení Carmichaelových čísel[28] rozšíříme podmínku.

Pro liché číslo $n \geq 3$, mějme sudé $n-1$, které vyjádříme jako součin určité mocniny čísla 2 a lichého čísla r. Potom $n - 1 = 2^s r$, $s > 0$. Algoritmus pak obsahuje výpočet zbytků po dělení číslem n posloupnosti mocnin $a^r, a^{2r}, \ldots, a^{2^{s-1}r}$ pro $1 \leq a \leq n-1$.

Princip testu:

Číslo n je liché prvočíslo, $p - 1 = 2^s r$, r je liché, a je celé číslo, že $gcd(a, p) = 1$, pak pro nějaké j, $0 \leq j \leq s-1$ platí:

$$a^r \equiv 1 \ (mod\ p) \quad nebo \quad a^{2^j} r \equiv -1 \ (mod\ p)$$

Číslo n je liché složené číslo, $n-1 = 2^s r$, r je liché a a je celé číslo, $1 \leq a \leq n-1$.

- Jestliže $a^r \not\equiv 1 \ (mod\ n)$ a $a^{2^j r} \not\equiv -1 \ (mod\ n)$ pro každé j, $0 \leq j \leq s-1$, pak se n nazývá *silný svědek složenosti* čísla n.

- Jinak $a^r \equiv 1 \ (mod\ n)$ nebo $a^{2^j r} \equiv -1 \ (mod\ n)$ pro každé j, $0 \leq j \leq s-1$, pak n se nazývá *silné pseudoprvočíslo* vzhledem k bázi a. Celé číslo a se nazývá *silný lhář* prvočíselnosti čísla n.

Pravděpodobnost prohlášení lichého složeného čísla n za prvočíslo je v tomto případě menší než $(1/4)^t$.

Řádová složitost algoritmu je $\Theta(t * log^3\ n)$.[9]

8.3.4. AKS

Je deterministický test prvočíselnosti s polynomiální časovou složitostí, který se neopírá o žádné neprokázané předpoklady ani faktorizaci[45]. Název je složen z počátečních písmen autorů z roku 2002 (Agrawal, Kayal a Saxena).

Používá zobecnění Malé Fermatovy věty[21] a vychází z rovnosti $(x-a)^n \equiv (x^n - a)(mod\ n)$, která platí pro nesoudělná přirozená čísla a a n pouze tehdy, když n je prvočíslo. Tato rovnost dává vždy správný výsledek, ale časová složitost je exponenciální.

Algoritmus používá příbuznou rovnost:

Jestliže n je prvočíslo, potom pro všechna a, kdy $1 \leq a \leq n$, platí:

$$(x-a)^n = (x^n - a)(mod\ n, x^r - 1)$$

kde $p(x)(mod\ n, x^r - 1)$ je polynom, který je zbytkem po dělení koeficientů modulo n a mocnin x modulo r.

Tuto rovnost již lze vyhodnotit v polynomiálním čase.

Postup testu při výpočtu

1. Vyhledání prvočísla r tak že platí $r = kq+1$, $q = P(r-1)$, $P(x)$ je největší prvočíselný dělitel x, $q \geq 4\sqrt{r\ log_2(n)}$ a $n^k \neq 1\ (mod\ r)$. V tomto kroku ověříme, že n není dělitelné žádným prvočíslem menším než r, jinak je n složené číslo.

2. Pro každé kladné přirozené číslo a, kdy $a \leq 2\sqrt{r\ log_2(n)}$, se ověří rovnost $(x-a)^n = (x^n - a)(mod\ n, x^r - 1)$. Pokud platí je n prvočíslo.

Časová složitost algoritmu je $\Theta(log^{10.5} n\ log\ log\ n)$[5].

9. Asymetrické šifry

9.1. Merkle-Hellman

Jeden z nejstarších asymetrických systémů, nazýván jako zavazadlový algoritmus nebo batoh. Objevili jej v roce 1978 Raplh Merkle a Martin Hellman. Lze jej využít pouze k šifrování. Algoritmus využívá "zavazadlový problém", který je NP-úplný.

Princip systému

Základem algoritmu jsou dvě posloupnosti nezáporných čísel. Jedna posloupnost, superrostoucí, tvoří soukromý klíč (lehké zavazadlo) a druhá posloupnost, normální, tvoří veřejný klíč (těžké zavazadlo). Pro lehké zavadlo je problém řešitelný v lineárním čase, pro těžké zavazadlo v nepolynomiálním. Obě zavazadla mají stejnou celkovou hmotnost, součet položek zavazadla. Zpráva je šifrována po blocích. Každý blok má počet bitů shodný s počtem prvků zavazadla. Prvky zavazadla reprezentují bity šifrovaných bloků zprávy a určují, zda položku zavazadlo obsahuje nebo ne.

Zavazadlový problém

Pro zavazadlo s definovanou kapacitou $S = \{c_i, c_{i+1}, \ldots, c_n\}$ a cílovou váhou T platí $c_i \geq 0$ a výběrový vektor $V = [v_1, v_2, \ldots, v_n]$ s prvky 0 a 1 platí:

$$\sum_{i=1}^{n}(c_i * v_i) = T$$

Dále pro každý prvek c_i superrostoucí posloupnosti platí, že je větší, než součet všech předcházejících prvků:

$$c_n > (c_1 + c_2 + \ldots, +c_{n-1})$$

Transformaci soukromého klíče na veřejný provedeme vynásobením prvků superrostoucí posloupnosti číslem $n \ mod \ m$, kdy m musí být větší než součet všech prvků posloupnosti a n nemá žádné společné součinitele s $mod \ m$.

$$S_1 = \{c_1, c_2, \ldots, c_n\}, \ S_2 = \{c_1 * \ n \ mod \ m, c_2 * \ n \ mod \ m, \ldots, c_n * \ n \ mod \ m\}$$

pro S_1 platí:

$$\sum_{i=1}^{n}(c_i) < m \ \wedge \ gcd(n, m) = 1$$

Transformaci šifrových bloků na položky lehkého zavazadla provedeme tak, že čísla šifrovaného textu, reprezentující šifrové bloky bitů, vynásobíme výrazem:

$$n^{-1} \ mod \ m$$

Bezpečnost systému

Pro váhu lehkého zavazadla je snadné zjistit složení zavazadla, pro těžké zavazadlo nikoliv. Obtížnost vyřešení problému roste exponenciálně s počtem položek zavazadla. Zdálo se, že pro zavazadlo s počtem položek řádově ve stovkách je algoritmus bezpečný.

Ale v procesu transformace těžkého zavazadla na lehké nalezli Shamir a Zippel bezpečnostní trhlinu. Podařilo se jim z šifrového textu získat první a poslední bit otevřeného textu. Dále Shamir ukázal, že za určitých okolností může být tento systém prolomen. Těžké zavazadlo vygenerované z lehkého není těžké, jen tak vypadá, a jedná se jen o speciální případ lehkého zavazadla.

Pro slabiny se algoritmus v praxi nepoužívá.

Příklad šifrování

Zvolí se soukromý klíč jako superrostoucí posloupnost (obsah lehkého zavazadla):

$$\{2, 3, 6, 13, 27, 52\}$$

Odvodí se veřejný klíč jako normální posloupnost (obsah těžkého zavazadla) vynásobením prvků superrostoucí posloupnosti zvoleným $n = 31$ a poté redukcí výsledků zvoleným modulem $m = 105$:

$$\{62, 93, 81, 88, 102, 37\}$$

Šifrovaná zpráva 011000110101101110 se rozdělí na bloky o stejném počtu prvků (bitů), jako je počet prvků zavazadla:

011000 110101 101110

Jedničkové prvky bloků zprávy se nahradí dílčími váhami těžkého zavazadla (veřejného klíče) na odpovídajících místech a stanoví se celkové váhy zavazade příslušející jednotlivým blokům:

$$\begin{aligned} 011000 &\rightarrow 93 + 81 = 174, \\ 110101 &\rightarrow 62 + 93 + 88 + 37 = 280, \\ 101110 &\rightarrow 62 + 81 + 88 + 102 = 333. \end{aligned}$$

Váhy zavazadel jednotlivých bloků otevřené zprávy se stávají šifrovým textem, který se odešle:

$$174, 280, 333$$

Příklad dešifrování

Šifrový text $174, 280, 333$ určený k dešifrování vyžaduje, aby příjemce znal *modul* 105 a číselný *násobitel* 31 pro výpočet inverzního prvku kongruence.

$$31^{-1} \equiv 1 \ (mod\ 105)\ dá\ inverzní\ prvek\ 61.$$

Vynásobením čísel šifrového textu $174, 280, 333$ výrazem 61 *mod* 105 a následným rozkladem výsledků se získají položky lehkého zavazadla (soukromého klíče)

$$\begin{aligned} 174 * 61\ mod\ 105 &= 9 = 3 + 6, \\ 280 * 61\ mod\ 105 &= 70 = 2 + 3 + 13 + 52, \\ 333 * 61\ mod\ 105 &= 48 = 2 + 6 + 13 + 27. \end{aligned}$$

Otevřeným textem jsou šestibitové bloky v nichž *jedničkové bity* jsou na místech položek lehkého zavazadla $\{2, 3, 6, 13, 27, 52\}$ určených rozklady a *nulové bity* na místech zbývajících:

$$\begin{aligned} \{3 + 6\} &= 011000, \\ \{2 + 3 + 13 + 52\} &= 110101, \\ \{2 + 6 + 13 + 27\} &= 101110. \end{aligned}$$

Příklady převzaty z [8]

9.2. RSA

Systém vytvořili v roce 1978 Ronald Rivest, Adi Shamir a Leonard Adlemann (RSA je zkratka jejich počátečních písmen). Pro výpočet šifrovacího klíče systém využívá součinu dvou velmi velkých prvočísel, řádově ve stovkách cifer. Bezpečnost systému je založena na obtížnosti rozkladu přirozeného čísla, které je součinem dvou prvočísel. RSA šifry jsou označovány podle délky klíče v bitech nebo v dekadických cifrách (např. RSA-129, klíč má 129 dekadických číslic což odpovídá 426 bitům). Systém lze použít pro šifrování i digitální podpis[17].

Postup vytvoření klíčů

Vybereme dvě prvočísla p a q a spočítáme modul N.

$$N = p * q$$

Spočítáme hodnotu Eulerovy funkce $\varphi(n)$[22].

$$\varphi(n) = (p-1)(q-1)$$

Náhodně zvolíme přirozené číslo e - šifrovací exponent, tak aby e bylo menší než $\varphi(n)$ a bylo s ním nesoudělné. Využijeme Euklidův algoritmus[24].

$$1 \leq e < \varphi(n),\ gcd(e, \varphi(n)) = 1$$

Vypočteme přirozené číslo d - dešifrovací exponent, které je menší než $\varphi(n)$. Existence čísla d je dána Bezoutovou větou[24].

$$1 \leq d < \varphi(n);\ e * d \equiv 1\ (mod\ \varphi(n)) \ \rightarrow\ d \equiv e^{-1}\ (mod\ \varphi(n))$$

Čísla (e, n) tvoří veřejný klíč, čísla (d, n) soukromý klíč.

Postup šifrování

Zprávu M transformujeme[19] na přirozené číslo m, které musí být menší než n.

$$0 \leq m < n$$

Z tohoto plyne, že nelze zašifrovat libovolně dlouhé zprávy. Řešením je rozdělit zprávu do bloků, aby se podmínka dodržela. Bloky se následně šifrují každý

zvášť. Výsledný kryptogram c vznikne aplikací veřejného klíče (e, n) na číslo m.

$$c = m^e \ (mod\ n)$$

Postup dešifrování

Dešifrování kryprogramu c provedeme s použitím soukromého klíče (d, n) dle:

$$m = c^d \ (mod\ n)$$

Platnost dešifrovací funkce je odvozena z tvrzení, že pro včechna m, kdy $0 \leq m < n$, platí:

$$(m^e)^d \equiv m \ (mod\ n)$$

1. Protože $ed \equiv 1 \ (mod\ (p-1)(q-1))$, existuje celé číslo l tak, že: $ed = 1 + l(p-1)(q-1)$
2. Dosazením získáme: $(m^e)^d = m^{ed} = m^{1+l(p-1)(q-1)} = m(m^{(p-1)(q-1)})^l$
3. Jestliže $gcd(p, m) = 1$,tak podle Malé Fermatovy věty platí $m^{p-1} \equiv 1 \ (mod\ p)$.
4. Dostaneme $(m^e)^d \equiv m \ (mod\ p)$. Jestliže p dělí m, jsou obě strany kongruentní 0. Obdobně pro q $(m^e)^d \equiv m \ (mod\ q)$.

Protože čísla p a q jsou prvočísla, podle Čínské věty o zbytcích[21] platí:

$$(m^e)^d \equiv m \ (mod\ n)$$

Bezpečnost systému

Protože není znám rychlý algoritmus na faktorizaci[45] čísla n, je algoritmus RSA bezpečný. Není ale dokázáno, že takový algoritmus neexistuje, ani že je nutno rozdrtit šifru pomocí modulární aritmetiky. Tedy je možné, že existuje postup, který rozluští zašifrovaný text jinak, než pomocí znalosti čísla n.

Příklad zjednodušeného šifrování

Zpráva "MARY HAD A LITLE LAMB" v ASCII kódu

77 65 82 89 32 72 65 68 32 65 32 76 73 84 84 76 69 32 76 65 77 66 46

Rozdělení na šestimístné bloky. *(00 slouží pro zachování stejné délky bloku)*

776582 893272 656832 653276 738484 766932 766577 664600

Jako šifrovací klíč $n = p * q$ je zvoleno číslo 94815109, kde $p = 7151$ a $q = 13259$ jsou prvočísla, $e = 3$. Číselné bloky zprávy se povýší na třetí mocninu a výsledky modulárně redukují klíčem n.

$(776582 * 776582 * 76582)$ *mod* 94815109 získáme první šifrový blok 71611947

Podobně zbytek šifrových bloků:

48484364 03944707 03741778 61544362 35331577 88278091 50439554

Příklad zjednodušeného dešifrování

Šifrový text určený k dešifrování:

71611947 48484364 03944707 03741778 61544362 35331577 88278091 50439554

Zvolený dešifrovací klíč $d = \frac{2*(p-1)*(q-1)+1}{3}$ je 63196467, kde $p = 7151$ a $q = 13259$ jsou prvočísla. Číselné bloky šifrového textu se umocní na dešifrovací klíč d a výsledky modulárně redukují klíčem e.

$71611947^{63196467}$ *mod* 94815109 získáme otevřený blok 776582

Podobně získáme zbývající bloky otevřeného číselného textu:

893272 656832 653276 738484 766932 766577 664600

Příklady převzaty z [8]

9.2.1. Urychlovací techniky

V této kapitole jsem čerpal z [2].

RSA velmi často používá k šifrování výpočty s umocňováním velmi dlouhých čísel. I když použijeme efektivní algoritmy pro tyto operace, například RSMA[25], jsou výpočty s operandy o délce 1024 bitů a víc velmi náročné. Proto byly vyvinuty následující obecné akcelerační techniky.

Rychlé šifrování s SPE

Fast Encryption with Short Public Exponents (SPE). Jedná se o velmi jednoduchý a účinný trik, který můžeme použít pokud provádíme operace s veřejným klíčem a přitom se nemusíme obávat o bezpečnost pro nízkou hodnotu exponentu e. V praxi tato situace nastává při ověření digitálního podpisu[17], kdy je použití klíčů opačné. Pro tento případ lze zvolit pro e velmi malé hodnoty. V praxi se často používají tři hodnoty e $\{3, 17, 2^{16}+1\}$, které mají zvláštní význam.

Public key e	e as binary string	#MUL + #SQ
3	11_2	3
17	10001_2	5
$2^{16}+1$	$1\ 0000\ 0000\ 0000\ 0001_2$	17

Tabulka 4. Složitost operací při použití SPE.[2]

Uvedené složitosti je třeba porovnat s přibližne 15-ti sty operacemi násobení a umocňování, které jsou nutné pro exponenty plné délky. V tomto případě je $t+1$ počet bitů modulu n, tj. $[log_2\ n] = t+1$. Všechny tři výše uvedené exponenty mají nízkou hmotnost, tzv. Hamming, což je počet jedniček v binárním reprezentaci. To má za následek mimořádně nízký počet operací prováděných při umocňování. Systém je stále bezpečný, i když jsou používány takto krátké exponenty. Soukromý klíč d má stále plnou bitovou délku $t-1$, přestože e je krátký.

S krátkými exponenty je šifrování RSA k ověření digitálního podpisu velmi rychlé. Kombinace obou urychlovacích technik činí RSA téměř ve všech praktických případech nejrychlejším systémem veřejného klíče, který je k dispozici. Bohužel, neexistuje způsob jak více urychlit RSA pokud je do výpočtu zapojen dešifrovací exponent d soukromého klíče, tj. pro dešifrování a vytváření podpisu. Z tohoto důvodu jsou tyto operace pomalé. Jiné systémy veřejného klíče bývají v těchto případech často mnohem rychlejší.

Rychlé dešifrování s využitím CRT

Fast Decryption with the Chinese Remainder Theorem (CRT). Pro dešifrování nelze zvolit krátký soukromý klíč, protože by tím byla ohrožena bezpečnost RSA. Pokud bychom zvolili krátký dešifrovací exponent d, útočník by mohl útokem hrubou silou klíč prolomit. Je pokázáno, že soukromý klíč musí mít délku nejméně 300 bitů, kde t je počet bitů modulu N. V praxi, se často e volí krátký a d má plnou bitovou délku. Pro urychlení výpočtu dešifrování a generováni digitálního podpisu s dlouhým n lze využít techniku , která je založena na čínské větě o zbytcích (CRT)[21].

Pro urychlení provedeme umocňování $x^d \ mod \ n$ efektivněji. Protože příjemce, který vlastní soukromý klíč, zná také prvočísla p a q, může místo počítání s jedním dlouhým modulem n počítat dva krátké moduly s prvočísly p a q. Zrychlený výpočet bude mít tři kroky: transformaci do zbytkových tříd dle CRT, výpočty v oblasti CRT a inverzní transformaci výsledku.

1. Transformace do zbytkových tříd - základní prvek x rozdělíme na dva faktory p a q na modulu n, a získáme modulární reprezentaci x.

$$x_p \equiv x \ mod \ p$$
$$x_q \equiv x \ mod \ q$$

2. Výpočet - pro snížené verze x provádíme následující umocnění:

$$y_p = x_p^{d_p} \ mod \ p$$
$$y_q = x_q^{d_q} \ mod \ q$$

   kde jsou dva nové exponenty dány vztahem:

$$d_p \equiv d \ mod \ (p-1)$$
$$d_q \equiv d \ mod \ (q-1)$$

   Oba exponenty d_p a d_q po transformaci do zbytkových tříd , jsou ohraničeny p a q. Totéž platí pro transformované výsledky y_p a y_q. Vzhledem k tomu, že pro dvě prvočísla jsme se rozhodli mít zhruba stejnou bitovou délku, dva exponenty jakož i y_p a y_q mají poloviční délku n bitů.

3. Inverzní transformace - zbývající krok k sestavení konečného výsledku y z modulární reprezentace (y_p, y_q), které vychází z CRT, provedeme:

$$y \equiv [qc_p]\, y_p + [pc_q]\, y_q \; mod \; n$$

kde koeficienty c_p a c_q vypočítame ta, že:

$$c_p \equiv q^{-1} \; mod \; p; \; c_p \equiv p^{-1} \; mod \; q$$

Prvočísla u konkrétní implemetace RSA se nemění často, proto dva výrazy v závorkách můžeme vypočítat předem. Potom celá zpětná transformace zahrnuje pouze dvě operace modulárního umocnění a jedno modulární sčítání.

Výpočetní složitost metody CRT.

Pokud se podíváme na tři kroky, které se podílejí na umocňování v CRT, dojdeme k závěru, že pro praktickou analýzu složitosti transformace a inverzní transformace může být ignorována, protože operace v nich prováděné jsou zanedbatelné ve srovnání s umocňováním ve zbytkových třídách.

Budeme předpokládat, že n má $t + 1$ bitů a p a q jsou asi $t/2$ bitů dlouhé. Pak všechna čísla zapojené do umocňování v CRT, x_p, x_q, d_p, d_q, jsou vázány k velikosti p a q, mají délku také asi $t/2$ bitů. Pokud budeme používat RSMA[25] algoritmus pro dvě umocnění, kdy každé z nich vyžaduje přibližně $1,5\ t/2$ modulárních násobení a mocnění, bude výsledný počet těchto operací:

$$\#SQ + \#MUL = 2 * 1,5\ t/2 = 1,5\ t$$

Zdá se, že je to stejná složitost jako u umocňování bez CRT. Nicméně, každé násobení a mocnění zahrnuje čísla, které mají délku pouze $t/2$ bitů, na rozdíl od operací bez CRT, kde se operace provádí s t-bity. Složitost násobení klesá kvadraticky s délkou bitů, každý $t/2$-bit násobení je čtyřikrát rychlejší než t-bit násobení. To znamená, že celkové zrychlení získané pomocí CRT, je 4. Toto zrychlení je velmi cenné v praxi. Vzhledem k tomu, že metoda nemá téměř žádné nevýhody, je umocňování v CRT používáno v mnoha kryptografických produktech, například webový prohlížeč.

9.3. Diffie-Hellman

Systém výměny veřejného klíče, vyvinutý profesory ze Stanfordu Whitfieldem Diffiem a Martinem Hellmanem. Systém umožňuje dvěma stranám bezpečnou výměnu šifrovacích klíčů nezabezpečeným kanálem. Tento systém bývá také označován jako exponenciální výměna klíče (Exponential Key Exchange).

Postup výměny a výpočet klíče

Před samotnou komunikací mezi dvěma účastníky se dohodne velké prvočíslo q a číslo α, kde α je primitivní modulo $mod\ q$. Čísla α a q nejsou tajná a může je znát každý. Jeden účastník vybere náhodné číslo X_a, které uloží jako tajné a vypočítá číslo Y_a podle vzorce:

$$Y_a = \alpha^{X_a}\ mod\ q$$

Potom první účastník-A odešle číslo Y_a druhému účastníkovi-B. Mezitím si B vybral své vlastní náhodné číslo X_b a vypočítal číslo Y_b a poslal jej A.

$$Y_b = \alpha^{X_b}\ mod\ q$$

Účastník A vypočítá číslo K_{ab}, které bude sdílený klíč, podle vzorce:

$$K_{ab} = Y_b^{X_a}\ mod\ q = \alpha^{X_b X_a}\ mod\ q$$

Stejné číslo K_{ab} vypočítá účastník B podle vzorce:

$$K_{ab} = Y_a^{X_b}\ mod\ q = \alpha^{X_b X_a}\ mod\ q$$

Číslo K_{ab} je bezpečné, neboť jeho výpočet vyžaduje znalost čísel X_a a X_b. A zná číslo X_a a B zná číslo X_b. Případný narušitel, který odposlouchává vzájemnou komunikaci, zná pouze čísla Y_a a Y_b. Vypočítat číslo X_a z čísla Y_a nebo X_b z Y_b je velmi obtížný problém.

Bezpečnost systému

K prolomení systému by bylo třeba najít diskrétní logaritmus $mod\ q$, nebo provést rozklad čísla o velikost q. Je-li q dost velké, nelze problém vyřešit.

Příklad výpočtu klíče

Použijeme čísla 5 a 563:

$$\alpha = 5\ ,\ q = 563$$

Alice si vybrala tajné číslo 9.

$$X_a = 9$$

Potom Bobovi odeslala číslo 78:

$$5^9\ mod\ 563 = 1953125\ mod\ 563 = 78$$

Bob si vybral své tajné číslo 14.

$$X_b = 14$$

Alici pak poslal číslo 534:

$$5^{14}\ mod\ 563 = 6103515625\ mod\ 534$$

Alice vypočítala číslo K:

$$K = 534^9\ mod\ 563 = 353078532821730886079846 4\ mod\ 563 = 117$$

A Bob vypočítal číslo K:

$$K = 78^{14}\ mod\ 563 = 308549209196654470906527744\ mod\ 563\ = 117$$

Alice a Bob teď spolu mohou komunikovat pomocí šifrovacího klíče 117.

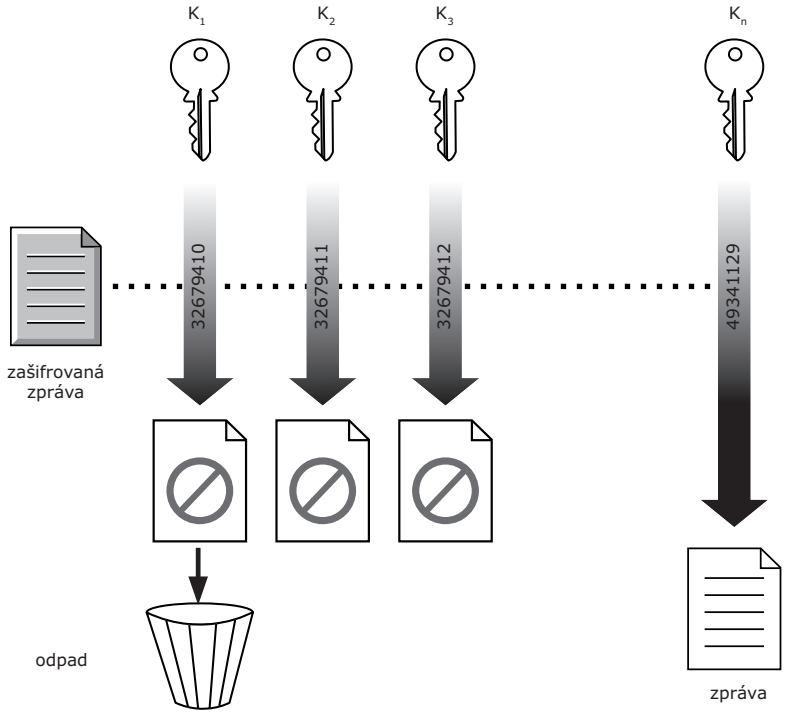
Příklad převzat z [7]

10. Kryptoanalýza

10.1. Způsoby prolomení

Použití hrubé síly

Nejjednodušší cesta jak šifrový kód prolomit je vyzkoušet všechny možné klíče, jeden po druhém, za předpokladu, že narušitel pozná výsledek správného klíče, například čitelná zpráva. Většina pokusů selže, ale jeden z pokusů bude úspěšný a ten umožní dešifrovat zprávu.



Obrázek 7. Použití hrubé síly

Použití hrubé síly je také nazýváno **útokem hledání klíče**. Principem je zjišťování všech klíčů, až po nalezení toho pravého. Použití hrubé síly není účinné, někdy dokonce ani možné. Bývá příliš velký počet klíčů, které se musí vyzkoušet, a není dostatek času na jejich vyzkoušení. [1]

[1] Pro 128-bitový klíč počet možných klíčů dosahuje hodnoty $2^{128}(3,4 * 10^{38})$. Při vyzkoušení miliardy možných klíčů za vteřinu souběžně na miliardě počítačů, by prověření všech možných klíčů trvalo 10^{13}let.[7]

”Pokud by jediným faktorem ovlivňující bezpečnost šifry byla délka klíče, pak by každý mohl jednoduše použít kódy se 128bitovými klíči a komunikace by byla bezpečná. Kryptografie by tak byla jen jednoduchou oblastí matematiky. Většina šifrovacích algoritmů se však nechová podle očekávání. Hrubá síla je k prolomení šifry používána jen zřídka. Dnes mohou být šifrovací algoritmy prolomeny použitím kombinace pokročilé matematiky a výkonu počítače. Výsledkem je pak to, že kdokoliv může rozšifrovat zprávu aniž by znal klíč. Zkušení kryptoanalytici někdy dokáží dešifrovat zprávu i bez znalosti šifrovacího algoritmu. A použití špatného šifrovacího algoritmu se silným klíčem bezpečnost šifrování algoritmu nezvýší.” [7]

Přístup se známým otevřeným textem

V tomto případě má před sebou kryptoanalytik část otevřeného textu a odpovídající část zašifrované zprávy. Tato zdánlivě nemožná okolnost je zcela běžná tehdy, když je šifrování použito pro ochranu elektronické pošty (s normovanými hlavičkami na začátku každé zprávy) nebo k ochraně dat pevných disků (se známými strukturami v předem daných umístěních na disku). Cílem tohoto útoku je nalezení šifrovacího klíče, který lze poté použít na zbývající část zprávy a dešifrovat ji.

Přístup s výběrem přímého textu

U tohoto případu je předmětem snažení kryptoanalytika neznámý blok zašifrovaných dat a výsledkem je, že tato data lze analyzovat, rozeznat. Tento způsob je snadnější provést než se může zdát, například pokud je šifrování použito pro radiové spojení, které přenáší telefonické zprávy. Cílem je najít šifrovací klíč, který se následně použije k dešifrování dalších zpráv.

Diferenční kryptoanalýza

Je druhem přístupu s vybraným otevřeným textem. Týká se šifrování mnoha textů, které jsou jen málo odlišné jeden od druhého a srovnávají se výsledky.

Frekvenční kryptoanalýza

Ve své podstatě je založená především na lingvistických znalostech konkrétního jazyka. Základem je zjištění frekvence výskytu jednotlivých i skupinových znaků v běžném textu daného jazyka a následným porovnáním se šifrovým textem za využití pokročilé matematiky.

Odolnost šifrovacího systému

"Šifrovací algoritmy jsou k výše uvedeným způsobům útoků různě citlivé. Obtížně prolomitelné se nazývají silné algoritmy, a ty které lze snadno odhalit jsou nazývány slabé. Vyvinout nový kryptografický algoritmus není vůbec snadné. O postupu návrhu silných algoritmů není moc zpráv anebo jsou utajeny. V mnoha případech, kdy byly algoritmy navrženy a považovány za silné, se po použití zjistilo, že mají slabiny. Jediný spolehlivý způsob jak určit, zda je algoritmus silný nebo slabý, je zveřejnění algoritmu a čekání, zda někdo slabinu najde. Tento proces rovnocenného přehledu o algoritmu není ideální, ale je lepší než žádný. Bezpečnost kryptografie spočívá v otevřenosti a rovnocenném zobrazení."[7]

10.2. Prolomení šifry Merkle-Hellman

Řešením hrubou silou je projít všechny možné kombinace součtů položek zavazadla dokud se neshodují s celkovou hmotností zavazadla. Algoritmus vždy vypočítá správné řešení. Má však exponeciální složitost $\Theta(2^n)$. Pro velký počet položek zavazadla je to časově velmi náročné.

Urychlení může přinést použití heuristiky. Položky zavazadla seřadíme podle poměru cena (pozice v zavazadle) / hmotnost. Poté zkoušíme přidávat nejprve položky s nejvyšším poměrem. Po nalezení správné kombinace položek se obnoví jejich původní pořadí. Úspora času je v tomto případě velká, ale není vždy optimální.

10.3. Prolomení šifry RSA

Řešením hrubou silou je pokusit se z veřejného klíče rozložit modul N a získat prvočísla p a q. Z tohoto rozkladu již lze dopočítat dešifrovací exponent dle:

$$d = i^{\phi((p-1)*(q-1))-1} \; mod \; (p-1)(q-1)$$

Dále lze k rozkladu využít možných slabin čísel p a q. Z malého rozdílu těchto čísel plyne, že $p \approx q$ a potom $p \approx \sqrt{n}$.

Pro velikost modulu v rozmezí 1024 - 2048 bitů je útok hrubou silou rozkladem modulu N v současné době z časových nároků na výpočet nemožný .

Dalším možným útokem proti systému RSA je využití postranních kanálů (časový, chybový) vzniklých při implementaci šifry. Popis těchto metod přesahuje rozsah práce.

11. Faktorizace čísel

Faktorizace čísla znamená rozklad na jeho prvočinitele. Tato procedura je podstatně složitější než testování prvočíselnosti. Faktorizace náleží mezi NP-úplné problémy.

Problém faktorizace byl v minulosti pokládán za nezajímavý. Používaná univerzální metoda postupného dělení prvočísly pro běžné rozklady postačovala. Pro velká čísla je tato metoda však nepoužitelná. Až nástup asymetrických šifrovacích systémů založených na součinech velkých prvočísel postavil tuto problematiku do popředí zájmu, zejména kryptoanalytiků. Problémem není pouze nalezení efektivního způsobu rozkladu, ale i ve výpočetní složitosti algoritmu.

V roce 1970 byly rozkládána 20-ti ciferná čísla. V roce 1980 umožnil rozvoj počítačů a vývoj metody řetězových zlomků rozklad 50-ti ciferného čísla. V dalších letech vedla metoda kvadratického síta k rozkladu 100-ciferných čísel. V polovině 90-tých let dovolilo použití distribuovaného zpracování prostřednictvím Internetu rozklad 129-ciferného čísla. Metoda obecného síta v číselném tělese umožňovala kolem roku 1996 rozklad 130-ti ciferných čísel. Stav v roce 2003 je zřejmý z toho, že nejmenší číslo zařazené do soutěže RSA-Challenge mělo 174 cifer.

počet cifer	bitů	datum	MIPS/rok	algoritmus rozkladu
100	332	duben 1991	7	Quadratic sieve
110	365	duben 1992	75	Quadratic sieve
120	398	červen 1993	830	Quadratic sieve
129	428	duben 1994	5000	Quadratic sieve
130	431	duben 1996	1000	Generalized number field sieve
140	465	únor 1999	2000	Generalized number field sieve
155	512	duben 1999	8000	Generalized number field sieve
160	530	duben 1996	-	Lattice sieve
174	576	duben 1996	-	Lattice sieve
200	663	duben 1996	-	Lattice sieve

Tabulka 5. Postup faktorizace[1]

Poslední rozložené číslo v rámci soutěže RSA Challenge je RSA-768.[54]
Graf časové náročnosti rozkladu velkých čísel metodami proséváním:[55]

11.1. Faktorizační metody

11.1.1. Metoda postupného dělení

je nejstarší metodou faktorizace. Postupně dělíme rozkládané číslo n čísly menšími než $\sqrt{n}$. Pokud vyjde dělení beze zbytku, je dělitel součástí rozkladu. Nemusíme však zkoušet všechna čísla. Pro urychlení algoritmu stačí dělit pouze prvočísly. Pokud bude číslo n dělitelné složeným číslem, bude také dělitelné všemi prvočísly z jeho prvočíselného rozkladu. Pro rozklad je vhodné mít k dispozici prvočísla z intervalu $< 2, \sqrt{n} >$.

Metoda je časově velmi náročná a pro velká čísla nepoužitelná. Složitost je pro dělení prvočísly $\Theta(2\sqrt{n}/ln\ n)$ a pro dělení lichými čísly $\Theta(\sqrt{n}/2)$.

11.1.2. Fermatova metoda

Algoritmus rozkladu popsaného Fermatem je velmi efektivní, když složené číslo n je součinem dvou přibližně velkých čísel, tedy s malým rozdílem $p - q$.

Jestliže n je kladné liché číslo, pak existuje vzájemně jednoznačné přiřazení mezi rozkladem n na součin dvou kladných čísel a rozdílem druhých mocnin dvou přirozených čísel rovnající se n.

$$n = a * b = s^2 - t^2,\ a > b$$

kde $s = \frac{a+b}{2}$ a $t = \frac{a-b}{2}$ jsou přirozená čísla. Jestliže $n = s^2 - t^2$ se rovná rozdílu druhých mocnin celých čísel, rozkladem je $n = (s + t) * (s - t)$.

Pro $n = a * b$ kde $a > b$ platí:

$$n = \left(\frac{a+b}{2}\right)^2 - \left(\frac{a-b}{2}\right)^2$$

Pro $D = \frac{p-q}{2}$ dostaneme $n + D^2 = \left(\frac{p-q}{2}\right)^2$. Pro malá D tedy zkoušíme, jestli $n + D^2$ je čtverec. Pokud takové D najdeme, dopočítáme p, q podle:

$$p,\ q = \sqrt{n + D^2} \pm D$$

Na bázi Fermatovy metody jsou postaveny dokonalejší algoritmy, které dávají mnohem efektivnější výsledky.[6]

11.1.3. Pollardova p-1 metoda

Pollardova $p-1$ metoda předpokládá, že alespoň pro jeden faktor p má číslo $p-1$ všechny své faktory relativně malé omezené nějakým b.

Přirozené číslo b je B-hladké jestliže pro libovolné prvočíslo p a libovolné přirozené číslo k platí: $p^k \mid n \Rightarrow p^k \leq B$.

Nyní můžeme odhadnout $k = b!$ jako násobek $p-1$, neboli $p-1|k$.

Dle Fermatovy věty pro dané $a,\ gcd(a,p) = 0$ platí:

$$a^{p-1} \equiv 1\ (mod\ p)$$

Protože $p-1 \mid k$, můžeme kongruenci umocnit do tvaru:

$$a^k \equiv\ 1\ (mod\ p)$$

Odtud $p \mid gcd(a^k - 1,\ n)$. Můžeme předpokládát, že $p = gcd(a^k - 1,\ n)$, k je kladný násobek $p-1$, za a volíme 2 nebo 3. Pokud nám vyjde 1, odhad k nebyl násobkem $p-1$ ani $q-1$. Jestliže vyjde odhad různý od 1 nebo roven n, odhad je dělitelem n a k byl násobkem $p-1$ a $q-1$[6].

11.1.4. Obecné faktorizační algoritmy

Jsou nezávislé na speciálních vlastnostech faktorizovaného čísla n. V posledních letech bylo vytvořeno několik úspěšných metod, například Dixonův algoritmus, Faktorizace metodou řetězových zlomků (CFRAC), Kvadratické síto (QS), Obecné síto číselného tělesa (GNFS), Shankova faktorizace kvadratickými formami (SQUare FOrm Factorization) a Shorova faktorizační metoda pro kvantový počítač. Popis těchto algoritmů přesahuje rozsah práce.

11.1.5. Srovnání metod

V současné době nelze jednoznačně určit, který z algoritmů je univerzální a nejefektivnější pro všechna čísla. Pro každý z nich existuje číselný rozsah pro který je právě efektivnější více než ostatní. Pro čísla řádově $10^{10} \sim 10^{18}$ je to SQUare FOrm Factorization, pro $10^{50} \sim 10^{120}$ Quadratic Sieve a pro čísla $> 10^{120}$ General Number Field Sieve.

12. Program AsymmCrypt

12.1. Popis programu

Program AsymmCrypt 1.1 je jednoduchá aplikace spustitelná přímo z disku počítače. Tato Windows Forms aplikace je naprogramována v jazyce C# a odladěna v prostředí Microsoft Visual Studio 2010 Express. Se systémových požadavků vyžaduje minimálně operační systém Microsoft Windows 7 a prostředí .NET verze 4.0 nebo vyšší. Vývoj aplikace a ladění probíhalo na mobilním zařízení s procesorem Atom 1,6 GHz a 1GB RAM s operačním systémrm Windosw 7 Starter.

Cílem vytvoření aplikace bylo ověření získaných znalostí z oblasti asymetrických šifrovacích systémů a přehledné znázornění postupu výpočtů během šifrovacích procesů těchto systémů. Aplikace umožňuje šifrovat a dešifrovat zprávu v textovém formátu dvěmi nejznámějšími asymetrickými systémy. Pro účely demonstrace funkce těchto systémů byly zvoleny zcela odlišné šifry RSA a Merkle-Hellman (zavazadlový algoritmus).

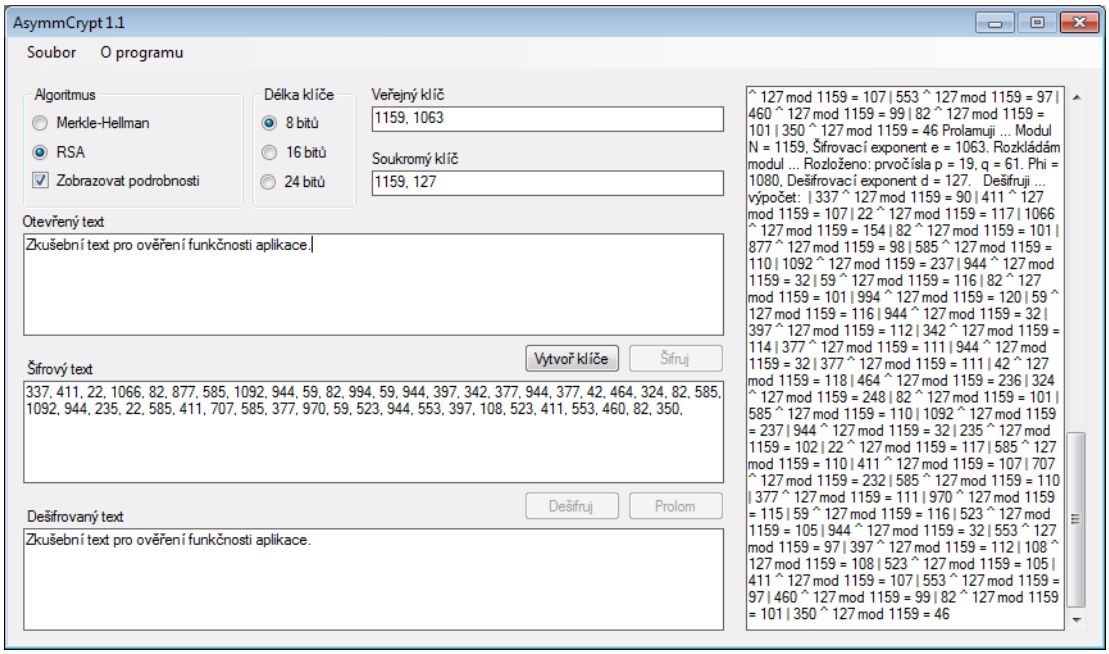
Hlavním rozdílem mezi těmito systémy je struktura šifrovacích klíčů a dílčí šifrovací výpočty. Pro klíče byly zvoleny 3 možnosti nastavení délky, bezpečnosti šifry, které se liší dle zvoleného šifrovacího systému. Maximální hodnota délky klíče byla volena s ohledem na výpočetní možnosti výkonnostně slabších stolních počítačů a přehlednost podrobných dílčích výpočtů během procesu kódování dat.

Dále byla pro volbu délky klíče algoritmu RSA rozhodující možnost předvést i prolomení kódu hrubou silou za použití faktorizace. Pro dlouhé klíče by možnosti k prolomení touto technikou byly značně ztížené. Pro transformaci textu bylo použito 8-bitové kódování znaků dle Win1250, což umožnilo snadněji identifikovat znaky ve výpočtu dle dekadické hodnoty znaku. Pro práci s prvočísly využívá aplikace vygenerovaný seznam prvočísel při startu programu.

12.2. Ovládání a výstup programu

Vzhledem k jednoduchosti a účelu aplikace bylo zvolen strohý návrh a přímočaré ovládání s minimem možností a nastavení pro šifrování. Pro oba algoritmy umožňuje program nastavit rozdílné dély šifrovacích klíčů. Dále je zde možnost zvolit zobrazování podrobností výpočtu. Vstupní data se zadávají do textového pole. Výstupy jsou zobrazovány taktéž v textových polích, včetně chybových hlášení.

Ovládání programu je přepínači a tlačítky, která se postupně aktivují a deaktivují dle kroku výpočtu. Při změně nastavení programu v dílčím kroku šifrového výpočtu jsou dosažené mezivýsledky zahozeny a program šifruje od začátku s novým nastavením. Volba "Prolom" je dostupná pouze u aloritmu RSA. Kompletní šifrovací výstup z programu včetně detailů výpočtu je možno uložit do textového souboru, nabídkou "Ulož" z menu.



Obrázek 8. Okno aplikace AsymmCrypt.

12.3. Implementované algoritmy

Pro výpočty s *modulo n* byly implementovány dva stěžejní algoritmy. Rozšířený Euklidův algoritmus (EEA) [24] pro výpočet inverzního prvku kongruence a rychlé modulární umocňování (RSMA) [25] pro výpočty s velkými hodnotami exponentů u výpočtů systému RSA. Původní předpoklad pro RSMA byl, že pro použité krátké šifrovací klíče, do 24-bitů, bude stačit datový typ *int*, bylo třeba poupravit a použít místo něho uvnitř funkce datový typ *long*.

Pro vygerování seznamu prvočísel v rozsahu $\leq$ 10000 bylo použito Eratosthenovo síto. Na faktorizaci modulu z veřejného klíče k prolomení šifry RSA postačilo postupné dělení prvočísly ze seznamu. Ostatní algoritmy použité v aplikaci jsou zcela běžné rutiny pro práci s textem, konverzi datových typů a aritmetické operace nebo ukládání dat do textových souborů. Nejpomalejšími

výpočty se nakonec, oproti předpokládané faktorizaci, ukázaly dekompozice položek u zavazadlového algoritmu v kombinaci s detailním zápisem dílčích kroků výpočtů do textových řetězců.

12.4. Struktura programu

Vyjma hlavního okna aplikace, formuláře Form1, jsou všechny naprogramované výpočtové třídy statické. K předávání argumentů a výsledků je užito převážně polí nebo kolekcí ArrayList.

Soubory a obsažené třídy programu

- `Form1.cs` - hlavní formulář aplikace se všemi obslužnými metodami pro vlastní běh programu, události, nastavení parametrů pro výpočet a výstupy.
- `Conversions.cs` - převodní fukce, převážně pro práci s textem a výstup.
  - `public static class Knapsack`
- `Cryptography.cs` - hlavní šifrovací fukce programu , obsahuje dvě třídy, pro každý šifrovací systém zvlášť.
  - `public class static Knapsack`
  - `public class static RSA`[56]
- `Modular.cs` - modulární aritmetika, rozšířený Euklidův algoritmus a rychlé modulární umocňování
  - `public class static Modular`[58]
- `Primes.cs` - generování seznamu prvočísel - Eratosthenovo síto, faktorizace
  - `public static class Primes`

Závěr

Podrobným studiem tématu bylo potvrzeno počáteční tvrzení, že kryptografie používá jednoduché principy a postupy k utajení zpráv za použití pokročilé matematiky. Jednotlivé kroky šifrování u různých asymetrických systémů jsou si velmi podobné, často i totožné. Rozdíly byly shledány především v dílčích aritmetických operacích. Základem těchto operací je vždy modulární aritmetika. Dostupnost veřejného klíče a jeho distribuce ukázala široké možnosti využití asymetrického systému v praxi, bez obav o jeho prolomení a ztrátu bezpečnosti systému. Následná implementace vybraných asymetrických systémů v jednoduchém demonstračním šifrovacím software tyto zjištění jen potvrdila. V operacích modulárního umocňování se už při 24-bitové délce klíče u algoritmu RSA projevily omezení původně zvoleného datového typu pro celá čísla, ač se u 6-ti ciferných prvočísel nepředpokládaly. Pokusy s volbou délky klíčů a řádovou velikostí prvočísel dokázaly exponenciálně stoupající náročnost výpočtu potřebného k prolomení šifry hrubou silou. Pro další zrychlení rozkladu modulu k prolomení šifry RSA se nabízí kromě distribuovaného výpočtu metodou prosívání i použití Shorova faktorizačního algoritmu. Úspěšné použití na kvantovém počítači by vedlo k složitosti rozkladu v polynomiálním čase. Pro současné doporučované délky modulu 1024 – 2048 bitů je však algoritmus RSA z hlediska náročnosti rozkladu modulu stále bezpečný a v nejbližší době obrat očekávat nelze.

Conclusions

By the detail of the study was confirmed initial statement that cryptography is using simple principles and methods for secret messages by using advanced mathematics. Particular steps of cryptography by different asymmetrical systems are very similar, often the same. The differences were found especially in partial arithmetical operations. Principle of these operations is always modular arithmetic. Availability of a public key and its distribution shows wide possibilities of using asymmetrical system, without an apprehension of its revelation and damage of safety system. Further implementation of chosen asymmetrical systems in simple demonstrational cryptography software these informations only confirmed. In operations of modular square by 24-bit key algorithm RSA shows restrictions of origin selected date type for integer numbers, though by six figures prime numbers were not supposed. Trials with choosing the key length and in line amount of prime numbers shows exponentially increase of demanding calculation needed to reveal cryptography by brute force. For furhter accelerate of decomposition modul to encrypt RSA is offering besides distributed sieve method also using a Shor factorization algorithm. Succesfull using on quantal computer leeds to complicacy decomposition in polynomial time. For the current recommended length modules 1024 – 2048 bites is algorithm RSA from the viewpoint difficulty decomposition still secure and we could not see the turnover in the nearest future.

Reference

[1] Stallings, William. *Cryptography and network security.* Prentice Hall, Boston, 2011

[2] Paar, Ch., Pelz, J. *Understanding Cryptography.* Springer-Verlag, Berlin, 2010

[3] Yan, Song Y. *Cryptanalytic attacks on RSA.* Springer Science + Business Media, LLC., New York, 2008

[4] Menezes, Alfred, J., Van Oorschot, Paul C., Vanstone, Scott A. *Handbook of Applied Cryptography.*, elektronický dokument, 2001

[5] Agrawal, M., Kayal, N., Saxena, N. *Primes is in P.*, elektronický dokument, 2002

[6] Wang, Baocang; Liu, Shuanggen; Hu Yupu. *N*ew weak keys in RSA. Wuhan University Journal of Natural Sciences, 2006

[7] Garfinkel, Simson. *PGP: Pretty Good Privaci, šifrování pro každého.* Computer Press, Brno, 1998

[8] Přibyl, J., Kodl, J.*Ochana dat v informatice.* České vysoké učení technické, Praha, 1996

[9] Ochodková, Eliška. *Matematické základy kryptografických algoritmů.* Vysoká škola báňská – Technická univerzita, Ostrava, 2011

[10] Ivánek, Jiří. *Základy kódování a kryptografie.* Ostravská univerzita, Ostrava, 2006

[11] EMC^2. *The RSA Challenge Numbers.* elektronický dokument

[12] Wikipedia. *Historie kryptografie.*, elektronický dokument

A. The RSA Challenge Numbers

RSA-768

Decimal Digits: 232

123018668453011775513049495838496272077285356959533479219732245215172640050726365751874520219978646938995647494277406384592519255732630345373154826850791702612214291346167042921431160222124047927473779408066535141959745985690214341 3

33478071698956898786044169848212690817704794983713768568912431388982883793878002287614711652531743087737814467999489
×
36746043666799590428244633799627952632279158164343087642676032283815739666511279233373417143396810270092798736308917

The effort took almost 2000 2.2GHz-Opteron-CPU years according to the submitters, just short of 3 years of calendar time.

RSA-640

Decimal Digits: 193

3107418240490043721350750035888567930037346022842727545720161948823206440518081504556346829671723286782437916272838033415471073108501919548529007337724822783525742386454014691736602477652346609

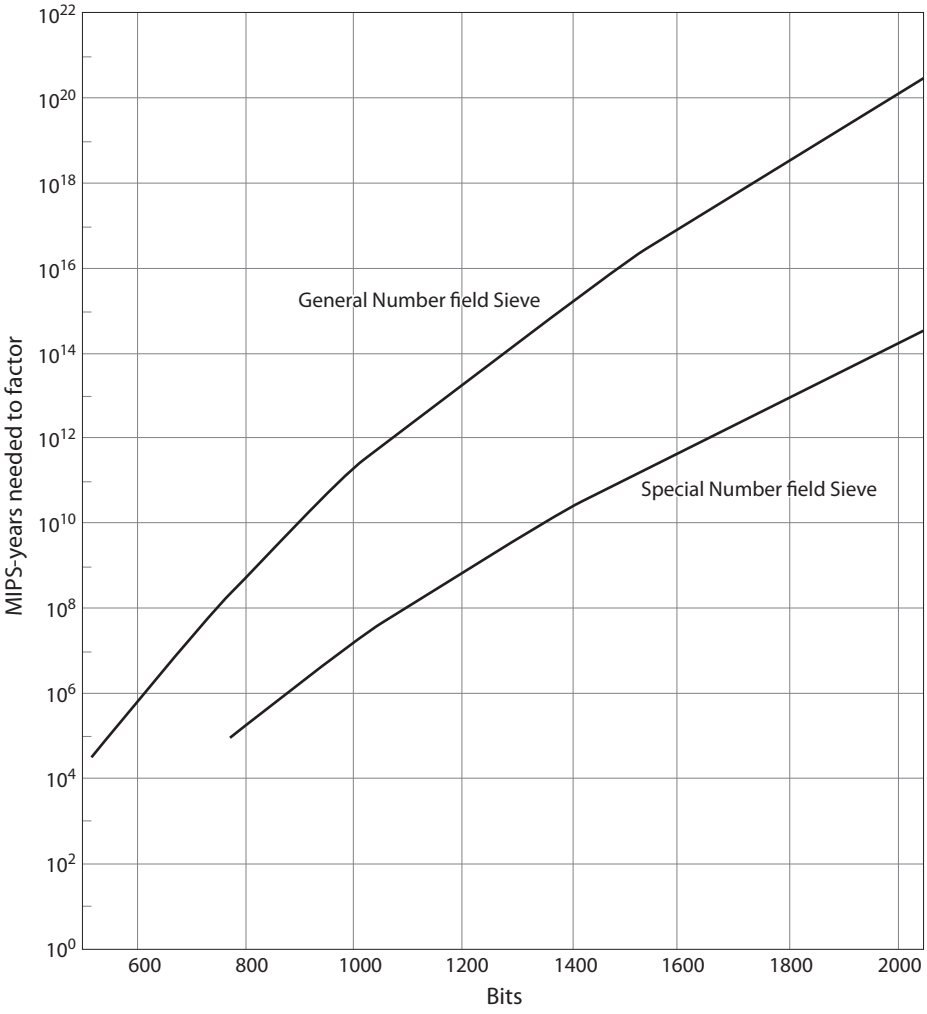
1634733645809253848443133883865090859841783670033092312181110852389331001045081512121181675115 79
×
1900871281664822113126851573935413975471896789968515493666638539088027103802104498957191261465571

The effort took approximately 30 2.2GHz-Opteron-CPU years according to the submitters, over five months of calendar time.

Převzato z [11].

B. MIPS-Year

Náročnost faktorizace velkých čísel pomocí síta[1].



Obrázek 9. MIPS-roky potřebné k faktorizaci

C. Zdrojové kódy

```
public class RSA
    {
        public static int modulN(int p, int q)
        { return p * q; }

        public static int eulerPhi(int p, int q)
        { return (p-1)*(q-1); }

        public static int expE(int phi)
        {
            int e;
            Random r = new Random();
            do e = r.Next(2, phi);
            while(Modular.GreatestCommonDivisor(phi, e) > 1);
            return e;
        }

        public static int expD(int e, int phi)
        { return Modular.InverseElement(e, phi); }

        public static ArrayList Encrypt
         (ArrayList openTextDec, int[] publicKey)
        {
            ArrayList encrypted = new ArrayList();
            string text = " \n Šifruji ... výpočet: ";
            int tmp = 0;
            int enc = 0;
            string e = publicKey[1].ToString();
            string n = publicKey[0].ToString();
            for(int i = 0; i < openTextDec.Count; i++)
            {
                tmp = Conversions.byte_to_int((byte)openTextDec[i]);
                enc = Modular.RepeatSquareAndMultiple
                  (tmp, publicKey[1], publicKey[0]);
                text += " | " + tmp.ToString() + " ^ " + e + " mod "
                 + n + " = " + enc.ToString();
                encrypted.Add(enc);
            }
            encrypted.Add(text);
            return encrypted;
```

```
    }

    public static ArrayList Decrypt
     (ArrayList cipherTextDec, int[] privateKey)
    {
        ArrayList decrypted = new ArrayList();
        string text = " \n Dešifruji ... výpočet: ";
        int dct = 0;
        string d = privateKey[1].ToString();
        string n = privateKey[0].ToString();
        for (int i = 0; i < cipherTextDec.Count; i++)
        {
            dct = Modular.RepeatSquareAndMultiple
             ((int)cipherTextDec[i], privateKey[1], privateKey[0]);
            text += " | " + ((int)cipherTextDec[i]).ToString()
             + " ^ " + d + " mod " + n + " = " + dct.ToString();
            decrypted.Add(dct);
        }
        decrypted.Add(text); return decrypted;
    }

    public static ArrayList Break
     (ArrayList cipherTextDec, int[] publicKey, ArrayList primes)
    {
        ArrayList results = new ArrayList();
        ArrayList factors = new ArrayList();
        int n = publicKey[0]; int e = publicKey[1];
        string text = "\n Prolamuji ... Modul N = " + n.ToString()
         + ", Šifrovací exponent e = " + e.ToString()
         + ". Rozkládám modul ... ";
        factors = Primes.Faktorisation(n, primes);
        int p = (int)factors[0]; int q = (int)factors[1];
        text += "Rozloženo: prvočísla p = " + p.ToString()
         + ", q = " + q.ToString() + ". ";
        int phi = RSA.eulerPhi(p, q);
        int d = RSA.expD(e, phi);
        text += "Phi = " + phi.ToString()
         + ", Dešifrovací exponent d = " + d.ToString() + ". ";
        int[] privateKey = new int[] { n, d };
        results = RSA.Decrypt(cipherTextDec, privateKey);
        results.Add(text);  return results;
    }
}
```

```
public class Modular
    {
        public static int GreatestCommonDivisor(int a, int b)
        {
            int tmp;
            do
            {
                tmp = a % b;
                a = b;
    b = tmp;
            } while (b != 0);
            return a;
        }

        public static int InverseElement(int n, int m)
        {
            int a = n; int b = m;
            int d, x, x1, x2, y, y1, y2, q, r;

            if (b == 0)
            {
                d = a; x = 1; y = 0;
                return x;
            }

            x2 = 1; x1 = 0; y2 = 0; y1 = 1;

            while (b > 0)
            {
                q = a / b; r = a - q * b; x = x2 - q * x1; y = y2 - q * y1;
                a = b; b = r; x2 = x1; x1 = x; y2 = y1; y1 = y;
            }

            d = a; x = x2; y = y2;

            if (x < 0) return x + m;
            else return x;
        }
```

```
    public static int RepeatSquareAndMultiple(int a, int k, int n)
    {
        //binární reprezentace obráceně v zásobníku
        int binary_k = k;
        Queue digits = new Queue();

        while (binary_k != 0)
        {
            digits.Enqueue(binary_k % 2);
            binary_k /= 2;
        }

        //výpočet
        long modul = (long)n;
        long sq = a; long sum = 1;
        if (k == 0) return (int)sum;

        int d = (int)digits.Dequeue();
        if (d == 1) sum = a;

        while (digits.Count > 0)
        {
            d = (int)digits.Dequeue();
            sq = sq * sq % modul;
            if (d == 1) sum = sq * sum % modul;
        }

        return (int)sum;
    }
}
```

D. Obsah přiloženého CD

bin/
: Spustitelný soubor programu AsymmCrypt 1.1

doc/
: Dokumentace práce ve formátu PDF, vytvořená dle závazného stylu KI PřF pro diplomové práce, včetně všech příloh, a všechny soubory nutné pro bezproblémové vygenerování PDF souboru dokumentace (v ZIP archivu).

src/
: Kompletní zdrojové texty programu AsymmCrypt 1.1 se všemi potřebnými zdrojovými texty, knihovnami a dalšími soubory pro bezproblémové vytvoření spustitelných verzí programu (v ZIP archivu).

readme.txt
: Instrukce pro instalaci a spuštění programu AsymmCrypt 1.1, včetně požadavků pro jeho provoz.