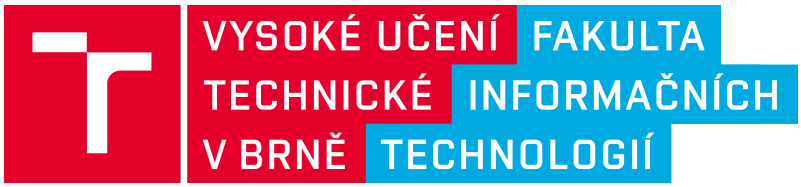
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



KRYPTOGRAFIE
2020/2021

Projekt 1: Vigenerova šifra

Tomáš Odehnal (xodehn08)

Brno, 4. dubna 2021

1 Zadání

Cílem tohoto projektu bylo si nastudovat testy klasických šifer a v jazyce C/C++ implementovat program, který bude hledat heslo k anglickému textu zašifrovenému pomocí šifry Vigenere. Hledání hesla zahrnuje provedení Friedmanova a Kasiského testu, ze kterých se dá odhadnout délka hesla. V další části projektu se hledá skutečná délka hesla a nakonec se zjišťuje samotné heslo (klíč).

2 Implementace

Projekt jsem implementoval v programovacím jazyce C++ verze C++17. Program mám rozdělený na tři třídy: Kasiského test (`Kasiski`), Friedmanův test a sloupcový test indexu konicidence (`Friedman`) a nalezení hesla (`KeyCracker`). Třída `Friedman` obsahuje implementaci Friedmanova testu i implementaci sloupcového testu indexu konicidence, který je využit pro získání skutečné délky hesla.

Program si nejprve uloží vstupní text tak, že odstraní všechny znaky, které nepatří do anglické abecedy, a zbylý text uloží jako malé znaky (tedy `a-z`). Následně se spustí tyto tři testy. Každý z těchto testů je prováděn ve svém vlastním vlákně.

2.1 Kasiského test

Test je spuštěn zavoláním metody `Kasiski::doTest()` a zahrnuje hledání řetězců znaků, které se v zašifrovaném textu opakují. Tyto řetězce by měly mít délku 3 znaky (dále trigramy) nebo více znaků. *V implementaci jsem použil pouze hledání trigramů, protože řetězce s více znaky výsledek příliš nezměnily, pouze test trval déle.* Následně vzdálenosti mezi následnými stejnými trigramy budou s velkou pravděpodobností násobky délky hesla. [1].

Trigramy jsou nalezeny pomocí metody `Kasiski::do_n_graph()`, kde metoda postupně projde celý zašifrovaný text a hledá trigramy. Nalezené trigramy jsou uloženy do proměnné `trigrams`, což je datová struktura Mapa a klíčem je řetězec (trigram) a záznam obsahuje datovou strukturu Dvojice (Pár). První záznam dvojice je index prvního výskytu nalezeného trigramu a druhý záznam jsou vzdálenosti dalších nalezených výskytů. Každý trigram (a jeho výskyty) je uložen pouze jednou.

Následně se hledá nejčastější výskyty největších společných dělitelů. Tito dělitelé a jejich výskyty jsou uloženy do mapy `dividers`. Pole tedy obsahuje kadidáty na skutečnou délku hesla. Následně je ale potřeba, aby metoda vrátila pole dělitelů seřazených podle výskytů. K tomu slouží metoda `sortMap()`, která vrátí vektor dvojic, protože datová struktura Mapa je automaticky řazena podle klíče. Dvojice v tomto vektoru obsahují dělitel a jeho počet výskytů. Tento vektor je z hlavní části programu dostupný prostřednictvím metody `Kasiski::getResult()`.

2.2 Friedmanův test

Test je spuštěn zavoláním metody `Friedman::doTest()`. U tohoto testu známe index koincidence κ_p. Pro anglický jazyk má hodnotu kolem $0,67$ (v mé implementaci má hodnotu $0,667$). Dále známe pravděpodobnost κ_r, což odpovídá náhodnému výběru znaku z náhodného textu (hodnota je $1/26 \approx 0,385$). Je potřeba spočítat index koincidence κ_0 zašifrovaného textu. [2]

Pro získání κ_0 je potřeba nejprve spočítat frekvenční charakteristiku zašifrovaného textu (počet výskytů jednotlivých znaků anglické abecedy v zašifrovaném textu). Následně se κ_0 spočítá pomocí následujicího vzorce:

$$\kappa_0 = \frac{\sum_{i=1}^{c} n_i(n_i - 1)}{N(N-1)} \tag{1}$$

kde c je velikost anglické abecedy (26 znaků), N je délka zašifrovaného textu a n_1 až n_c jsou frekvence jednotlivých znaků (celá čísla).

Odhad délky klíče se potom vypočítá pomocí vzorce:

$$\frac{\kappa_p - \kappa_r}{\kappa_0 - \kappa_r}$$

Výsledek testu je dostupný pomocí metody `Kasiski::getFriedmanKey()`. A odhadnutá vzdálenost není použita pouze pro požadovaný výstup, ale maximální délku klíče ve sloupcovém testu indexu koincidence.

2.3 Sloupcový test indexu koincidence

Test je spuštěn zavoláním metody `Friedman::doColumnWiseCi()`. Metoda musí na začátku vyčkat na výsledek Friedmanova testu pro maximální odhadovanou velikost hesla. Podstatou tohoto testu je, že postupně odhaduji délku h hesla. Postupně inkrementuji délku od 1 po dvojnásobek odhadnuté délky hesla z Friedmanova testu. Zašifrovaný text zapíši do h sloupců a pokud je odhad délky hesla správný, pak všechny znaky v každém sloupci byly zašifrovány stejným znakem hesla. Následně se podle vzorce 1 spočítá index koincidence každého sloupce zvlášť. Poté se spočítají průměrné hodnoty indexů koincidence pro jednotlivé délky hesla. Stejně jako u Kasiského testu je výsledkem sloupcového testu pole (vektor) kandidátů na délku hesla.

Mezivýsledky sloupců odpovídajícím délkám hesla jsou uloženy v mapě `columnsCI`, kde první záznam je délka hesla a druhý záznam je vektor jednotlivých sloupců. Výpočty jednoho sloupce jsou ukládány do struktury `columnInfo`.

Nejprve se pro každý sloupec spočítají frekvence (četnost) jednotlivých znaků (`letterMap`) a celkový počet znaků (`letterCnt`). Následně se pro daný sloupce spočítá index koincidence (`kappa0`). Nakonec se pro každou délku hesla spočítá průměrná hodnota indexu koincidence. Výsledný vektor kandidátů délky hesla je dostupný prostřednictvím metody `Friedman::getColumnCI()`.

2.4 Skutečná délka hesla a heslo

Jakmile jsou všechny testy provedené, tak je potřeba získat skutečnou délku hesla. Ta se získá z kandidátních délek Kasiského i sloupcového testu. Problém u kandidátních délech Kasiského testu je, že mezi kandidáty patří, jak násobky skutečné délky, tak jeho dělitelé (např. skutečná délka hesla je 6 a mezi kandidáty patří i čísla 2 a 3). Vektor těchto kandidátů je seřazen podle četnosti a ve většině případů skutečná délka hesla nemá největší četnost, největší četnost většinou mívají jeho dělitelé (většinou čísla 2 a 3).

U kandidátů délek sloupcového testu je situace podobná. Zde mezi kandidáty patří pouze násobky skutečné délky hesla a ne jeho dělitelé. V implementaci mezi kandidáty počítám délky, které mají index koincidence větší než $0,059$ (což mají: skutečná délka hesla a jeho násobky). Stejně jako u Kasiského kandidátů i zde nemá skutečná délka hesla největší hodnotu indexu koincidence. **Skutečná délka hesla je tedy nejmenší číslo z průniku těchto dvou vektorů (množin).**

Jakmile mám skutečnou délku hesla, je potřeba uhodnout samotné heslo. To se provede pomocí frekvenční analýzy. Stejně jako u sloupcového testu indexu koincidence se zapíše zašifrovaný text do sloupců. Nyní již, ale je známá skutečná délka hesla. Každý sloupec obsahuje otevřený text, který byl zašifrován monoalfabetickou Caesarovou šifrou (viz [3]). Je tedy potřeba pouze zjistit o kolik znaků je daný sloupec zašifrovaného textu posunutý od znaků otevřeného textu.

Spočítám si frekvenci jednotlivých znaků anglické abecedy. Následně provedu korelaci 26-ti variant posunutí této frekvence s frekvencí anglického textu (`engLettersFreq`) [4]. Varianta s největší shodností s frekvencí anglického textu odpovídá konkrétnímu znaku hesla. Toto se provede pro všechny sloupce, odpovídající konkrétním pozicím znaků hesla a získáme skutečné heslo. [2]

Reference

[1] Wikipedia contributors. Kasiski examination — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Kasiski_examination&oldid=1003359133, 2021. [Online; accessed 29-March-2021].

[2] Wikipedia contributors. Vigenère cipher — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Vigen%C3%A8re_cipher&oldid=1011049884, 2021. [Online; accessed 29-March-2021].

[3] Wikipedia contributors. Caesar cipher — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Caesar_cipher&oldid=1014979830, 2021. [Online; accessed 30-March-2021].

[4] LEWAND Robert Edward. Relative frequencies of letters in general english plain, Cryptographical Mathematics. http://cs.wellesley.edu/~fturbak/codman/letterfreq.html. [Online; accessed 30-March-2021].