Postkvantová kryptografie

Roman Perutka

Bakalářská práce
2022

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2021/2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Roman Perutka**
Osobní číslo:	**A19084**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Postkvantová kryptografie**
Téma práce anglicky:	**Post-quantum Cryptography**

Zásady pro vypracování

1. Uveďte základní matematické pojmy potřebné ke studiu kryptografie.
2. Shrňte problematiku dnes používaných šifry a jejich silné a slabé stránky.
3. Popište některé šifry postkvantové kryptografie.
4. Srovnejte výhody a nevýhody vůči dnes používaným šifrám.
5. Popište a ilustrativně implementujte některé algoritmy spojené s poskvantovou kryptografií.

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. STANOVSKÝ, David, 2010. Základy algebry. Praha: Matfyzpress. ISBN 978-80-7378-105-7.
2. STANOVSKÝ, David a Libor BARTO, 2017. Počítačová algebra. Druhé, upravené vydání. Praha: Matfyzpress. ISBN 978-80-7378-340-2.
3. HOFFSTEIN, Jeffrey, Jill PIPHER a Joseph H. SILVERMAN, 2008. An introduction to mathematical cryptography. New York: Springer. ISBN 978-0-387-77993-5.
4. DZURENDA, Petr a Jan HAJNÝ, 2014. Techniky homomorfního šifrování a jejich praktické využití. Elektrorevue. 16(2), 53-60. ISSN 1213-1539.

Vedoucí bakalářské práce: **Mgr. Jan Krňávek, Ph.D.**
Ústav matematiky

Datum zadání bakalářské práce: **3. prosince 2021**
Termín odevzdání bakalářské práce: **23. května 2022**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 24. ledna 2022

Jméno, příjmení: Roman Perutka

Název bakalářské práce: Postkvantová kryptografie

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 16.5.2022

Roman Perutka v. r.
podpis studenta

ABSTRAKT

Práce si klade za cíl seznámit čtenáře s nejpopulárnějšími standardy kryptografie dnešní doby a jejich postkvantovými alternativami. Poukázat na jejich silné a slabé stránky, demonstrovat použití na konkrétních příkladech a vysvětlit matematické principy, na kterých tyto šifry pracují.

Dále práce obsahuje kapitolu o homomorfním šifrování, ve které je zdůvodněn vznik, nastíněná logika a rozdělení šifer.

Na konci práce jsou stručně popsány algoritmy, které jsem naprogramoval a příloha obsahuje odkaz na veřejné GIT úložiště.

Klíčová slova: AES, RSA, NTRU, NTRUEncrypt, homomorfní šifrování

ABSTRACT

The thesis aims to inform readers about the most popular standards of current cryptography and their postquantum alternatives. Point out their strengths and weaknesses, demonstrate their use in specific examples and explain the mathematical principles on which these ciphers work.

Furthermore, the work contains a chapter on homomorphic encryption, in which the origin, outlined logic and distribution of ciphers are justified.

At the end of the work, the algorithms that have I programmed are briefly described and the appendix contains a link to the public GIT repository.

Keywords: AES, RSA, NTRU, NTRUEncrypt, homomorphic encryption

Chtěl bych poděkovat svému vedoucímu práce panu magistru Janu Krňávkovi, Ph.D. za jeho odbornou konzultaci práce i rad týkajících se zpracování práce samotné. Děkuji mu za jeho trpělivost, ochotu a vstřícnost.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

V práci se zaměříme na dnes nejpoužívanější šifry v kontextu vyhlídek na jejich užitečnost s příchodem kvantových počítačů, popřípadě si představíme jejich alternativy.

Vysvětlení šifer se bude odvíjet od pochopení matematiky, která se skrývá za funkcionalitou šifer, konkrétně AES, RSA a NTRUEncrypt. Tuto matematiku si podrobně vysvětlíme, nejprve obecně a poté i na příkladech.

Příklady budou také u samotných šifer.

Zmíněny budou i kryptologické pojmy spojené s tématem.

Dále se podíváme na takzvané homomorfní šifrování, proč je potřeba a jak funguje.

Na závěr práce jsou uvedeny stručné popisy mnou programovaných algoritmů z prostředí kryptografie a kryptoanalýzy. Odkazy ke zdrojovým kódům se nachází v příloze **P I**.

I. TEORETICKÁ ČÁST

1 MATEMATICKÝ ZÁKLAD

1.1 Lineární kombinace

O číslu c můžeme říct, že je lineární kombinací čísel a a b, pokud existuje vztah:

$$c = x \cdot a + y \cdot b$$

Kde:

x a y jsou celá čísla

a a b jsou nesoudělná celá čísla

Příklad:

Zapište číslo 20 jako lineární kombinaci čísel 3 a 10:

$$20 = -10 \cdot 3 + 5 \cdot 10$$

1.2 Číselné soustavy

V běžném životě používáme desítkovou soustavu, to nikoho nepřekvapí.

Co to ale znamená?

To znamená, že máme k dispozici deset číslic, než budeme muset zleva připsat novou, aby nám stačila k vyjádření číselné hodnoty.

Stejný princip mají i všechny ostatní používané soustavy.

Dvojková (binární) má k dispozici pouze dvě číslice – 0 a 1.

Osmičková (oktálová) má rozsah 0 až 7.

Šestnáctková (hexadecimální) má k dispozici číslice 0 až 9 a na ně navazuje písmeny A až F pro vyjádření hodnot 10 až 15.

1.2.1 Převody mezi soustavami

Pro naše potřeby stačí umět převody mezi dvojkovou a hexadecimální prostřednictví desítkové.

Příklad:

Převeďte číslo z binární 1101 do hexadecimální.

Řešení:

Každou cifru vynásobíme základem soustavy umocněnou exponentem, který se zvyšuje s cifrou, počínaje zprava nulou a jednotlivé cifry takto sečteme:

$$1101_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

$$1101_2 = 8 + 4 + 1$$

$$1101_2 = 13_{10}$$

Nyní jsme ve stavu, že bylo převedeno číslo z dvojkové do desítkové soustavy, což je užitečné, ale to nebyl náš úkol.

Nyní musíme dělit číslo v desítkové soustavě základem soustavy, na kterou chceme převádět (šestnáctková):

$$13_{10} : 16 = 0(13)$$

Výsledek je zbytek po dělení, při dělení orientovaný „odspodu nahoru“, v našem konkrétním jednořádkovém dělení to není důležité vědět, ale ukážeme si i opačný postup zpět na dvojkovou, kde už bude nutné tuto informaci znát.

Teď, když už známe hodnotu v šestnáctkové soustavě, ještě musíme tuto hodnotu převést na odpovídající symbol, protože číslo 13 je z prostoru desítkové soustavy dvojciferné číslo.

$$13_{10} = D_{16}$$

Příklad:

Převeďte číslo D z hexadecimální do binární soustavy.

Řešení:

Postup bude identický s tím, že nyní je základ soustavy šestnáct, takže:

Každou cifru vynásobíme základem soustavy umocněnou exponentem, který se zvyšuje s cifrou, počínaje zprava nulou a jednotlivé cifry takto sečteme:

$$D_{16} = 13 \cdot 16^0$$

$$D_{16} = 8 + 4 + 13$$

$$D_{16} = 13_{10}$$

Nyní musíme dělit číslo v desítkové soustavě základem soustavy, na kterou chceme převádět (binární):

$$13:2 = 6(1)$$

$$6:2 = 3(0)$$

$$3:2 = 1(1)$$

$$1:2 = 0(1)$$

Výsledkem jsou zbytky zapisované „odspodu nahoru“. Tady už je nutné zapisovat zbytky správně, jinak dostaneme špatný výsledek.

$$D_{16} = 1101_2$$

Poznámka: Existují různé triky a pomůcky, jak si převody soustav ulehčit. Například, že hexadecimální číslici tvoří vždy čtyři číslice binární nebo tři číslice oktálové, počínaje zprava. A tudíž:

$$1111 \;\; 1110_2 = FE_{16}$$

$$111 \;\; 111_2 = 77_8$$

Poznámka: Pokud číslic není přesný počet, aby vycházelo rozdělení na trojice nebo čtveřice, můžeme si zleva přidat kolik nul potřebujeme.

1.3 Byty a bity

Hned ze začátku je nutno si vyjasnit, že byty (bajty), jsou násobkem bitů (bitů) ve vztahu:

$$1\ byte\ =\ 8\ bitů$$

Bit je základní jednotka nosiče informací, může nabývat hodnot 1 (taky pravda, vysoké napětí) nebo 0 (taky nepravda, nízké napětí). Toto bipolární fungování dnešních počítačů je důvodem nutnosti převádět čísla z jedné soustavy do druhé.

Byte je poté kombinace hodnot osmi bitů. Tyto hodnoty bitů poté dávají bytu vlastní výpovědní hodnotu. Například díky tomu, že osm bitů vnímáme jako jeden byte, můžeme vyjádřit daleko více stavů než ano nebo ne. Najednou máme k dispozici celou škálu možných výsledků od 00000000_2 až 11111111_2.

Například intenzita světla se jeví jako vhodný ilustrační příklad – 00000000_2 je absolutní tma, 11111111_2 je oslepující světlo a například 10000000_2 je ideální hladina světla.

Pro ještě přesnější vyjádření se i byty spojují dohromady:

Dva byty, tedy šestnáct bitů

Čtyři byty, tedy třicet-dva bitů

Osm bytů, tedy šedesát-čtyři bitů.

1.4 Logická operace – XOR

Jedna z logických operací, které je možno provádět na úrovni bitů a je používání pro šifrování dat je operace exkluzivní OR – XOR. Výsledek operace je pravda (1), jen když máme lichý počet jedniček. (Obecný OR je vždy pravda už pokud je přítomna alespoň jedna jednička.)

Tabulka 1: Pravdivostní tabulka logické operace OR a XOR

A	B	OR	XOR
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

Ukázka: Spočítejte výsledek operace XOR pro čísla $6D_{16}$ a $8C_{16}$.

Nejdříve převedeme čísla z hexadecimální do binární hodnoty.

$$6D_{16} = 01101101_2$$

$$8C_{16} = 10001100_2$$

Teď provedeme operaci XOR na posloupnostech nul a jedniček na odpovídajících pozicích.

$$\begin{array}{r} 01101101 \\ \underline{10001100} \\ 11100001 \end{array}$$

Převedeme výsledek zpět na hexadecimální číslo a máme hotovo.

$$11100001_2 = E1_{16}$$

Poznámka: Jak si můžete všimnout, pokud výsledek XORu "XORneme" se stejnou druhou hodnotou, dostaneme hodnotu původní:

$$\begin{array}{r} 11100001 \\ \underline{10001100} \\ 01101101 \end{array}$$

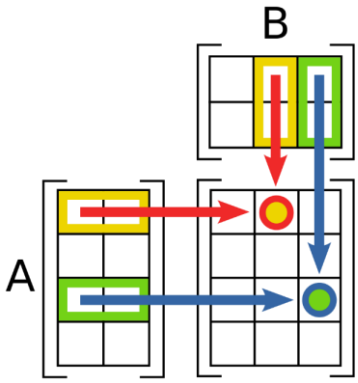
1.5 Maticové násobení

Násobení matic je jedna z hlavních operací s maticemi, stručné vysvětlení jsem získal na wikipedii [1].

Násobení matic probíhá postupně prvek po prvku nové matice tak, že bereme prvky první matice z řádku počítaného prvku a prvky druhé matice ze sloupce počítaného prvku. Tyto prvky spolu vynásobíme a tyto jednotlivé součiny k sobě přičteme. (První prvek řádku násobíme s prvním prvkem sloupce atd.)

Tento postup opakujeme pro každý prvek výsledné matice.

$$\begin{pmatrix} 3 & 2 \\ 4 & 5 \end{pmatrix} * \begin{pmatrix} 2 & 8 \\ 5 & 4 \end{pmatrix} = \begin{pmatrix} 3 \cdot 2 + 2 \cdot 5 & 3 \cdot 8 + 2 \cdot 4 \\ 4 \cdot 2 + 5 \cdot 5 & 4 \cdot 8 + 5 \cdot 4 \end{pmatrix} = \begin{pmatrix} 16 & 32 \\ 33 & 52 \end{pmatrix}$$



Obrázek 1: Znázornění maticového násobení [1]

1.5.1 Maticové násobení v AES

Speciální typ násobení je využíván v AES jako jeden z nástrojů, který napomáhá šifrovat data. Logiku jsem odvodil z dokumentace AES [2].

Pro toto násobení je potřeba poukázat na několik změn.

Prvky matic jsou zapsané v hexadecimální číselné soustavě.

Jednotlivé součiny nesčítáme, ale provádíme mezi nimi operaci XOR.

Platí zde několik pravidel pro počítání:

$$01 = 1$$

$$02 = x$$

$$03 = x + 1$$

$$x^8 = x^4 + x^3 + x + 1$$

Máme předdefinovanou matici, kterou násobíme libovolnou matici.

$$M = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$$

Příklad:

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} * \begin{pmatrix} 25 \\ F1 \\ C5 \\ 04 \end{pmatrix}$$

Spočítáme prvek sloupce na nulté pozici (S):

$$S = 02 \cdot 25 \oplus 03 \cdot F1 \oplus 01 \cdot C5 \oplus 01 \cdot 04$$

Nejdřív musíme spočítat jednotlivé součiny s využitím našich pravidel.

Je na místě si čísla převést do dvojkové soustavy a přepsat tyto čísla jako mocniny x:

$25_{16} = 00100101_2 = 0 \cdot x^7 + 0 \cdot x^6 + 1 \cdot x^5 + 0 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0 = x^5 + x^2 + 1$

$F1_{16} = 11110001_2 = 1 \cdot x^7 + 1 \cdot x^6 + 1 \cdot x^5 + 1 \cdot x^4 + 1 \cdot x^0 = x^7 + x^6 + x^5 + x^4 + 1$

$C5_{16} = 11000101_2 = 1 \cdot x^7 + 1 \cdot x^6 + 1 \cdot x^2 + 1 \cdot x^0$

$04_{16} = 00000100_2 = 1 \cdot x^2$

Teď spočítáme jednotlivé binární mezivýsledky:

$$02 \cdot 25_{16} = x \cdot (x^5 + x^2 + 1) = x^6 + x^3 + x = 01001010_2$$

$$03 \cdot F1_{16} = (x + 1) \cdot (x^7 + x^6 + x^5 + x^4 + 1)$$
$$= (x^8 + x^7 + x^6 + x^5 + x) + (x^7 + x^6 + x^5 + x^4 + 1)$$

Poznámka: Pokud se nám v součtu vyskytne sudý počet jedné mocniny, tuto mocninu škrtneme a dále s ní nepočítáme.

$$(x^8 + \cancel{x^7} + \cancel{x^6} + \cancel{x^5} + x) + (\cancel{x^7} + \cancel{x^6} + \cancel{x^5} + x^4 + 1) = x^8 + x^4 + x + 1$$

Teď je potřeba rozepsat x^8 jak je napsáno v pravidlech:

$$x^8 + x^4 + x + 1 = \cancel{x^4} + x^3 + \cancel{x} + \cancel{1} + \cancel{x^4} + \cancel{x} + \cancel{1} = x^3$$

$$x^3 = 00001000_2$$

Zbylé dva prvky násobíme jedničkou, takže jen přepíšeme jejich binární hodnoty:

$$C5_{16} = 11000101_2$$

$$04_{16} = 00000100_2$$

Teď na těchto binárních hodnotách provedeme hromadný XOR:

$$\begin{array}{l} 01001010 \\ 00001000 \\ 11000101 \\ \underline{00000100} \\ 10000011 \end{array}$$

Tuto hodnotu převedeme zpět na hexadecimální a máme spočítaný první prvek matice:

$$10000011_2 = 83_{16}$$

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} * \begin{pmatrix} 25 \\ F1 \\ C5 \\ 04 \end{pmatrix} = \begin{pmatrix} 83 \\ 8C \\ 49 \\ 53 \end{pmatrix}$$

Abychom získali zpět původní matici, musíme toto násobení provést s inverzní maticí k matici M:

$$M^{-1} = \begin{pmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{pmatrix}$$

Poznámka: Kvůli velikosti prvků matice M^{-1} už nelze využít tohoto výpočetního zjednodušení. Původní stavovou matici získáme tak, že spočítáme: výsledek součinu modulo $(x^8 + x^4 + x^3 + x + 1)$. Modulární aritmetika je vysvětlena v další kapitole.

1.6 Modulární aritmetika

V modulární aritmetice není předmětem zájmu to, co je výsledkem dělení dvou čísel. Předmět našeho zájmu je zbytek po tomto dělení, zajímají nás takzvané zbytkové třídy, kterých je stejný počet jako číslo, kterým dělíme.

1.6.1 Kongruence

Vysvětlení, co je to kongruence, jsem získal od pana Dominika Chládka, který na webu Isibalo vyučuje matematiku pro střední a vysoké školy [3].

Čísla a a b můžeme označit jako kongruentní, pokud mají pro nás za určitých podmínek stejný výpočetní význam.

Čísla a a b jsou vzhledem k modulu n kongruentní, právě když je rozdíl těchto čísel dělitelný číslem n.

$$a \equiv b \; mod \; (n) \Leftrightarrow n|(a-b)$$

Kde:

a a b jsou celá čísla

n je přirozené číslo

Příklad:

Čísla 3 a 15 jsou kongruentní vzhledem k dělení číslem 12:

$$12|(15-3)$$

$$12|(12)$$

Nebo druhou variantou:

$$3:12 = 0(3)$$

$$15:12 = 1(3)$$

Zbytek po dělení je v obou případech 3, takže čísla 3 a 15 jsou kongruentní vzhledem k modulo 12.

$$3 \equiv 15 \; mod \; (12)$$

Obě čísla spadají do zbytkové třídy 3.

1.6.2 Množina zbytkových tříd

Vysvětlení zbytkových tříd jsem získal od pana Dominika Chládka, který na webu Isibalo vyučuje matematiku pro střední a vysoké školy [4].

Tato množina nám říká, že se pohybujeme v modulární aritmetice a kolik zbytkových tříd existuje.

Značíme Z_n kde:

Z označuje, že se jedná o množinu zbytkových tříd

n označuje počet těchto tříd

Příklad: Do jaké zbytkové třídy spadá číslo 153 v Z_{15}?

Řešení:

$$153 \; mod \; 15 = 3$$

Z toho také vyplývá, že

$$153 \equiv 3 \; mod \; (15)$$

Číslo 153 v Z_{15} spadá do zbytkové třídy 3.

1.6.3 Eulerova funkce

Vysvětlení Eulerovy funkce jsem získal od pana Dominika Chládka, který na webu Isibalo vyučuje matematiku pro střední a vysoké školy [5].

Eulerova věta nám říká, že pro každá dvě nesoudělná přirozená čísla a, n platí:

$$a^{\varphi(n)} \equiv 1 \; mod \; (n)$$

Kde:

a, n jsou nesoudělná přirozená čísla

$\varphi(n)$ je hodnota Eulerovy funkce pro číslo n

$mod \; (n)$ je zbytek po dělení číslem n

Výsledek Eulerovy funkce nám říká, kolik přirozených čísel je nesoudělných s číslem, které jsme funkci předali a zároveň jsou menší než toto číslo.

Hodnotu Eulerovu funkci můžeme spočítat pomocí následujících vzorců:

$$\varphi(1) = 1$$

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

$$\varphi(p^a) = (p-1) \cdot p^{a-1}$$

$$\varphi(a \cdot b) = \varphi(a) \cdot \varphi(b)$$

$$\varphi(n^N) = n^{N-1} \cdot \varphi(n)$$

Kde:

n, N jsou přirozená čísla

p je prvočíslo

a, b jsou nesoudělná čísla

1.6.4 Rozšířený Euklidův algoritmus

Algoritmus je využívaný k nalezení největšího společného dělitele, ale vedlejší produktem je i multiplikativní inverze našich čísel. Tuto inverzi se budeme snažit pro naše potřeby zjistit. Algoritmus je podrobněji popsán na webech [6] a [7].

Multiplikativní inverze čísla a je takové číslo b, že vynásobením těchto čísel dostaneme výsledek 1:

$$a \cdot b = 1$$

Například pro racionální číslo 4 je multiplikativní inverzí číslo $\frac{1}{4}$.

Pro naše potřeby budeme ale muset spočítat multiplikativní inverzi ve spojení s modulární aritmetikou, což znamená, že výsledek 1 má být zbytek po dělení tohoto součinu číslem n.

$$a \cdot b \equiv 1 \, mod(n)$$

Kde:

a a n jsou nesoudělná přirozená čísla

Poznámka: Nesoudělná čísla jsou taková, jejichž největší společný dělitel je číslo 1.

Poznámka: Číslo a má inverzi v Z_n , protože čísla a a n jsou nesoudělná.

Příklad:

Najděte multiplikativní inverzi čísla 61 v $Z_{97}\ mod\ (97)$, příklad jsem převzal z webu fimatek, součástí webu [6] je i kalkulačka pro jakákoliv dvě čísla.

Začneme s tím, že si jako první členy posloupností označíme dělitele a jako druhý naše číslo a.

$$a_0 = 97$$
$$a_1 = 61$$

Dále budeme dokola využívat vlastnosti, že každé následující a je lineární kombinací našich počátečních čísel a_0 a a_1 a každé následující a_i je menší než předchozí.

Každé další a_i zjistíme tak, že provedeme výpočet:

$$a_i = a_{i-2} - y_i \cdot a_{i-1}$$

Koeficient y_i volíme tak, aby výsledkem bylo co nejmenší přirozené číslo.

$$a_2 = a_0 - y_2 \cdot a_1 = 97 - 1 \cdot 61 = 36$$

$$a_3 = a_1 - y_3 \cdot a_2 = 61 - 1 \cdot 36 = 25$$

$$a_4 = a_2 - y_4 \cdot a_3 = 36 - 1 \cdot 25 = 11$$

$$a_5 = a_3 - y_5 \cdot a_4 = 25 - 2 \cdot 11 = 3$$

$$a_6 = a_4 - y_6 \cdot a_5 = 11 - 3 \cdot 3 = 2$$

$$a_7 = a_5 - y_7 \cdot a_6 = 3 - 1 \cdot 2 = 1$$

První část máme hotovou, už známe a_i které je rovno jedné.

Teď toto číslo musíme vyjádřit pomocí našich počátečních čísel a_0 a a_1.

$$a_2 = 36 = 97 - 61$$

$$a_3 = 25 = 61 - 36 = 61 - (97 - 61) = 2 \cdot 61 - 97$$

$$a_4 = 11 = 36 - 25 = (97 - 61) - (2 \cdot 61 - 97) = 2 \cdot 97 - 3 \cdot 61$$

$$a_5 = 3 = 25 - 2 \cdot 11 = (2 \cdot 61 - 97 - 2) \cdot (2 \cdot 97 - 3 \cdot 61) = 8 \cdot 61 - 5 \cdot 97$$

$$a_6 = 2 = 11 - 3 \cdot 3 = (2 \cdot 97 - 3 \cdot 61) - 3 \cdot (8 \cdot 61 - 5 \cdot 97) = 17 \cdot 97 - 27 \cdot 61$$

$$a_7 = 1 = 3 - 2 = (8 \cdot 61 - 5 \cdot 97) - (17 \cdot 97 - 27 \cdot 61) = \mathbf{35} \cdot 61 - 22 \cdot 97$$

Číslo 35 je naše hledané číslo. Pro kontrolu můžeme vyzkoušet výpočet:

$$(35 \cdot 61)\ mod\ 97 = 1$$

Poznámka: Pokud by nám číslo vyšlo záporně, přičteme k němu naše a_0.

II. PRAKTICKÁ ČÁST

2 KRYPTOGRAFIE DNEŠNÍ DOBY

Kryptografie po internetu dnes probíhá ve dvou fázích, v první fázi je potřeba navázat bezpečné spojení vzhledem k možnosti odposlechu komunikace a komunikace samotné. V této práci se podíváme podrobně na nejrozšířenější dvojici, která tuto činnost vykonává, kterou je AES pro šifrovanou komunikaci a RSA pro navázání bezpečného spojení.

Nejdříve se ale podíváme na základní terminologii používanou v kryptografii.

2.1 Pojmy kryptografie

2.1.1 Kryptografie

Obor kryptografie má za úkol vymýšlet takové postupy, aby zajistila nezjistitelnost posílaných dat třetí stranou, spadá sem vývoj šifer a systémů, které tyto šifry využívají.

2.1.2 Kryptoanalýza

Obor kryptoanalýza má opačný úkol než kryptografie, hledat bezpečnostní díry a snažit se rozluštit šifry, následně upozornit na nefunkčnost systému, tím zamezit únikům a zneužitím dat, kdyby onen systém byl přijat do provozu.

2.1.3 Zdrojová abeceda

Abeceda je soubor znaků, kterými se snažíme vyjádřit informaci. Zdrojová abeceda je ta, ze které přepisujeme text do jiné. Pro naši běžnou mluvu je zdrojová abeceda jednoduše česká abeceda písmen.

2.1.4 Kódová abeceda

Kódová abeceda může být jen přeházená zdrojová abeceda, nebo přepis na úplně jiné znaky.

Informaci přepíšeme z jedné abecedy do druhé podle jasných, a hlavně jednoznačných pravidel za účelem kódování nebo šifrování.

2.1.5 Kódování

Proces, kdy se například informace z čitelného formátu přepisují do jazyka počítače, tedy posloupností nul a jedniček. Tento proces nemá nic společného se zabezpečením informace proti zneužití třetí stranou.

2.1.6 Šifrování

Proces, kdy se data v jazyku počítače mixují a přepisují, aby byla zajištěna nezjistitelnost přenášené informace třetí stranou (odposlechem). Abychom byli schopní zpětně získat zašifrovaná data, tzv. je dešifrovat, musíme znát postup a potřebné informace. Pokud se pokusíme posloupnosti nul a jedniček přeložit bez dešifrování, získáme jen shluk nesmyslných znaků, obecně nazvaný rozsypaný čaj.

Právě při tomto procesu se využívají algoritmy a šifry, které zkoušejí útočníci (třetí strany) prolomit různými kryptoanalytickými nástroji.

2.1.7 Klíč

Nejdůležitější část šifer jsou klíče. Klíč představuje onu „klíčovou informaci“, jejíž znalost je pro šifrování a dešifrování naprosto nezbytná. Je důležité si uvědomit, že klíč je jakákoliv informace, která je potřeba k šifrování nebo dešifrování. Může být v jakémkoliv formátu – číslo, písmena, samotná znalost použitého šifrovacího algoritmu.

Jako příklad můžeme zmínit Caesarovu šifru, kdy klíčem je samotná znalost, že byl text zašifrován tímto způsobem: Písmeno v šifře je ve skutečnosti písmeno o tři místa vlevo v abecedě.

Příklad: Ahoj se zašifruje jako Dkrm (používáme anglickou abecedu)

2.1.8 Délka klíče

Délka klíče je jeden z parametrů, kterým můžeme kategorizovat šifry, nebo diskutovat o jejich efektivitě, případně bezpečnosti. U jednodušších šifer se jako délka klíče může brát počet znaků v klíčovém slově, u pokročilejších a dnes používaných šifer se spíše bavíme v počtu bitů.

2.1.9 Veřejný klíč

Veřejný klíč je jeden z klíčů tvořící takzvaný klíčový pár pro asymetrické šifrování. Je důležité zdůraznit, že tato informace je veřejně přístupná. Tento klíč využívá odesílatel dat.

Označení klíč je zde lehce zavádějící, pro lepší představu je rozumnější si veřejný klíč představit jako zámek, který příjemce rozdává odesílatelům a ten si tento zámek odemkne svým privátním klíčem.

2.1.10 Soukromý/privátní klíč

Druhým prvkem klíčového páru je právě soukromý klíč. Tato informace je tajná a není nikdy zveřejněna příjemcem. Tento způsob zajišťuje, že jen zamýšlený příjemce je schopen dešifrovat příchozí zprávu. Čili i kdyby odesílatel poslal zprávu zašifrovanou odpovídajícím veřejným klíčem všem možným příjemcům, jen tento příjemce je schopen zprávu přeložit zpět a přečíst si ji.

2.1.11 Symetrické šifrování

Označuje způsob šifrování, kdy se používá symetrický klíč. Zprávu lze dešifrovat stejným klíčem jakým byla šifrována s použitím „opačného“ postupu.

Výhodou těchto šifer je obecně vysoká šifrovací rychlost pro velké objemy posílaných dat.

Zjevnou nevýhodou je předání klíče, který musí mít k dispozici obě strany. Buď klíč pošleme nezašifrovaný, toto předání je velmi náchylné k odposlechu, nebo musíme rozšířit naši komunikaci o metodu pro bezpečný přenos klíče.

2.1.12 Asymetrické šifrování

Způsob šifrování využívající klíčový pár, každá strana má rozdílný klíč. Tyto klíče jsou ale ve vztahu, kdy jeden klíč šifruje a druhý je schopen získat informaci zpět.

Jasnou výhodou je neposílání klíče přes přenosové médium čili potencionální útočník ani nemá příležitost klíč odposlechnout. Šifrovaná data odesílatelů mohou doputovat i ke třetím stranám, ale bez znalosti soukromého klíče není možné tyto informace získat.

Nevýhodou jsou nižší rychlosti šifrování než u symetrických šifer, díky tomu je vhodná jen na posílání menších objemů dat.

2.2 Symetrické šifry

2.2.1 AES

AES neboli Advanced Encryption Standard je nástupce šifry DES, která již není v dnešní době považována za bezpečnou z důvodu malé délky klíče.

AES je už v dnešní době přejaté jméno šifry pod původním jménem Rijndael, vytvořené belgickými kryptografy Joanem Daemenem a Vincentem Rijmenem. Informace o této šifře jsem čerpal z dokumentace AES [2], webu Passportalmsp, kde se autor Solarwinds věnuje

hlavním otázkám o AES [8] a webu TechTarget, kde se autoři Bernstein a Cobb opět snaží přiblížit problematiku AES [9].

Tato šifra vyhrála soutěž o nový standard mezi ostatními uchazeči a tím jí byl přisouzen název AES.

Šifra pracuje s klíči o délce 128, 192, 256 bitů.

Jedná se o blokovou šifru, což znamená, že šifrování dat probíhá v každém bloku dat jinak – v jednotlivých blocích je písmeno zašifrováno různými písmeny.

Jakožto u symetrické šifry probíhá šifrování a dešifrování za pomocí stejného klíče, a tedy je zde už zmíněný problém s bezpečným přenosem klíče k příjemci.

2.2.1.1 Kde se používá

Šifra AES je populární všude, kde chcete, aby byla data bezpečně zašifrována, například ukládání na disk, přenos dat přes internet. Najdete ji ve WiFi, mobilních aplikacích, státních službách, VPN, souborových systémech operačních systému.

2.2.1.2 Matematika za AES

Šifrování probíhá v blocích o pevné velikosti 128 bitů. Celá zpráva je rozdělena do bloků o této velikosti a tyto bloky jsou postupně šifrovány a odesílány.

Znaky (In) jsou po šestnácti poskládány do matice.

$$\begin{pmatrix} In_0 & In_4 & In_8 & In_{12} \\ In_1 & In_5 & In_9 & In_{13} \\ In_2 & In_6 & In_{10} & In_{14} \\ In_3 & In_7 & In_{11} & In_{15} \end{pmatrix}$$

Podle vybraného typu AES si vytvoříme klíč požadovaných rozměrů.

Poznámka: Číslo v označení AES odpovídá velikosti použitého klíče, například AES-128 má klíč o velikosti 128 bitů, což je 16 bytů a zapisuje se do matice 4x4.

Pro AES-128 hodnoty klíče (K) mají formát matice 4x4:

$$\begin{pmatrix} K_0 & K_4 & K_8 & K_{12} \\ K_1 & K_5 & K_9 & K_{13} \\ K_2 & K_6 & K_{10} & K_{14} \\ K_3 & K_7 & K_{11} & K_{15} \end{pmatrix}$$

Pro AES-192 hodnoty klíče (K) mají formát matice 4x6:

$$\begin{pmatrix} K_0 & K_4 & K_8 & K_{12} & K_{16} & K_{20} \\ K_1 & K_5 & K_9 & K_{13} & K_{17} & K_{21} \\ K_2 & K_6 & K_{10} & K_{14} & K_{18} & K_{22} \\ K_3 & K_7 & K_{11} & K_{15} & K_{19} & K_{23} \end{pmatrix}$$

Pro AES-256 hodnoty klíče (K) mají formát matice 4x8:

$$\begin{pmatrix} K_0 & K_4 & K_8 & K_{12} & K_{16} & K_{20} & K_{24} & K_{28} \\ K_1 & K_5 & K_9 & K_{13} & K_{17} & K_{21} & K_{25} & K_{29} \\ K_2 & K_6 & K_{10} & K_{14} & K_{18} & K_{22} & K_{26} & K_{30} \\ K_3 & K_7 & K_{11} & K_{15} & K_{19} & K_{23} & K_{27} & K_{31} \end{pmatrix}$$

Jako první je provedena operace XOR s klíčem:

$$\begin{pmatrix} In_0 & In_4 & In_8 & In_{12} \\ In_1 & In_5 & In_9 & In_{13} \\ In_2 & In_6 & In_{10} & In_{14} \\ In_3 & In_7 & In_{11} & In_{15} \end{pmatrix} \oplus \begin{pmatrix} K_0 & K_4 & K_8 & K_{12} \\ K_1 & K_5 & K_9 & K_{13} \\ K_2 & K_6 & K_{10} & K_{14} \\ K_3 & K_7 & K_{11} & K_{15} \end{pmatrix}$$

Poznámka: I při odlišných délkách klíče vezmeme první čtyři sloupce.

Nyní je už původní text zašifrován klíčem, od této chvíle už čistý text není čitelný. Od této chvíle se matici, která je šifrována říká **stavová matice –** označení pro současnou podobu šifrovaných dat.

Nyní se naše stavová matice dostává do takzvaných rund, což je opakovaní čtyř kroků dokola. Kolikrát tyto kroky opakujeme souvisí opět s vybranou velikostí klíče.

Pro AES-128 je to 9x.

Pro AES-192 je to 11x.

Pro AES-256 je to 13x.

Tyto čtyři kroky jsou:

- Nahrazení bytů.
- Posun řádků.
- Modifikace sloupců.
- Přidání klíče rundy.

Nahrazení bytů

Máme k dispozici tabulku, ve které máme přesně definované hodnoty, kterými nahrazujeme. Této tabulce se říká S-box, konkrétně zde máme S-box pro šifru AES.

Tabulka 2: Rijndael (AES) S-box

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Pomocí této tabulky zaměníme všechny hexadecimální hodnoty v matici. Pokud máme například hexadecimální číslo 1a, je zaměněno za a2 – levá číslice určuje řádek, pravá sloupec.

Tabulka 3: Rijndael (AES) S-box s vyznačeným postupem

	0	1	2	3	4	5	6	7	8	9	0a	0b	0c	0d	0e	0f
0	63	7c	77	7b	f2	6b	6f	c5	30	1	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	4	c7	23	c3	18	96	5	9a	7	12	80	e2	eb	27	b2	75
40	9	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	0	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	2	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	6	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	8
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	3	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Stejným způsobem jsou nahrazena i ostatní čísla za jiná.

Posun řádků

Prvky stavové matice (S_i) jsou přeházeny podle následující logiky:

První řádek zůstane nedotčený.

Druhý řádek posune prvky o jeden vlevo.

Třetí řádek posune prvky o dva vlevo.

Čtvrtý řádek posune prvky o tři vlevo.

$$\begin{pmatrix} S_0 & S_4 & S_8 & S_{12} \\ S_1 & S_5 & S_9 & S_{13} \\ S_2 & S_6 & S_{10} & S_{14} \\ S_3 & S_7 & S_{11} & S_{15} \end{pmatrix} \rightarrow \begin{pmatrix} S_0 & S_4 & S_8 & S_{12} \\ S_5 & S_9 & S_{13} & S_1 \\ S_{10} & S_{14} & S_2 & S_6 \\ S_{15} & S_3 & S_7 & S_{11} \end{pmatrix}$$

Poznámka: Pro přehlednost indexů matice budeme po každém kroku obnovovat indexy tak, aby odpovídali umístění v matici.

Modifikace sloupců

Šifra AES má předdefinovanou matici, která ve používá v následujícím výpočtu.

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix}$$

S touto maticí a na naší současnou stavovou maticí provede následující matematický úkon:

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} * \begin{pmatrix} S_0 & S_4 & S_8 & S_{12} \\ S_1 & S_5 & S_9 & S_{13} \\ S_2 & S_6 & S_{10} & S_{14} \\ S_3 & S_7 & S_{11} & S_{15} \end{pmatrix}$$

Tímto výpočtem získáme novou stavovou matici.

Přidání klíče rundy

Jako čtvrtý krok provedeme opět operaci XOR stavové matice s klíčem. Ovšem tento klíč je úplně jiný než náš původní vymyšlený klíč. Na to, jak se tvoří tento klíč, se podíváme později.

Prvky klíče rundy (Kr_i)

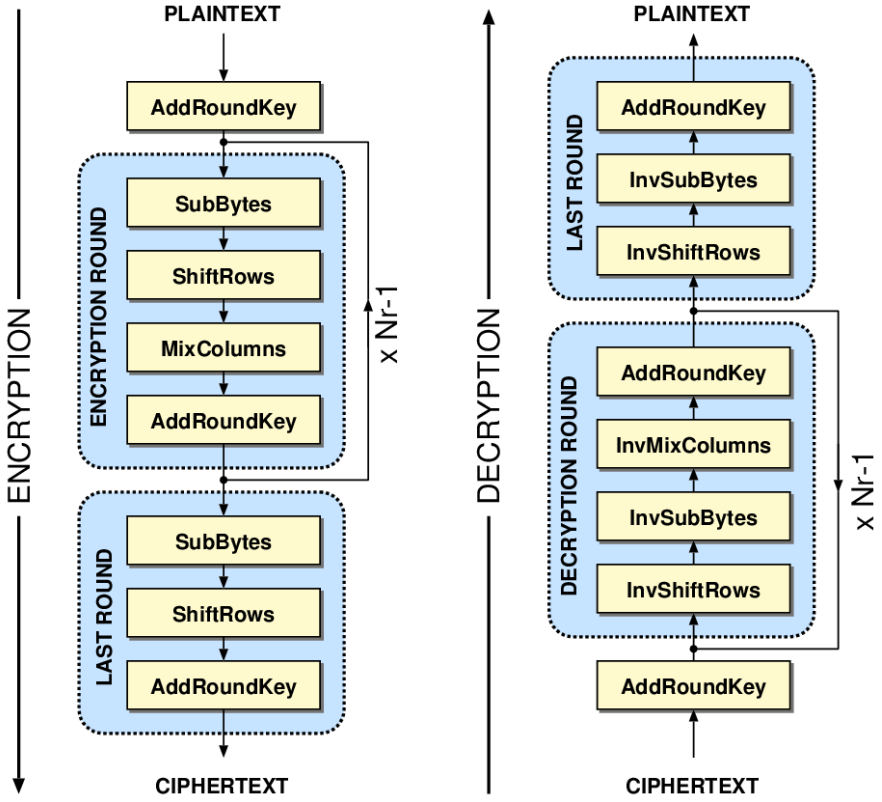
$$\begin{pmatrix} S_0 & S_4 & S_8 & S_{12} \\ S_1 & S_5 & S_9 & S_{13} \\ S_2 & S_6 & S_{10} & S_{14} \\ S_3 & S_7 & S_{11} & S_{15} \end{pmatrix} \oplus \begin{pmatrix} Kr_0 & Kr_4 & Kr_8 & Kr_{12} \\ Kr_1 & Kr_5 & Kr_9 & Kr_{13} \\ Kr_2 & Kr_6 & Kr_{10} & Kr_{14} \\ Kr_3 & Kr_7 & Kr_{11} & Kr_{15} \end{pmatrix}$$

Výsledkem je opět nová stavová matice.

Tyto čtyři kroky se několikrát opakují podle délky klíče.

Závěrečné úpravy

Poslední úpravy jsou tvořené stejným způsobem jako rundy, ale je vynechán krok modifikace sloupců.



Obrázek 2: Postup AES šifrování a dešifrování [11]

Tvorba rundových klíčů

Na začátku máme náš klíč K (Pro AES-128):

$$\begin{pmatrix} K_0 & K_4 & K_8 & K_{12} \\ K_1 & K_5 & K_9 & K_{13} \\ K_2 & K_6 & K_{10} & K_{14} \\ K_3 & K_7 & K_{11} & K_{15} \end{pmatrix}$$

Dále máme předdefinovanou matici Rc (Round constants):

$$\begin{pmatrix} 01 & 02 & 04 & 08 & 10 & 20 & 40 & 80 & 1B & 36 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \end{pmatrix}$$

Dále máme k dispozici náš S-Box, který jsme už používali pro nahrazovaná bytů ve stavové matici viz. Tabulka 2.

Náš úkol je vygenerovat s pomocí těchto nástrojů tolik klíčů, kolik je potřeba, aby při každé rundě i závěrečné úpravě byl použit jiný klíč. Pro AES-128 je to deset nových klíčů.

Postup: Vezmeme poslední sloupec našeho klíče a v něm provedeme rotaci o jeden prvek.

$$\begin{pmatrix} K_{12} \\ K_{13} \\ K_{14} \\ K_{15} \end{pmatrix} \rightarrow \begin{pmatrix} K_{13} \\ K_{14} \\ K_{15} \\ K_{12} \end{pmatrix}$$

Následně nahradíme každý byte pomocí S-boxu (viz. Tabulka 2).

$$\begin{pmatrix} K_{13} \\ K_{14} \\ K_{15} \\ K_{12} \end{pmatrix} \rightarrow \begin{pmatrix} K'_{13} \\ K'_{14} \\ K'_{15} \\ K'_{12} \end{pmatrix}$$

Následně provede dvojitou operaci XOR s prvním sloupcem klíče a prvním sloupce matice Rc:

$$\begin{pmatrix} K_0 \\ K_1 \\ K_2 \\ K_3 \end{pmatrix} \oplus \begin{pmatrix} K'_{13} \\ K'_{14} \\ K'_{15} \\ K'_{12} \end{pmatrix} \oplus \begin{pmatrix} 01 \\ 00 \\ 00 \\ 00 \end{pmatrix}$$

Výsledkem je první sloupec prvního rundového klíče:

$$\begin{pmatrix} Kr_0 \\ Kr_1 \\ Kr_2 \\ Kr_3 \end{pmatrix}$$

Druhý sloupec vznikne XOR operací prvního sloupce rundového klíče (Kr) a druhého sloupce našeho klíče:

$$\begin{pmatrix} K_4 \\ K_5 \\ K_6 \\ K_7 \end{pmatrix} \oplus \begin{pmatrix} Kr_0 \\ Kr_1 \\ Kr_2 \\ Kr_3 \end{pmatrix} = \begin{pmatrix} Kr_4 \\ Kr_5 \\ Kr_6 \\ Kr_7 \end{pmatrix}$$

Třetí sloupec vznikne XOR operací druhého sloupce rundového klíče (Kr) a třetího sloupce našeho klíče:

$$\begin{pmatrix} K_8 \\ K_9 \\ K_{10} \\ K_{11} \end{pmatrix} \oplus \begin{pmatrix} Kr_4 \\ Kr_5 \\ Kr_6 \\ Kr_7 \end{pmatrix} = \begin{pmatrix} Kr_8 \\ Kr_9 \\ Kr_{10} \\ Kr_{11} \end{pmatrix}$$

Čtvrtý sloupec vznikne XOR operací třetího sloupce rundového klíče (Kr) a čtvrtého sloupce našeho klíče:

$$\begin{pmatrix} K_{12} \\ K_{13} \\ K_{14} \\ K_{15} \end{pmatrix} \oplus \begin{pmatrix} Kr_8 \\ Kr_9 \\ Kr_{10} \\ Kr_{11} \end{pmatrix} = \begin{pmatrix} Kr_{12} \\ Kr_{13} \\ Kr_{14} \\ Kr_{15} \end{pmatrix}$$

V tento moment jsme úspěšně vytvořili klíč pro první rundu

$$\begin{pmatrix} Kr_0 & Kr_4 & Kr_8 & Kr_{12} \\ Kr_1 & Kr_5 & Kr_9 & Kr_{13} \\ Kr_2 & Kr_6 & Kr_{10} & Kr_{14} \\ Kr_3 & Kr_7 & Kr_{11} & Kr_{15} \end{pmatrix}$$

Další rundové klíče jsou analogicky vytvořené tak, že klíč předchozí rundy v tomto postupu zaujme funkci našeho počátečního klíče.

2.2.1.3 Ukázkový příklad

Pro jednoduchost budeme text psát velkými písmeny bez diakritiky a bez tečky.

Mějme čistý text: „DNES JE PEKNE.“

Tento text převedeme na hexadecimální znaky: 44 4E 45 53 20 4A 45 20 50 45 4B 4E 45 2E

Máme čtrnáct hexadecimálních znaků, ale potřebujeme jich šestnáct, tento problém vyřešíme přidáním dalších znaků, například znaků reprezentující písmeno X (58).

Náš text: 44 4E 45 53 20 4A 45 20 50 45 4B 4E 45 2E 58 58

Tabulka 4: Část ASCII tabulky

Znak	ASCII Hx	Znak	ASCII Hx	Znak	ASCII Hx	Znak	ASCII Hx
.	2E	G	47	N	4E	U	55
A	41	H	48	O	4F	V	56
B	42	I	49	P	50	W	57
C	43	J	4A	Q	51	X	58
D	44	K	4B	R	52	Y	59
E	45	L	4C	S	53	Z	5A
F	46	M	4D	T	54		

Nyní tento řetězec znaků převedeme do matice po sloupcích odshora dolů:

$$Vstup = \begin{pmatrix} 44 & 20 & 50 & 45 \\ 4E & 4A & 45 & 2E \\ 45 & 45 & 4B & 58 \\ 53 & 20 & 4E & 58 \end{pmatrix}$$

Teď je potřeba si vytvořit klíč pro AES-128.

28 48 2B 4B 62 52 65 53 68 56 6D 59 71 33 74 36

$$Klíč = \begin{pmatrix} 28 & 62 & 68 & 71 \\ 48 & 52 & 56 & 33 \\ 2B & 65 & 6D & 74 \\ 4B & 53 & 59 & 36 \end{pmatrix}$$

Můžeme začít šifrovat. První krok je provést operaci XOR mezi naším textem a klíčem:

$$\begin{pmatrix} 44 & 20 & 50 & 45 \\ 4E & 4A & 45 & 2E \\ 45 & 45 & 4B & 58 \\ 53 & 20 & 4E & 58 \end{pmatrix} \oplus \begin{pmatrix} 28 & 62 & 68 & 71 \\ 48 & 52 & 56 & 33 \\ 2B & 65 & 6D & 74 \\ 4B & 53 & 59 & 36 \end{pmatrix} = \begin{pmatrix} S_0 & S_4 & S_8 & S_{12} \\ S_1 & S_5 & S_9 & S_{13} \\ S_2 & S_6 & S_{10} & S_{14} \\ S_3 & S_7 & S_{11} & S_{15} \end{pmatrix}$$

$$S_0 = 44_{16} \oplus 28_{16}$$

$$S_0 = 01000100_2 \oplus 00101000_2$$

$$S_0 = 01101100_2 = 6C_{16}$$

Po vypočítání všech prvků máme stavovou matici:

$$\begin{pmatrix} 6C & 42 & 38 & 34 \\ 06 & 18 & 13 & 1D \\ 6E & 20 & 26 & 2C \\ 18 & 73 & 17 & 6E \end{pmatrix}$$

Tato matice vstupuje do rund:

Poznámka: Pro stručnost bude ukázána jedna runda, postup by byl totožný pro všechny další.

Nahrazení bytů

Pomocí S-boxu nahradíme všechny byty za nové (viz. Tabulka 2).

$$\begin{pmatrix} 6C & 42 & 38 & 34 \\ 06 & 18 & 13 & 1D \\ 6E & 20 & 26 & 2C \\ 18 & 73 & 17 & 6E \end{pmatrix} \rightarrow \begin{pmatrix} 50 & 2C & 07 & 18 \\ 6F & AD & 7D & A4 \\ 9F & B7 & F7 & 71 \\ AD & 8F & F0 & 9F \end{pmatrix}$$

Posun řádků

Teď je čas na posun řádků: První řádek necháme, každý další se rotuje vždy o jeden prvek navíc.

$$\begin{pmatrix} 50 & 2C & 07 & 18 \\ 6F & AD & 7D & A4 \\ 9F & B7 & F7 & 71 \\ AD & 8F & F0 & 9F \end{pmatrix} \rightarrow \begin{pmatrix} 50 & 2C & 07 & 18 \\ AD & 7D & A4 & 6F \\ F7 & 71 & 9F & B7 \\ 9F & AD & 8F & F0 \end{pmatrix}$$

Modifikace sloupců

S využitím maticového násobení v AES (kapitola 1.5.1) spočítáme výsledek násobení.

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} * \begin{pmatrix} 50 & 2C & 07 & 18 \\ AD & 7D & A4 & 6F \\ F7 & 71 & 9F & B7 \\ 9F & AD & 8F & F0 \end{pmatrix} = \begin{pmatrix} 25 & 03 & E9 & C6 \\ 8C & A8 & 61 & F4 \\ B2 & 4F & 0C & 09 \\ 9F & 39 & 37 & 0B \end{pmatrix}$$

$$S_0 = 2 \cdot 50 \oplus 3 \cdot AD \oplus F7 \oplus 9F$$

$$S_0 = 25$$

Stejným způsobem dopočítáme všechny prvky nové stavová matice.

Přidání klíče rundy

$$Klíč = \begin{pmatrix} 28 & 62 & 68 & 71 \\ 48 & 52 & 56 & 33 \\ 2B & 65 & 6D & 74 \\ 4B & 53 & 59 & 36 \end{pmatrix}$$

Postup: Vezmeme poslední sloupec našeho klíče a v něm provedeme rotaci o jeden prvek.

$$\begin{pmatrix} 71 \\ 33 \\ 74 \\ 36 \end{pmatrix} \rightarrow \begin{pmatrix} 33 \\ 74 \\ 36 \\ 71 \end{pmatrix}$$

Následně nahradíme každý byte pomocí S-boxu (viz. Tabulka 2).

$$\begin{pmatrix} 33 \\ 74 \\ 36 \\ 71 \end{pmatrix} \rightarrow \begin{pmatrix} C3 \\ 92 \\ 05 \\ A3 \end{pmatrix}$$

Následně provede dvojitou operaci XOR s prvním sloupcem klíče a prvním sloupce matice Rc:

$$\begin{pmatrix}28\\48\\2B\\4B\end{pmatrix}\oplus\begin{pmatrix}C3\\92\\05\\A3\end{pmatrix}\oplus\begin{pmatrix}01\\00\\00\\00\end{pmatrix}$$

$$Kr_0 = 28 \oplus C3 \oplus 01$$

$$Kr_0 = 00101000 \oplus 11000011 \oplus 00000001$$

$$Kr_0 = 11101010$$

$$Kr_0 = EA$$

Výsledkem je první sloupec prvního rundového klíče:

$$\begin{pmatrix}EA\\DA\\2E\\E8\end{pmatrix}$$

Druhý sloupec vznikne XOR operací prvního sloupce rundového klíče (Kr) a druhého sloupce našeho klíče:

$$\begin{pmatrix}62\\52\\65\\53\end{pmatrix}\oplus\begin{pmatrix}EA\\DA\\2E\\E8\end{pmatrix}=\begin{pmatrix}88\\88\\4B\\BB\end{pmatrix}$$

Třetí sloupec vznikne XOR operací druhého sloupce rundového klíče (Kr) a třetího sloupce našeho klíče:

$$\begin{pmatrix}68\\56\\6D\\59\end{pmatrix}\oplus\begin{pmatrix}88\\88\\4B\\BB\end{pmatrix}=\begin{pmatrix}E0\\DE\\26\\E2\end{pmatrix}$$

Čtvrtý sloupec vznikne XOR operací třetího sloupce rundového klíče (Kr) a čtvrtého sloupce našeho klíče:

$$\begin{pmatrix}71\\33\\74\\36\end{pmatrix}\oplus\begin{pmatrix}E0\\DE\\26\\E2\end{pmatrix}=\begin{pmatrix}91\\ED\\52\\D4\end{pmatrix}$$

V tento moment jsme úspěšně vytvořili klíč pro první rundu.

$$\begin{pmatrix} EA & 88 & E0 & 91 \\ DA & 88 & DE & ED \\ 2E & 4B & 26 & 52 \\ E8 & BB & E2 & D4 \end{pmatrix}$$

Teď s tímto klíčem provedeme stejnou XOR operaci jako s naším původním klíčem.

$$\begin{pmatrix} 25 & 03 & E9 & C6 \\ 8C & A8 & 61 & F4 \\ B2 & 4F & 0C & 09 \\ 9F & 39 & 37 & 0B \end{pmatrix} \oplus \begin{pmatrix} EA & 88 & E0 & 91 \\ DA & 88 & DE & ED \\ 2E & 4B & 26 & 52 \\ E8 & BB & E2 & D4 \end{pmatrix} = \begin{pmatrix} CF & 8B & 09 & 57 \\ 56 & 20 & BF & 19 \\ 9C & 04 & 2A & 5B \\ 77 & 82 & D5 & DF \end{pmatrix}$$

Po průchodu první rundou se náš čistý text zašifroval takto:

$$\begin{pmatrix} 44 & 20 & 50 & 45 \\ 4E & 4A & 45 & 2E \\ 45 & 45 & 4B & 58 \\ 53 & 20 & 4E & 58 \end{pmatrix} \rightarrow \begin{pmatrix} CF & 8B & 09 & 57 \\ 56 & 20 & BF & 19 \\ 9C & 04 & 2A & 5B \\ 77 & 82 & D5 & DF \end{pmatrix}$$

Tento proces proběhne ještě osmkrát při AES-128 (plus závěrečné úpravy, kde chybí krok modifikace sloupců), ale můžeme vidět, že už po prvním kole je šifrovaný text naprosto odlišný.

Dešifrování probíhá stejnými kroky v opačném pořadí a s inverzní logikou (řádky posouváme na opačnou stranu, v S-boxu hledáme, ze kterého sloupce a řádku je hexadecimální číslo složeno apod.)

Ve zdroji [10] je uveden ještě další příklad.

2.2.1.4 Slabiny a možnosti útoku na AES

Dosud kromě útoku postranním kanálem (což není vina šifry, jedná se o únik informací) nebyl zjištěn úspěšný útok na AES. Šifra je dokonce tak bezpečná, že se o ní mluví i jako o postkvantové, pak obzvlášť její varianta AES-256. Útočník by musel zkusit všechny klíče což je 2^{256} možných klíčů, pokud by správný klíč byl ten poslední. Aby útočník tolik klíčů vyzkoušel, potřeboval by miliardy let zkoušení. Pokud by klíč našel v půlce zkoušení, stále je to 2^{255} možností. Z těchto skutečností vyplývá, že útok na AES hrubou silou je vzhledem počtu možností téměř nemožný. Ostatní úspěšné možnosti útoku zatím nebyly objeveny.

2.3 Asymetrické šifry

2.3.1 RSA

Dnešní standard asymetrického šifrování RSA nese jméno podle svých tvůrců – Rives, Shamir a Adleman. Tito informatici přišli s algoritmem fungujícím na základě Eulerovy funkce a problému faktorizace. Vědomosti jsem čerpal z webu pana Neckáře Algoritmy [12], web, kde autoři rozebírají i výhody a nevýhody RSA [13] a web BinaryTerms od autora Neha T, který popisuje, jak funguje RSA [14].

Příklad jsem převzal z webu pana Neckáře [12].

Klíč je tvořen dvojicí čísel a dvě tyto dvojce tvoří klíčový pár.

2.3.1.1 Kde se používá

Klíčová pár je velmi důležitá vlastnost při praktickém využití této šifry, kterým je výměna klíče pro komunikaci pomocí symetrické šifry, typicky AES.

Zpráva posílaná pomocí RSA je zašifrována veřejným klíčem příjemce a ten následně využije svůj privátní klíč k dešifrování.

Z tohoto postupu vyplývá, že je nutná znalost soukromého klíče k přečtení zprávy – máme ochranu před odposlechem třetí strany.

Problémem této šifry je pomalá rychlost šifrování a omezená velikost šifrované zprávy. Proto je v praxi využívána hlavně jako bezpečný způsob výměny symetrického klíče, pomocí kterého probíhá samotná komunikace.

2.3.1.2 Matematika za RSA

Tato šifra je realizována s využitím modulární aritmetiky.

- Příjemce si zvolí dvě prvočísla p a q a spočítají jejich součin n.
- Spočítá hodnotu Eulerovi funkce φ pro součin p a q.
- Zvolí si další prvočíslo e, které je nesoudělné a menší než hodnota Eulerovy funkce pro naše prvočísla p a q.

  Tj. $e \nmid \varphi(p \cdot q)$

- Spočítá multiplikativní inverzi d prvočísla e v $Z_{\varphi(p \cdot q)}$.

  Tj. $d \cdot e \equiv 1 \, mod \, (\varphi(p \cdot q))$

Dvojice n a e tvoří veřejný klíč.

Dvojice n a d tvoří soukromý klíč.

2.3.1.3 Ukázkový příklad

Vybereme si dvě prvočísla:

$$p = 17$$

$$q = 19$$

Spočítáme jejich součin:

$$n = p \cdot q = 323$$

S využitím vzorce

$$\varphi(a \cdot b) = \varphi(a) \cdot \varphi(b)$$

spočítáme hodnotu Eulerovy funkce $\varphi(n)$ pro n:

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

$$\varphi(323) = (17 - 1) \cdot (19 - 1)$$

$$\varphi(323) = 16 \cdot 18$$

$$\varphi(323) = 288$$

Teď si zvolíme další prvočíslo. Musí splňovat, že je nesoudělné s $\varphi(n)$ a je menší než $\varphi(n)$.

$$e = 37$$

Nakonec potřebujeme spočítat multiplikativní inverzi čísla e v $Z_{\varphi(n)}$ s použitím rozšířeného Euklidova algoritmu.

$$d = e^{-1} \, mod(\varphi(n))$$

$$d = 37^{-1} \, mod(288)$$

$$d = 109 \, mod(288)$$

$$d = 109$$

Nyní už jsme spočítali všechno, co jsme potřebovali znát, v tuto chvíli můžeme zveřejnit veřejný klíč, který je tvořen uspořádanou dvojicí (n, e).

Soukromý klíč tvořený dvojicí (n, d) si ponecháme a čekáme na příchozí zprávy zašifrované veřejným klíčem.

Příklad komunikace:

Chci poslat číslo 74 příjemci. Vezmu si veřejně přístupný veřejný klíč $(323, 37)$.

Provedu matematickou operaci:

$$Zpráva = (\check{c}ist\acute{y}\ text)^e mod\ (n)$$

$$Zpráva = 74^{37} mod\ (323)$$

$$Zpráva = 245$$

Toto číslo doputuje k příjemci, ten provede analogický postup s využitím soukromého klíče (323, 109):

$$\check{C}ist\acute{y}\ text = (Zpráva)^d mod\ (323)$$

$$\check{C}ist\acute{y}\ text = 245^{109} mod\ (323)$$

$$\check{C}ist\acute{y}\ text = 74$$

Hotovo – zpráva byla zašifrována, dešifrována, doputovala a nebyl posílán žádný klíč.

Na závěr je nutné si uvědomit, že tento matematický výpočet probíhá na obou koncích, pokud chce příjemce odpovědět zpět odesílateli, používá jeho veřejný klíč. V žádném případě nesmí komunikovat zpět pomocí svého privátního klíče – zprávu by si mohl přečíst kdokoliv, protože druhá půlka klíčového páru je veřejná pro všechny, a tudíž každý je potencionálním vlastníkem potřebné informace k dešifrování.

Dá se říct, že při jedné komunikaci jsou ve hře dva klíčové páry, každá strana má svůj privátní a veřejný klíč.

2.3.1.4 Slabiny a možnosti útoku na RSA

Síla nebo slabina RSA spočívá ve výběru prvočísel p a q, jak jsme si před chvílí ukázali. Princip toho, jak prolomit šifru je najít číslo, kterým je tvořený soukromý klíč. K dispozici

jako útočník máme veřejný klíč, který je tvořen součinem n a voleným prvočíslem e. Víme, že číslo, které hledáme je multiplikativní inverzí prvočísla e. Zatím to nevypadá moc složitě.

$$(n, e) \rightarrow (n, d)$$

$$(323{,}37) \rightarrow (323, ?)$$

$$d = e^{-1}\, mod(\varphi(n))$$

Při další úvaze si ale uvědomíme, že abychom získali multiplikativní inverzi prvočísla e (v našem případě 37) musíme být schopní provést výpočet hodnoty Eulerovy funkce pro součin počátečních prvočísel p a q:

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

$$\varphi(323) = (p-1) \cdot (q-1)$$

Kde a jak ale zjistíme prvočísla, ze kterých tento součin vzniknul?

Víme, že součin n vzniknul násobením dvou prvočísel, což znamená, že jen tyto dvě prvočísla budou schopné součin n dělit.

Můžeme tedy zkoušet dělit součin n postupně každým prvočíslem od 2 dokud nenarazíme na situaci, že zbytek po dělení bude nula:

$$323\, mod(2) = 1$$

$$323\, mod(3) = 2$$

$$323\, mod(5) = 3$$

$$323\, mod(7) = 1$$

$$323\, mod(11) = 4$$

$$323\, mod(13) = 11$$

$$323\, mod(17) = 0$$

Teď, když jsme postupným zkoušením našli jedno z prvočísel, je snadné dopočítat druhé:

$$323 : 17 = 19$$

Teprve teď jsme schopní spočítat hodnotu Eulerovy funkce pro součin n, který jsme znali na začátku:

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

$$\varphi(323) = (17-1) \cdot (19-1)$$

$$\varphi(323) = 16 \cdot 18$$

$$\varphi(323) = 288$$

Teď stejným způsobem, jako při tvorbě klíčů, soukromý klíč dopočítáme:

$$d = e^{-1}\, mod(\varphi(n))$$

$$d = 37^{-1}\, mod(288)$$

$$d = 109\, mod(288)$$

$$d = 109$$

V tento moment jsme prolomili ochranu RSA a můžeme rozšifrovat odposlechnutá data jako útočník.

Co stojí na naší straně jako vývojářů RSA je problém faktorizace, což je proces, při kterém rozkládáme čísla na jejich prvočíselné činitele. Tento problém spočívá v tom, že není zatím znám (je možné, že existuje) postup jak v reálném (dosažitelném) čase rozložit velká čísla na součin.

V našem případě to bylo zjistit prvočísla, ze kterých vzniknul součin $n = 323$.

Pokud bychom jako prvočísla zvolili taková prvočísla, aby jejich součinem bylo obrovské číslo (v rámci stovek číslic). Proces zjišťování prvočísel by byl poté pro útočníka neřešitelný problém (v reálném čase).

Obecně se tento problém dá zjednodušeně formulovat jako: Je snadné spočítat součin dvou čísel, ale je extrémně náročné zjistit, z jakých čísel součin vzniknul.

Výhody:

Je to skvělý doplněk k blokové AES šifře, kdy AES je rychlejší, bezpečnější, ale nemá, jak dostat na druhou stranu svůj symetrický klíč. RSA při komunikaci zajistí bezpečný přenos symetrického klíče za pomoci svého asymetrického fungování a tím řeší hlavní nedostatek šifry AES. Společně dnes tvoří zlatý standard šifrování dat po internetu.

Nevýhody:

Šifra RSA je velmi zranitelná proti útokům kvantových počítačů, problém faktorizace je pro ně mnohonásobně jednodušší. Při uvedení kvantových počítačů do běžného provozu je nutno od šifry RSA okamžitě upustit a hledat náhradu v postkvantové kryptografii.

3 KRYPTOGRAFIE BUDOCNOSTI

3.1 Postkvantové algoritmy

Algoritmy, jejichž bezpečnost není významně oslabena příchodem kvantových počítačů, označujeme jako postkvantové. Matematické problémy, na kterých jsou tyto algoritmy založeny musí být takové, aby ani mnohonásobně výkonnější kvantové počítače, vůči dnešním binárním, nebyly dostatečné, aby problém vyřešily reálném čase.

Zde si dopodrobna rozebereme, jak funguje šifra NTRU, přesněji její část NTRUEnrypt, která je slibný kandidát pro nahrazení šifry RSA, v případě, že se kvantové počítače začnou běžně využívat.

3.1.1 NTRU

Konkrétně NTRUEncrypt, což je algoritmus pro šifrování dat založený na problému nejkratšího vektoru, je jedním z algoritmů, které nepodlehnou kvantové výpočetní rychlosti.

Pro tuto kapitolu o NTRU jsem informace získával z knížky od tvůrců NTRU pana Hoffsteina, paní Pipherové a pana Silvermana [15].

3.1.1.1 Matematika za NTRU

Strana příjemce si vybere čtyři přirozená čísla (N, p, q, d).

Tyto čísla jsou veřejná pro všechny, kde:

N je prvočíslo a určuje maximální stupeň polynomu tak, že maximální stupeň je nejvýše

$N - 1$ řádu

q musí být nesoudělné s p a N

musí platit: $q > (6 \cdot d + 1) \cdot p$

Dále si příjemce sestrojí dva polynomy F a G, kde:

koeficienty těchto polynomů jsou -1, 0 nebo 1

Počet těchto koeficientů nám určuje naše číslo d

Pro *F* platí:

počet kladných koeficientů je $d + 1$

počet záporných koeficientů je d

zbytek koeficientů je roven 0

pro tento polynom existuje inverzní polynom v Z_q a Z_p

Poznámka: Jak je myšlen inverzní polynom je ukázáno v příkladu.

Poznámka: Pokud inverze neexistuje, sestrojíme si nový polynom a zkusíme znovu.

Pro *G* platí:

počet kladných koeficientů je d

počet záporných koeficientů je d

zbytek koeficientů je roven 0

Spočítáme si inverze polynomu F: $F_q^{'}$ a $F_p^{'}$.

Provedeme výpočet polynomu H:

$$H = F_q^{'} * G$$

Tento polynom H poskytneme odesílatelům jako veřejný klíč.

Strana odesílatele si získá veřejný klíč H a svou zprávu M.

Koeficienty M se upraví tak, aby splňovaly: $-\frac{1}{2}p < M_i \leq \frac{1}{2}p$

Odesílatel si zvolí polynom R, kde:

koeficienty tohoto polynomu jsou -1, 0 nebo 1.

počet kladných koeficientů je d

počet záporných koeficientů je d

Odesílatel provede výpočet šifrované zprávy e:

$$e \equiv p \cdot R \cdot H + M \; mod \; (q)$$

Tento polynom E odešle příjemci.

Příjemce má nyní k dispozici svůj soukromý klíč $(F, F_p^{'})$ a šifrovanou zprávu E.

Jako první provede výpočet mezivýsledku A:

$$A \equiv (F \cdot E) \; mod \; (q)$$

Nakonec provedeme následující výpočet, abychom získali B.

$$A \equiv (F_p^{'} \cdot A) \; mod \; (p)$$

Pokud jsme dodrželi všechny pravidla, B je totožné s původní zprávou.

3.1.1.2 Ukázkový příklad

Jakožto příjemce si zvolíme naše čtyři čísla N, p, q, d tak, aby splňovali všechny výše uvedené podmínky:

$$(N, p\,, q\,, d) = (7, 3, 41, 2)$$

Zkontrolujeme si nerovnost pro q a d:

$$q > (6 \cdot d + 1) \cdot p$$

$$41 > 39$$

Teď si vymyslíme polynomy F a G v souladu s podmínkami:

$$F = x^6 - x^4 + x^3 + x^2 - 1$$

$$G = x^6 + x^4 - x^2 - x$$

Spočítáme si inverze polynomu F v modulu q a modulu p:

$$F_q^{'} = 8x^6 + 26x^5 + 31x^4 + 21x^3 + 40x^2 + 2x + 37$$

$$F_p^{'} = x^6 + 2x^5 + x^3 + x^2 + x + 1$$

Můžeme si tyto inverze ověřit vynásobením těchto polynomů a vydělením odpovídajícím modulem, například $F_q^{'}$:

$$F \cdot F_q^{'} \equiv 1 \; mod \; (q)$$

$$\begin{aligned}(x^6 - x^4 + x^3 + x^2 - 1) \cdot (8x^6 + 26x^5 + 31x^4 + 21x^3 + 40x^2 + 2x + 37) \\ \equiv (8x^{12} - 26x^{11} + 31x^{10} + 21x^9 + 40x^8 + 2x^7 + 37x^6) + (-8x^{10} \\ - 26x^9 - 31x^8 - 21x^7 - 40x^6 - 2x^5 - 37x^4) + (8x^9 + 26x^8 + 31x^7 \\ + 21x^6 + 40x^5 + 2x^4 + 37x^3) + (8x^8 + 26x^7 + 31x^6 + 21x^5 + 40x^4 \\ + 2x^3 + 37x^2) + (-8x^6 - 26x^5 - 31x^4 - 21x^3 - 40x^2 - 2x - 37)\end{aligned}$$

Teď ovšem nastává problém, máme v našem získaném polynomu mocninu až dvanáctého řádu, ale naše N mocniny omezuje na $N - 1$ řád. Řešením je začít mocniny počítat „nanovo“ od mocniny sedmého řádu, takže $x^7 = x^0$, $x^8 = x^1$ atd.

Polynom tedy podle tohoto přepíšeme:

$$\begin{aligned}(8x^5 - 26x^4 + 31x^3 + 21x^2 + 40x + 2 + 37x^6) \\ + (-8x^3 - 26x^2 - 31x - 21 - 40x^6 - 2x^5 - 37x^4) \\ + (8x^2 + 26x + 31 + 21x^6 + 40x^5 + 2x^4 + 37x^3) \\ + (8x + 26 + 31x^6 + 21x^5 + 40x^4 + 2x^3 + 37x^2) + (-8x^6 - 26x^5 \\ - 31x^4 - 21x^3 - 40x^2 - 2x - 37)\end{aligned}$$

Teď dáme dohromady koeficienty jednotlivých mocnin:

$$(37 - 40 + 21 + 31 - 8)x^6 = 41x^6$$

$$(8 - 2 + 40 + 21 - 26)x^5 = 41x^5$$

$$(-26 - 37 + 2 + 40 - 31)x^4 = 0x^4$$

$$(31 - 8 + 37 + 2 - 21)x^3 = 41x^3$$

$$(21 - 26 + 8 + 37 - 40)x^2 = 0x^2$$

$$(40 - 31 + 26 + 8 - 2)x^1 = 41x^1$$

$$2 - 21 + 31 + 26 - 37 = 1$$

Výsledný polynom:

$$(41x^6 + 41x^5 + 41x^3 + 41x^1 + 1)\ mod\ (41)$$

$$(0x^6 + 0x^5 + 0x^3 + 0x^1 + 1)\ mod\ (41)$$

$$(1) mod (41) = 1$$

Ano, zkontrolovali jsme, že tento polynom $F_q^{'}$ je skutečně multiplikativní inverzí k polynomu F vzhledem k modulu q.

Teď spočítáme polynom H:

$$H = F_q^{'} \cdot G$$

$$H = 20x^6 + 40x^5 + 2x^4 + 38x^3 + 8x^2 + 26x + 30$$

Polynom H tvoří veřejný klíč, soukromý klíč jsou naše polynomy $(F, F_p^{'})$.

Odesílatel nám chce poslat zprávu, která je reprezentovaná jako polynom M:

$$M = -x^5 + x^3 + x^2 - x + 1$$

Dále si vymyslí polynom R, splňující všechny podmínky, který slouží pro zašifrování zprávy pro přenos:

$$R = x^6 - x^5 + x - 1$$

Následně provede výpočet šifrované zprávy e:

$$E \equiv p \cdot R \cdot H + M\ mod\ (q)$$

$$E = 31x^6 + 19x^5 + 4x^4 + 2x^3 + 40x^2 + 3x + 25$$

Tuto šifrovanou zprávu E posílá příjemci.

Příjemce dostane zprávu E a má k dispozici svůj soukromý klíč $(F, F_p^{'})$.

První spočítáme mezivýsledek A:

$$A \equiv F \cdot E\ (mod\ q)$$

$$A \equiv x^6 + 10x^5 - 8x^4 - x^3 - x^2 + x + 1$$

Tento mezivýsledek vynásobíme $F_p^{'}$ a podělíme modulem p.

$$B \equiv F_p^{'} \cdot A\ (mod\ p)$$

$$B \equiv -x^5 + x^3 + x^2 - x + 1 \equiv M$$

3.1.1.3 Srovnání s RSA

Jelikož je pohlíženo na NTRU jako náhradu za RSA, je na místě srovnávat tyto algoritmy vůči sobě.

Problém, na kterém je NTRU založen, nebude vyřešen příchodem kvantových počítačů na rozdíl od problému RSA.

NTRU je rychlejší než RSA ve výpočtech – U RSA počet operací zvyšuje spolu s délkou klíče kubicky (x^3), u NTRU jen kvadraticky (x^2).

Problém s nahrazením RSA tímto algoritmem je, že se jedná o poměrně novou šifru, takže garance bezpečnosti není ještě zcela potvrzena (je možné, že existují slabiny, o kterých ještě nikdo neví).

3.2 Homomorfní šifrování

3.2.1 Úvod do problematiky

Znalosti k této kapitole jsem čerpal z odborného článku VUT v Brně [16].

První myšlenka o takzvaném homomorfním šifrování prezentovala část autorů šifry RSA Rivest a Adleman spolu s profesorem Michaelem Dertouzem v roce 1978.

Zatímco se ryze postkvantové algoritmy pokouší řešit bezpečnostní problémy stávajících šifer, které se objeví s příchodem kvantových počítačů, se algoritmy homomorfního šifrování snaží utajit posílané informace i před příjemcem.

Jednoduchou představou, proč je to praktické a podporuje to soukromí na internetu, je příklad ze života, když chceme přes poštu poslat balík. Nechceme, aby si obsah balíku četl nebo prohlížel zaměstnanec pošty, ale chceme, aby s ním mohl nakládat, aby díky němu doputoval ke koncovému příjemci.

Ve světě internetu je to hlavně prostředí cloudových úložišť, kde jsou naše, někdy i velmi soukromá, data uložena v čistém formátu, i když by nemusely, jelikož si je nikdo správně číst nemá.

Aby ovšem něco takové fungovalo, musí být data zašifrována takovým způsobem, aby bylo operacemi s kryptogramem (zašifrovaným textem) dosaženo stejných výsledků jako operacemi s čistým (otevřeným) textem.

Hlavní problém, který se snaží řešit homomorfní šifrování je, že vlastně příjemci našich dat nevěříme, ale zároveň potřebujeme, aby s nimi pracoval.

Příklad: Představme si online kalkulačku. Máme čísla 10 a 5 chceme od kalkulačky zjistit podíl těchto čísel. Zároveň ale nechceme, aby kalkulačka tato naše čísla znala.

Řešení: Umocníme čísla číslem 2, a tato čísla kalkulačce pošleme.

Kalkulačka spočítá $100 : 25 = 4$

Vrátí nám číslo 4.

My ale víme, že jsme skutečné čísla skryli tím, že byly umocněny číslem 2. Provedeme tedy opačnou operaci a máme náš výsledek 2.

Kalkulačka neznala skutečná čísla.

Toto byl velmi zjednodušený příklad na ukázku fungování homomorfního šifrování.

V současné době existují dvě možnosti, jak ukládat data na internet:

1. Data jsou šifrovaná na straně odesílatele (u nás)
2. Data jsou šifrovaná na straně (úložného) serveru

   Homomorfní šifrování umožňuje třetí možnost:

3. Data jsou šifrována tak, že server je schopen práce

Poznámka: V této kapitole se bavíme hlavně o šifrování dat k dlouhodobému uložení – samotná komunikace přes internet je samozřejmě šifrována způsoby, které již byly dříve v práci vysvětleny.

3.2.1.1 Data jsou šifrovaná na straně odesílatele

Data posíláme na server zašifrovaná a tím pádem chráněná proti zneužití ze strany serveru. Pokud nechceme, aby server s daty jakkoli interagoval, je to v pořádku. Pokud se ale vrátíme k našemu příkladu s kalkulačkou, nebyla by schopná nic spočítat, protože data vlastně nezná.

3.2.1.2 Data jsou šifrovaná na straně serveru

Server v tomto případě vlastní klíč k dešifrování uložených dat a může s nimi podle našich požadavků pracovat. Ovšem to znamená, že má k dispozici všechny nástroje k dešifrování a tím pádem zná obsah uložených dat. V takovém případě už ani nemůžeme mluvit o soukromých datech.

3.2.1.3 Data jsou šifrována tak, že server je schopen práce

Homomorfní zašifrování dat umožňuje serveru provádět operace nad daty, získávat relevantní výsledky operací bez znalosti dat.

3.2.2 Rozdělení homomorfních šifer

Šifry dělíme podle toho, jaké nebo kolik operací můžeme s kryptogramem provádět, dokud jsou data v šumu operací nenávratně ztracena.

3.2.2.1 Částečně homomorfní šifrování – PHE

Jako PHE označujeme taková schémata šifer, kdy jsme schopní provádět v šifrách operaci sčítání, nebo násobení ale ne obě operace.

Příkladem je naše známá šifra RSA, jelikož na ní můžeme provádět operaci násobení, dále ElGamal a Pailer, který je aditivně homomorfní (je možno provádět sčítání).

3.2.2.2 Poněkud homomorfní šifrování – SWHE

Pokud je schéma šifry označené jako SWHE, můžeme nad daty provádět omezený počet obou operací – sčítání i násobení.

3.2.2.3 Plně homomorfní šifrování – FHE

Všechny FHE šifry jsou ve skutečnosti SWHE s tím, že v průběhu provádění operací převádí kryptogram na původní data a následně data opět zašifruje, čímž eliminuje šum, který v důsledku provádění operací narůstá.

Ovšem tady nastává náš starý problém – straně k dešifrování tím pádem musíme poskytnout klíč, takže celý smysl homomorfního šifrování by byl pošlapán.

Druhá možnost by byla si nechávat průběžně kryptogram posílat ze serveru zpět a znovu ho šifrovat na naší straně. To také není ideální řešení.

Tento problém řeší takzvaný bootstrapping.

3.2.3 Bootstrapping

Bootstrapping má na starosti snižovat šum („noise“, chybovost) kryptogramu jeho rešifrováním. Existuje více technik, které může využívat:

1. **Squash** – Provedeme klasické dešifrování, kde nepoužijeme soukromý klíč, ale veřejný, ve kterém je část informací o soukromém klíči.
2. **Chimeric** – Informace o soukromém klíči jsou zavedeny do veřejného klíče, ale ten vstupuje do speciálního dešifrovacího cyklu pro redukci šumu.

3.2.4 Homomorfní šifrování a kvantové počítače

Schémata, na základě kterých jsou šifry konstruovány, se zakládají na různých matematických problémech. Jedním z těchto problémů je již v dřívější kapitole zmíněný CVP – Closest Vector Problem, na kterém se zakládá i postkvantová šifra NTRU, která, jak už víme, odolává kvantové výpočetní rychlosti. Šifry homomorfní založené na stejných nebo podobný principech se tudíž také jeví jako postkvantové.

Nedá se tedy říct, že by homomorfní šifrování spadalo do postkvantové kryptografie, ale jednotlivý zástupci tam kategoricky patřit mohou.

4 PROGRAMOVACÍ PRÁCE

V rámci práce jsem naprogramoval následující algoritmy a jejich implementace jsou veřejně přístupné na odkazu uvedeném jako příloha P I.

4.1 Lenstrův-Lenstrův-Lovászův algoritmus

Zkráceně nazýván LLL algoritmus je jedním z nástrojů využívaný kryptoanalytiky pro prolomování šifer. Pro nás je zásadní, že mezi tyto šifry patří RSA i NTRUEnrypt.

V rámci práce jsem tento algoritmus naprogramoval a jeho implementace je veřejně přístupná na odkazu uvedeném jako příloha P I.

Jak krok po kroku tento algoritmus funguje jsem se naučil v knížce od tvůrců NTRU pana Hoffsteina, paní Pipherové a pana Silvermana [18].

4.2 Euklidův algoritmus a rozšířený Euklidův algoritmus

Euklidův algoritmus slouží k nalezení největšího společného dělitele dvou přirozených čísel. Tuto vlastnost čísel využívá šifra RSA i NTRU.

Rozšířený Euklidův algoritmus je verze, kdy sledujeme i vývoj koeficientů při postupném zjišťování největšího dělitele. Tento algoritmus se využívá pro získání multiplikativní inverze.

Algoritmus jsem zpracoval podle tabulky na wikipedii [19].

ZÁVĚR

V této práci jsem se snažil vytvořit materiál pro čtenáře, které zajímá dnešní kryptografie a její budoucnost, popřípadě si chtějí doplnit své znalosti o dané problematice.

Práce vysvětluje potřebnou matematiku pro pochopení a základní pojmy z oboru kryptologie.

Vysvětlil jsem, jak fungují šifry AES, RSA a NTRUEncrypt a demonstroval jejich použití na ukázkových příkladech.

Dále jsem představil homomorfní šifrování, které se snaží o zvýšení bezpečnosti dat zejména na internetovém úložišti.

SEZNAM POUŽITÉ LITERATURY

[1] Násobení matic. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001, 8. 8. 2021 [cit. 2022-05-15]. Dostupné z: https://cs.wikipedia.org/wiki/N%C3%A1soben%C3%AD_matic

[2] ADVANCED ENCRYPTION STANDARD (AES). *NIST Technical Series Publications* [online]. 2001 [cit. 2022-04-26]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

[3] CHLÁDEK, Dominik. Kongruence čísel. *Isibalo* [online]. 2022 [cit. 2022-05-10]. Dostupné z: https://isibalo.com/matematika/modularni-aritmetika/kongruence-cisel

[4] CHLÁDEK, Dominik. Zbytkové třídy. *Isibalo* [online]. 2022 [cit. 2022-05-10]. Dostupné z: https://isibalo.com/matematika/modularni-aritmetika/zbytkove-tridy

[5] CHLÁDEK, Dominik. Eulerova věta a Eulerova funkce. *Isibalo* [online]. 2022 [cit. 2022-05-10]. Dostupné z: https://isibalo.com/matematika/modularni-aritmetika/eulerova-veta-a-eulerova-funkce

[6] Euklidův algoritmus | FIMATEK. *FIMATEK - FInance MATematika EKonomie* [online]. [cit. 2022-04-26]. Dostupné z: http://fimatek.chciweb.eu/eea_vysl.html?Arg0=288&Arg1=37

[7] NECKÁŘ, Jan. Euklidův algoritmus. In: *Algoritmus* [online]. 2016 [cit. 2022-04-27]. Dostupné z: https://www.algoritmy.net/article/44/Eukliduv-algoritmus

[8] SOLARWINDS MSP. Understanding AES 256 Encryption. *N-able Passportal* [online]. 2019 [cit. 2022-04-27]. Dostupné z: https://www.passportalmsp.com/blog/aes-256-encryption-algorithm

[9] BERNSTEIN, Corinne a Michael COBB. Advanced Encryption Standard. In: *TechTarget* [online]. 2021 [cit. 2022-04-27]. Dostupné z: https://www.techtarget.com/searchsecurity/definition/Advanced-Encryption-Standard

[10] AES Example. In: *Kavaliro* [online]. [cit. 2022-04-27]. Dostupné z: https://www.kavaliro.com/wp-content/uploads/2014/03/AES.pdf

[11] AES-128. *Poisonninja* [online]. 2016, Dec 7, 2016 [cit. 2022-05-09]. Dostupné z: https://poisonninja.github.io/2016/12/07/AES-in-Google-Sheets/

[12] NECKÁŘ, Jan. Algoritmus RSA. *Algoritmy* [online]. 2016 [cit. 2022-04-26]. Dostupné z: https://www.algoritmy.net/article/4033/RSA

[13] YPSJNV2013 a AS5853535. RSA Full Form. *GeeksforGeeks* [online]. [cit. 2022-04-26]. Dostupné z: https://www.geeksforgeeks.org/rsa-full-form/

[14] NEHA T. RSA Algorithm in Cryptography. *Binary Terms* [online]. 2022 [cit. 2022-04-26]. Dostupné z: https://binaryterms.com/rsa-algorithm-in-cryptography.html

[15] HOFFSTEIN, Jeffrey, Jill PIPHER a Joseph H. SILVERMAN. The NTRU Public Key Cryptosystem. *An Introduction to Mathematical Cryptography*. Druhé vydání. New York: Springer, 2014, s. 416-422. ISBN 978-1-4939-1710-5.

[16] DZURENDA, Petr a Jan HAJNÝ. Techniky homomorfního šifrování. *Elektrorevue* [online]. Fakulta elektrotechniky a komunikačních technologií VUT v Brně, 2014 [cit. 2022-05-02]. Dostupné z: http://www.elektrorevue.cz/cz/clanky/informacni-technologie/0/techniky-homomorfniho-sifrovani-a-jejich-prakticke-vyuziti--techniques-of-homomorphic-encryption-and-their-practical-usage-/

[17] Moderní šifrování. *Šifrování a dešifrování* [online]. [cit. 2022-04-26]. Dostupné z: https://lide.uhk.cz/fim/student/havrdda1/moderni_sifrovani.html

[18] HOFFSTEIN, Jeffrey, Jill PIPHER a Joseph H. SILVERMAN. The NTRU Public Key Cryptosystem. *An Introduction to Mathematical Cryptography*. Druhé vydání. New York: Springer, 2014, s. 436-470. ISBN 978-1-4939-1710-5.

[19] Rozšířený Eukleidův algoritmus. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001, 9. 8. 2021 [cit. 2022-05-15]. Dostupné z: https://cs.wikipedia.org/wiki/Roz%C5%A1%C3%AD%C5%99en%C3%BD_Eukleid%C5%AFv_algoritmus

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AES Advanced Encryption Standard

CVP Closest Vector Problem

DES Data Encryption Standard

FHE Fully Homomorphic Encryption

LLL Lenstrův-Lenstrův-Lovász (algoritmus)

mod modulo

NTRU N-th degree Truncated polynomial Ring Units

PHE Partially Homomorphic Encryption

RSA Rivest, Shamor, Adleman (Název šifry složený z iniciálů zakladatelů)

SWHE SomeWhat Homomorphic Encryption

S-box Substitution(nahrazovací) box

VPN Virtual Private Network

VUT Vysoké Učení Technické

XOR eXkluzivní OR

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: GIT ODKAZY

https://github.com/DeadArrow1/LLLAlgorithmDOTNET

https://github.com/DeadArrow1/Euklid