Matematika v pozadí šifrovacích algoritmov

Matej Černohorský

Bakalárska práca
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

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

Akademický rok: 2022/2023

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	Matej Černohorský
Osobní číslo:	A20519
Studijní program:	B0613A140020 Softwarové inženýrství
Forma studia:	Kombinovaná
Téma práce:	Matematika v pozadí šifrovacích algoritmů
Téma práce anglicky:	The Mathematics behind Cryptographic Algorithms

Zásady pro vypracování

1. Vysvětlete principy teorie čísel využívané v kryptografii.
2. Popište princip a význam jednosměrné funkce.
3. Popište matematickou teorii a uveďte příklad výpočtu vybraných jednosměrných funkcí využívaných v komerční bezpečnosti.
4. Názorně popište způsob použití jednosměrné funkce v systému s veřejným klíčem.
5. Popište a ilustrativně implementujte některé algoritmy teorie čísel využívané v kryptografii.

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

Seznam doporučené literatury:

1. HOFFSTEIN, Jeffrey, Jill PIPHER a Joseph H. SILVERMAN. An Introduction to mathematical cryptography. New York: Springer, 2008. ISBN 978-0-387-77993-5.
2. WONG, David. Real-world cryptography. Shelter Island: Manning, 2021. ISBN 978-1-617-29671-0.
3. CRANDALL, Richard, and Carl B. POMERANCE. Prime Numbers. Springer Science & Business Media, 2006. ISBN 978--0387-25282-7.
4. KOHNO, Tadayoshi, FERGUSON, Niels a SCHNEIER, Bruce. Cryptography engineering : design principles and practical applications. Indianapolis, In: Wiley Pub., Inc. 2010. ISBN 978-0-470-47424-2.
5. H. COHEN et al., Handbook of Elliptic and Hyperelliptic Curve Cryptography. CRC Press, 2005. ISBN 978-1-58488-518--4.

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

Datum zadání bakalářské práce: **2. prosince 2022**
Termín odevzdání bakalářské práce: **26. května 2023**

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

doc. Ing. Jiří Vojtěšek, Ph.D.
děkan

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

Ve Zlíně dne 7. prosince 2022

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 bakalářské 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 Univerzita Tomáše Bati 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 23.5.2023 Matej Černohorský,v.r.

podpis studenta

ABSTRAKT

Táto práca sa venuje základnej matematickej teórii, ktorá stojí za modernými kryptografickými systémami a šifrovacími algoritmami. V prvej časti sa zameriava na teóriu čísel, kde sú predstavené základné koncepty ako deliteľnosť, prvočísla, kongruencie, modulárna aritmetika a ďalšie. Obsahuje aj implementácie niektorých algoritmov v jazyku Python. Druhá časť sa zaoberá jednosmernými funkciami, ich vlastnosťami a využitím v oblasti kryptografie. Následné kapitoly sú venované konkrétnym kryptosystémom Diffie-Hellman a RSA a ich názorným výpočtom.

Klíčová slova: Teória čísel, Prvočísla, Modulárna aritmetika, Kongruencie, Jednosmerné funkcie, Kryptografia, Diffie-Hellman, RSA.

ABSTRACT

This thesis focuses on the fundamental mathematical theory behind modern cryptographic systems and encryption algorithms. The first part centers around number theory, introducing fundamental concepts such as divisibility, prime numbers, congruences, modular arithmetic, and more. It also includes implementations of some algorithms in Python. The second part addresses one-way functions, their properties, and their application in the field of cryptography. Subsequent chapters are dedicated to specific cryptosystems - Diffie-Hellman and RSA and their illustrative calculations.

Keywords: Number theory, Prime numbers, Modular arithmetic, Congruences, One-way functions, Cryptography, Diffie-Hellman, RSA.

OBSAH

ÚVOD

V súčasnej digitálnej dobe sa bezpečnosť stáva jednou z najdôležitejších tém. Z tohto dôvodu sa kryptografia stala neoddeliteľnou súčasťou moderných informačných systémov, poskytujúc bezpečnosť a dôvernosť komunikácie. Táto disciplína, aj keď na prvý pohľad výrazne technologická, je v skutočnosti hlboko zakorenená v matematike. Je to matematika, ktorá poskytuje pevný základ pre konštrukciu kryptografických algoritmov a systémov.

Táto práca sa ponára do matematiky stojacej za šifrovacími algoritmami a kryptografickými systémami a skúma, ako abstraktné matematické koncepty, predovšetkým z oblasti teórie čísel, nachádzajú svoje uplatnenie v praxi. Cieľom je vytvoriť most medzi matematickým základom a jeho praktickým využitím, teda ukázať, ako vlastnosti čísel umožňujú zvýšiť bezpečnosť v digitálnom svete.

Teória čísel, neoddeliteľná súčasť matematiky, je kľúčovým pilierom tejto práce, kde sa čitateľovi predstavujú základné matematické koncepty, ako sú deliteľnosť, prvočísla, kongruencie, modulárna aritmetika a mnoho ďalších. Táto sekcia nie je iba teoretickým exkurzom, ale predstavuje aj množstvo konkrétnych príkladov a úloh, ktoré čitateľovi pomôžu lepšie pochopiť a vstrebávať matematické princípy. V kapitole sa nájdu aj ukážky implementácie niektorých algoritmov teórie čísel v programovacom jazyku Python.

Ďalšia časť práce je zameraná na jednosmerné funkcie. Tieto funkcie sú kľúčové v moderných šifrovacích algoritmoch a systémoch s verejnými kľúčmi. Táto práca sa zameriava na ich vlastnosti a ich rozmanité využitie v komerčnej bezpečnosti.

Tretia kapitola práce sa venuje protokolu Diffie-Hellman, ktorý je základom mnohých moderných kryptosystémov a je dôležitým stavebným kameňom v kryptografii. Predstavuje kľúčovú metódu zabezpečenia komunikácie medzi dvoma stranami bez potreby predchádzajúcej bezpečnej výmeny kľúčov. Táto kapitola poskytuje podrobné vysvetlenie mechanizmu tohto protokolu, vysvetľuje jeho matematické základy a názorne demonštruje použitie jednosmernej funkcie založenej na probléme diskrétneho logaritmu. Súčasne sa venuje aj útokom, ktorým je tento systém vystavený, ako napríklad útoku typu "man in the middle"a útoku na malé podgrupy. V závere kapitoly je naznačené, ako zvýšiť bezpečnosť protokolu výberom vhodných prvočísel.

Štvrtá kapitola je venovaná RSA, najznámejšiemu a široko používanému kryptosystému s verejným kľúčom. Táto kapitola poskytuje názornú ukážku použitia jednosmernej funkcie založenej na probléme faktorizácie, na ktorej je tento systém založený. Súčasne sa venuje aj bezpečnostným aspektom RSA a diskutuje o jeho budúcnosti v kontexte rýchleho rozvoja kvantových počítačov.

V celej práci je použité množstvo príkladov a ilustrácií s cieľom čo najviac zjednodušiť a zobrazovať matematické koncepty a teórie. Cieľom je, aby tieto koncepty boli zrozumiteľné a prístupné nielen pre odborníkov v oblasti matematiky a kryptografie, ale aj pre širšiu verejnosť so záujmom o tieto témy.

1 TEÓRIA ČÍSEL V KRYPTOGRAFII

Teória čísel je odvetvie matematiky, ktoré skúma vlastnosti a vzťahy medzi celými číslami. V kontexte kryptografie nám teória čísel poskytuje kľúčové nástroje a poznatky, ktoré umožňujú navrhovať a analyzovať šifrovacie algoritmy a protokoly. Táto oblasť je základom pre veľké množstvo šifrovacích metód, vrátane asymetrických kryptosystémov, ako je RSA, kryptografických protokolov založených na eliptických krivkách a ďalších bezpečnostných technológií.

Prvá kapitola je zameraná na základné pojmy a princípy teórie čísel, ktoré sú dôležité pre pochopenie ich úlohy v kryptografii. Popisuje celočíselné delenie, najväčší spoločný deliteľ, Euklidov algoritmus, a základy modulárnej aritmetiky. Ďalej prechádza ku matematickým konceptom, ktoré sú priamo využívane v kryptografii, ako prvočísla, Eulerova funkcia, malá Fermatova veta, riešenie kongruencií, problém diskrétneho logaritmu a iné.

1.1 Deliteľnosť. Najväčší spoločný deliteľ

Operácia delenia celého čísla celým číslom produkuje podiel a zvyšok. Práca so zvyškom vedie k veľmi dôležitej časti matematiky pre kryptografiu, modulárnej aritmetike. Preto má význam venovať sa vlastnostiam deliteľnosti.

Definícia 1.1. Nech a a b sú celé čísla a $b \neq 0$. Hovoríme, že b delí a, alebo že a je deliteľné b, ak existuje celé číslo c také, že $a = bc$. Píšeme $b \mid a$, aby sme označili, že b delí a. Ak b nedelí a, potom píšeme $b \nmid a$.

Príklad 1.1. Platí $748 \mid 712844$, pretože platí $712844 = 748 \cdot 953$. Platí $256 \nmid 27277$, pretože ak delíme 27277 číslom 256 dostaneme zvyšok 141, tzn. $27277 = 256 \cdot 106 + 141$, takže 27277 nie je presný násobok čísla 256. △

Nasledujúca veta popisuje základné vlastnosti deliteľnosti. Dôkazy môžu byť nájdené v [1].

Veta 1.1. Pre ľubovoľné $a, b, c \in \mathbb{Z}$ platí

(a) Ak $a \mid b$ a $a \mid c$, tak $a \mid (b + c)$

(b) Ak $a \mid b$, tak $a \mid bc$ pre všetky c

(c) Ak $a \mid b$ a $b \mid c$, tak $a \mid c$

Veta 1.2. Nech a je celé číslo a d je celé kladné číslo. Potom existujú jedinečné celé čísla k a r, pričom $0 \leq r < d$, také, že $a = dk + r$.

Na základe Vety 1.2 je možné definovať operátor pre zvyšok - modulo:

$$r = a \bmod d$$

Definícia 1.2. Spoločný deliteľ dvoch kladných celých čísel a a b je kladné celé číslo d také, že platí $d \mid a$ a $d \mid b$.

Definícia 1.3. Nech a a b sú celé čísla, potom najväčší spoločný deliteľ $NSD(a, b)$ je najväčšie celé číslo d, ktorým sa dajú vydeliť čísla a a b bez zvyšku a teda platí

$$NSD(a, b) = \max\{d \in \mathbb{N} : d \mid a \wedge d \mid b\}$$

Najväčší spoločný deliteľ je veľmi dôležitý koncept využívaný v kryptografických systémoch. Existuje mnoho postupov a algoritmov pre výpočet NSD.

Pre výpočet $NSD(a, b)$ sa v praxi používa **Euklidov algoritmus**. Jedná sa o opakované delenie so zvyškom a rekurzívna verzia sa dá formálne zapísať takto:

$$\mathrm{NSD}(a, b) = \begin{cases} a, & \text{ak } b = 0 \\ \mathrm{NSD}(b, a \bmod b), & \text{ak } b \neq 0 \end{cases}$$

Nasleduje pseudokód Euklidovho algoritmu:

Algoritmus 1 Euklidov Algoritmus

Vstup: nezáporné celé čísla a, b
Výstup: najväčší spoločný deliteľ čísel a a b

```
funkcia NSD(a, b)
    ak b = 0 potom                          ▷ základná podmienka
        vráť a
    else
        vráť NSD(b, a mod b)
```

Príklad 1.2. Vypočet najväčšieho spoločného deliteľa čísel 1314 a 288:

Začneme tým, že väčšie číslo 1314 vydelíme menším číslom 288 a zapíšeme kvocient a zvyšok:

$$1314 \div 288 = 4 \text{ so zvyškom } 162$$

Potom vydelíme deliteľa z predchádzajúceho kroku 288 zvyškom 162 a opäť zapíšeme kvocient a zvyšok:

$$288 \div 162 = 1 \text{ so zvyškom } 126$$

Takto pokračujeme v delení predchádzajúceho deliteľa predchádzajúcim zvyškom, až kým zvyšok nie je nula. V tomto bode je predchádzajúci deliteľ najväčší spoločný deliteľ pôvodných dvoch čísel. Takýmto spôsobom pokračujeme:

$$\begin{aligned} 162 \div 126 &= 1 \text{ so zvyškom } 36 \\ 126 \div 36 &= 3 \text{ so zvyškom } 18 \\ 36 \div 18 &= 2 \text{ bez zvyšku} \end{aligned}$$

Keďže zvyšok je teraz nula, končíme. Predchádzajúci deliteľ 18 je najväčší spoločný deliteľ 1314 a 288.

$$NSD(1314, 288) = 18$$

△

Príklad 1.3. Predchádzajúci príklad teraz vypočítame trochu iným spôsobom, a to pomocou funkcie $NSD(a, b)$:

$$\begin{aligned} &NSD(1314, 288) \\ &NSD(288, 1314 \bmod 288) = NSD(288, 162) \\ &NSD(162, 288 \bmod 162) = NSD(162, 126) \\ &NSD(126, 162 \bmod 126) = NSD(126, 36) \\ &NSD(36, 126 \bmod 36) = NSD(36, 18) \\ &NSD(18, 36 \bmod 18) = NSD(18, 0) \end{aligned}$$

Druhý parameter je rovný 0 a funkcia vracia prvý parameter ako výsledok

$$NSD(1314, 288) = 18$$

△

Funkciu môžeme prepísať do jazyka Python následovne:

Kód 1.1 (Príloha 1)**.**

```
def NSD(a, b):
    if b == 0:
        return a
    else:
```

```
return NSD(b, a % b)
```

Definícia 1.4. Dve celé čísla a a b sú **súdeliteľné**, ak $NSD(a, b) > 1$, t.j. ak existuje $d \in \mathbb{Z}$ také, že $a = kd$ a $b = ld$ pre niektoré $k, l \in \mathbb{Z}$.

Dve celé čísla a a b sú **nesúdeliteľné**, ak neexistuje žiadne celé číslo $d > 1$, ktoré by delilo obe čísla bez zvyšku, t.j. ak pre každé $d \in \mathbb{Z}$ platí, že d nedelí ani a, ani b.

1.2 Vlastnosti prvočísel

Štúdium vlastností prvočísel je dôležitá časť diskrétnej matematiky. Prvočísla hrajú významnú úlohu v počítačovej bezpečnosti, keďže dnešné najpoužívanejšie kryptografické systémy sú založené na prvočíslach.

Definícia 1.5. Celé číslo p sa nazýva prvočíslo, ak $p \geq 2$ a ak jediné kladné celé čísla, ktoré delia p, sú 1 a p.

Napríklad číslo 2 je prvočíslo, pretože je deliteľné len číslami 1 a 2. Podobne číslo 5 je prvočíslo, pretože je deliteľné len číslami 1 a 5. Číslo 6 však nie je prvočíslo, pretože je deliteľné číslami 1, 2, 3 a 6.

Veta 1.3. Nech n je kladné celé číslo väčšie ako 1 a d je najmenší deliteľ čísla n, ktorý je väčší ako 1. Potom d je prvočíslo.

Dôkaz. Predpokladajme, že d nie je prvočíslo. Potom existuje deliteľ a čísla d taký, že $1 < a < d$. Pretože d je deliteľ čísla n, platí $n = d \cdot k$ pre nejaké celé číslo k. Potom môžeme zapísať n ako:
$n = (a \cdot b) \cdot k$, kde $b = \frac{d}{a}$.
Keďže $1 < a < d$, potom a je deliteľom čísla n, ktorý je menší než d. To je však v rozpore s predpokladom, že d je najmenší deliteľ čísla n väčší ako 1. Z toho vyplýva, že predpoklad o tom, že d nie je prvočíslo je nepravdivý. Preto musí byť d prvočíslo. □

Dôležitou vlastnosťou prvočísel je, že ich existuje nekonečne veľa. Toto tvrdenie dokázal Euklides v roku 300 pred n.l., keď bol profesorom v Alexandrii. Tento úspech možno považovať za začiatok abstraktnej teórie prvočísel, ako je spomenuté v [2]. Slávny dôkaz nasledujúcej vety je Euklidovým dielom.

Veta 1.4 (Euklides)**.** Prvočísel existuje nekonečne veľa.

Dôkaz. Predpokladajme, že existuje konečný počet prvočísel a označíme ich $p_1, p_2, \ldots, p_k$, kde k je počet prvočísel. Definujeme číslo $n = p_1 p_2 \ldots p_k + 1$, čo je súčin všetkých prvočísel plus jedna. Uvažujme deliteľa $d > 1$. Podľa Vety 1.3 je d prvočíslo a $d \mid n$ Teraz n nemôže byť deliteľné žiadnym z prvočísel 2 až p, pretože každé takéto delenie vráti

zvyšok 1. Predpokladali sme, že 2 až p tvoria všetky prvočísla, ale d je prvočíslo a nie je v zozname. To je v rozpore s predpokladom, že zoznam obsahuje všetky prvočísla a preto je prvočísel nekonečne veľa. □

Definícia 1.6. Nech $x \in \mathbb{N}$ a $x > 1$, potom **prvočíselný rozklad** je každý zápis, ktorý splňuje:

- $x = p_1^{m_1} \cdot p_2^{m_2} \cdot \ldots \cdot p_n^{m_n}$
- $k, m_1, m_2, \ldots, m_k \in \mathbb{Z}^*$
- $p_1, \ldots, p_k$ sú prvočísla

Prvočíselný rozklad známy ako **faktorizácia** znamená jednoznačne rozložiť kladné celé číslo na súčin prvočísel. To znamená, že každé zložené číslo je možné vyjadriť ako súčin prvočísel a pre každé číslo sú prvočísla vo faktorizácii až na poradie jedinečné. Toto tvrdenie je dokázané v [3]. Napríklad číslo 30 môžeme rozložiť ako $2 \cdot 3 \cdot 5$, a to je jediný spôsob, ako vyjadriť číslo 30 ako súčin prvočísel.

Funkcia `faktorizacia(n)` je navrhnutá tak, aby rozložila dané celé číslo n na jeho prvočíselné faktory. Toto je základný algoritmus prvočíselnej faktorizácie, ktorý pracuje tak, že postupne skúša deliteľnosť čísla n číslami začínajúcimi od 2.

Kód 1.2 (Príloha 1)**.**

```
def faktorizacia(n):
    i = 2
    faktory = []
    while i * i <= n:
        if n % i:
            # Ak je n nedeliteľné i, zvýšime i o 1
            i += 1
        else:
            # Ak je n deliteľné i, vydelíme n i
            # a pridáme i do zoznamu faktorov
            n //= i
            faktory.append(i)
    if n > 1:
        # Ak zostane n väčšie ako 1, pridáme ho ako ďalší faktor
        faktory.append(n)
    return faktory
```

Príklad 1.4. Priebeh funkcie `faktorizacia(n)` pre $n = 315$ je takýto:

Začneme s $i = 2$, ale 315 nie je deliteľné 2, takže $i = 3$.

315 je deliteľné 3, takže pridáme 3 do zoznamu a $315/3 = 105$.

105 je deliteľné 3, takže pridáme 3 do zoznamu a $105/3 = 35$.

35 nie je deliteľné 3 ani 4, takže $i = 5$.

35 je deliteľné 5, takže pridáme 5 do zoznamu a $35/5 = 7$.

7 je prvočíslo, pridáme ho do zoznamu a skončíme.

Výsledok je [3, 3, 5, 7], ktoré sú prvočíselné faktory čísla 315. △

1.3 Generovanie prvočísel

Pre filtrovanie prvočísel v určitom intervale je dôležité poznamenať, že neexistuje jednoduchý a efektívny vzorec. Zdá sa, že prvočísla sa vyskytujú pomerne náhodne. Existujú rôzne metódy, ktoré dokážu zistiť, či je číslo prvočíslo, tie však vyžadujú značné výpočtové zdroje, najmä pri analýze veľkých čísel.

Eratosthenov sito

Eratosthenov sito je jedným z najstarších a najjednoduchších spôsobov generovania prvočísel. Tento algoritmus bol navrhnutý starovekým gréckym matematikom Eratosthenom z Cyrene. Algoritmus Eratosthenovo sito je založený na postupnom eliminovaní násobkov prvočísel, čím zostanú iba prvočísla samotné [4].

Predpokladáme, že chceme generovať prvočísla menšie alebo rovné n. Postup je nasledovný:

- Vytvoríme zoznam čísel od 2 po n.
- Najmenšie číslo v zozname je označené ako prvočíslo p.
- Následne sú označené ako zložené čísla všetky násobky čísla p.
- Postup sa opakuje pre ďalšie neoznačené čísla v zozname.
- Po prejdení celého zoznamu zostávajú iba prvočísla.

Kód 1.3 (Príloha 1)**.**

```
def eratosthenovo_sito(n):
    prvocislo = [True] * (n + 1)
    p = 2
    while p * p <= n:
        if prvocislo[p] == True:
            # Označíme všetky násobky čísla p ako zložené čísla
            for i in range(p * p, n + 1, p):
```

```
                prvocislo[i] = False
        p += 1
    # Vyberieme iba prvočísla z poľa prvocislo
    prvocisla = [p for p in range(2, n) if prvocislo[p]]
    return prvocisla
```

Príklad 1.5. Chceme nájsť všetky prvočísla menšie ako 25.

Vytvoríme si zoznam čísel

$$\{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25\}$$

Začneme s prvočíslom 2 a označíme ho ako prvočíslo.

Vyškrtneme zo zoznamu všetky jeho násobky a v zozname zostanú čísla:

$$\{2, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25\}$$

Nájdeme ďalšie najmenšie prvočíslo v zozname a opakujeme krok 3, teda vyškrtneme všetky jeho násobky. V našom prípade ďalším prvočíslom je číslo 3, ktoré zostane v zozname a jeho násobky vymažeme:

$$\{2, 3, 5, 7, 11, 13, 17, 19, 23, 25\}$$

Pokračujeme, kým nevyškrtneme všetky násobky všetkých prvočísel menších ako n. V našom prípade zostanú v zozname len prvočísla

$$\{2, 3, 5, 7, 11, 13, 17, 19, 23\}$$

Týmto spôsobom sme úspešne použili Eratosthenovo sito na nájdenie všetkých prvočísel menších ako 25. △

Eratosthenovo sito je algoritmus efektívny iba na generovanie malých prvočísel, avšak aj ten sa využíva v kryptografii v kombinácii s ďalšími metódami.

Pravdepodobnostné testy prvočíselnosti zahŕňajú metódy, ktoré s vysokou pravdepodobnosťou určujú, či je číslo prvočíslom alebo nie. Tieto testy nie sú úplne presné, ale s dostatočným počtom opakovaní testov môžeme dosiahnuť veľmi vysokú úroveň istoty. Príklady pravdepodobnostných testov prvočíselnosti zahŕňajú Fermatov test alebo Miller-Rabinov test [5].

Deterministické testy prvočíselnosti Na rozdiel od pravdepodobnostných testov, deterministické testy prvočíselnosti poskytujú 100% istotu, že dané číslo je prvočíslom alebo nie. Medzi tieto testy patrí AKS (Agrawal-Kayal-Saxena) test[5].

1.4 Kongruencia

Carl Friedrich Gauss bol významný nemecký matematik a fyzik, ktorý sa narodil 30. apríla 1777 a zomrel 23. februára 1855. Venoval sa mnohým oblastiam matematiky a vedy, ako sa píše v [6]. Mimo iné vytvoril základy modernej teórie čísel a zaviedol pojem kongruencie. Je to analógia k rovnosti v klasickej aritmetike.

Definícia 1.7. Nech a a b sú celé čísla a m je kladné celé číslo, potom a je **kongruentné** s b modulo m, práve vtedy, keď $a - b$ je deliteľné m. Hovoríme, že $a \equiv b \pmod{m}$ je **kongruencia** a že m je jej **modul**. Ak a a b nie sú kongruentné modulo m, píšeme $a \not\equiv b \pmod{m}$.

Nasledujúce tvrdenia sú ekvivalentné:

- $a \equiv b \pmod{m}$
- $a \bmod m = b \bmod m$
- $m \mid (a - b)$
- $a = b + km$, kde k je celé číslo

Príklad 1.6.

(a) Platí, že $13 \equiv 8 \pmod{5}$, pretože $5 \mid (13 - 8)$

(b) Platí, že $18 \not\equiv 15 \pmod{6}$, pretože $6 \nmid (18 - 15)$

△

Definícia 1.8. Množinu všetkých celých čísel b, ktoré sú kongruentné číslu a modulo m nazývame **triedu kongruencie** a zapisuje ju ako:

$$[a]_m = \{b \in \mathbb{Z} \mid b \equiv a \bmod m\}$$

1.5 Modulárna aritmetika

Modulárna aritmetika je odvetvie matematiky, ktoré sa zaoberá celými číslami a ich zvyškami pri delení pevným kladným celým číslom nazývaným modul. Modulárna aritmetika je obzvlášť užitočná pri riešení problémov, ktoré zahŕňajú periodicitu alebo opakovanie. V modulárnej aritmetike vykonávame aritmetické operácie na triedach kongruencií, nie na samotných celých číslach. To znamená, že sa zaoberáme len konečnou množinou čísel. Modulárna aritmetika má široké uplatnenie v informatike, najmä v oblastiach ako je hashovanie, detekcia chýb a šifrovanie.

Operácie modulárnej aritmetiky sú na prvý pohľad veľmi podobné od klasickej aritmetiky, zavádza však niekoľko nových pojmov.

Keď vykonávame sčítanie, odčítanie alebo násobenie v modulárnej aritmetike, jednoducho vykonáme operáciu ako v klasickej aritmetike, ale navyše vypočítame výsledok modulo daným modulom.

Platia nasledovné vzťahy [1]:

- $a +_m b = (a + b) \bmod m = [(a \bmod m) + (b \bmod m)] \bmod m$
- $a -_m b = (a - b) \bmod m = [(a \bmod m) - (b \bmod m)] \bmod m$
- $a \cdot_m b = (a \cdot b) \bmod m = [(a \bmod m) \cdot (b \bmod m)] \bmod m$

Príklad 1.7.

(a) $(23 + 48) \bmod 11 = [(23 \bmod 11) + (48 \bmod 11)] \bmod 11 = (1 + 4) \bmod 11 = 5$

(b) $(19 - 26) \bmod 9 = [(19 \bmod 9) - (26 \bmod 9)] \bmod 9 = (1 - 8) \bmod 9 = 2$

(c) $(17 \cdot 31) \bmod 7 = [(17 \bmod 7) \cdot (31 \bmod 7)] \bmod 7 = (3 \cdot 3) \bmod 7 = 2$

△

V modulárnej aritmetike neexistuje analogická vlastnosť pre delenie. Pri pokuse vypočítať $(8 \div 4) \bmod 7$, z analógie vychádza, že výsledok je 2, ale výraz $((8 \bmod 7) \div (2 \bmod 7)) \bmod 7$, t.j. $1/2 \bmod 7$ nedáva zmysel, pretože $1/2$ nepatrí do oboru hodnôt, v ktorom pracujeme.

1.6 Modulárne umocňovanie

V kryptografii je dôležité efektívne vypočítavať hodnotu $b^n \pmod m$, kde b, n a m sú veľké čísla. Nie je praktické najprv vypočítať b^n a potom nájsť zvyšok po delení m, pretože b^n bude obrovské číslo. Preto sú v praxi využívané efektívnejšie metódy na modulárne umocňovanie [7]. Jednou z týchto metód je **binárne umocňovanie** .

Základná myšlienka binárneho umocňovania spočíva v rozklade exponentu n na binárnu reprezentáciu a následne v opakovanom násobení a umocňovaní základu b. Binárna reprezentácia exponentu $n = (a_{k-1} \dots a_1 a_0)_2$ nám umožní rozdeliť problém umocňovania na niekoľko častí:

$$b^n = b^{a_{k-1} \cdot 2^{k-1} + \dots + a_1 \cdot 2 + a_0} = b^{a_{k-1} \cdot 2^{k-1}} \cdot \dots \cdot b^{a_1 \cdot 2} \cdot b^{a_0}$$

Z toho vyplýva, že na výpočet hodnoty b^n potrebujeme iba získať hodnoty b, b^2, b^4, b^8, $\dots$, b^{2^k}. Keď máme tieto hodnoty, vynásobíme členy b^{2^j} v tomto zozname, kde $a_j = 1$.

Príklad 1.8. Určíme $x = 3^{10} \bmod 13$ pomocou binárneho umocňovania.

Konvertujeme exponent 10 do binárnej sústavy: $10 = (1010)_2$

Postupne vypočítame mocniny čísla 3 (mod 13):

$$\begin{aligned} 3^1 \pmod{13} &= 3 \\ 3^2 \pmod{13} &= 9 \\ 3^4 \pmod{13} &= 3 \\ 3^8 \pmod{13} &= 9 \end{aligned}$$

Výsledok je súčin všetkých mocnín čísla 3, ktoré sme vypočítali a vynásobili na číselnej hodnote jednotlivých binárnych číslic exponentu:

$$x = 9 \cdot 9 \pmod{13} = 3$$

△

Pre efektivitu, po vynásobení každým členom, zredukujeme výsledok modulo m. Týmto získame algoritmus na binárne umocňovanie[1].

Algoritmus 2 Binárne umocňovanie

```
1: funkcia BINARNEUMOCNOVANIE(b, n, m)
2:     x ← 1
3:     mocnina ← b mod m
4:     for i ← 0 to k − 1 opakuj
5:         ak a_i = 1 potom
6:             x ← (x · mocnina) mod m
7:         mocnina ← (mocnina · mocnina) mod m
8:     vráť x                                  ▷ x sa rovná b^n mod m
```

Tento pseudokód reprezentuje binárne umocňovanie v modulárnej aritmetike, kde b je základ, n je exponent v binárnej forme $n = (a_{k-1} \dots a_1 a_0)_2$ a m je modul. Algoritmus inicializuje premennú x na hodnotu 1 a premennú *mocnina* na hodnotu $b \bmod m$. Následne sa použije cyklus **for**, ktorý prechádza všetky binárne číslice exponentu n sprava doľava. Ak je aktuálna binárna číslica 1, vynásobí sa x s hodnotou *mocnina* a výsledok sa redukuje modulo m. Potom sa aktualizuje hodnota premennej *mocnina* na jej druhú mocninu modulo m. Po ukončení cyklu sa výsledok x rovná $b^n \bmod m$ a je vrátený z funkcie.

Príklad 1.9. Použijeme binárne umocňovanie na výpočet $3^{11} \pmod{17}$.

Najprv zapíšeme exponent 11 v binárnej forme: $11 = (1011)_2$.

Inicializujeme výsledok ako $x = 1$.

Prechádzame binárnymi číslicami exponentu 11 (zprava doľava): 1, 1, 0, 1.

Ak je číslica rovná 1, vynásobíme x aktuálnou hodnotou základu (3) a zredukujeme modulo 17:

$$x = (1 \cdot 3) \pmod{17} = 3$$

Aktualizujeme základ:

$$3^2 \pmod{17} = 9$$

Ďalšia číslica je 1. Vynásobíme x aktuálnou hodnotou základu:

$$x = (3 \cdot 9) \pmod{17} = 10$$

Aktualizujeme základ:

$$9^9 \pmod{17} = 13$$

Ďalšia číslica je 0.

Ak je číslica rovná 1, vynásobíme x aktuálnou hodnotou základu, ale v tomto prípade to nerobíme.

Aktualizujeme základ:

$$13^2 \pmod{17} = 16$$

Posledná číslica je 1.

a. Vynásobíme x aktuálnou hodnotou základu:

$$x = (10 \cdot 16) \pmod{17} = 7$$

Keďže sme prešli všetkými číslicami algoritmus vracia výsledok:

$$3^{11} \equiv 7 \pmod{17}$$

△

V jazyku Python sa dá funkcia binárneho umocňovania napísať nasledovne:

Kód 1.4 (Príloha 1)**.**

```
def binarneUmocnovanie(zaklad, exponent, modulo):
    vysledok = 1
    while exponent > 0:
        if exponent % 2 == 1:
            # Ak je exponent nepárny, násobíme aktuálny výsledok
            # s aktuálnym základom a modulo výsledku.
            vysledok = (vysledok * zaklad) % modulo
        # Umocňujeme základ na druhú a berieme modulo výsledku.
        zaklad = (zaklad**2) % modulo
        # Delíme exponent dvojkou.
        exponent = exponent // 2
    return vysledok
```

1.7 Riešenie kongruencií

1.7.1 Lineárna kongruencia

Lineárna kongruencia je rovnica v tvare:

$$ax \equiv b \pmod{m}$$

kde a, b sú celé čísla a m je kladné celé číslo a x je neznáma promemenná. Jedným zo spôsobov ako riešiť túto rovnicu je hľadať také celé číslo $\overline{a}$, že platí $\overline{a}a \equiv 1 \pmod{m}$.

Také číslo $\overline{a}$ je inverzný prvok a v modulárnej aritmetike ho nazývame **modulárna inverzia** a existuje len vtedy, ak platí $NSD(a, m) = 1$. Ak a a m sú súdeliteľné, t.j. $NSD(a, m) \neq 1$, potom modulárna inverzia neexistuje.

Ak existuje, nahradzuje operáciu delenia v modulárnej aritmetike. Vykonáva sa tak, že najprv sa vypočíta inverzný prvok čísla b modulo m, t.j. také číslo c, pre ktoré platí $b \cdot c \equiv 1 \pmod{m}$. Potom môžeme vyjadriť podiel dvoch čísel v modularnej aritmetike ako súčin čísla a a inverzného prvku čísla b modulo m, t.j. $d = a \cdot c \pmod{m}$.

Jedným zo spôsobov, ako nájsť modulárnu inverziu, je použiť **rozšírený Euklidov algoritmus**. Tento algoritmus sa používa na nájdenie najväčšieho spoločného deliteľa dvoch celých čísel a pri tomto procese sa nájdu aj koeficienty, ktoré spĺňajú tzv. **Bézoutovu rovnosť** [8].

Definícia 1.9 (Bézoutova rovnosť)**.** Nech a a b sú celé čísla, potom existujú také celé čísla s a t, že platí

$$NSD(a, b) = sa + tb$$

V rozšírenom Euklidovom algoritme postupne vypočítavame NSD, ale snažíme sa aj uchovávať zvyšky po delení, ktoré použijeme na výpočet koeficientov Bézouto-

vej rovnosti. Výstupom tejto rekurzívnej funkcie sú hodnoty $(NSD(a,b), s, t)$, kde $as + bt = NSD(a,b)$. Nasleduje pseudokód rekurzívnej verzie algoritmu.

Algoritmus 3 Rozšírený Euklidov Algoritmus

Vstup: celé čísla a, b
Výstup: Najväčší spoločný deliteľ čísel a, b
celé čísla s, t také, že $sa + tb = NSD(a,b)$

```
1: funkcia ROZSIRENYNSD(a, b)
2:     ak b = 0 potom
3:         vráť a, 1, 0
4:     else
5:         d, s, t ← ROZSIRENYNSD(b, a mod b)
6:         vráť d, t, s − ⌊a/b⌋ · t
```

Princíp výpočtu je uvedený v nasledujúcom príklade.

Príklad 1.10. Chceme nájsť najväčšieho spoločného deliteľa čísel 180 a 128 a zodpovedajúce Bézoutove koeficienty s a t.
Začneme výpočtom $NSD(180, 128)$:

$$180 = 1 \cdot 128 + 52$$
$$128 = 2 \cdot 52 + 24$$
$$52 = 2 \cdot 24 + 4$$
$$24 = 6 \cdot 4 + 0$$

Posledný nenulový zvyšok je výsledok: $NSD(180, 128) = 4$
V nasledujúcom kroku pracujeme od konca a postupne vyjadrujeme NSD ako lineárnu kombináciu predchádzajúcich zvyškov, až kým sa nedostaneme k pôvodným číslam. Začneme tretím riadkom a vyjadríme zvyšok:

$$4 = 52 - 2 \cdot 24$$

Z druhého riadka vyjadríme 24 a dosadíme

$$4 = 52 - 2 \cdot (128 - 2 \cdot 52)$$

Zjednodušíme

$$4 = -2 \cdot 128 + 5 \cdot 52$$

Znovu dosadíme vyjadrený zvyšok 52

$$4 = -2 \cdot 128 + 5 \cdot (180 - 128)$$

Po zjednodušení dostávame výsledok

$$4 = 5 \cdot 180 - 7 \cdot 128$$

To znamená, že najväčšieho spoločného deliteľa čísel 180 a 128, ktorý sa rovná 4, je možné vyjadriť ako lineárnu kombináciu čísel 180 a 128. △

Príklad 1.11. Chceme nájsť modulárnu inverznú hodnotu čísla 7 mod 26. Inými slovami, chceme nájsť číslo x také, že $7x \equiv 1 \pmod{26}$.
Najprv vypočítame NSD pomocou Euklidovho algoritmu: Potom vykonáme Euklidov algoritmus tak, že od väčšieho čísla odčítame násobky menšieho čísla, kým nedostaneme zvyšok 1:

$$\begin{aligned} 26 &= 3 \cdot 7 + 5 \\ 7 &= 1 \cdot 5 + 2 \\ 5 &= 2 \cdot 2 + 1 \\ 2 &= 2 \cdot 1 + 0 \end{aligned}$$

Vidíme, že posledný nenulový zvyšok je 1, čo znamená, že 7 a 26 sú nesúdeliteľné a preto existuje modulárna inverzia k 7 mod 26. Modulárnu inverziu vypočítame pomocou rozšíreného Euklidovho algoritmu:

$$\begin{aligned} 1 &= 5 - 2 \cdot 2 \\ &= 5 - 2 \cdot (7 - 5 \cdot 1) \\ &= 3 \cdot 5 - 2 \cdot 7 \\ &= 3 \cdot (26 - 3 \cdot 7) - 2 \cdot 7 \\ &= 3 \cdot 26 - 11 \cdot 7 \end{aligned}$$

Keďže chceme vypočítať modulárnu inverziu k 7 mod 26, odčítame koeficient násobiaci číslo 7 a pripočítaním hodnoty 26 dostávame kladný výsledok 15.

$$7 \cdot 15 \equiv 1 \pmod{26}$$

△

Následuje implementácia rozšíreného Euklidovho algoritmu a modulárnej inverzie.

Kód 1.5 (Príloha 1)**.**

```
def rozsireny_NSD(a, b):
    if a == 0:
        return b, 0, 1
    else:
        g, x, y = rozsireny_NSD(b % a, a)
        # vráti NSD(a, b) a Bézoutove koeficienty
        return g, y - (b // a) * x, x

def modularna_inverzia(a, m):
    gcd, x, _ = rozsireny_NSD(a, m)
    if gcd != 1:
        raise ValueError("Modulárna inverzia neexistuje.")
    else:
        return (x % m + m) % m
```

Definícia 1.10. Eulerova funkcia $\varphi(n)$, kde $n \geq 2$, je definovaná ako počet kladných celých čísel, ktoré sú menšie ako n a sú s n nesúdeľné [9].

Napríklad, ak n = 10, potom celé čísla medzi 1 a 10, ktoré sú nesúdeliteľné s 10, sú 1, 3, 7 a 9. Preto $\varphi(10) = 4$.

Pre Eulerovu funkciu $\varphi(n)$ platí:

- $\varphi(1) = 1$.
- $\varphi(p) = p - 1$ pre každé prvočíslo p.
- $\varphi(p^k) = p^k - p^{k-1} = p^k \left(1 - \frac{1}{p}\right)$ pre každé prvočíslo p a každé celé kladné číslo k.
- $\varphi(nm) = \varphi(n) \cdot \varphi(m)$, ak n a m sú nesúdeliteľné.
- $\varphi(n) = \varphi(p \cdot q) = \varphi(p-1) \cdot \varphi(q-1)$, kde $n = p \cdot q$ a p a q sú rôzne prvočísla.
- $\sum_{d|n} \varphi(d) = n$, kde n a d sú kladné celé čísla.

Veta 1.5 (Eulerova veta)**.** Nech $NSD(a, n) = 1$, potom

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

Veta 1.6 (Malá Fermatova veta)**.** Ak p je prvočíslo a a je celé číslo, ktoré nie je deliteľné p, potom $a^{p-1} \equiv 1 \pmod{p}$. Okrem toho pre každé celé číslo a platí $a^p \equiv a \pmod{p}$.

V prípade Malej Fermatovej vety je $\varphi(p) = p - 1$ a platí

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

Malá Fermatova veta je teda špeciálnym prípadom Eulerovej vety pre prvočísla p. Je veľmi efektívna pri výpočte zvyšku po delení celých čísel s veľkou mocninou prvočíslom, výpočtoch modulárnej inverzie a testoch prvočíselnosti.

Príklad 1.12. Nájdime $7^{222} \bmod 11$ pomocou Fermatovej malej vety[1].
Čísla 7 a 11 sú nesúdeliteľné a 11 je prvočíslo, preto je môžné použiť malú Fermatovu vetu.
$7^{10} \equiv 1 \pmod{11}$, takže $(7^{10})^k \equiv 1 \pmod{11}$ pre každé kladné celé číslo k. Aby sme využili túto poslednú kongruenciu, vydelíme exponent 222 číslom 10 a zistíme, že $222 = 22 \cdot 10 + 2$. Teraz vidíme, že $7^{222} = 7^{22 \cdot 10+2} = (7^{10})^{22} \cdot 7^2 \equiv (1)^{22} \cdot 49 \equiv 5 \pmod{11}$. Z toho vyplýva, že $7^{222} \bmod 11 = 5$ △

Príklad 1.13. Vypočítame inverznú hodnotu 9 modulo 11, t.j. hľadáme také číslo, že platí $9x \equiv 1 \,(\mathrm{mod}\ 11)$
Čísla 9 a 11 sú nesúdeliteľné a 11 je prvočíslo, preto je môžné použiť malú Fermatovu vetu:

$$9^{(11-1)} \equiv 1 \,(\mathrm{mod}\ 11)$$
$$9^{10} \equiv 1 \,(\mathrm{mod}\ 11)$$

Pri výpočte inverznej hodnoty upravíme kongruenciu $9x \equiv 1 \,(\mathrm{mod}\ 11)$ tak, že vynásobíme obe strany číslom 9^9

$$9^9 \cdot 9x \equiv 9^9 \cdot 1 \pmod{11}$$
$$x \equiv 9^9 \pmod{11}$$

Na ľavej strane sa koeficient 9^{10} vďaka malej Fermatovej vete zjednoduší na 1, a tak dostávame

$$x \equiv 9^9 \pmod{11}$$

Výraz $9^9 \pmod{11}$ vyriešime postupným umocňovaním

$$\begin{aligned} 9^2 &\equiv 4 \pmod{11} \\ 9^4 \equiv (9^2)^2 &\equiv 4^2 \equiv 5 \pmod{11} \\ 9^8 \equiv (9^4)^2 &\equiv 5^2 \equiv 3 \pmod{11} \\ 9^9 \equiv 9^8 \cdot 9 &\equiv 3 \cdot 9 \equiv 5 \pmod{11} \end{aligned}$$

Z toho plynie, že inverzná hodnota 9 modulo 11 je $x = 5$ a môžeme to potvrdiť spätným dosadením

$$9 \cdot 5 \equiv 1 \pmod{11}$$

△

Frematov test je jednoduchá pravdepodobnostná metóda na určenie, či dané číslo je prvočíslo. Tento test je založený na malej Fermatovej vete.

Algoritmus Fermatovho testu prebieha nasledovne:

- Vyberieme číslo p, ktoré chceme testovať na prvočíselnosť.
- Zvolíme náhodné číslo a také, že platí $1 < a < p$.
- Vypočítame $NSD(a, p)$. Ak $NSD(a, p) \neq 1$, p nie je prvočíslo a nemá zmysel pokračovať.
- Vypočítame $a^{(p-1)} \pmod{p}$.
- Ak $a^{(p-1)} \equiv 1 \pmod{p}$, potom je p pravdepodobne prvočíslo. Ak nie, p je zložené číslo.
- Opakujte kroky 2-5 s rôznymi hodnotami a pre väčšiu istotu.

Príklad 1.14. Chceme otestovať číslo $p = 19$ na prvočíselnosť Fermatovým testom.

Vyberieme náhodné číslo a, napríklad $a = 3$ $(1 < a < 19)$.

Skontrolujeme, či sú čísla a a p nesúdeliteľné: $NSD(3, 19) = 1$.

Vypočítame $3^{19-1} \pmod{19}$ pomocou binárneho umocňovania:

Binárny tvar exponentu 18: 10010_2.

$$\begin{aligned} 3^1 &\pmod{19} = 3 \\ 3^2 &\pmod{19} = 9 \\ 3^4 &\pmod{19} = 5 \\ 3^8 &\pmod{19} = 6 \\ 3^{16} &\pmod{19} = 17 \end{aligned}$$

Vynásobíme výsledky pre exponenty s hodnotou 1 v binárnej reprezentácii:

$$3^{18} \equiv 3^{16} \cdot 3^2 \pmod{19} \equiv 17 \cdot 9 \pmod{19} \equiv 1 \pmod{19}$$

Keďže $3^{18} \equiv 1 \pmod{19}$, číslo $p = 19$ je pravdepodobne prvočíslo podľa Fermatovho testu.

Môžeme test zopakovať s inými hodnotami a pre väčšiu istotu, napríklad:

$$\begin{aligned} &\text{Pre } a = 2,\ 2^{18} \equiv 1 \pmod{19} \\ &\text{Pre } a = 4,\ 4^{18} \equiv 1 \pmod{19} \end{aligned}$$

Pre všetky testované hodnoty a platí, že $a^{(p-1)} \equiv 1 \pmod{p}$. Teda, číslo $p = 19$ je pravdepodobne prvočíslo. V tomto prípade vieme, že 19 je skutočne prvočíslo. △

Treba poznamenať, že existujú čísla, ktoré prechádzajú Fermatovým testom, aj keď nie sú prvočíslami. Tieto čísla sa nazývajú **Carmichaelove čísla**. Fermatov test je pravdepodobnostná metóda, takže nie je garantované, že poskytne správny výsledok pre každé číslo [10]. Ak výsledok testu indikuje, že p je prvočíslo, potom je s veľkou pravdepodobnosťou správny. Ak výsledok testu ukazuje, že p nie je prvočíslo, potom je to vždy správne. Preto sa Fermatov test zvyčajne používa ako prvá kontrola, pred použitím náročnejších a spoľahlivejších testov na prvočíselnosť.

1.7.2 Generátor pseudonáhodných čísel

Jednou z možností, ako využiť modulárnu aritmetiku a kongruencie, je generovanie pseudonáhodných čísel. Pseudonáhodné čísla sú číselné sekvencie, ktoré vyzerajú náhodne, ale generujú sa deterministickým procesom. Tieto sekvencie sú pseudonáhodné, pretože sa dajú presne reprodukovať, ak je známy východzí bod alebo seed. Pseudonáhodné čísla sú nevyhnutné v mnohých oblastiach, vrátane simulácií, počítačových hier a kryptografie.

Jedným z najjednoduchších a najrýchlejších generátorov pseudonáhodných čísel je **li-**

neárny kongruenčný generátor(LCG) [1]. Je to typ generátora, ktorý je definovaný rekurentným vzťahom:

$$x_{n+1} = (ax_n + c) \pmod{m}$$

kde:

- x_{n+1} je $(n+1)$-té číslo v generovanej sekvencii,
- x_n je n-té číslo v sekvencii (s počiatočnou hodnotou x_0, známou ako "seed"),
- a, c a m sú konštanty.

Parametre a, c a m musia byť správne zvolené, aby generátor produkoval dobre premiešanú sekvenciu čísel. Základným požiadavkom je, aby platilo

$$0 \leq c < m$$
$$0 \leq a < m$$
$$0 \leq x_0 < m$$

Generovaná sekvencia je periodická s periódou najviac m a táto perióda sa opakuje po každých m číslach. Pri vhodnej voľbe parametrov môže LCG generovať sekvenciu s maximálnou možnou periódu m, čo je jeden z kľúčových cieľov pri návrhu a analýze LCG.

Príklad 1.15. Vytvoríme si jednoduchý príklad lineárneho kongruenčného generátora (LCG). Zvolíme nasledovné parametre:

$$m = 9$$
$$a = 2$$
$$c = 5$$
$$x_0 = 0$$

Teraz môžeme generovať sekvenciu pseudonáhodných čísel:

$$\begin{aligned}
x_0 &= 0 \\
x_1 &= (ax_0 + c) \pmod m = (2 \cdot 0 + 5) \pmod 9 = 5 \\
x_2 &= (ax_1 + c) \pmod m = (2 \cdot 5 + 5) \pmod 9 = 6 \\
x_3 &= (ax_2 + c) \pmod m = (2 \cdot 6 + 5) \pmod 9 = 8 \\
x_4 &= (ax_3 + c) \pmod m = (2 \cdot 8 + 5) \pmod 9 = 3 \\
x_5 &= (ax_4 + c) \pmod m = (2 \cdot 3 + 5) \pmod 9 = 2 \\
x_6 &= (ax_5 + c) \pmod m = (2 \cdot 2 + 5) \pmod 9 = 0
\end{aligned}$$

Vidíme, že $x_6 = x_0$ a keďže generátor závisí iba od hodnoty predchádzajúceho prvku, dostávame sekvenciu

$$0, 5, 6, 8, 3, 2, 0, 5, 6, 8, \ldots$$

v ktorej sa bude opakovať 6 rôznych čísel. △

Kód 1.6 (Príloha 1)**.**

```
def lcg(modul, a, c, seed):
    while True:
        seed = (a * seed + c) % modul
        yield seed
```

Funkcia `lcg(modul, a, c, seed)` je implementácia generátora LCG, ktorý neustále generuje nové hodnoty v nekonečnej slučke. Môžeme ju opätovne volať pomocou konštrukcie `next(generator)`, kde `generator = lcg(modul, a, c, seed)` alebo ho použiť v cykle for.

Lineárne kongruenčné generátory sú populárne vďaka svojej jednoduchosti a rýchlosti, ale musia byť používané opatrne, pretože niektoré voľby parametrov môžu viesť k sekvenciám s krátkymi periódami alebo s jasne viditeľnými vzormi v generovaných číslach.

1.7.3 Sústavy lineárnych kongruencií a Čínska veta o zvyškoch

Veta 1.7 (Čínska veta o zvyškoch)**.** Nech $m_1, m_2, \ldots, m_k$ je množina vzájomne nesúdeliteľných celých čísel. To znamená, že $NSD(m_i, m_j) = 1$ pre všetky $i \neq j$.

Nech $a_1, a_2, \ldots, a_k$ sú ľubovoľné celé čísla. Potom systém lineárnych kongruencií

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ &\vdots \\ x &\equiv a_k \pmod{m_k} \end{aligned}$$

má riešenie x a všetky riešenia x sú navzájom kongruentné.

Čínska zvyšková veta hovorí, že existuje jedinečné riešenie pre x modulo $M = m_1 m_2 \cdots m_n$, a toto riešenie možno nájsť pomocou vzorca:

$$x \equiv \sum_{i=1}^{n} a_i M_i y_i \pmod{M}$$

kde $M_i = \frac{M}{m_i}$ a y_i je inverzná hodnota M_i modulo m_i, t.j. $M_i y_i \equiv 1 \pmod{m_i}$.
Inými slovami, aby sme našli riešenie pre x, musíme najprv vypočítať M, M_i a y_i pre každé i a potom ich dosadiť do vyššie uvedeného vzorca. Výsledná hodnota x bude jedinečným riešením, ktoré spĺňa všetky dané kongruencie.

Príklad 1.16. Vyriešme sústavu lineárnych kongruencií

$$\begin{aligned} x &\equiv 3 \pmod{5} \\ x &\equiv 1 \pmod{7} \\ x &\equiv 6 \pmod{8} \end{aligned}$$

Najprv hodnoty a_i, M_i a y_i postupne vyplníme do tabuľky. Hodnoty a_i sú dané a iba ich opíšeme. Vypočítame výsledný modul

$$M = m_1 m_2 m_3 = 5 \cdot 7 \cdot 8 = 280$$

M_i vypočítame podľa vzťahu $M_i = \frac{M}{m_i}$ a vhodnou metódou vypočítame hodnoty y_i. Následne vypočítame súčin týchto hodnôt.

Tabulka 1.1 Príklad 1.9

a_i	M_i	y_i	$a_i M_i y_i$
3	56	1	168
1	40	3	120
6	35	3	630

Sumou týchto súčinov získavame riešenie $x = 168 + 120 + 630 = 918$.

$$x \equiv 918 \pmod{280}$$
$$x \equiv 78 \pmod{280}$$

△

Kód 1.7 (Príloha 1)**.**

```
def cinska_zvyskova_veta(n, a):
    M = 1
    for cislo in n:
        M *= cislo
    vysledok = 0
    for mi, ai in zip(n, a):
        Mi = M // mi
        _, yi, _ = rozsireny_NSD(Mi, mi)
        vysledok += ai * yi * Mi % M
    return vysledok % M
```

V tejto implementácii sa najprv vypočíta hodnota M, ktorá je súčinom všetkých čísel v zozname n. Potom sa iteruje cez všetky čísla v sústave a pre každé číslo sa vypočíta hodnota Mi ako M delené mi. Následne sa použije `rozsireny_NSD` na výpočet inverzného koeficientu yi pre každé číslo mi. Sčítajú sa hodnoty výrazov $ai \cdot yi \cdot Mi \pmod{M}$ a vráti sa výsledok modulo M.

1.8 Grupy

Definícia 1.11. Nech G je množina a $*$ je binárna operácia na G. Potom $(G, *)$ sa nazýva **grupa**, ak spĺňa nasledujúce podmienky:

- **Uzavretosť**: Pre všetky $a, b \in G$ platí, že $a * b \in G$.
- **Asociativita**: Pre všetky $a, b, c \in G$ platí, že $(a * b) * c = a * (b * c)$.
- Existencia **neutrálneho prvku**: Existuje prvok $e \in G$, taký, že pre každý prvok $a \in G$ platí $a * e = e * a = a$.
- Existencia **inverzného prvku**: Pre každý prvok $a \in G$ existuje inverzný prvok $a^{-1} \in G$, taký, že $a * a^{-1} = a^{-1} * a = e$, kde e je neutrálny prvok.

Pokiaľ $(G, *)$ spĺňa tieto podmienky, $(G, *)$ nazývame grupou.

Symbol $\mathbb{Z}_p^*$ reprezentuje multiplikatívnu grupu kladných celých čísel modulo p, kde p je prvočíslo.

Definícia 1.12. Počet prvkov grupy G sa nazýva **rád grupy** a označujeme ho $|G|$. Ak je G nekonečná, potom $|G| = \infty$.

Definícia 1.13. Nech G je grupa a nech $g \in G$. **Rád prvku** g, označený ako $|g|$, je najmenšie kladné celé číslo n, také, že $g^n = e$, kde g^n znamená aplikáciu skupinovej operácie $*$ na prvok g celkom n-krát a e je neutrálny prvok grupy (taký prvok, že $e * g = g * e = g$ pre všetky $g \in G$). Ak neexistuje takéto kladné celé číslo n, hovoríme, že rád prvku g je nekonečný.

Kód 1.8 (Príloha 1)**.**

```
def rad_grupy(p):
    # Pre prvočíslo p je rád grupy jednoducho p-1.
    return p - 1


def rad_prvku_grupy(a, p):
    # Musíme skontrolovať všetky čísla od 1 do p-1.
    for n in range(1, p):
        if binarneUmocnovanie(a, n, p) == 1:
            return n
    return None
```

Príklad 1.17. Pre multiplikatívnu grupu zvyškov po delení modulo 7, označovanú ako $(\mathbb{Z}_7^*, \cdot)$ vypočítame rád grupy a rád prvku 3.

Táto grupa obsahuje prvky $\{1, 2, 3, 4, 5, 6\}$ a počet prvkov je 6. Rád grupy je teda $|(\mathbb{Z}_7^*, \cdot)| = 6$.

Uvažujme prvok $g = 3 \in \mathbb{Z}_7^*$. Postupne aplikujeme operáciu grupy(násobenie), až kým nezískame neutrálny prvok:

$$
\begin{aligned}
3^1 &= 3 \pmod 7 \\
3^2 &= 9 \pmod 7 = 2 \pmod 7 \\
3^3 &= 27 \pmod 7 = 6 \pmod 7 \\
3^4 &= 81 \pmod 7 = 4 \pmod 7 \\
3^5 &= 243 \pmod 7 = 5 \pmod 7 \\
3^6 &= 729 \pmod 7 = 1 \pmod 7
\end{aligned}
$$

Keďže neutrálny prvok v tejto grupe je 1 a $3^6 = 1$, rád prvku $g = 3$ je 6. To znamená, že potrebujeme šesť opakovaní skupinovej operácie (násobenie modulo 7) na prvok g, aby sme dostali neutrálny prvok grupy. △

Definícia 1.14. Nech $(G, *)$ je grupa a H je podmnožina G. H je **podgrupou** G (značíme $H \leq G$), ak spĺňa tieto podmienky:

- H je **neprázdna**: $H \neq \emptyset$.
- **Uzavretosť**: Pre všetky $a, b \in H$, platí, že $a * b \in H$.
- Existencia **inverzného prvku**: Pre každý prvok $a \in H$, platí, že $a^{-1} \in H$.

Poznámka: Neutrálny prvok e grupy G automaticky patrí do H, keďže H je neprázdna a uzavretá na inverzné prvky.

Ak podmnožina H spĺňa tieto podmienky, potom H je podgrupou G a môžeme ju označiť ako $(H, *)$.

Definícia 1.15. Nech $(G, *)$ je grupa. Hovoríme, že G je **cyklická grupa**, ak existuje prvok $g \in G$, taký, že každý prvok v G môže byť zapísaný ako g^n pre nejaké celé číslo n. Prvok g sa potom nazýva **generátor** cyklickej grupy G. Formálne zapisujeme, že $G = \langle g \rangle = \{g^n | n \in \mathbb{Z}\}$, kde $\mathbb{Z}$ je množina celých čísel a g^n je aplikácia skupinovej operácie $*$ na prvok g, n-krát.

Generátory sú dôležité, pretože nám poskytujú jednoduchý spôsob, ako študovať a popisovať štruktúru grupy. Ak vieme, že grupa je cyklická a poznáme jej generátor, môžeme jednoducho vytvoriť všetky prvky tejto grupy a analyzovať ich vlastnosti.

Nie všetky grupy majú generátory; iba cyklické grupy môžu byť generované jediným prvkom. Je dôležité poznamenať, že nie všetky prvky cyklickej grupy sú generátory. Generátor musí byť taký, že jeho mocniny generujú všetky prvky grupy.

Kód 1.9 (Príloha 1)**.**

```
def najdi_generatory(p):
    phi = p - 1
    faktory = faktorizacia(phi)
    generatory = []

    for g in range(2, p):
        if all(pow(g, phi // faktor, p) != 1 for faktor in faktory):
            generatory.append(g)

    return generatory
```

Táto funkcia hľadá generátory grupy modulo p. Vypočíta hodnotu Eulerovej funkcie $\varphi(p)$ pre p a faktorizuje ju pomocou funkcie `faktorizacia(phi)`. Potom iteruje cez čísla od 2 do p a overuje, či sú generátormi grupy.

Príklad 1.18. Uvažujme konečnú multiplikatívnu cyklickú grupu $(\mathbb{Z}_7^*, \cdot) = \{0, 1, 2, 3, 4, 5, 6\}$. Chceme nájsť generátor tejto grupy, teda prvok, ktorým sa dá vygenerovať celá grupa. Ak by sme postupovali skúšobne, vyskúšali by sme postupne všetky prvky a počítali by sme ich mocniny, kým by sme nenašli prvok, ktorým sa dá vygenerovať celá grupa. Ak je náš prvok g generátorom cyklickej grupy, potom $g^1, g^2, g^3, ..., g^{n-1}$ sú všetky rôzne prvky množiny grupy.
Nájdime teda generátor cyklickej grupy $(\mathbb{Z}_7^*, \cdot)$. Začneme s prvkom 2:

$$2^1 = 2,\ 2^2 = 4,\ 2^3 = 1,\ 2^4 = 2,\ 2^5 = 4,\ 2^6 = 1$$

Prvok 2 nie je generátorom, pretože jeho mocniny sa opakujú a nevytvárajú celú grupu. Skúsime ďalší prvok:

$$3^1 = 3,\ 3^2 = 2,\ 3^3 = 6,\ 3^4 = 4,\ 3^5 = 5,\ 3^6 = 1$$

Prvok 3 vytvoril celú zadanú grupu, preto je generátorom cyklickej grupy $(\mathbb{Z}_7^*, \cdot)$. △

V príklade je naznačené, ako nájsť generátor pre cyklickú grupu hrubou silou. V praxi sa však často používajú iné algoritmy, ktoré sú efektívnejšie a rýchlejšie pre veľké grupy, ako napríklad Pollardov ró algoritmus, Baby-step/Giant-step algoritmus a algoritmy založené na faktorizácii čísel.

1.9 Problém diskrétneho logaritmu

Definícia 1.16. Nech G je cyklická grupa s generátorom grupy g a nech y je prvok tejto grupy. Problém diskrétneho logaritmu spočíva v nájdení takého celého čísla x, že platí:

$$g^x \equiv y \pmod{p}$$

kde p je prvočíslo, ktoré je rádom grupy G, a x je hľadaný diskrétny logaritmus y o základe g modulo p.

Zatiaľ čo v obore reálnych čísel $\mathbb{R}$ je výpočet a^x rádovo rovnako časovo zložitý ako spätný výpočet logaritmu, v konečnej multiplikatívnej grupe G síce existuje niekoľko efektívnych algoritmov pre výpočet a^x, ale spätný výpočet je časovo veľmi náročný a v súčasnosti neexistuje algoritmus, ktorý by počítal diskrétny logaritmus v polynomiálnom čase.

Príklad 1.19. Máme dané prvočíslo $p = 17$ a generátor $g = 3$. Máme tiež číslo $y = 13$ a chceme nájsť také celé číslo x, aby $g^x \equiv y \pmod{p}$. To znamená riešime kongruenciu $3^x \equiv 13 \pmod{17}$.

Tento problém budeme riešiť hrubou silou, takže postupne budeme skúšať všetky možné hodnoty x, až kým nenájdeme takú, ktorá spĺňa kongruenciu.

$$\begin{aligned} 3^1 &\equiv 3 \pmod{17} \\ 3^2 &\equiv 9 \pmod{17} \\ 3^3 &\equiv 10 \pmod{17} \\ 3^4 &\equiv 15 \pmod{17} \\ 3^5 &\equiv 11 \pmod{17} \end{aligned}$$

Takto postupujeme ďalej, až nakoniec zistíme, že

$$3^{12} \equiv 13 \pmod{17}$$

Riešením úlohy je $x = 12$, tzn. že diskrétny logaritmus 13 pri základe 3 je 12. △

Príklad naznačuje, že pre veľké čísla je tento spôsob hľadania diskrétneho logaritmu veľmi neefektívny, čo podnietilo vznik mnohých kryptografických systémov, ktoré využívajú problém diskrétneho logaritmu.

2 JEDNOSMERNÁ FUNKCIA

Jednosmerná funkcia (z angličtiny "one-way function") je koncept z kryptografie a teoretickej informatiky. Ide o funkciu, ktorá je ľahko vypočítateľná pre každý vstup, ale ťažko reverzibilná, teda je ťažko nájditeľná jej inverzná hodnota. Inými slovami, ak máme funkciu $f(x) = y$, je ľahké nájsť hodnotu y pre dané x, ale nájsť x pre dané y je veľmi ťažké.

Jednosmerné funkcie sú základom mnohých kryptografických algoritmov, ako sú napríklad hashovacie funkcie, asymetrické šifrovanie (RSA) a digitálne podpisy. Bezpečnosť týchto algoritmov závisí od toho, ako ťažké je 'zlomiť' jednosmernú funkciu, teda nájsť jej inverziu v prijateľnom čase.

Jedným z prvých výsledkov v tejto oblasti bola zásadná práca Diffieho a Hellmana [11], ktorí deklarovali, že existencia jednosmerných funkcií je ekvivalentná s existenciou tzv. trapdoor funkcií. Výskum v oblasti teórie výpočtovej zložitosti poskytol ďalšie podmienky. Existencia jednosmerných funkcií úzko súvisí so zložitosťou problému P vs. NP, či každý problém, ktorý sa dá verifikovať v polynomiálnom čase (t. j. problémy v NP), sa dá aj vyriešiť v polynomiálnom čase (t. j. problémy v P). Inými slovami, pýta sa, či P = NP. Ak by sa podarilo potvrdiť, že $P \neq NP$, znamenalo by to existenciu jednosmerných funkcií. Na druhú stranu, ak by sa potvrdilo, že $P = NP$, malo by to nedozierne následky pre súčasné kryptografické systémy. Bezpečnosť systémov založených na jednosmerných funkciách je dokonca silne spochybňovaná, ako napríklad v tejto práci [12]

Všeobecný dôkaz o existencii jednosmerných funkcií však zostáva otvoreným problémom, a teda nejedná sa o potvrdený matematický aparát. V kryptografii je pojem jednosmerná funkcia teoretický konštrukt, ktorým definujeme vlastnosti funkcií s určitými vlastnosťami. Táto kapitola je podporená zdrojmi [7], [13] a [14].

2.1 Vlastnosti jednosmerných funkcií

Jednosmernú funkciu je možné popísať ako funkciu f, ktorá mapuje vstup x z množiny X na výstup y z množiny Y a má niektoré základné vlastnosti, ktoré ich robia užitočnými pre kryptografické účely. Tieto vlastnosti zahŕňajú:

Jednosmernosť. Nech f: $X \to Y$ je jednosmerná funkcia. Funkcia f je jednosmerná, ak platí:

$\forall x \in X, f(x)$ je ľahko vypočítateľná

$\forall y \in Y$, nájsť $x \in X$ také, že $f(x) = y$ je výpočtovo náročné

Príklad 2.1. Nech máme dve prvočísla p a q, kde $p = 89$ a $q = 97$.
Vytvoríme jednosmernú funkciu:

$$f(p, q) = p \cdot q$$

V tomto prípade môžeme ľahko vypočítať výstup funkcie:

$$f(x) = 89 \cdot 97 = 8633$$

Získať pôvodné prvočísla p a q na základe výstupu funkcie $f(x) = 8633$ je problém celočíselnej faktorizácie, ktorý sa považuje za výpočtovo náročný pre veľké hodnoty p a q. △

Jednoznačnosť. Nech f: $X \to Y$ je jednosmerná funkcia. Funkcia f je jednoznačná, ak pre všetky prvky $x_1, x_2 \in X$, kde $x_1 \neq x_2$, platí, že $f(x_1) \neq f(x_2)$.

Príklad 2.2. Definujme funkciu f ako $f(x) = 2x + 3 \pmod 7$
Teraz uvažujme dva vstupy v celom rozsahu
Vypočítajme hodnoty $f(x)$ pre rôzne x:

$$\begin{aligned}
f(0) &= (2 \cdot 0 + 3) \pmod 7 = 3 \\
f(1) &= (2 \cdot 1 + 3) \pmod 7 = 5 \\
f(2) &= (2 \cdot 2 + 3) \pmod 7 = 0 \\
f(3) &= (2 \cdot 3 + 3) \pmod 7 = 2 \\
f(4) &= (2 \cdot 4 + 3) \pmod 7 = 4 \\
f(5) &= (2 \cdot 5 + 3) \pmod 7 = 6 \\
f(6) &= (2 \cdot 6 + 3) \pmod 7 = 1
\end{aligned}$$

Výstupy $f(x)$ sú rôzne pre rôzne vstupy x. Teda táto funkcia f spĺňa vlastnosť jednoznačnosti. △

Táto vlastnosť je dôležitá pre udržanie integrity dát a predchádzanie kolíziám, najmä v kryptografii a hašovaní.

Efektívnosť. Nech f: $X \to Y$ je jednosmerná funkcia a $T_f(n)$ je čas potrebný na výpočet f pre vstup veľkosti n. Funkcia f je efektívna, ak platí:

$$\exists c > 0, \forall x \in X, T_f(|x|) = O(|x|^c)$$

Funkcia f je efektívna, ak existuje konštanta $c > 0$, taká že pre všetky vstupy $x \in X$, čas potrebný na výpočet f pre vstup veľkosti $|x|$ je zhora ohraničený funkciou $O(|x|^c)$. To znamená, že čas potrebný na výpočet f rastie najviac polynomiálne s veľkosťou vstupu.

Symbol O v zápise $O(|x|^c)$ označuje "big O"notáciu, ktorá sa používa na popis rastu funkcií výpočtovej zložitosti. V tomto prípade $O(|x|^c)$ hovorí, že čas potrebný na výpočet f rastie najviac polynomiálne s veľkosťou vstupu.

Lavínový efekt zabezpečuje, že útočník nemôže odvodiť žiadne informácie o vstupe na základe výstupu funkcie, keďže aj najmenšia zmena na vstupe spôsobí veľkú zmenu výstupu. Lavínový efekt je dôležitá vlastnosť jednosmerných funkcií, najmä v hashovacích funkciách a blokových šifrách. Lavínový efekt sa dá hodnotiť pomocou **Hammingovej vzdialenosti**.

Definícia 2.1. Nech x a y sú reťazce rovnakej dĺžky n, potom Hammingova vzdialenosť $d_H(x, y)$ je funkcia, ktorá vracia počet pozícií v ktorých sa reťazce líšia, teda platí:

$$d_H(x, y) = \sum_{i=1}^{n} I(x_i \neq y_i)$$

kde x_i a y_i sú zodpovedajúce bity (alebo znaky) na pozícii i v reťazcoch x a y, a $I(\cdot)$ je indikátorová funkcia, ktorá má hodnotu 1, ak je podmienka v zátvorke splnená (t.j. $x_i \neq y_i$), a 0 v opačnom prípade.

Príklad 2.3. Uvažujme jednosmernú funkciu f: $0,1^8 \to 0,1^8$, ktorá pracuje s 8-bitovými binárnymi reťazcami. Aby sme demonštrovali rýchlu zmenu výstupu, uvažujme, že máme nasledujúce hodnoty pre f:

$$\begin{aligned} f(00000000) &= 11001100 \\ f(00000001) &= 10101010 \\ f(00000010) &= 01110011 \\ &\vdots \\ f(11111111) &= 11111100 \end{aligned}$$

Teraz uvažujme dva rôzne vstupy, ktoré sa líšia iba v jednom bite, napríklad:

$$x_1 = 00000000$$
$$x_2 = 00000001$$

Výstupy pre tieto vstupy sú:

$$f(00000000) = 11001100$$
$$f(00000001) = 10101010$$

Hammingova vzdialenosť medzi výstupmi je $d_H(11001100, 10101010) = 4$. V tomto zjednodušenom príklade môžeme pozorovať, že meniace sa bity na vstupe spôsobujú väčšie zmeny na výstupe, čo je žiadaná vlastnosť hashovacích funkcií. △

Pseudonáhodnosť: Nech $f : \{0,1\}^n \to \{0,1\}^n$ je jednosmerná funkcia. Funkcia f je pseudonáhodná, ak pre všetky algoritmy D, pravdepodobnosť, že D dokáže rozoznať výstupy z funkcie f od náhodných výstupov je zanedbateľná.

2.2 Využitie v kryptografii

Keďže chýba jednoznačný popis, je prakticky nemožné deklarovať univerzálnu jednosmernú funkciu, ktorá by bola použiteľná vo všetkých sférach informatiky. Namiesto toho sa používajú špecializované funkcie, ktoré sú vhodné iba v určitých prípadoch použitia. Tento prehľad je podporený zdrojmi [15] a [16].

Hashovacia funkcia je jednosmerná funkcia, ktorá prijíma vstup ľubovoľnej dĺžky a ako výstup vracia reťazec bajtov pevnej veľkosti, nazývaný digest správy. Hašovacie funkcie sa v kryptografii široko používajú na overovanie správ, digitálne podpisy a hašovanie hesiel. Medzi najčastejšie používané hašovacie funkcie patria SHA, MD, BLAKE a iné.

Problém diskrétneho logaritmu sa využíva ako jednosmerná funkcia. Ide o úlohu nájsť hodnotu exponentu x z rovnice $a^x \equiv b \pmod p$, kde a, b a p sú celé čísla a p je prvočíslo. Tento problém je ťažký na riešenie, najmä pre veľké hodnoty p, a v súčasnosti neexistuje žiadny známy algoritmus, ktorý by ho riešil efektívne pre všetky prvočíselné moduly. Problém diskrétneho logaritmu je využívaný v systémoch Diffie-Hellman, ElGamal, DSA a iné.

Problém celočíselnej faktorizácie spočíva v hľadaní prvočiniteľov zloženého čísla. Vynásobenie dvoch prvočísel je jednoduchá operácia, ale nájdenie pôvodných prvočíselných činiteľov, ak je daný len súčin, sa považuje za výpočtovo náročné. Táto jednosmerná vlastnosť tvorí základ kryptosystému RSA a Rabinovho systému.

Kryptografia eliptických kriviek (ECC) je založená na algebraickej štruktúre eliptických kriviek nad konečnými poľami. Jednosmernou funkciou v ECC je násobenie bodov eliptickej krivky, ktoré sa dá ľahko vypočítať v jednom smere, ale ťažko v opačnom. Táto vlastnosť sa využíva pri výmene kľúčov pomocou eliptickej krivky Diffie-Hellman (ECDH) a algoritme digitálneho podpisu pomocou eliptickej krivky (ECDSA).

V **Mriežková kryptografia** je oblasť kryptografie, ktorá sa zakladá na probléme vyhľadávania najbližšieho bodu v mriežke (nazývaný aj problém najbližšieho vektora, alebo NVP v angličtine). Všeobecne povedané, mriežka v tomto kontexte je množina bodov v n-rozmernom priestore, ktoré majú celočíselné súradnice vzhľadom k niektorým základným vektorom. Základný problém, na ktorom je mriežková kryptografia založená, spočíva v tom, že hoci je pomerne jednoduché generovať bod v mriežke blízko k nejakému cieľovému bodu, je výpočtovo veľmi ťažké zistiť, ktorý bod v mriežke je najbližšie k danému cieľovému bodu, ak nemáte k dispozícii základné vektory.

Problém súčtu podmnožín je rozhodovací problém, v ktorom je daná množina celých čísel a cieľové celé číslo a cieľom je určiť, či existuje podmnožina danej množiny, ktorej súčet je rovný cieľovému celému číslu. Hoci sa tento problém dá ľahko formulovať, považuje sa za NP-úplný problém, čo znamená, že je vo všeobecnom prípade výpočtovo ťažko riešiteľný. Problém súčtu podmnožín bol použitý ako základ pre niekoľko kryptografických schém vrátane kryptosystémov s verejným kľúčom založených na probléme batohu.

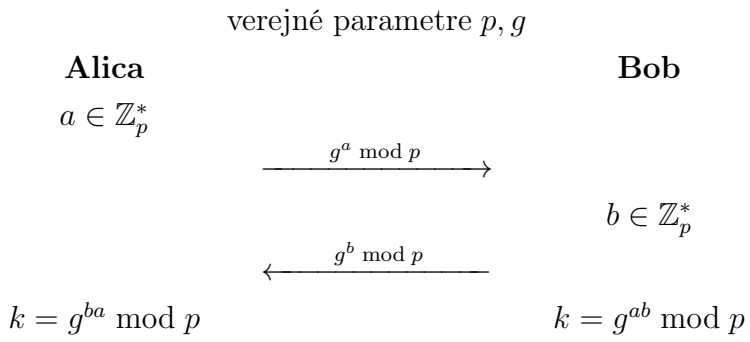
Permutačná funkcia je typ transformačnej funkcie, ktorá mení usporiadanie prvkov vstupných dát. Permutačné funkcie sa používajú v blokových šifrách, čo sú kryptografické algoritmy, ktoré šifrujú bloky otvoreného textu pevnej dĺžky. Medzi príklady permutačných funkcií patria Advanced Encryption Standard (AES) a Data Encryption Standard (DES).

3 DIFFIE-HELLMAN

V roku 1976 Whitfield Diffie a Martin Hellman v článku [11] predstavili metódu na výmenu tajného kľúča medzi dvoma stranami, ktoré nemajú predtým žiadny zdieľaný tajný kľúč. Bol to prvý algoritmus na výmenu kľúčov pre asymetrické šifrovanie, ktorý umožňuje bezpečne posielať správy cez nezabezpečenú komunikačnú linku.

Základný DH algoritmus pozostáva z nasledujúcich krokov[17]:

- Strany Alica a Bob si dohodnú čísla $p > 1$ a g, kde p je prvočíslo a g je prirodzené číslo, ktoré generuje celú grupu $(\mathbb{Z}_p^*, \cdot)$. Tieto čísla sú známe obidvom stranám aj potenciálnym útočníkom.
- Alica náhodne vygeneruje súkromný kľúč a, kde $1 < a < p - 1$.
  Bob náhodne vygeneruje súkromný kľúč b, kde $1 < b < p - 1$.
  Oba súkromné kľúče musia byť utajené a nesmú sa s nikým zdieľať.
- Alica vypočíta svoj verejný kľúč $A = g^a \pmod p$. Bob vypočíta svoj verejný kľúč $B = g^b \pmod p$. Obe strany si vymenia svoje verejné kľúče cez nezabezpečený komunikačný kanál.
- Alica vypočíta zdieľaný tajný kľúč $k_A = B^a \pmod p$. Bob vypočíta zdieľaný tajný kľúč $k_B = A^b \pmod p$. Obe strany majú teraz rovnaký zdieľaný tajný kľúč k, ktorý použijú na šifrovanú komunikáciu.



Príklad 3.1. Vypočítame kľúčovú výmenu DH so skutočnými číslami. Zvolíme si verejné prvočíslo $p = 23$ a generátor $g = 5$.

Alica si náhodne zvolí tajné číslo $a = 4$ a vypočíta verejný kľúč A:

$$\begin{aligned} A &= g^a \pmod p \\ A &= 5^4 \pmod{23} \\ A &= 4 \end{aligned}$$

Alica pošle číslo A Bobovi.

Bob si náhodne zvolí tajné číslo $b = 3$ a vypočíta verejný kľúč B:

$$\begin{aligned} B &= g^b \pmod{p} \\ B &= 5^3 \pmod{23} \\ B &= 10 \end{aligned}$$

Alica vypočíta zdieľaný tajný kľúč:

$$\begin{aligned} k_A &= B^a \pmod{p} \\ k_A &= 10^4 \pmod{23} \\ k_A &= 18 \end{aligned}$$

Bob vypočíta zdieľaný tajný kľúč:

$$\begin{aligned} k_B &= A^b \pmod{p} \\ k_B &= 4^3 \pmod{23} \\ k_B &= 18 \end{aligned}$$

Výsledok je rovnaký: $k = k_A = k_B = 18$. Teraz Alica a Bob majú zdieľaný tajný kľúč, ktorý môžu použiť na šifrovanie a dešifrovanie správ. △

Tento príklad ilustruje základný DH protokol kľúčovej výmeny. Vo skutočnosti by sa pri výbere p a g použili väčšie čísla, aby bola komunikácia bezpečnejšia. Napríklad, pri použití 2048-bitového prvočísla by bolo ťažké pre útočníka získať tajné kľúče a a b pomocou bežných metód.

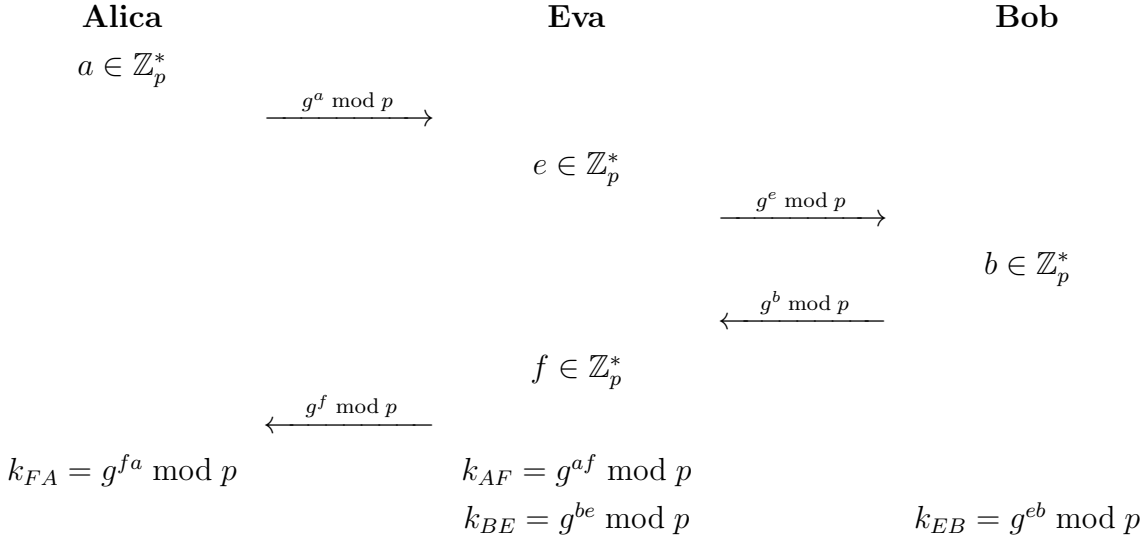
3.1 Man in the middle

Pri útoku man-in-the-middle útočník Eva zachytí komunikáciu medzi Alicou a Bobom a v komunikácii sa pokúsi vydávať za obe strany. Eva vygeneruje svoj vlastný pár kľúčov a použije ho na vytvorenie dvoch samostatných výmen kľúčov, jedného s Alicou a jedného s Bobom, pričom sa vydáva za druhú stranu. To umožňuje Eve odpočúvať a manipulovať s akoukoľvek následnou komunikáciou medzi Alicou a Bobom[17].

Predpokladajme, že Alica chce poslať správu Bobovi pomocou výmeny kľúčov Diffie-Hellman. Alica vygeneruje súkromný kľúč a a pošle verejný kľúč $g^a \bmod p$ Bobovi. Bob vygeneruje súkromný kľúč b a pošle verejný kľúč $g^b \bmod p$ Alici.

Eva zachytí komunikáciu a predstiera, že je Alica pri výmene kľúčov s Bobom. Eva

vygeneruje súkromný kľúč e a pošle verejný kľúč $g^e \bmod p$ Bobovi, pričom predstiera, že je Alica. Eva sa tiež vydáva za Boba pri výmene kľúčov s Alicou a vygeneruje ďalší súkromný kľúč f a verejný kľúč $g^f \bmod p$ a pošle ho Alici vydávajúc sa za Boba.



Takto Eva vytvorila dve samostatné výmeny kľúčov, jednu s Alicou a jednu s Bobom. Eva teraz môže zachytiť akúkoľvek komunikáciu medzi Alicou a Bobom, dešifrovať a upraviť správy podľa potreby a potom ich znovu zašifrovať pomocou príslušného zdieľaného tajného kľúča K, ktorý Eva vytvorila s každou stranou.

Príklad 3.2. Vypočítame kľúčovú výmenu DH so skutočnými číslami a naznačíme útok MITM.

Alica a Bob zvolia verejné parametre $p = 29$ a $g = 11$.

Alica náhodne zvolí $a = 6$ a vypočíta verejný kľúč A:

$$\begin{aligned} A &= g^a \pmod p \\ A &= 11^6 \pmod{29} \\ A &= 9 \end{aligned}$$

Bob náhodne zvolí $b = 7$ a vypočíta verejný kľúč B:

$$\begin{aligned} B &= g^b \pmod p \\ B &= 11^7 \pmod{29} \\ B &= 12 \end{aligned}$$

Eva odpočúva komunikáciu medzi Alicou a Bobom a zasahuje do nej. Zvolí si hodnoty

e a f a vypočíta verejné kľúče E a F, tak, aby platilo:

$$F^a \pmod p = A^e \pmod p$$
$$E^b \pmod p = B^f \pmod p$$

Eva si teda zvolí hodnoty:

$$e = 4$$
$$f = 5$$

Eva ďalej vypočíta(napr. hrubou silou) verejné kľúče E a F:

$$F^a \pmod p = A^e \pmod p$$
$$F^6 \pmod{29} = 9^4 \pmod{29}$$
$$\text{Pre } F = 4:$$
$$4^6 \pmod{29} = 9^4 \pmod{29}$$
$$E^b \pmod p = B^f \pmod p$$
$$E^7 \pmod{29} = 12^5 \pmod{29}$$
$$\text{Pre } E = 3:$$
$$3^7 \pmod{29} = 12^5 \pmod{29}$$

Eva pošle Bobovi číslo E namiesto A a Alici pošle číslo F namiesto B.
Alica dostane číslo F a vypočíta zdieľaný tajný kľúč:

$$k_{FA} = F^a \pmod p$$
$$k_{FA} = 4^6 \pmod{29}$$
$$k_{FA} = 7$$

Bob dostane číslo E a vypočíta zdieľaný tajný kľúč:

$$k_{EB} = E^b \pmod p$$
$$k_{EB} = 3^7 \pmod{29}$$
$$k_{EB} = 12$$

Eva vypočíta dva zdieľané kľúče k_{AF} a k_{BE}, ktoré by mali byť rovnaké ako k_{FA} a k_{EB}:

$$k_{AF} = A^e \pmod p = 9^4 \pmod{29} = 7$$
$$k_{BE} = B^f \pmod p = 12^5 \pmod{29} = 12$$

Eva úspešne zmanipulovala komunikáciu medzi Alicou a Bobom. Teraz Alica a Bob majú nesprávne zdieľané kľúče, ktoré sú 7 a 12. V skutočnosti by mal byť ich zdieľaný kľúč 28. △

Tento príklad ukazuje, ako Man-in-the-Middle útok môže kompromitovať Diffie-Hellman kľúčovú výmenu. Na ochranu proti takýmto útokom sa často používajú digitálne podpisy, certifikáty a protokoly ako Transport Layer Security (TLS), ktoré zaručujú autentifikáciu komunikačných strán a zabezpečujú komunikáciu proti MITM útokom[16].

3.2 Útok na malé podgrupy

Útok na malé podgrupy je známy typ útoku na kryptografický protokol Diffie-Hellman, ktorý sa týka slabých hodnôt verejných parametrov[17].

- Útočník Eva sleduje komunikáciu medzi Alicou a Bobom a získava verejné hodnoty A, B, p a g.
- Eva identifikuje, že generátor g vytvára malú podgrupu rádu r v $\mathbb{Z}_p^*$.
- Vypočíta hodnoty $g^x \pmod p$ pre x z množiny $\{1, 2, ..., r\}$.
- Porovná hodnoty A a B s vypočítanými hodnotami $g^x \pmod p$ a hľadá zhody.
- Ak Eva nájde zhody, získa tajné kľúče a a b, ktoré zodpovedajú hodnotám x.
- Eva vypočíta zdieľaný tajný kľúč k, buď ako $A^b \pmod p$ alebo $B^a \pmod p$.
- S vypočítaným spoločným tajným kľúčom k môže Eva dešifrovať a kompromitovať správy medzi stranami Alicou a Bobom.

Príklad 3.3. Vypočítame kľúčovú výmenu DH so skutočnými číslami a predvedieme útok na malé podgrupy.

Zvolíme verejné parametre $p = 23$ a generátor $g = 3$.

Alica a Bob si náhodne vyberú tajné kľúče $a = 4$ a $b = 7$ a vypočítajú verejné kľúče:

$$A = g^a \pmod p = 3^4 \pmod{23} = 12$$
$$B = g^b \pmod p = 3^7 \pmod{23} = 2$$

Strany si vymenia hodnoty A a B a vypočítajú spoločný tajný kľúč:

$$k_a = B^a \pmod p = 2^4 \pmod{23} = 16$$
$$k_b = A^b \pmod p = 12^7 \pmod{23} = 16$$

Eva sleduje komunikáciu a vypočíta rád generátora(hrubou silou):

$$\begin{aligned} 3^1 &\equiv 3 \pmod{23} \\ 3^2 &\equiv 9 \pmod{23} \\ 3^3 &\equiv 4 \pmod{23} \\ &\vdots \\ 3^{11} &\equiv 1 \pmod{23} \end{aligned}$$

Generátor $g = 3$ generuje podgrupu rádu 11 v $\mathbb{Z}_p^*$ a Eva môže využiť slabý generátor na útok na malé podgrupy. Identifikuje verejné kľúče A a B v prvkoch podgrupy

$$\{3, 9, 4, 12, 13, 16, 2, 6, 18, 8, 1\}$$

Eva môže vidieť, že hodnota $A = 12$ sa nachádza v podgrupe a zodpovedá hodnote $3^4 \bmod 23$. To znamená, že získala tajný kľúč strany A, $a = 4$.

Podobne Eva zistí, že hodnota $B = 2$ sa nachádza v podgrupe a zodpovedá hodnote $3^7 \bmod 23$. Získala teda tajný kľúč strany $B, b = 7$.

S týmito informáciami môže útočník teraz vypočítať spoločný tajný kľúč k rovnako ako strany A a B:

$$k_A = B^a \mod p = 2^4 \mod 23 = 16$$

alebo

$$k_B = A^b \pmod p = 12^7 \pmod{23} = 16$$

Eva úspešne získala tajný kľúč $k = 16$. △

3.3 Výber prvočísel

Veľké prvočísla. Pre praktické použitie volíme veľké prvočíslo p (minimálne 2048 bitov, ale odporúča sa 3072 bitov alebo viac)[18]. Čím väčšie prvočíslo zvolíme, tým ťažšie bude pre útočníka rozlúštiť kryptosystém pomocou metód, ako je útok hrubou silou.

Bezpečné prvočíslo je špeciálny typ prvočísla, ktorý sa často používa v kryptografii.

Definícia 3.1. Prvočíslo p je bezpečné prvočíslo, ak platí: $p = 2q + 1$, kde q je tiež prvočíslo.

Príklad 3.4. Nech $p = 59$. Potom

$$q = \frac{p-1}{2} = \frac{59-1}{2} = \frac{58}{2} = 29$$

Obe čísla $p = 59$ a $q = 29$ sú prvočísla, takže p je bezpečné prvočíslo. △

Použitím bezpečného prvočísla sú prakticky znemožnené útoky založené na malých podgrupách, pretože to znamená, že multiplikatívna grupa rádu $p = 2q + 1$ má iba dve malé podgrupy, {1} a {1, -1}, ale tým sa dá ľahko vyhnúť. Stačí ak pre algoritmus nevyberieme pre generátor g čísla 1 alebo -1, potom g generuje grupu rádu q, a ak je p obrovské, tak aj q. Keďže q je prvočíslo, podgrupa rádu q nemá žiadne ďalšie podgrupy[18].
Pre výber konkrétneho prvočísla sa prakticky využívajú dva princípy:

- **Náhodné generovanie prvočísel**: Tento prístup spočíva v použití kryptograficky silného generátora náhodných čísel na náhodný výber prvočísla p. Výhodou tohto prístupu je, že generované prvočísla sú nepredvídateľné a ťažko odhadnuteľné. To zvyšuje bezpečnosť protokolu, pretože útočníkovi sťažuje analýzu a útoky založené na predvídaní prvočísel. Nevýhodou je, že proces generovania náhodných prvočísel môže byť časovo náročný a závisí na kvalite generátora náhodných čísel.

- **Preddefinované prvočísla**: Tento prístup spočíva v použití preddefinovaných prvočísel a generátorov, ktoré boli navrhnuté špeciálne pre Diffie-Hellmanov protokol. Tieto prvočísla sú zahrnuté v rôznych štandardoch, ako sú RFC 3526 (MODP Grupy) alebo NIST SP 800-56A (tzv. Oakley prvočísla). Výhodou tohto prístupu je, že tieto prvočísla a generátory sú široko používané a testované, čo zaručuje ich bezpečnosť. Nevýhodou je, že ich použitie môže byť predvídateľné, čo môže v niektorých prípadoch zvýšiť riziko útokov.

4 RSA

RSA je kryptografický algoritmus, ktorý sa používa pre šifrovanie a digitálne podpisovanie správ. Je založený na matematickej ťažkosti faktorizácie súčinu dvoch veľkých prvočísel. RSA (Rivest-Shamir-Adleman) bol vynájdený v roku 1977 Ronom Rivestom, Adim Shamirom a Leonardom Adlemanom.

4.1 Postup výpočtu RSA

Princíp RSA zahŕňa niekoľko kľúčových krokov:

Generovanie kľúčov

- Výber dvoch veľkých prvočísel p a q. Veľkosť týchto čísel určuje mieru bezpečnosti šifrovania.
- Výpočet ich súčinu $n = p \cdot q$.
- Výpočet Eulerovej funkcie $\varphi(n) = (p-1)(q-1)$.
- Výber verejného exponentu e, takého, že platí $1 < e < \varphi(n)$ a e je nesúdeliteľný s $\varphi(n)$.
- Výpočet súkromného exponentu d, takého, že platí $d \cdot e \equiv 1 \pmod{\varphi(n)}$.

Verejný kľúč je dvojica (n, e) a súkromný kľúč je dvojica (n, d). Verejný kľúč môže byť zverejnený a použitý na šifrovanie správ, zatiaľ čo súkromný kľúč musí zostať tajný a slúži na dešifrovanie správ.

Šifrovanie

Pre šifrovanie správy X sa vypočíta šifrový text S pomocou verejného kľúča (n, e) a vzorca $S = X^e \pmod{n}$.

Dešifrovanie

Pre dešifrovanie šifrového textu S sa vypočíta pôvodná správa X pomocou súkromného kľúča (n, d) a vzorca $X = S^d \pmod{n}$.

Generátor	Verejný kľúč	Súkromný kľúč
$\downarrow$	$\downarrow$	$\downarrow$
p, q	(n, e)	(n, d)
$\downarrow$	$\downarrow$	$\downarrow$
Generovanie kľúčov	Šifrovanie	Dešifrovanie
$\downarrow$	$\downarrow$	$\downarrow$
(n, e, d)	$S = X^e \pmod{n}$	$X = S^d \pmod{n}$

V kryptosystéme RSA je využitá jednosmerná funkcia násobenia prvočísel. Pri generovaní kľúčov v RSA, násobíme dve veľké prvočísla p a q, aby sme získali číslo $n = p \cdot q$. Násobenie týchto dvoch čísel je jednoduchý proces, rozklad čísla n na jeho prvočíselné faktory (t.j. zistenie hodnôt p a q) je veľmi zložitý a časovo náročný proces, najmä ak sú p a q veľké prvočísla.

Príklad 4.1. V tomto príklade vypočítame parametre RSA. Pre jednoduchosť výpočtu budú použité malé čísla.

Vyberieme dve rôzne prvočísla, p a q:

$$p = 11$$
$$q = 17$$

Vypočítame $n = p \cdot q$:

$$n = 11 \cdot 17 = 187$$

Vypočítame Eulerovu funkciu:

$$\varphi(n) = (p-1)(q-1)$$
$$\varphi(n) = (11-1)(17-1) = 10 \cdot 16 = 160$$

Vyberieme verejný exponent e tak, aby platilo $1 < e < \varphi(n)$ a aby boli e a $\varphi(n)$ nesúdeliteľné

$$e = 7$$

Vypočítame súkromný exponent d, taký, že $de \equiv 1 \pmod{\varphi(n)}$, to znamená

$$d \equiv 7^{-1} \pmod{160}$$

Je to problém hľadania modulárnej inverzie, preto použijeme rozšírený Euklidov algoritmus.

Najprv vypočítame najväčšieho spoločného deliteľa pomocou Euklidovho algoritmu:

$$160 = 22 \cdot 7 + 6$$
$$7 = 6 \cdot 1 + 1$$
$$1 = 1 \cdot 1 + 0$$

Spätne vypočítame Bézoutove koeficienty

$$1 = 7 \cdot 1 - 6 \cdot 1$$
$$1 = 7 \cdot 1 - (160 - 22 \cdot 7)$$
$$1 = 23 \cdot 7 - 1 \cdot 160$$

Z toho platí

$$d = 23$$

Teraz máme kľúče:

$$\text{Verejný kľúč: } (n = 187, e = 7)$$

$$\text{Súkromný kľúč: } (n = 187, d = 23)$$

Predpokladajme, že chceme zašifrovať správu $X = 123$ pomocou verejného kľúča. Vypočítame šifrovanú správu S:

$$S = X^e \pmod{n} = 123^7 \pmod{187} = 183$$

Teraz dešifrujeme šifrovanú správu S pomocou súkromného kľúča a získame pôvodnú správu X:

$$X = S^d \pmod{n} = 183^{23} \pmod{187} = 123$$

△

Pri dešifrovaní v RSA môžeme zrýchliť výpočet tým, že použijeme čínsku vetu o zvyškoch. Namiesto priameho výpočtu správy modulo $n = pq$, vypočítame dešifrovanú správu modulo p a modulo q. Týmto spôsobom sa dá výpočet dešifrovania výrazne zrýchliť, najmä pri použití veľkých čísel.

Príklad 4.2. Chceme dešifrovať šifrovanú správu $S = 183$ z predošlého príkladu. Namiesto toho, aby sme priamo vypočítali $S^d \pmod{n}$, použijeme čínsku vetu o zvyškoch. Rozdelíme súkromný exponent d na d_p a d_q:

$$d_p = d \pmod{\varphi(p)} = d \pmod{p-1} = 23 \pmod{10} = 3$$
$$d_q = d \pmod{\varphi(p)} = d \pmod{q-1} = 23 \pmod{16} = 7$$

Vypočítame dešifrovanú správu modulo p a modulo q:

$$x_p = S^{d_p} \pmod{p} = 183^3 \pmod{11} = 2$$

$$x_q = S^{d_q} \pmod{q} = 183^7 \pmod{17} = 4$$

Teraz vypočítame modulárne inverzné hodnoty potrebné na kombináciu vyššie uvedených výsledkov:

$$y_p = p^{-1} \pmod{q} = 11^{-1} \pmod{17} = 14$$
$$y_q = q^{-1} \pmod{p} = 17^{-1} \pmod{11} = 2$$

Pomocou čínskej vety o zvyškoch kombinujeme výsledky a nájdeme text X. Riešime nasledujúce kongruencie:

$$X \equiv x_p \cdot q \cdot y_q \pmod{n}$$
$$X \equiv x_q \cdot p \cdot y_p \pmod{n}$$

Následne tieto dve kongruencie sčítame a aplikujeme modulo n:

$$X = (x_p \cdot q \cdot y_q + x_q \cdot p \cdot y_p) \pmod{n}$$
$$X = (2 \cdot 17 \cdot 2 + 4 \cdot 11 \cdot 14) \pmod{187} = 123$$

Čo vedie k výsledku:

$$X \equiv 123 \pmod{187}$$

Takže pomocou čínskej vety o zvyškoch a poskytnutých hodnôt sme úspešne dešifrovali šifrovaný text $S = 183$ na text $X = 123$. △

4.2 Bezpečnosť RSA

Odolnosť RSA voči útokom závisí od veľkosti použitých kľúčov. Dlhšie kľúče zvyšujú čas potrebný na faktorizáciu, čím zvyšujú aj bezpečnosť šifry. Z tohto dôvodu sa odporúča používať kľúče s dĺžkou aspoň 2048 bitov. V prípade potreby vyššej bezpečnosti, ako napríklad pre štátnu tajomstvo alebo finančné transakcie, sa môžu použiť aj dlhšie kľúče, napríklad 3072 alebo 4096 bitov[18].

Aj keď je RSA považovaný za bezpečný algoritmus, jeho bezpečnosť závisí od voľby parametrov. Niektoré z možných slabín a útokov zahŕňajú:

- Útoky hrubou silou: Keby sa útočník pokúsil vygenerovať všetky možné súkromné kľúče, časom by našiel správny kľúč. Avšak pre dostatočne veľké kľúče je tento

útok nerealistický z dôvodu obrovskej časovej náročnosti.

- Útoky súvisiace s malými verejnými exponentmi: Ak by sa použil malý verejný exponent, napr. $e = 3$, môže byť RSA zraniteľný voči útokom na základe matematických vlastností. Preto sa často používa štandardný verejný exponent $e = 65537$, ktorý poskytuje dobrú bezpečnosť a zároveň je výpočtovo efektívny.
- Útoky na základe slabých náhodných čísel: Ak by generátor náhodných čísel vytváral predvídateľné alebo slabé prvočísla p a q, útočník by mohol ľahšie rozložiť n a kompromitovať súkromný kľúč. Preto je dôležité používať kvalitné generátory náhodných čísel pri výbere prvočísel pre RSA kľúče.

4.3 Aplikácie RSA

RSA je široko používaný v mnohých oblastiach, kde je potrebná digitálna bezpečnosť. Medzi hlavné aplikácie patrí[16]:

- **Bezpečná komunikácia**: RSA sa často používa na šifrovanie a dešifrovanie správ v elektronickej komunikácii, ako sú e-maily, chaty alebo sociálne siete.
- **Digitálne podpisy**: RSA je využívaný na vytváranie a overovanie digitálnych podpisov, čo umožňuje overiť integritu a autentickosť správ alebo dokumentov.
- **SSL/TLS**: RSA je súčasťou SSL (Secure Sockets Layer) a jeho nástupcu TLS (Transport Layer Security), ktoré sa používajú na zabezpečenie webových stránok a komunikácie medzi klientom a serverom.
- **VPN (Virtual Private Networks)**: RSA je bežne používaný v protokoloch VPN na zabezpečenie komunikácie medzi zariadeniami a sieťou, čím zaručuje súkromie a bezpečnosť pri prenose dát.
- **Elektronické platby**: RSA sa využíva pri zabezpečovaní platobných kariet a elektronických platobných systémov, ako sú napríklad PayPal či Google Pay.
- **Autentifikácia a autorizácia**: RSA sa používa na overenie identity užívateľov a zariadení, čo zvyšuje bezpečnosť pri prístupe k citlivým údajom a službám.

4.4 Budúcnosť RSA a kvantové počítače

Kvantové počítače predstavujú hrozbu pre súčasné kryptografické algoritmy, vrátane RSA. Kvantové počítače využívajú kvantovú mechaniku na výrazne rýchlejšie riešenie niektorých matematických problémov, ako je napríklad faktorizácia veľkých čísel. V

prípade, že by sa v budúcnosti dosiahlo vytvorenie prakticky využiteľného kvantového počítača, bezpečnosť RSA by bola ohrozená[15].

Takýto vývoj by znamenal potrebu prechodu na tzv. post-kvantovú kryptografiu, ktorá je odolná voči útokom kvantovými počítačmi. V súčasnosti sa vyvíjajú alternatívne kryptografické algoritmy založené na iných matematických problémoch, ktoré sú odolné voči kvantovým útokom.

Pre zabezpečenie budúcich komunikácií a ochranu citlivých údajov je dôležité sledovať vývoj kvantových počítačov a post-kvantovej kryptografie a prispôsobiť sa novým technologickým hrozbám.

ZÁVER

Cieľom práce bolo vytvoriť most medzi matematickým základom a jeho praktickým využitím, teda ukázať, ako určité vlastnosti čísel umožňujú projektovať bezpečné kryptografické systémy.

V kapitole venovanej teórii čísel sú konkrétne príklady navrhnuté tak, aby čitateľovi pomohli získať hlbšie pochopenie deliteľnosti, prvočísel, kongruencií, modulárnej aritmetiky a ďalších matematických konceptov. Okrem toho je v tejto časti práce prezentovaných niekoľko implementácií algoritmov teórie čísel v jazyku Python, ktoré umožňujú praktickú aplikáciu týchto konceptov. Súčasťou prílohy sú unit testy(Príloha 2), ktoré dokladujú správnosť implementácií.

V druhej časti práce, ktorá sa zaoberá jednosmernými funkciami, sú popísané základné vlastnosti, ktoré sa dajú zovšeobecniť pre veľké množstvo funkcií, ktoré sú súčasťou systémov súčasnej komerčnej bezpečnosti. Následne je vymenované množstvo rôznych typov jednosmerných funkcií, čím je poukázané na rozmanitosť matematických vlastností, ktoré sú využívané v špecializovaných kryptografických systémoch.

V rámci kapitol venovaných protokolu Diffie-Hellman a RSA sú uvedené riešené príklady a ilustrácie, ktoré názorne ukazujú princíp použitia jednosmerných funkcií v systéme s verejným kľúčom, ale aj riziká spojené s výmenou kľúčov.

Najväčším prínosom tejto práce je vytvorenie kompaktného textu, ktorý zrozumiteľne popisuje matematické koncepty nevyhnutné pre štúdium kryptografie. S dôrazom na jasnú a zrozumiteľnú prezentáciu, táto práca poskytuje čitateľovi základ a potrebné matematické vedomosti na pochopenie a zvládnutie štúdia kryptografických systémov. Tento učebný text môže poslúžiť študentom kryptografie, ale aj čitateľom, ktorí chcú bez hlbokých matematických znalostí ľahšie porozumieť danej problematike.

ZOZNAM POUŽITEJ LITERATÚRY

[1] ROSEN, K.: *Discrete mathematics and its applications*. New York: Random House, 8 vydání, 2018, ISBN 978-1-260-09199-1.

[2] BURTON, D.: *Elementary Number Theory*. Princeton University Press, 6 vydání, 2007, ISBN 978-0-073-38314-9.

[3] CRANDALL, R.; POMERANCE, C.: *Prime Numbers: A Computational Perspective*. Springer Science Business Media, první vydání, 2006, ISBN 978-0387-25282-7.

[4] Eratosthenes [online]: Plzeň: © Techmania Science Center. [cit. 2023-05-14], dostupné z: http://edu.techmania.cz/cs/encyklopedie/vedec/1131/eratosthenes.

[5] ZIXING, Wang: Methods of Primality Testing. [online]. [cit. 2023-03-29], dostupné z: https://www.researchgate.net/publication/348899360_Methods_of_Primality_Testing.

[6] HOBST, E.; HOBSTOVÁ, M.: Carl Friedrich Gauss — zakladateľ modernej matematiky. *Pokroky matematiky, fyziky a astronomie*, ročník 52, č. 4, 2007: s. 296–307, ISSN 0032-2423.

[7] HOFFSTEIN, J.; PIPHER, J.; SILVERMAN, J.: *An introduction to mathematical cryptography*. New York: Springer, první vydání, 2008, ISBN 978-0-387-77993-5.

[8] Bézout's Identity [online]: PAULI, Sebastian. [cit. 2023-04-21], dostupné z: https://mathstats.uncg.edu/sites/pauli/112/HTML/secbezout.html.

[9] Eulerova věta [online]: Algoritmy.net, Jan Neckář © 2016. [cit. 2023-04-13], dostupné z: https://www.algoritmy.net/article/57/Eulerova-veta.

[10] Fermatův test prvočíselnosti [online]: Algoritmy.net, Jan Neckář © 2016. [cit. 2023-04-13], dostupné z: https://www.algoritmy.net/article/61/Fermatuv-test.

[11] DIFFIE, W.; HELLMAN, M.: New Directions in Cryptography. *IEEE Transactions on Information Theory*, ročník 6, č. 6, november 1976: s. 644–654, ISSN 0018–9448.

[12] VEGA, F.: The existence of one-way functions. *HAL-open science*, 2014, dostupné z: https://hal.science/hal-01020088v2.

[13] HOLDEN, J.: *The Mathematics of Secrets*. Princeton University Press, první vydání, 2018, ISBN 978-0-691-18331-2.

[14] GALBRAITH, S.: *Mathematics of Public Key Cryptography.* Cambridge University Press, první vydání, 2012, ISBN 978-1-107-01392-6.

[15] WONG, D.: *Real-world cryptography.* Shelter Island: Manning, první vydání, 2021, ISBN 978-1-61729-671-0.

[16] SCHNEIER, B.: *Applied cryptography: protocols, algorithms, and source code in C. 20th anniversary edition.* Indianapolis: Wiley, 2015, ISBN 978-1-119-09672-6.

[17] KOHNO, F. N. a. S. B., Tadayoshi: *Cryptography engineering : design principles and practical applications.* Indianapolis, In: Wiley Pub., první vydání, 2010, ISBN 978-0-470-47424-2.

[18] FERGUSON, N.; SCHNEIER, B.: *Practical Cryptography.* New York: Wiley, 2003, ISBN 978-0-471-22357-3.

ZOZNAM TABULIEK

ZOZNAM PRÍLOH

Príloha 1: TeoriaCisel.py
Príloha 2: UnitTesty.py