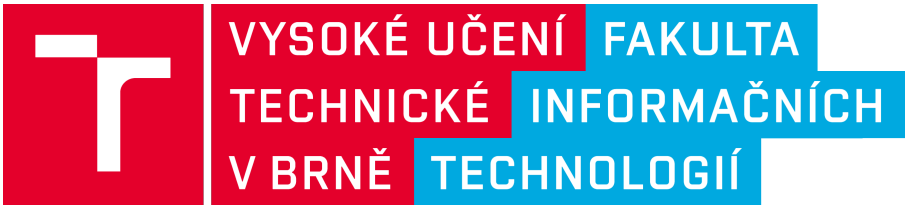

KRY Projekt 2023

Vojtěch Fiala (**xfiala61**)

1 Afinní šifra

Afinní šifra je jedna z existujících substitučních šifer, která umožňuje šifrovat a dešifrovat znaky na základě dvou klíčů – **a** a **b** s využitím následujícího výpočtu:

$$\text{Šifrování: } E(x) = (a \times x + b)\ mod\ n$$

$$\text{Dešifrování: } D(x) = a^{-1}(x - b)\ mod\ n$$

V rovnicích výše značí *E(x)* zašifrovaný znak, *D(x)* dešifrovaný (původní) znak, *n* značí počet písmen v abecedě, *x* značí původní znak a a^{-1} značí multiplikativní inverzi[1]. Klíč a musí být nesoudělné číslo s číslem n, tedy s počtem znaků v abecedě. To znamená, že jediným dělitelem čísla a a m může být pouze číslo 1. Klíč b může nabývat libovolné hodnoty.

Jelikož patří afinní šifra mezi substituční šifry, je možné ji poměrně jednoduše prolomit s využitím frekvenční analýzy, čímž se budou zabývat následující řádky.

2 Prolomení šifry

Základním stavebním prvkem programu na prolomení šifry je frekvenční analýza. S jejím využitím jsou poté odhaleny odpovídající znaky ve výchozím textu a znalost jak výchozího, tak zašifrovaného textu vede k možnosti získání klíče. Následující řádky popisují jednotlivé kroky, které můj algoritmus využívá k prolomení šifry.

2.1 Frekvence písmen v textu

Prvním krokem, na základě kterého bude později možné provést frekvenční analýzu, je výpočet samotné frekvence jednotlivých znaků v textu. K tomu slouží v mé implementaci tabulka obsahující jako klíč dané písmeno a jako hodnotu počet jeho výskytů v textu. Celý text je poté ve smyčce analyzován a jsou zjištěny odpovídající počty jednotlivých znaků.

Aby bylo možné frekvenční analýzu provést, je nutné vědět, v jakém jazyce je zašifrovaný text napsaný a pro tento jazyk mít k dispozici četnost výskytu jednotlivých písmen[2].

[1] `https://www.algoritmy.net/article/46/Multiplikativni-inverze`
[2] `https://www.matweb.cz/frekvencni-analyza/`

Jak četnost písmen v zašifrovaném textu, tak četnost písmen v českém jazyce jsou poté seřazeny od nejvíce použitého po nejméně použité za účelem pozdějšího porovnání.

2.2 Hledání správného klíče

Jak bylo již zmíněno v popisu šifry, klíč a může, jelikož známe jazyk a tedy délku n, nabývat pouze určitých, předem známých hodnot. Klíč b může nabývat libovolné hodnoty, prakticky to ovšem z důvodu modulárního dělení znamená, že jeho hodnotu lze převést na libovolnou hodnotu z intervalu $\langle 0, n-1 \rangle$.

V mém algoritmu jsou tedy 2 smyčky – jedna iteruje přes všechna možná a a druhá přes všechna možná b. V těle vnitřní smyčky poté dochází k samotné frekvenční analýze.

2.3 Frekvenční analýza

Frekvenční analýza probíhá nad určitým počtem nejčastějších písmen zašifrovaného textu a českého jazyka, která jsou vůči sobě porovnávána pro každou dvojici klíčů. Výchozí hodnota, která byla zvolena na základě nejlepší úspěšnosti, je 20 nejčastějších písmen. V případě, že tolik různých písmen vstupní text neobsahuje, je hodnota snižována, což ovšem vede k nižší přesnosti.

Tato písmena jsou, stejně jako znaky v české abecedě, seřazeny od nejčastějších po nejméně časté. V rámci frekvenční analýzy dochází ve 2 smyčkách k srovnání, zda jedno písmeno české abecedy vede s danými klíčy a a b k jednomu z písmen zašifrovaného textu. Dojde k jeho zašifrování s využitím jedné z kombinací klíčů a v případě, že odpovídá zašifrovanému písmenu, je navýšeno počítadlo. Tento postup ilustruje následující funkce:

```
int getMatches(vector<char> most_commons,
vector<char> my_commons, int a, int b) {
    int matches = 0;
    for (auto c : most_commons) {
        for (auto c2 : my_commons) {
            if ((c * a + b) % 26 == c2) {
                matches++;
            }
        }
    }
```

```
    return matches;
}
```

Jako správná kombinace klíče a a b jsou zvoleny ty hodnoty, pro které bylo počítadlo správných výsledků nejvyšší. Byl proveden i pokus o zpřesnění algoritmu s využitím porovnávání bigramů, což ovšem přesnost mého algoritmu spíše zhoršilo než naopak.

3 Výsledky

Testování probíhalo na poskytnutém souboru se zašifrovanými texty, kdy můj algoritmus v případě porovnávání 20 písmen pro každý klíč dosahoval 100% úspěšnosti. V případě, že bylo porovnáváno písmen méně, snižovala se i úspěšnost algoritmu a některé texty byly analyzovány nesprávně.