Teorie čísel a šifrování

J. Hlava, P. Šebek
Gymnázium Jiřího Ortena Kutná Hora, Gymnázium Uherské Hradiště
honzahlava@seznam.cz, artimis@seznam.cz

Abstrakt:
Naše práce se zabývá šifrováním zpráv. Ukazuje hlavně základní symetrické šifrovací metody. Součástí naší práce byl šifrovač a dešifrovač, který pracuje na metodě šifrování pomocí kódovací matice. Kódovací program je relativně jednoduchý a výsledná šifra těžko rozluštitelná při neznalosti klíče.

1 Úvod

Kryptologie neboli šifrování se používá pro bezpečný přenos informací mezi dvěma lidmi. Její součástí jsou kryptografie (šifrování) a kryptoanalýza (dešifrování) [1]. Problém v šifrování je ten, že vždycky existuje cesta jak dešifrovat kódovou zprávu, tudíž čím jednodušší máme kryptografickou metodu, tím jednodušeji může potencionální nepřítel zprávu rozluštit.

K rozluštění zprávy je nutná znalost klíče, tedy parametru funkce, kterou použijeme ke kódování. Šifrovací metody se dělí na symetrické, které jsou starší a tím pádem jednodušší, a na asymetrické, které už vyžadují dva klíče (jeden klíč je veřejný, pomocí kterého se zpráva zašifruje, a druhý je soukromý, který zprávu dešifruje uživateli). My jsme se zabývali symetrickou šifrovací metodou s pomocí kódovací matice.

2 Šifrování

Pozadí šifrování je většinou založené na matematice, proto je důležité převést zprávu na číselný kód. Písmenům přiřadíme numerickou hodnotu (například písmenu A přiřadíme hodnotu 0, B 1, …, Z 25). S tímto kódem se potom provede matematická operace, číselné hodnoty se převedou zpět na písmena a zašifrovaná zpráva je připravená k odeslání. Dešifrovací program používá inverzi matematické operace, která byla použita při šifrování. Tato operace je pochopitelně použita na číselnou podobu zašifrované zprávy.

3 Symetrické šifrování

Metoda posunutí

Nejjednodušší metoda, která se nazývá posunutí, funguje na principu, že se jednomu písmenu přiřadí číslo, ke kterému se potom přičte klíčové číslo a výsledek je zpátky převeden na

písmeno a zprávu máme zašifrovanou [2]. Při dešifrování se písmena převedou na číselnou hodnotu a od ní se odečte klíčové číslo. Potom se již opět převede číselná hodnota na písmena a zpráva je dešifrována. Všechny tyto operace musíme provádět modulo 26 (počet písmen abecedy bez mezer), abychom zůstali v určeném intervalu. Tato metoda je jedna z nejjednodušších a znali ji již za dob Říma (Julius Caesar ji často používal, jeho klíčové číslo bývalo 3). Tato metoda šifrování je celkem bez obtíží dešifrovatelná, i když neznáme klíčové číslo. Stačí si všimnout písmene, které se tam nejvíce vyskytuje a také písmena s nejnižším výskytem nebo žádným, a pak porovnat s jazykem ve kterém je pravděpodobně zpráva zašifrována. Každý jazyk má některá písmena více používaná než jiná a zase některá písmena nepoužívá skoro vůbec (v češtině například W a X). Porovnáme posunutí nejpoužívanějšího písmene v nezašifrované podobě a zašifrované podobě a číslo, které nám vyjde, by mělo být hledaným klíčovým číslem. Se znalostí klíčového čísla je to již jednoduché dešifrovat.

Šifrování pomocí matic

Zašifrovanou zprávu pomocí této metody je již obtížnější dešifrovat. Klíč je tvořen maticí (v našem případě matice 2 x 2 tvořená přirozenými čísly). První krok je stejný jako u předcházejícího typu šifrování a to, že písmena převedeme do číselné podoby. Na rozdíl od minulého typu zde kódujeme vždy 2 po sobě jdoucí čísla najednou pomocí maticového násobení podle následující rovnice

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} z_i \\ z_{i+1} \end{pmatrix} = \begin{pmatrix} k_i \\ k_{i+1} \end{pmatrix}.$$

Matice nemůže mít libovolný tvar, musí být tvořena vhodnými čísly, protože jinak může nastat problém s nejednoznačným šifrováním a tím pádem nemožným dešifrováním. Jestli je naše matice vhodná si ověříme tím, že zjistíme, zda největší společný dělitel determinantu D (kde $D = ad - bc$) a počtu písmen v abecedě je 1.

Po násobení maticí jsme dostali číselnou hodnotu, kterou opět musíme vzít modulo 26, abychom byli zpět v požadovaném intervalu. Teď již převedeme číselné hodnoty na písmena a zprávu máme chráněnou před nežádanými čtenáři.

Následná dešifrace je již složitější, opět musíme provést inverzi k provedené operaci (tedy k násobení matic). Přesněji řečeno, zprávu – opět po dvojicích písmen – násobíme maticí inverzní k matici kódové. Tuto inverzní matici vypočteme podle následujícího vzorce

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \begin{pmatrix} D^{-1}c & -D^{-1}b \\ -D^{-1}c & D^{-1}a \end{pmatrix},$$

kde D^{-1} je inverzní prvek k determinantu D (v množině $\{0,1,\ldots,25\}$). Tento inverzní prvek hledáme pomocí rozšířeného Euklidova algoritmu [3]. Poté musíme použít modulo 26, abychom byli v požadovaném intervalu a poslední krok je převést číselné hodnoty zpět na písmena.

4 Asymetrické šifrování

Tento způsob šifrování se vyznačuje použitím dvou klíčů (veřejný a soukromý). Pokud chce někdo zašifrovat zprávu pro nás, použije náš veřejný klíč, který může být např. vystaven na webu. Pokud chceme naopak zprávu rozšifrovat, musíme použít soukromý klíč, který pochopitelně držíme v tajnosti. Na tomto principu jsou založeny např. také digitální podpisy.

5 Shrnutí

Naším úkolem bylo sestrojit program, který využíval k šifrování matice. Když jsme porozuměli principu této metody, již bylo velmi jednoduché naprogramovat program, který toto zvládne. Tento program je tvořen dvěma programy. První program zajišťuje zakódování zprávy a následné uložení kódované zprávy do textového souboru a druhý program načte zakódovanou zprávu z textového souboru, rozšifruje ji a pak zobrazí rozšifrovanou podobu. Za zmínku stojí, že první program je napsán v Pascalu a druhý je v C++.

Poděkování

Děkujeme ČVUT a FJFI za zprostředkování fyzikálního týdne. Také děkujeme našemu supervizorovi Ing. Petru Ambrožovi, Ph.D., který nám s prací významně pomohl.

Reference:

[1] KRYPTOGRAFIE. *http://cs.wikipedia.org/wiki/Kryptografie*

[2] KOBLITZ, N.: *A Course in Number Theory and Cryptography* Springer, 1994, pp. 235.

[3] EXTENDED EUCLIDEAN ALGORITHM. *http://en.wikipedia.org/wiki/Extended_Euclidean_algorithm*