VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

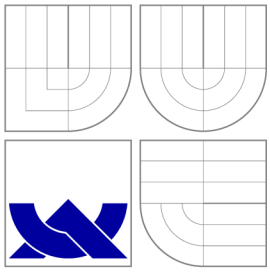
LUŠTĚNÍ SUBSTITUČNÍCH ŠIFER V KLASICKÉ KRYPTOGRAFII

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

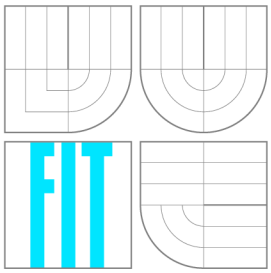
AUTOR PRÁCE
AUTHOR

MARTIN KULICH

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

LUŠTĚNÍ SUBSTITUČNÍCH ŠIFER V KLASICKÉ KRYPTOGRAFII

CRACKING OF SUBSTITUTION CIPHERS IN CLASSICAL CRYPTOGRAPHY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MARTIN KULICH
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAROSLAV STRUŽKA
SUPERVISOR

BRNO 2012

Abstrakt

Práce uvádí programový nástroj pro automatické luštění monoalfabetických substitučních šifer s využitím slovníkového útoku. Vychází z již uvedených řešení pro anglický jazyk, které však měly omezení na vlastnosti šifrového textu. Práce se zaměřuje zejména na šifry, jejichž šifrový text není dělený na slova. Také se dotýká specifik českého jazyka – skloňování a jeho vlivu na velikost slovníku.

Abstract

The thesis provides a tool for automatic monoalphabetic substitution ciphers cracking using most common words dictionary. It is based on some previous solutions for english language which were not designed for cracking ciphertext without word divisions. This thesis is focused on cracking ciphertexts without word divisions. It also looks at specifics of czech language – declension and its influence on dictionary size.

Klíčová slova

šifry, luštění, slovníkový útok

Keywords

ciphers, cracking, dictionary search

Citace

Martin Kulich: Luštění substitučních šifer
v klasické kryptografii, bakalářská práce, Brno, FIT VUT v Brně, 2012

Luštění substitučních šifer v klasické kryptografii

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jaroslava Stružky.

.....
Martin Kulich
16. května 2012

Poděkování

Děkuji panu Mgr. Tobiáši Šmolkovi z Fakulty informatiky Masarykovy univerzity za doporučení dobrých zdrojů, ze kterých práce čerpá a na něž navazuje. Dále děkuji Výzkumné skupině zpracování přirozeného jazyka za poskytnutí podkladových dat pro vytvoření slovníku.

Obsah

Úvod

Již od nepaměti pociťují lidé potřebu sdělovat si informace důvěrně. Nemohou-li použít osobní styk a jsou nuceni předávat si důvěrnou zprávu písemně, musí ji zabezpečit tak, aby náhodný čtenář nebyl schopen zprávu přečíst. Klasická kryptografie řeší možnosti šifrování, dešifrování a luštění zpráv, které lze provádět bez pomoci výpočetní techniky. Dnes se již používá zřídka, ale ještě v poměrně nedávné minulosti měla své zastoupení, zejména ve vojenských záležitostech.

Zprávy z různých vojenských konfliktů bývají uloženy v archivech, lze podle nich rekonstruovat průběhy bitev, politické pozadí konfliktu i činnost odboje. V archivech jsou však pouze zašifrované zprávy bez klíče. I když je známý způsob dešifrování šifry, nelze jej využít bez klíče. Ten může být ukryt v jiné zprávě, nebo se nemusel vůbec dochovat.

Musíme se tedy uchýlit k metodě, kterou používá během války nepřítel: pokusit se šifru prolomit bez znalosti klíče. Tato práce poskytuje za tím účelem účinný nástroj, který ušetří luštiteli mnoho hodin času.

Členění práce

V kapitole 1 budou vysvětleny základní pojmy z oblasti klasické kryptografie a matematicky formulován pojem kryptografického systému Také budou představeny základní principy, na kterých pracují šifry klasické kryptografie včetně příkladů (vyjma substitučních šifer).

Kapitola 2 se zabývá kryptografickými systémy, založenými na principu substituce. Ukazuje základní typy substituce včetně příkladů, zejména se zaměřuje na monoalfabetickou substituci. Také představuje historické i současné požadavky na bezpečnost kryptosystémů.

Ve 3. kapitole budou ukázány způsoby kryptoanalýzy monoalfabetické substituce pomocí tradičních metod „tužky a papíru“ s využitím charakteristik jazyka, ale i způsoby algoritmické, které byly k luštění monoalfabetické substituce již využity. Zároveň zde bude vysvětlena složitost českého jazyka z hlediska automatizované kryptoanalýzy.

Kapitola 4 se zabývá samotným problémem automatizované kryptoanalýzy. Dělí problém na podproblémy a pro každý z nich navrhuje řešení – ať už na základě algoritmů zmíněných v kapitole 3 nebo řešení zcela nové. Výsledkem je algoritmus, který je schopen luštit i monoalfabetické šifry s šifrovým textem bez zřetelného oddělení slov.

V kapitole 5 jsou rozvedeny některé implementační detaily navrženého algoritmu. Zejména je zde probrán slovník, z kterého algoritmus při luštění čerpá.

Kapitola 6 se zabývá testováním implementovaného algoritmu.

Konvence zápisu

V textu se budou objevovat ukázky různých šifer a šifrovacích postupů. Je dodržován jednotný vzhled šifer, aby se čtenář mohl rychle orientovat.

Otevřený text (OT) neboli text před zašifrováním je psán vždy malými písmeny a neproporciálním písmem.

```
priklad otevreneho textu
```

Šifrový text (ŠT) neboli text po zašifrování je psán vždy velkými písmeny a neproporciálním písmem. Je zapisován ve standardizované formě po pěti znacích. Ta se používá kvůli lepší kontrole zprávy, neboť pět znaků si je člověk schopen při přepisu šifrového textu snadno zapamatovat.

```
QSJLM BETJG SPWFI PUFYU V
```

Jsou-li otevřený a šifrový text psány pod sebou, mohou být řádky uvozeny zkratkami OT a ŠT. Otevřený text může být v tom případě také dělen po pěti znacích.

```
OT:   textu vozen yzkra tkouo t
ŠT:   QHNQK CPUHV EUMGJ QMPKP Q
```

Kapitola 1

Klasická kryptografie

V následující kapitole budou vyjasněny některé základní pojmy a vysvětleno, co je kryptografický systém. Následně bude vysvětlen pojem klasická kryptografie, předvedeno několik kryptografických systémů a principů v klasické kryptografii a budou představeny podmínky, které musí být splněny, aby mohl být kryptografický systém považován za bezpečný.

1.1 Základní pojmy

Nežli načneme téma klasické kryptografie, je vhodné zmínit, co se skrývá pod pojmy kryptologie, kryptografie a kryptoanalýza [2, 6]. Také se podíváme na vztah pojmů šifrování, dešifrování a luštění.

1.1.1 Kryptologie, kryptografie, kryptoanalýza

Co mají tyto tři pojmy společného? Není náhodou, že je to právě stejný základ slova. *Krypto-* totiž pochází z řeckého slova $\kappa\rho\iota\pi\tau\omega\sigma$ – skrýt.

Kryptologie je vědecký obor, který studuje způsoby ochrany zpráv před útočníkem a testování těchto způsobů proti útokům. Zahrnuje podobory: kryptografii a kryptoanalýzu.

Kryptografie je disciplína, která se zabývá šifrováním a dešifrováním zpráv neboli transformací otevřeného textu (obecně dat) na šifrový a naopak. Také se zaměřuje na zabezpečení jejich přenosu.

Kryptoanalýza se naopak zabývá luštěním šifrových textů, tedy prolomením jejich ochrany. Existují různé druhy takových útoků:

- *Útok na šifrový text (ciphertext only attack)* používá k luštění pouze šifrový text.
- *Útok se znalostí otevřeného textu (known plaintext attack)* využívá znalosti odpovídající dvojice otevřeného a šifrového textu.
- *Útok se zvoleným otevřeným textem (chosen plaintext attack)* využívá možnosti vybrat si libovolný otevřený text a dostat k němu odpovídající šifrový text. Útočník má dočasný přístup k šifrovacímu mechanismu, který je však pro něj černou skříňkou.

- *Útok se zvoleným šifrovým textem (chosen ciphertext attack)* využívá možnosti vybrat si libovolný šifrový text a dostat k němu odpovídající otevřený text. Útočník má dočasný přístup k dešifrovacímu mechanismu, který je však pro něj černou skříňkou.

1.1.2 Šifrování, dešifrování, luštění...

Je velmi důležité rozumět rozdílům mezi těmito pojmy. Každý znamená něco jiného a jejich záměnou mohou vznikat různá nedorozumění. Bude zmíněn také pojem kódování, který je často nesprávně zaměňován se šifrováním.

Kódování je přepis textu z jedné abecedy[1] do jiné podle všeobecně známého pravidla. Zpravidla se používá pro usnadnění uložení nebo zpracovávání. Kódováním je např. ASCII kód (pro ukládání v paměti počítače), Morseova abeceda (pro přenos telegrafem), nebo index písmene v abecedě.

Šifrování je transformace otevřeného textu na šifrový text. K jejímu provedení je třeba znát použitý kryptosystém (viz oddíl 1.2) a klíč.

Dešifrování je transformace šifrového textu na otevřený za znalosti použitého kryptosystému a klíče.

Luštění je pokus o transformaci šifrového textu na otevřený bez znalosti kryptosystému nebo klíče.

1.2 Kryptografický systém

Kryptografický systém (také kryptosystém) je popis kryptografického mechanismu, zahrnující omezující podmínky na otevřený text, šifrový text a klíč a dále popis algoritmů pro šifrování a dešifrování. V následujícím textu bude používána definice kryptosystému podle D. R. Stinsona [14]:

Definice 1.1 *Kryptografický systém je pětice* $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$*, která splňuje následující podmínky:*

1. $\mathcal{P}$ *je konečná množina možných otevřených textů,*
2. $\mathcal{C}$ *je konečná množina možných šifrových textů,*
3. $\mathcal{K}$ *je konečná množina možných klíčů,*
4. *Pro každé* $k \in \mathcal{K}$ *existuje šifrovací pravidlo* $e_k \in \mathcal{E}$ *a odpovídající dešifrovací pravidlo* $d_k \in \mathcal{D}$*. Každá dvojice pravidel* $e_k : \mathcal{P} \to \mathcal{C}$ *a* $d_k : \mathcal{C} \to \mathcal{P}$ *je taková funkce, že* $d_k(e_k(x)) = x$ *pro každý otevřený text* $x \in \mathcal{P}$*.*

Jako příklad uveďme kryptosystém posuvné šifry[2]. Jako šifrovací algoritmus je použit posun o předem daný počet znaků (klíč) v abecedě. Dešifrování probíhá opačným způsobem, tj. posunem v abecedě o zápornou hodnotu klíče.

[1] Abeceda je zde míněna z matematického pohledu, tzn. jako konečná neprázdná množina symbolů.
[2] Posuvná šifra byla během vývoje aplikace používána pro testovací účely, např. na testování slovníku.

Používají se pouze písmena anglické abecedy, kódovaná na celá čísla v rozsahu 0-25. Písmeno A je tak kódováno na číslo 0, písmeno B na číslo 1 atd. Pro práci s těmito čísly se používá modulární aritmetika se základem 26, označovaná $\mathbb{Z}_{26}$.

Definice 1.2 *Kryptosystém posuvné šifry je pětice* $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$*:*

1. $\mathcal{P}$ *i* $\mathcal{C}$ *obsahuje písmena anglické abecedy kódovaná jako čísla 0–25,*
2. $\mathcal{K} = \mathbb{Z}_{26}$*,*
3. $e_k(x) = (x + k) \bmod 26 \qquad k \in \mathcal{K}, x \in \mathcal{P}$*,*
4. $d_k(y) = (y - k) \bmod 26 \qquad k \in \mathcal{K}, y \in \mathcal{C}$*.*

Příklad 1.3 *Předpokládejme posuvnou šifru, klíč* $k = 5$ *a otevřený text*

```
sbohem a diky za vsechny ryby
```

Takový otevřený text bude pomocí šifrovacího pravidla e_k *zašifrován do šifrového textu*

```
XGTMJ RFINP DEFAX JHMSD WDGD
```

1.3 Kryptosystémy klasické kryptografie

Nyní již čtenář ví, co je to kryptografie i co je to kryptosystém. Zatím však ještě nebylo řečeno, co se rozumí pojmem klasická kryptografie.

Klasická kryptografie je soubor takových kryptografických systémů, které jsou určeny primárně pro ruční šifrování. Klasické šifry lze tedy šifrovat a dešifrovat jen za pomoci tužky a papíru, zpravidla je tak lze i vyluštit. Dalším výrazným znakem je, že při šifrování ani dešifrování nejsou použity binární operace.

Klíče jsou symetrické, pro dešifrování se používá stejný klíč jako pro šifrování. To je v protikladu s moderními šiframi, které používají dvojici klíčů – veřejný (sdílený) a soukromý. Moderní šifry nejsou předmětem této práce, proto se jimi nebudeme dále zabývat.

1.3.1 Základní používané principy

Klasická kryptografie využívá tři různé základní principy šifer, které mohou být následně kombinovány.

Steganografie

Asi nejstarším principem jsou *steganografické šifry*. Šifrový text vypadá na první pohled jako běžný text, např. jako novinový článek. Může jít o čtení prvních písmen slov nebo vět, použití různých neviditelných inkoustů apod.

Dnes jsou tyto systémy využívány spíše pro rekreační účely, nachází své uplatnění např. v šifrovacích hrách. Příklad takové šifry čerpá ze sedmého ročníku asi nejznámější české šifrovací hry TMOU[3], konaného roku 2006 v Brně. [4]

[3] Viz http://www.tmou.cz/

Příklad 1.4 *Následující steganografická šifra byla použita na šifrovací hře TMOU. Vyluštily ji skoro všechny soutěžní týmy, ponecháváme ji proto na samostatné vyluštění. V textu samotném je obsaženo množství nápověd, jak šifru vyluštit. Kdyby se čtenáři nedařilo šifru vyluštit, řešení nalezne v příloze A.1.*

Tato šifra není maso
Klíč k ní jistě najdete
Stačí nápad, co jak laso
Klid Vám dodá, začnete

Stále nic, nebo snad ano?
Další verš je potvora
Roh má čert, rým je tvé lano
Další sloka opora

Rohlík dej si, co se stalo?
Balvan v srdci, skřeky vran
Dech se krátí, pomačkalo
Balík mozku mnoho hran

Debil nebuď ani pako
Rdí se ten, kdo chytá vzduch
Naše šifra nosí sako
Rdousí Tě jen malý duch

Najdi slova mezi pěnou
Zezelenals? To chce klid
Oltář šifry je za stěnou
Ze slov jako epoxid

Olemuj si veršů konce
Stanov, proč je zvláštní šev
A to stačí, zazní zvonce
Stabilní je tento jev
A vyluštění zklidní krev

Substituce

Na jiném principu fungují *substituční šifry*. Způsob vyzrazuje již samotný název (z angl. *substitute* – záměna). Jedná se o záměnu částí šifry, např. znaků nebo slov, za jiné znaky. Může jít o monoalfabetické substituce (záměna znaku za znak), polyalfabetické substituce (šifruje podle více různých abeced), homofonní substituce (znak má více možností, na které může být zašifrován) nebo o polygrafické substituce (šifrování probíhá po skupinách znaků). Diskutabilně lze řadit mezi substituční šifry i použití kódových knih.

Čím delší je text substituční šifry, tím je jednodušší na rozluštění. Z tohoto pohledu jsou lépe využitelné polyalfabetické šifry, které šifrují podle klíče. Je-li klíčem dostatečně dlouhé heslo, je šifra bezpečná. To je ale zejména u dlouhých textů v ostrém rozporu se třetím Kerckhoffovým principem (viz oddíl 1.3.2), který říká, že klíč musí být snadno zapamatovatelný. Více o substitučních šifrách se může čtenář dozvědět v 2. kapitole, která je jim celá věnována.

Transpozice

Poslední často používaný princip klasické kryptografie využívají *transpoziční šifry*, jejichž označení pochází rovněž z angličtiny ze slova *transpose* – přesunout, přemístit. V šifrovém textu se vyskytují přesně stejné znaky jako v otevřeném, je však zaměněno jejich pořadí. V šifrovém textu se mohou vyskytovat i jiné znaky jako výplň.

Transpoziční šifru, známou pod jménem *Skytalé*, používali již ve starověku Řekové. Základními stavebními prvky byla tyč o přesně stanoveném průměru a kožený pásek, který byl na tuto tyč navinut. Na takový stočený pásek se napsal otevřený text, po rozmotání nečitelný. Přijímací strana měla k dispozici tyč o stejném průměru, pásek na tyč namotala a přečetla zprávu.

Transpozičních šifer existuje velké množství, poměrně bezpečná (úměrně době vzniku) je transpozice podle hesla. Je nezbytné, aby toto heslo mělo dostatečnou délku – tím pádem bude množství permutací nad takovým počtem prvků velmi vysoké a šifra bude splňovat požadavky na výpočetní bezpečnost (viz oddíl 1.3.2). Princip je následující: Šifrér napíše na jeden řádek heslo. Pod heslo vypisuje do řádků o stejné délce jako délka hesla otevřený text. Následně seřadí sloupce tak, že seřadí písmena hesla a jim příslušející sloupce podle abecedy a zapisuje sloupce za sebe. Dešifrér si po přijetí sestaví podle délky hesly a známé délky šifrového textu mřížku a vyplňuje ji po sloupcích. Po řádcích následně přečte otevřený text.

Příklad 1.5 *Předpokládejme, že chceme zašifrovat podle sedmnáctipísmenného hesla* `SBOHEMADIKYZARYBY` *otevřený text:*

```
spojte se s osobou kterou naleznete v obci olbramkostel nedaleko znojma
cislo popisne 67. bude vas ocekavat.
```

Napíšeme tedy nejprve heslo a pod něj budeme vpisovat otevřený text. Čísla zapsaná nad heslem označují řazení sloupců podle abecedy. Tečka je pro lepší čitelnost zapisována jako pomlčka.

```
13 3  11 7  6  10 1  5  8  9  14 17 2  12 15 4  16
S  B  O  H  E  M  A  D  I  K  Y  Z  A  R  Y  B  Y
-------------------------------------------------
S  P  O  J  T  E  S  E  S  O  S  O  B  O  U  K  T
E  R  O  U  N  A  L  E  Z  N  E  T  E  V  O  B  C
I  O  L  B  R  A  M  K  O  S  T  E  L  N  E  D  A
L  E  K  O  Z  N  O  J  M  A  C  I  S  L  O  P  O
P  I  S  N  E  6  7  -  B  U  D  E  V  A  S  O  C
E  K  A  V  A  T  -
```

Po seřazení zapisujeme sloupce za sebou:

```
SLMO7 -BELS VPROE IKKBD POEEK J-TNR ZEASZ OMBON SAUEA AN6TO OLKSA OVNLA
SEILP ESETC DUOEO STCAO COTEI E
```

Co udělá luštitel při získání tohoto šifrového textu za předpokladu, že ví, o jakou šifru se jedná? Může se pokusit vyzkoušet všechny permutace i různé délky klíče, je to však velmi časově náročné. Například pro klíč délky 17, použitý v příkladu 1.5 je počet možných permutací $17! \approx 3,56 \cdot 10^{14}$. To je velké číslo i pro poměrně výkonný počítač, zvláště za předpokladu, že bude muset každý možný otevřený text ověřit podle slovníku.

Praktičtější možností, využívanou i německými luštiteli během 2. světové války, je tzv. *metoda anagramů*. K jejímu využití je potřeba mít nejméně dva šifrové texty stejné délky zašifrované podle stejného hesla, neboť znaky těchto dvou šifrových textů jsou přeházeny podle stejné permutace. Luštitel při této metodě zkouší permutovat sloupce více šifrových textů najednou (podle různých heuristik) tak, aby našel ve všech smysluplný text.

Lze tedy dojít k závěru, že při jednorázovém používání klíčů nebo alespoň různé délce zpráv šifrovaných podle jednoho klíče je transpozice podle hesla výpočetně bezpečnou metodou. Z historického vývoje je však známo, že bezpečný přenos klíčů nebyl vždy možný, proto se šifrovalo mnoho zpráv dle jednoho klíče, příp. byly nové klíče vysílány rádiem. Byly sice zašifrované, ovšem podle starého klíče. Znal-li tedy nepřítel starý klíč, byl schopen zachytit i nový, čehož němečtí luštitelé zhusta využívali. [7]

Kombinované šifry

V historii docházelo ke kombinaci výše uvedených principů, příkladem mohou být například šifry TTS (dvojitá transpozice, následovaná substitucí) a STT (substituce písmen na dvojciferná čísla a jejich následná dvojitá substituce podle hesel), které využíval československý odboj během 2. světové války pro komunikaci s Ministerstvem národní obrany (MNO) a exilovou vládou v Londýně.

Více informací o historických šifrách se čtenář může dozvědět z knih historika Jiřího Janečka. [7, 8]

1.3.2 Požadavky na bezpečnost

Výpočetní bezpečnost *(computational security)* – kryptosystém je považován za výpočetně bezpečný, je-li k jeho prolomení nejlepším lušticím algoritmem potřeba alespoň n operací, kde n je velmi velké (předem stanovené) číslo. Tento druh bezpečnosti je obtížně prokazatelný, neboť nelze předem určit, který lušticí algoritmus je nejlepší. V praxi se proto vztahuje jen na některé typy útoků, např. *exhaustive key search* (vyzkoušení všech možných klíčů).

Prokazatelná bezpečnost *(provable security)* – pro prokazování je kryptosystém zjednodušen na nějaký (nejčastěji matematický) dobře známý problém. Prokázána je ovšem pouze bezpečnost vůči tomuto zjednodušení.

Nepodmíněná bezpečnost *(unconditional security)* – uvažuje se, že útočník má k dispozici neomezené množství výpočetních zdrojů.

Roku 1883 publikoval nizozemský kryptolog Auguste Kerckhoffs v časopise *Journal des sciences militaires* článek *La cryptographie militaire* [9], ve kterém zmiňuje praktické požadavky na bezpečnost kryptosystému, známé jako **Kerckhoffovy principy:**[4]

1. Kryptosystém musí být prakticky, když už ne matematicky, nerozluštitelný.
2. Bezpečnost systému nesmí být založena na utajení principu šifry. Musí být předpokládáno, že princip šifry bude nepříteli znám.

[4] Zajímavé rozšíření Kerckhoffových principů pro dnešní dobu sepsal Chad Perrin. K vidění je na blogu *techrepublic.com*:
http://www.techrepublic.com/blog/security/six-principles-of-practical-ciphers/1833

3. Sdělování a uchovávání klíčů musí být možné bez pořizování záznamu.
4. Systém musí být přizpůsoben pro telegrafickou komunikaci.
5. Systém musí být přenositelný a k jeho obsluze musí stačit jedna osoba.
6. S přihlédnutím k okolnostem, za kterých je používán, musí být systém snadno použitelný a nesmí vyžadovat velkou psychickou námahu nebo zapamatování dlouhé série pravidel.

Shrnutí

V kapitole byly vysvětleny základní pojmy a především pojem kryptografického systému. Ten byl demonstrován na příkladě posuvné šifry. Dále byly ukázány základní principy, se kterými pracuje klasická kryptografie – steganografie, substituce a transpozice – a požadavky na tyto šifry.

Kapitola 2

Substituční kryptografické systémy

Jak již bylo řečeno, substituční kryptosystémy používají pro šifrování záměnu částí otevřeného textu za části šifrového textu bez změny pořadí. V následující kapitole budou ukázány různé typy substitučních kryptosystémů, zejména pak monoalfabetická substituční šifra, jejíž luštěním se zabývá 3. kapitola.

2.1 Monoalfabetická substituce

Monoalfabetická substituční šifra (MSŠ) je zobecněním posuvné šifry. Zatímco u posuvné šifry je znak nahrazován znakem posunutým o hodnotu klíče, u monoalfabetické substituční šifry je každý znak nahrazen jiným znakem podle substituční tabulky. Precizněji popisuje MSŠ následující definice.

Definice 2.1 *Monoalfabetická substituce je takový kryptosystém, který splňuje:*

- *abecedy otevřeného a šifrového textu jsou stejně mohutné* $|\Sigma(\mathcal{P})| = |\Sigma(\mathcal{C})|$,
- *klíč je množina všech bijektivních zobrazení abecedy otevřeného textu na abecedu šifrového textu* $K = \{k : k = \Sigma(\mathcal{P}) \mapsto \Sigma(\mathcal{C})\}$.

Při vybrání jednoho klíče $k \in \mathcal{K}$ *určíme:*

- *šifrovací pravidlo* $e_k(x) = k(x)$,
- *dešifrovací pravidlo* $d_k(y) = k^{-1}(y)$.

Velmi často se používá shodná abeceda šifrového a otevřeného textu $\Sigma(\mathcal{P}) = \Sigma(\mathcal{C})$. V dalším textu budeme uvažovat pouze takový druh klíče. V takovém případě je klíčem permutace abecedy vstupního textu. Klíčem může být například:

Příklad 2.2 *Náhodně vygenerovaný klíč:*

```
OT:   a b c d e f g h i j k l m n o p q r s t u v w x y z
ŠT:   V I E U Q W G C S L O F K Z T Y X M A D P B H N R J
```

Jak již bylo řečeno v oddíle 1.3.2, klíč musí být spolehlivě uchovatelný i bez použití záznamu. Takový klíč však není dobře zapamatovatelný, proto se používá tvorba klíče podle nějakého předpisu. Tím předpisem může být heslo, následované nevyužitými písmeny abecedy. Při psaní hesla se vynechávají již použité znaky.

Příklad 2.3 *Klíč vytvořený použitím hesla* `PREZIDENTMASARYK`*:*

```
OT:   a b c d e f g h i j k l m n o p q r s t u v w x y z
ŠT:   P R E Z I D N T M A S Y K B C F G H J L O Q U V W X
```

Vidíme však, že ke konci abecedy se již jedná o pouhý posun o 2 znaky zpět. V případě kratšího hesla neobsahujícího znaky z konce abecedy (Y, Z) by se jednalo dokonce o jednoznačné přiřazení. Proto je vhodné použít i počáteční písmeno vpisování hesla.

Příklad 2.4 *Klíč vytvořený použitím hesla* `PREZIDENTMASARYK` *a posunu (začátek od* `i`*):*

```
OT:   a b c d e f g h i j k l m n o p q r s t u v w x y z
ŠT:   J L O Q U V W X P R E Z I D N T M A S Y K B C F G H
```

Takový klíč už je dostatečně silný, podobně jako náhodně vygenerovaný klíč.
Další možnosti tvorby klíče může čtenář nalézt v knize [2], str. 97.

2.1.1 Bezpečnost

Šifra je již odolná proti prolomení metodou *exhaustive key search*. Oproti posuvné šifře (viz 1.2) má řádově mnohem mohutnější množinu možných klíčů $\mathcal{K}$. Zatímco posuvná šifra měla mohutnost množiny klíčů rovnu mohutnosti abecedy otevřeného i šifrového textu $|\mathcal{K}| = |\Sigma(\mathcal{P})| = |\Sigma(\mathcal{C})|$, u monoalfabetické substituce lze každý znak nahradit libovolným jiným znakem, počet klíčů je tedy $|\mathcal{K}| = |\Sigma(\mathcal{P})|! = |\Sigma(\mathcal{C})|!$.

Příklad 2.5 *Uvažujeme-li anglickou abecedu o 26 znacích, je možný počet klíčů, který by musel být při prolamování metodou* exhaustive key search *vyzkoušen* $|\mathcal{K}| = 26! \approx 4 \cdot 10^{26}$.

Můžeme tedy již nyní říci, že monoalfabetická substituční šifra nesplňuje požadavky na nepodmíněnou bezpečnost, ale splňuje (při uvažovaní současného výkonu počítačů) podmínky výpočetní bezpečnosti s omezením na metodu *exhaustive key search*.

znak	výskyt	znak	výskyt	znak	výskyt
a	8,740 %	j	1,963 %	s	5,383 %
b	1,649 %	k	3,715 %	t	5,537 %
c	3,589 %	l	4,056 %	u	3,807 %
d	3,596 %	m	3,230 %	v	4,335 %
e	10,395 %	n	6,683 %	w	0,071 %
f	0,390 %	o	8,233 %	x	0,092 %
g	0,339 %	p	3,420 %	y	2,667 %
h	2,280 %	q	0,006 %	z	3,114 %
i	7,598 %	r	5,113 %		

Tabulka 2.1: Četnost znaků v českém jazyce

Největší bezpečnostní slabinou monoalfabetické substituce je to, že zachovává poměr četnosti znaků. V běžném (otevřeném) textu jsou jednotlivé znaky zastoupeny nerovnoměrně. Průměrnou četnost znaků v českém jazyce ukazuje tabulka 2.1 [3][1]. Toho se využívá

[1]Tabulka je přepočítána, aby neobsahovala diakritiku a písmeno ch, které je pro potřeby kryptoanalýzy monoalfabetické substituce nepotřebné.

při *znakové frekvenční analýze* šifrového textu. Kdybychom zjistili počty jednotlivých znaků v ŠT a vypočítali jejich relativní četnosti, dostali bychom podobnou tabulku, ovšem s jiným procentuálním zastoupením znaků. Potom k sobě můžeme přiřazovat znaky OT – ŠT s podobnými relativními výskyty. Také můžeme využít další charakteristiky jazyka, jakými jsou frekvenční analýza digramů, trigramů, nebo slovník nejčastějších slov.

Kapitola 3 se věnuje tématu kryptoanalýzy monoalfabetické substituce podrobněji.

2.2 Polyalfabetické substituce

Polyalfabetická substituce využívá více klíčů (více zobrazení abecedy otevřeného textu na abecedu šifrového textu $\Sigma(\mathcal{P}) \mapsto \Sigma(\mathcal{C})$). To, který klíč bude použit, závisí na pozici v textu. Oproti monoalfabetické substituci tak výrazně ztěžuje luštění, neboť nelze použít jednoduchou znakovou frekvenční analýzu.

Asi nejznámějším zástupcem polyalfabetických substitucí je *Vigenèrova šifra*. Z té jsou odvozeny některé další šifry, z nichž si představíme šifru *Autoclave*.

2.2.1 Vigenèrova šifra

Jak již bylo řečeno výše, polyalfabetické substituce používají více klíčů $\Sigma(\mathcal{P}) \mapsto \Sigma(\mathcal{C})$. Takovým klíčem je v případě Vigenèrovy šifry stejný klíč jako v případě posuvné šifry (viz definici 1.2). O kolik znaků bude posunut znak šifrového textu oproti znaku otevřeného textu závisí na jeho pozici v textu.

Klíč bývá zpravidla reprezentován jako heslo. Při šifrování i-tého znaku OT k němu přičteme i-tý znak hesla podle Vigenèrova čtverce, někdy též nazývaného *tabula recta* (viz obrázek 2.1). Jde v zásadě o posuvnou šifru, tedy o sčítání hodnot znaků modulo 26.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Obrázek 2.1: Vigenèrův čtverec neboli *tabula recta* [15]

Příklad 2.6 *Chceme zašifrovat text* `zboznujusifrovanichtelbychsifrovatporad` *podle hesla* `bicykl`*. Napíšeme si tedy otevřený text a opakovaně heslo pod sebe a budeme sčítat:*

```
zbozn ujusi frova nicht elbyc hsifr ovatp orad
bicyk lbicy klbic yklbi cyklb icykl bicyk lbic
-----------------------------------------------
AJQXX FKCUG PCPDC LSNIB GJLJD PUGPC PDCRZ ZSIF
```

Při dešifrování budeme postupovat obdobně – od znaku šifrového textu odečítáme hodnotu znaku hesla.

Při kryptoanalýze není oproti monoalfabetické substituci možné provést jednoduchou frekvenční analýzu znaků. Je však možné pomocí různých metod[2] zjistit délku hesla a poté luštit jako několik (počet odpovídá délce hesla) monoalfabetických substitucí.

2.2.2 Šifra Autoclave

Jak již název napovídá (ze španělského *clave* = klíč), jedná se o šifru, která si klíč generuje sama na základě hesla. Klíč tvoří heslo následované otevřeným textem.

Příklad 2.7 *Budeme-li chtít šifrou Autoclave zašifrovat stejný šifrový text podle stejného hesla jako v příkladě 2.6, napíšeme otevřený text a pod něj heslo, následované otevřeným textem. Následně znaky sečteme podle Vigenèrova čtverce.*

```
zbozn ujusi frova nicht elbyc hsifr ovatp orad
bicyk lzboz nujus ifrov anich telby chsif rova
-----------------------------------------------
AJQXX FIVGH SLXPS VNTVO EYJAJ AWTGP QCSBU FFVD
```

I tuto šifru je však možné rozluštit. Bezpečnou polyalfabetickou šifrou by byla jen taková, která by využívala heslo stejné délky jako šifrový text, a to pouze za předpokladu, že by pro každou zprávu byl použit jedinečný klíč. To by znamenalo mít generátor nekonečného náhodného klíče. V minulosti se využívaly např. knihy (klíčem pak bylo číslo stránky a řádku, podle kterého se začínalo šifrovat), telefonní seznamy, desetinné rozvoje iracionálních čísel.

2.3 Polygrafické substituce

Všechny již popsané substituce pracovaly s jednotlivými znaky. Polygrafické substituce šifrují po blocích znaků (nejčastěji dvojicemi nebo trojicemi; pak se jedná o digrafickou, resp. trigrafickou substituci).

Definice 2.8 *Nechť n je délka bloku. Polyalfabetická substituce je takový kryptosystém, který splňuje:*

- *množina všech možných n-tic znaků otevřeného textu je stejně mohutná jako množina všech možných prvků šifrového textu* $|\Sigma^n(\mathcal{P})| = |\Sigma(\mathcal{C})|$,

[2] Pro zjišťování délky hesla se používá ponejvíce Kasiskiho test a Friedmanův index shody. Více informací lze nalézt např. v knize R. Stinsona *Cryptography: Theory and practice* [14] nebo ve slovenské knize *Klasické šifry* [2].

- *mezi těmito dvěma množinami existuje jednoznačné přiřazení; množina klíčů obsahuje všechna tato přiřazení* $\mathcal{K} = \{k : k = \Sigma^n(\mathcal{P}) \mapsto \Sigma(\mathcal{C})\}$.

Při vybrání jednoho klíče $k \in \mathcal{K}$ *určíme:*

- *šifrovací pravidlo* $e_k(x) = k(x)$,
- *dešifrovací pravidlo* $d_k(y) = k^{-1}(y)$.

Již při použití dvojic znaků OT ($n = 2$) je mohutnost množiny prvků šifrového textu $n^2 = 26^2 = 676$. Takové číslo je příliš velké pro pamatování nebo uchovávání klíče v podobě převodní tabulky. Proto se používají předpisy a metody, jak tato zjednodušit zapamatování si těchto přiřazení.

Jako příklad polygrafické substituce bude představena šifra *Playfair*. Další velmi známou šifrou uváděnou v literatuře jako polygrafická je šifra *bifid*. Ta však nepatří striktně do kategorie polygrafických šifer, neboť kombinuje polygrafickou substituci s transpozicí. Dalším příkladem může být *Hillova šifra*, v této práci však nebude více rozebírána.

2.3.1 Šifra Playfair

Digrafická šifra, nesoucí jméno svého propagátora Lyona Playfaira, pochází z poloviny 19.století. Pro vytvoření transformačních pravidel využívá následující algoritmus.

Mějme čtvercovou tabulku, obsahující $5 \cdot 5 = 25$ písmen (standardně jsou sdružena písmena `I` a `J`, pro český jazyk lze vynechávat např. písmeno `Q` – nahrazeno `KV`, nebo `W` – nahrazeno `VV`). Tato tabulka začíná smluveným heslem, následují zbylé znaky abecedy. Označme si řádky i sloupce čísly 0-4.

Následně odebíráme z otevřeného textu dvojice znaků (A, B) a převádíme je na dvojici znaků šifrového textu (A', B'):

- Leží-li oba znaky OT ve stejném řádku v různých sloupcích $A[r, c_A],\ B[r, c_B]$, budou znaky ŠT ležet na souřadnicích $A'[r, (c_A + 1) \bmod 5],\ B'[r, (c_B + 1) \bmod 5]$.
- Leží-li oba znaky OT v různých řádcích ve stejném sloupci $A[r_A, c],\ B[r_B, c]$, budou znaky ŠT ležet na souřadnicích $A'[(r_A + 1) \bmod 5, c],\ B'[(r_B + 1) \bmod 5, c]$.
- Leží-li znaky OT v různých řádcích i v různých sloupcích $A[r_A, c_A],\ B[r_B, c_B]$, budou znaky ŠT na stejném řádku jako původní znak, ale ve sloupci druhého znaku OT $A'[r_A, c_B],\ B'[r_B, c_A]$.

Jak si pozorný čtenář mohl všimnout, za sebou nemohou ležet dva stejné znaky, neboť by je nebylo možné zašifrovat. Proto se mezi stejné znaky vkládá smluvený oddělovací znak, zpravidla `X`. Pokud má otevřený text po doplnění oddělovacích znaků lichý počet znaků, je na konec doplněn další oddělovací znak.

Dešifrování probíhá opačným způsobem.

Příklad 2.9 *Budeme šifrovat zprávu podle klíče* `ESKYMAK`. *Nejprve si sestavíme převodní tabulku. Abychom dodrželi počet 25 znaků, vynecháme písmeno Q.*

	0	*1*	*2*	*3*	*4*
0	*E*	*S*	*K*	*Y*	*M*
1	*A*	*B*	*C*	*D*	*F*
2	*G*	*H*	*I*	*J*	*L*
3	*N*	*O*	*P*	*R*	*T*
4	*U*	*V*	*W*	*X*	*Z*

Nyní zašifrujeme zprávu `chtel bych se jmenovat jan byt pritel divek a dam`. *Protože text neobsahuje žádné stejné za sebou se vyskytující znaky, nebudeme vkládat žádný oddělovací znak. Dokonce má i sudý počet znaků, oddělovací znak na konci tedy rovněž vynecháme.*

```
ch te lb yc hs ej me no va tj an by tp ri te ld iv ek ad am
BI NM HF KD OB YG ES OP UB RL GU DS NR PJ NM JF HW SY BF FE
```

Po přepsání na standardizovaný tvar ŠT dostaneme:

```
BINMH FKDOB YGESO PUBRL GUDSN RPJNM JFHWS YBFFE
```

Příklad 2.10 *Podle klíče z příkladu 2.9 je zašifrován následující šifrový text:*

```
UAUKW JMOXG SKNCN OKNPB BELPR HAUUP SYNMP JGJAY HYNVK OFNUA VHGMA KPG
```

Zkuste si jej dešifrovat, řešení naleznete v příloze A.2. Výsledkem je severské rčení, jako oddělovací znak je použito `W`.

2.4 Další známé substituce

Existují také různé další modifikace substitučních šifer, zejména monoalfabetických substitucí. Většina z nich se snaží o zvýšení bezpečnosti monoalfabetické substituce. S těmito modifikacemi byly monoalfabetické substituce využívány velmi dlouhou dobu jako téměř jediný způsob šifrování.

Homofonní substituce

Homofonní substituce přiřazuje každému znaku z abecedy otevřeného textu jeden nebo více znaků z abecedy šifrového textu. Jeden znak OT má tedy několik možných variant, jak může být zobrazen v ŠT. Toho se nejvíce využívá u často se vyskytujících znaků OT (samohlásek apod.).

Z toho vyplývá, že abeceda šifrového textu musí být větší než abeceda otevřeného textu. Využívány byly kupříkladu převodní tabulky z anglické abecedy na dvouciferná čísla 00–99.

Homofonní substituce zvyšuje náročnost luštění tím, že neumožňuje provádět frekvenční analýzu. Zároveň zvyšuje náročnost na uchovatelnost a přenositelnost klíče.

Klamače a nomenklátory

Klamače jsou znaky, které se před zašifrováním vloží do otevřeného textu. Nemají žádný praktický význam, slouží pouze pro zmatení luštitele. Nomenklátor je speciální znak, sloužící rovněž pro zmatení nepřítele, má však význam – nahrazuje nějaké (zejména často používané) slovo.

Shrnutí

Mezi substituční šifry řadíme ty šifry, které využivají pro šifrování mechanismus záměny částí otevřeného textu za části šifrového textu. V kapitole bylo ukázáno několik mechanismů, kterými může být substituce provedena, včetně příkladů šifer, které tyto principy využívají.

Kapitola 3

Luštění monoalfabetické substituce

V předchozí kapitole byl nastíněn postup, kterým se luští monoalfabetické substituce ručně. V této kapitole bude rozveden princip, na kterém je ruční způsob luštění založen a budou ukázány některé algoritmické postupy, kterými byly v minulosti tyto šifry luštěny. U jednotlivých autorů budou zmíněny části algoritmů, z nichž čerpá tato práce.

Nakonec bude ukázáno, jaký má dopad na luštění skutečnost, že jazyk otevřeného textu je *flektivní* (ohebný).

3.1 Charakteristiky jazyka

Chceme-li luštit monoalfabetickou substituci, musíme vědět (nebo alespoň hádat), v kterém jazyce byl napsán otevřený text. Každý jazyk má své charakteristiky, které se od sebe mohou značně odlišovat. Charakteristikami, které nás budou zabývat nejvíce, bude četnost jednotlivých znaků v jazyce, nejčastější bigramy a trigramy. V neposlední řadě také slovník nejčastěji používaných slov.

Charakteristiky jazyka jsou zpravidla získávány statisticky na nějaké velké množině textů. Při získávání je důležité, aby se nejednalo o odborné texty pouze z jednoho oboru, neboť zde hrozí zanesení chyby opakovaným výskytem oborových pojmů, zejména cizích slov.

Pro český jazyk jsou výskyty jednotlivých znaků zobrazeny v tabulce 2.1 na straně 13. Vidíme, že se často vyskytují samohlásky `e`, `a`, `o`, `i`, následované souhláskami `n`, `t`, `s`, `v`. Při provádění kryptoanalýzy lze předpokládat, že nejčastěji se vyskytující znaky šifrového textu budou odpovídat právě samohláskám v otevřeném textu.

Při luštění delších textů můžeme využít také nejčastější bigramy (dvojice znaků) a trigramy (trojice znaků). V českém jazyce jsou nejčastějšími dvojicemi znaků `st`, `ni`, `po`, `ov`, `ro`, nejčastějšími trojicemi `pro`, `ost`, `sta`, `pre`, `ter` [3]. Více bigramů a trigramů může čtenář nalézt v [3].

Poslední běžně používanou charakteristikou jazyka je slovník nejčastěji se vyskytujících slov. Tato slova, resp. jejich fragmenty, může luštitel (ať už člověk nebo počítač) v částečně vyluštěném textu rozeznávat a podle nich zkoušet další přiřazení znaků mezi otevřeným a šifrovým textem.

Příklad 3.1 *Uvažujme následující šifrový text:*

```
VYRER GDCQD PHQDV YRERG QHSUR KODVL WCHGY HDGYH MVRXF WBULM
HVWOL CHWRW RMHGD QRYVH FKQRR VWDWQ LCWRK RYBSO BQH
```

Z tohoto šifrového textu sestavíme tabulku výskytu znaků:

znak	*výskyt*	*výskyt*	*znak*	*výskyt*	*výskyt*	*znak*	*výskyt*	*výskyt*
A	*0*	*0,00 %*	*J*	*0*	*0,00 %*	*S*	*2*	*2,15 %*
B	*3*	*3,23 %*	*K*	*3*	*3,23 %*	*T*	*0*	*0,00 %*
C	*4*	*4,30 %*	*L*	*4*	*4,30 %*	*U*	*2*	*2,15 %*
D	*7*	*7,53 %*	*M*	*3*	*3,23 %*	*V*	*7*	*7,53 %*
E	*2*	*2,15 %*	*N*	*0*	*0,00 %*	*W*	*8*	*8,60 %*
F	*2*	*2,15 %*	*O*	*3*	*3,23 %*	*X*	*1*	*1,08 %*
G	*5*	*5,38 %*	*P*	*1*	*1,08 %*	*Y*	*6*	*6,45 %*
H	*10*	*10,75 %*	*Q*	*7*	*7,53 %*	*Z*	*0*	*0,00 %*
I	*0*	*0,00 %*	*R*	*13*	*13,98 %*			

Z frekvenční analýzy ŠT můžeme usoudit, že znaky ŠT `R` *a* `H` *budou zřejmě v OT reprezentovat nějakou samohlásku. Vyluštění celé zprávy je opět ponecháno na čtenáři, řešení nalezne v příloze A.3.*

3.2 Algoritmické luštění

Z hlediska automatizovaného luštění je monoalfabetická substituce poměrně známým problémem. Algoritmické způsoby luštění můžeme rozdělit na dvě rodiny. První rodina algoritmů využívá jako základní stavební kámen charakteristiky jazyka. Z našeho přehledu mezi ně spadá algoritmus pánů Pelega a Rosenfelda (oddíl 3.2.1). Druhá rodina řeší luštění jako slovníkový útok s nejrůznějšími optimalizacemi. Sem řadíme zbylé algoritmy z následujícího přehledu.

Všechny zmíněné algoritmy umí luštit pouze šifrový text dělený na slova. Cílem této práce je jejich modifikací vyvinout algoritmus, který bude schopen luštit i šifru, kde nebude oddělení slov zřetelné.

3.2.1 Peleg a Rosenfeld (1979) [13]

Zřejmě prvními průkopníky v algoritmickém luštění monoalfabetické substituce byli pánové Shmuel Peleg a Azriel Rosenfeld. Pro luštění využili relaxační algoritmus nad grafem. Graf je modelem šifrového textu: uzly reprezentují znaky abecedy a hrany trigramy v ŠT. Algoritmus pracuje s anglickou abecedou a znakem mezery, počet uzlů je tedy 27. Hran v grafu je stejný počet jako trigramů v šifrovém textu (vyskytuje-li se v ŠT trigram vícekrát, objeví se vícekrát i v grafu).

Dále využívá tabulky četnosti znaků pro porovnávání pravděpodobností výskytu jednotlivých znaků v textu.

Peleg-Rosenfeldův algoritmus vyžaduje poměrně dlouhý šifrový text (autoři uvádí příklady s šifrovými texty délek kolem 1000 znaků), který je dělený na slova.

3.2.2 Lucks (1990) [10]

Obsáhlý slovník je základem algoritmu Michaela Luckse. Algoritmus je nezávislý na jazyce textu za předpokladu, že slovník obsahuje všechna slova v daném jazyce. Pracuje s textem děleným na slova.

Luštění je implementováno jako prohledávání stavového stromu metodou *depth first search* (DFS, slepé prohledávání do hloubky) s omezujícími podmínkami. Ty jsou tři:

- délka slova,
- již známé dvojice OT – ŠT,
- opakující se znaky ve slově.

Slovník je databáze slov optimalizovaná na vyhledávání podle uvedených omezujících podmínek. Díky známé délce slov je možné heuristicky určit, kterými slovy začít vyhledávání, aby bylo prohledávání co nejúčinnější.

Sám autor zmiňuje, že při použití hybridního přístupu (použití frekvenční analýzy a stanovení počátečních omezení) by mohl být algoritmus účinnější. Tato práce navazuje na práci Michaela Luckse a úpravy jeho algoritmu budou popsány ve 4. kapitole.

3.2.3 Hart (1994) [5]

George W. Hart používá při luštění slovník 135 nejběžnějších anglických slov. Jedná se z velké části o zájmena, předložky, částice, spojky, modální slovesa a některá další slova. Aby si čtenář mohl udělat lepší představu, předkládáme obsah slovníku [5]:

> *THE OF AND TO A IN THAT IS WAS HE FOR IT WITH AS HIS ON BE AT BY I THIS HAD NOT ARE BUT FROM OR HAVE AN THEY WHICH ONE YOU WERE HER ALL SHE THERE WOULD THEIR WE HIM BEEN HAS WHEN WHO WILL MORE NO IF OUT SO SAID WHAT UP ITS ABOUT INTO THAN THEM CAN ONLY OTHER NEW SOME COULD TIME THESE TWO MAY THEN DO FIRST ANY MY NOW SUCH LIKE OUR OVER MAN ME EVEN MOST MADE AFTER ALSO DID MANY BEFORE MUST THROUGH BACK YEARS WHERE MUCH YOUR WAY WELL DOWN SHOULD BECAUSE EACH JUST THOSE PEOPLE MR HOW TOO LITTLE STATE GOOD VERY MAKE WORLD STILL OWN SEE MEN WORK LONG GET HERE BETWEEN BOTH LIFE BEING UNDER NEVER DAY SAME ANOTHER KNOW WHILE LAST*

Tato slova tvoří přes 50 % slov z libovolného anglického textu. Hart dále uvažuje, že moderní angličtina využívá přibližně 100 000 slov. Z toho vyplývá, že slovo, které není uvedeno ve slovníku, má pravděpodobnost $\approx 10^{-6}$, že se vyskytne v textu. Všechna slova, která nejsou ve slovníku, mohou být proto zanedbána.

Po zjednodušení je počítáno, že slovo má pravděpodobnost výskytu 10^{-2}, pokud se vyskytuje ve slovníku, a 10^{-6}, pokud se v něm nevyskytuje. Pravděpodobnost výskytu věty je počítána jako součin pravděpodobnosti výskytu jejích jednotlivých slov. Každé slovo (i pokud se opakuje) je ve větě počítáno pouze jednou.

Luštění probíhá opět na šifrovém textu děleném na slova. Algoritmus se pokouší najít takový klíč, pomocí kterého by byla většina vyluštěného textu obsažena ve slovníku. Luštění probíhá jako prohledávání stromu, kde úroveň zanoření ve stromu odpovídá pořadí slova ve větě a uzel stromu odpovídá buď příslušnému slovu ve slovníku, nebo stavu, kdy se slovo ve slovníku nevyskytuje.

Jedná se o účinný a velmi efektivní lusticí algoritmus. Bohužel je přímo závislý na charakteristikách anglického jazyka, zejména na velmi malém slovníku nejčastějších slov, který se skládá z „pomocných“ slov charakteristických pro angličtinu. Z důvodů uvedených v oddílu 3.3 by pro luštění zpráv v českém jazyce musel obsahovat mnohem větší slovník, čímž by ztratil svou efektivitu. Stejně jako již uvedené algoritmy je závislý na šifrovém textu děleném na slova.

3.2.4 Olson (2007) [12]

Algoritmus Edwina Olsona pracuje rovněž jako slovníkový útok na šifrový text dělený na slova. Zjednodušuje však prohledávání stromu na provádění množinových operací. Pro každé slovo šifrového textu vytvoří množinu možných překladů jednotlivých písmen. Nad takto vytvořenými množinami udělá průnik. Výslednou množinu možných překladů písmen následně využívá jako omezující podmínku pro vyhledávání kandidátů na překlad dalších slov šifrového textu.

Pro výběr šifrových slov k překladu zkoušel autor několik ohodnocovacích funkcí. Nakonec využil velmi jednoduchou:

$$cost(N, L) = N, \tag{3.1}$$

kde N je počet možných kandidátů a L je počet nových překladů znaků šifrového textu, které přinese výběr tohoto slova. Vybírá to šifrové slovo, jehož použiti přinese nejmenší počet kandidátů.

Vyhledávání ve slovníku je implementováno poněkud rozdílně než u Michaela Luckse. Slovník obsahuje ke každému slovu jeho vzor, v kterém jsou stejná písmena nahrazena stejným znakem. To je efektivní zejména při vyhledávání duplicit ve slově, vzor lze totiž použít jako index pro vyhledávání.

Příklad 3.2 *Vzor pro slovo* `anglictina` *by byl* `ABCDEFGEBA`*, pro slovo* `terminologie` *potom* `ABCDEFGHGIEB`*.*

Olsonův algoritmus vrací nejen úplná řešení, nýbrž i částečná. Všechna řešení jsou ohodnocena a uživateli jsou zobrazeny pouze nejlepší výsledky. Jedná se o velmi účinný algoritmus využívající několik různých optimalizací. Tato práce využívá ohodnocovací funkce (3.1) pro výběr nejlepší délky slova a vychází z Olsonova rekurzivního lušticího algoritmu.

3.3 Specifika českého jazyka

Výše uvedené algoritmy (kromě Luckse a Olsona) počítají s angličtinou jako s použitým jazykem. Moderní angličtina je z hlediska počítačového zpracování velice jednoduchá. Český jazyk má oproti anglickému svá specifika, která ztěžují kryptoanalýzu. Z hlediska automatizované kryptoanalýzy jsou zřejmě největším problémem češtiny (a ostatně všech slovanských jazyků) ohebné slovní druhy, tedy slovní druhy, které lze skloňovat, nebo časovat[1]. Jazyk, který využívá ohebné slovní druhy, je označován jako *flektivní (flexe* = ohýbání) [16].

Germánské jazyky, mezi které patří i angličtina, se vyznačují minimální flexí. Místo ohýbání slov skloňují pomocí koncovek. Angličtina je oproti jiným germánským jazyků, např. němčině, ještě zjednodušena – využívá pouze 2 slovní druhy. Význam ve větě určuje umístění slova ve větě (angličtina používá pevný slovosled) a předložka, s kterou je slovo použito. Časování sloves závisí na tom, zda se jedná o slabé (pravidelné) nebo silné (nepravidelné) sloveso. Tvar slovesa se liší pouze v minulém a předminulém čase, v ostatních časech a v infinitivu je neměnný.

Z uvedeného vyplývá, že anglický jazyk používá velmi málo tvarů slov. Pro automatizované luštění pomocí slovníku si vystačíme s poměrně málo obsáhlým slovníkem. Zbylé tvary není složité odvodit.

[1] Mezi ohebné slovní druhy v českém jazyce patří podstatná jména, přídavná jména, zájmena, číslovky a slovesa.

Budeme-li chtít vytvořit slovník pro češtinu, musíme do slovníku zahrnout všechny, nebo alespoň ty nejpoužívanější, tvary slov. „Naučit“ program skloňovat a časovat by jednak bylo značně obtížné, jednak by bylo skloňování a časování během vyhledávání slova ve slovníku velmi neefektivní.

Shrnutí

V kapitole byly popsány charakteristiky jazyka, které lze využít pro ruční i algoritmické luštění – četnost znaků, bigramů a trigramů. Tyto charakteristiky byly demonstrovány na českém jazyce.

Poté byly představeny dvě rodiny lušticích algoritmů: první využívá charakteristiky jazyka, druhá slovníkový útok. Pozorný čtenář si mohl všimnout závislosti použitého principu luštění na době vzniku. Je logické, že v dřívějších dobách, kdy se běžná kapacita úložných zařízení pohybovala v řádu kilobajtů až megabajtů, nebyl dostatek úložného prostoru pro dostatečně obsáhlý slovník. Postupem času se zvýšila kapacita úložných zařízení, což umožnilo uložení obsáhlejšího slovníku a provedení slovníkového útoku.

V posledním oddílu byly ukázány ty rozdíly mezi anglickým a českým jazykem, které mají zásadní vliv na slovníkový útok. Zejména se jedná o skutečnost, že český jazyk je flektivní. To zásadně zvyšuje požadavky na velikost slovníku nebo na lingvistické schopnosti programu.

Kapitola 4

Návrh lušticího algoritmu

Algoritmické luštění monoalfabetické substituce představuje zajímavý optimalizační problém z oblasti prohledávání stavového prostoru. Přesněji a formálněji problém formulujeme v oddílu 4.1. Následně bude v oddílu 4.2 představeno několik úvah, jak problém algoritmicky řešit. Další oddíly tyto úvahy rozvíjí a hledají řešení dílčích problémů.

4.1 Formulace problému

Označme jako stavový prostor dvojici (S, O), kde S (z angl. *states)* je množina všech stavů a O (z angl. *operations)* je množina všech operací, kterými lze stav měnit. Dále označme stav $s_0 \in S$ jako počáteční stav úlohy a $G \subset S,\ G \neq \emptyset$ (z angl. *goales)* jako množinu všech cílových stavů úlohy [17].

Stavový prostor hledání řešení monoalfabetické substituce má podobu grafu. Stav $s_i \in S$ je dvojice $s_i = (k_i, c_i)$, kde $k_i \subset \Sigma(\mathcal{C}) \mapsto \Sigma(\mathcal{P})$ je část klíče a $c_i \in \Sigma(\mathcal{C})^*$ je zbývající šifrový text. Zbývající šifrový text se skládá ze znaků $c_i = \{c_{i,1}, \ldots, c_{i,n}\}$, kde n je délka zbývajícího šifrového textu. Operací $o \in O$ se rozumí změna klíče, která je ve směru od kořene vždy aditivní, tzn. mohutnost klíče se s rostoucí hloubkou zanoření zvětšuje.

Počáteční stav úlohy $s_0 = (k_0, c_0)$ je takový, že klíč $k_0 = \emptyset$ neobsahuje žádná mapování a zbývající šifrový text $c_0 = c$ obsahuje celý šifrový text.

Cílovými stavy úlohy $G \subset S$ jsou ty stavy $g \in G$, ve kterých je klíč úplný, tzn. obsahuje všechna mapování z abecedy šifrového textu do abecedy otevřeného textu $k_g = \Sigma(\mathcal{C}) \mapsto \Sigma(\mathcal{P})$, a zároveň je otevřený text p_g vzniklý použitím dešifrovacího pravidla s tímto klíčem $p_g = d_{k_g}(c)$ platným řetězcem v jazyce otevřeného textu $p_g \in L(\mathcal{P})$.

Provedení operace

Přechod ze stavu s_i do stavu s_{i+1} může být proveden, existuje-li slovo $w = \{w_1, \ldots, w_{l_w}\}$ délky l_w, které může být překladem části šifrového textu $\{c_{i,1}, \ldots, c_{i,l_w}\}$ v tom smyslu, že překlad odpovídá mapováním klíče k_i.

Provedením operace se rozumí přechod ze stavu $s_i = (k_i, c_i)$ do stavu $s_{i+1} = (k_{i+1}, c_{i+1})$. Při přechodu je vytvořen nový klíč k_{i+1}, který vznikne sjednocením klíče a mapováními mezi slovem w a částí šifrového textu $\{c_{i,1}, \ldots, c_{i,l_w}\}$:

$$k_{i+1} = k_i \cup \bigcup_{j=1}^{l_w} w_j \mapsto c_{i,j}$$

Ve stavu s_{i+1} je použita pouze část šifrového textu:

$$c_{i+1} = \{c_{i,l_w+1}, \ldots, c_{i,n}\}$$

4.2 Obecné úvahy

Nejspolehlivější a nejpřesnější metodou vyřešení problému formulovaného v předchozím oddílu by bylo úplné prohledání stavového prostoru klíčů. S přihlédnutím k velikosti stavového prostoru (při uvažování anglické abecedy o 26 znacích obsahuje stavový prostor $26! \approx 4 \cdot 10^{26}$ stavů) by šlo o výpočetně značně náročnou a neefektivní metodu.

Frekvenční charakteristiky lze pro hrubý odhad použít, ale jak zmiňuje G. W. Hart [5], *„luštění pouze s pomocí četnosti znaků nemusí být úspěšné ani při textu o 10 000 znacích.“* Je to způsobeno tím, že ani takto dlouhý text nemusí svými charakteristikami plně odpovídat statisticky zjištěným charakteristikám jazyka.

Jako vhodná volba se jeví použití slovníkového útoku. Na principu útoku se slovníkem pracují algoritmy popsané v oddílech 3.2.2, 3.2.3 a 3.2.4. Všechny však vyžadují text dělený na slova. Vzhledem k tomu, že cílem práce bylo navrhnout a implementovat algoritmus, který by byl schopen luštit i monoalfabetické substituce bez zřetelného oddělení slov, jevilo se jako vhodné vybrat z těchto metod nejlépe použitelné a na jejich základě provést návrh nového algoritmu.

Rovnou můžeme zavrhnout přístup G. W. Harta [5]. Jeho vyhledávání ve velmi malém slovníku je natolik charakteristické pro angličtinu, že není možné jej pro český jazyk použít. Jak již bylo řečeno, čeština je jazyk flektivní a jako taková nepotřebuje pomocná slova, která tvoří většinu Hartova slovníku.

Organizace slovníku Michaela Luckse [10] a vyhledávání v něm je dostatečně rychlé i pro luštění šifrového textu bez mezer. Návrh slovníku bude více popsán v oddílu 4.4.

Algoritmus Edwina Olsona [12] zase obsahuje zajímavou optimalizaci výběru slova šifrového textu pro překlad. Tu sice nelze při šifrovém textu bez mezer přímo použít, lze však aplikovat stejný princip na výběr délky slova. Toto téma bude rozvedeno v oddílu 4.5.

4.3 Luštění

Základní algoritmus vychází z algoritmu Edwina Olsona [12], je však upraven pro vyhledávání optimální délky slova. Řídicí částí je rekurzivně volaná procedura `proceedCrack`, kterou ukazuje algoritmus 4.1. Ten si nyní vysvětlíme.

Na řádcích 2–5 je řetězec šifrového textu prázdný, nemá proto smysl vyhledávat ve slovníku. Současný klíč je zaznamenán jako úplné řešení a algoritmus se navrací do předchozího stupně zanoření.

Na řádku 7 vidíme sestavení seřazeného seznamu délek slov. V oddílu 4.5 bude vysvětleno, jakým způsobem je tento seznam sestavován.

Dále jsou na řádku 9 pro každou délku (ve stanoveném pořadí) vyhledána všechna kandidátní slova pro překlad části šifrového textu délky *length* (to budeme dále označovat jako šifrové slovo). To zajišťuje funkce searchCandidates. Její náplní je provést vyhledávání v databázi podle omezení stanovených klíčem a duplicitami v šifrovém textu.

Každé z těchto slov je na řádku 12 pomocí funkce isWordConsistent ověřeno, zda splňuje všechny předpoklady pro to, aby mohlo být překladem šifrového slova. Jedná se o to, aby klíč splňoval požadavky bijektivního zobrazení.

```
1:  procedure proceedCrack(ciphertext, key, rank)
2:      if ciphertext.length() = 0 then
3:          ReportFullSolution(rank, key)
4:          return
5:      end if
6:      hasChild ← false
7:      lengthList ← planLengthList(ciphertext, key)
8:      for ∀length ∈ lengthList do
9:          candidates ← searchCandidates(ciphertext, length, key)
10:         for ∀word ∈ candidates do
11:             if isWordConsistent(key, word, ciphertext) then
12:                 newKey ← addMappings(key, word, ciphertext)
13:                 newRank ← rank + length · word.rank
14:                 proceedCrack(ciphertext.from(length), newKey, newRank)
15:                 hasChild ← true
16:             end if
17:         end for
18:     end for
19:     if hasChild then
20:         ReportPartialSolution(rank, key)
21:     end if
22: end procedure
```

Algoritmus 4.1: Lušticí algoritmus

Může-li být slovo překladem šifrového slova, je vytvořen nový klíč obsahující mapování vzešlá z tohoto slova a vyhledávání je na řádku 14 zanořeno do další úrovně. Na řádku 13 je vypočtena nová hodnota ohodnocení slova, *rank*. Oddíl 4.6 popisuje, jakým způsobem je počítáno ohodnocení řešení.

Na řádcích 19–21 můžeme vidět, že nebylo-li nalezeno úplné řešení, je oznámeno i částečné řešení. Je to z toho důvodu, aby mohl uživatel vidět správné řešení i za předpokladu, že nejsou všechna slova obsažena ve slovníku.

Všimněme si, že algoritmus neukončí prohledávání po nalezení prvního úplného řešení, neboť to nemusí být nejlepším možným řešením. Místo toho se pokouší najít všechna řešení a následně z nich vybrat to nejlepší. Je však zřejmé, že tento přístup značně prodlouží dobu výpočtu. Je vhodné, aby plná i částečná řešení byla průběžně zobrazována uživateli, kterému by se tak dostala možnost algoritmus ve vhodný čas zastavit.

4.3.1 Složitost algoritmu

Lušticí algoritmus vychází z grafového algoritmu DFS, v každém zanoření však provádí prohledávání různých délek slov.

V nejhorším možném případě může algoritmus prohledávat pro každý znak šifrového textu všechny délky slov a dosáhnout tak faktoriálové časové složitosti $O(L!)$, kde L je délka šifrového textu. Tyto případy se však algoritmus snaží eliminovat tak, aby dosáhl polynomiální složitosti $O(L^N)$, kde N by bylo nejnižší možné.

Prostorová složitost je lineární $O(L)$, závislá na délce slova, neboť algoritmus DFS prohledává vždy jen jednu větev.

4.4 Slovník

Pro úspěšné luštění šifry v rozumném čase je klíčové, aby vyhledávání slov ve slovníku bylo co nejrychlejší. Zvažovali jsme, zda použít slovník Lucksův [10] (viz 3.2.2), nebo Olsonův [12] (viz 3.2.4), nebo zda vyvinout zcela nový přístup. Lucksův algoritmus se jeví jako velmi rychlý pro vyhledávání podle již známých částí klíče. Olsonův algoritmus oproti tomu vyhledává podle vzorů slov (zde zohledňuje duplicity) a slova nekonzistentní s klíčem k_i ignoruje až při zpracování stavu s_i, nikoli při výběru z databáze. Protože bylo na výběr ze dvou funkčních variant slovníku, byla zavržena možnost vyvíjet slovník vlastní.

Vzhledem k tomu, že v době návrhu nebyla ještě dostatečně doceněna důležitost rychlosti vyhledávání duplicit, byl zvolen Lucksův model slovníku, jelikož je jednodušší i dostatečně efektivní. Při vyhledávání duplicit není třeba z šifrového slova sestavovat jeho vzor, jako by to bylo nutné při použití Olsonova modelu.

4.5 Výběr délky slova

V každém stavu $s_i \in S$ musí algoritmus rozhodnout, pro kolik znaků ze zbývajícího šifrového textu c_i bude hledat ve slovníku substituci. Cílem je, aby zvolený počet znaků přinesl co největší užitek s co nejmenším množstvím procházených stavů. Tento užitek lze do jisté míry předpovídat a předem ohodnotit. To lze provést několika způsoby.

Na algoritmizaci nejjednodušší, ale zároveň nejméně efektivní, je *statický výběr* pořadí vyhledávání délky slova. V každém stavu bude tedy pořadí vyhledávání slov podle délky stejné. Způsob, jakým je statický výběr proveden, může být různý. Ohodnocovací funkci pro sestavení pořadí délek slov označíme jako $cost(N, L)$, kde N je počet slov délky L ve slovníku. Čím nižší hodnotu (cenu) má pořadí, tím je pro vyhledávání lepší. Implementací této funkce může být:

- Počet slov podle délky $cost(N, L) = N$. Při použití této metriky jsou ovšem upřednostňována krátká slova (např. jednopísmenná), kterých je ve slovníku málo, což může vést k nesmyslným výsledkům luštění.
- Počet slov podle délky dělený délkou slova $cost(N, L) = \frac{N}{L}$. I tato metrika upřednostňuje kratší slova. Je však použitelná, omezíme-li délku slov na 3 znaky a více.
- Podle délky slov sestupně $cost(N, L) = \frac{1}{L}$. Výběr této metriky však může způsobit neúměrné prodloužení doby běhu programu, neboť jsou vybírána dlouhá slova, kterých je mnoho.

Protipólem statického výběru pořadí je *dynamický výběr*. Dynamický výběr je prováděn v každém stavu $s_i \in S$ a závisí jak na zbývajícím šifrovém textu c_i, tak na aktuálním obsahu klíče k_i. Zaměřuje se na to, aby délka slova, která bude vybrána, přidala do klíče k_i co nejvíce nových mapování z abecedy otevřeného textu do abecedy šifrového textu. V každém stavu je tedy nutné vypočítat hodnotu ohodnocovací funkce $cost(N, L)$, kde N je počet slov délky L s omezeními vyplývajícími ze stavu s_i. Tím je myšleno, že kandidátní slovo pro překlad musí odpovídat klíči k_i a vzorem slova zbývajícímu šifrovému textu c_i.

I pro dynamický výběr existuje množství těchto metrik. Edwin Olson je popisuje pro výběr nejlepšího slova z šifrového textu, který je dělený na slova [12]. V podstatě v nezměněné podobě je však lze použít i pro výběr nejlepší délky slova.

- Efektivní faktor větvení *(effective branching factor – EBF):* $cost(N, L) = N^{1/L}$. EBF poskytuje dobré výsledky, je však výpočetně náročný.
- Lineární metrika $cost(N, L) = \frac{N}{L}$. Tato metrika velmi upřednostňuje délky slov s menším počtem kandidátních slov. Je výpočetně jednodušší než EBF.
- Zjednodušená lineární metrika $cost(N, L) = N$ má podobné vlastnosti jako lineární metrika, je však ještě jednodušší na výpočet.

V rámci práce byly vyzkoušeny dvě metriky: lineární a zjednodušená lineární. Obě poskytují přibližně stejné výsledky, proto byla nakonec použita zjednodušená lineární metrika, neboť je výpočetně méně náročná. Poznamenejme zde, že i pro počáteční stav dosahuje dynamická metrika lepších výsledků než statická, neboť zohledňuje i duplicitní znaky v šifrovém textu.

4.6 Ohodnocení výsledků

Jak již bylo řečeno, algoritmus nekončí po nalezení prvního použitelného řešení, nýbrž hledá další, i neúplná řešení. Aby bylo možné výsledky seřadit podle relevance, provádí algoritmus hodnocení *(ranking)* každého výsledku neboli přiřazuje každému řešení ohodnocení *(rank).*

Při každém zanoření je k současné hodnotě ranku přičtena četnost $freq_w$ vybraného slova w násobená jeho délkou l_w:

$$rank_{i+1} = rank_i + freq_w \cdot l_w$$

Bohužel toto řešení velmi zvýhodňuje krátká slova, neboť jejich výskyt v jazyce je mnohonásobně vyšší než výskyt slov delších. Nalezení vhodnějšího řešení je jedním z témat, které by mohlo tuto práci rozvíjet.

Shrnutí

Návrh programu vychází částečně z návrh dvou různých autorů, Luckse [10] a Olson [12], jejichž metody řešení monoalfabetické substituce však počítají s šifrovým textem děleným na slova.

Kombinací jejich řešení dílčích problémů vznikl nový algoritmus, který je schopen řešit i šifry bez zjevných oddělovačů slov. Základem tohoto algoritmu je prohledávání stavového prostoru metodou depth-first search s upřednostněním výpočetně nejméně náročných cest. Algoritmus vyhledává více řešení, jež následně hodnotí a vybírá z nich jen ty nejlepší.

Kapitola 5

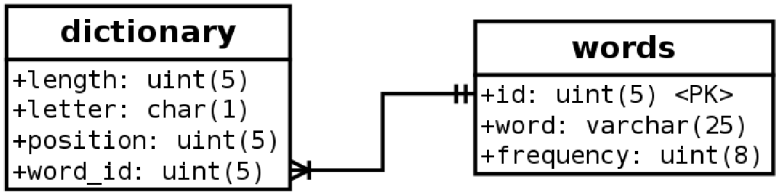
Implementace

Jako implementační jazyk byl zvolen univerzální objektový jazyk C++. Ačkoli úloha samotná je procedurální a stačil by pro ni jednodušší jazyk C, jazyk C++ umožňuje využívat ze standardní knihovny běžné abstraktní datové typy jako vázaný seznam, hashovací tabulku nebo textový řetězec.

Dále byla použita knihovna Qt ve verzi 4.7, která usnadňuje přenositelnost programu mezi různými operačními systémy. Také poskytuje jednoduché nástroje pro tvorbu uživatelského rozhraní.

5.1 Slovník

Vzhledem ke své databázové podstatě je slovník implementován jako relační databáze nad DBMS[1] MySQL. Databáze obsahuje dvě tabulky: *words* a *dictionary*; relační schéma je zobrazeno na obrázku 5.1. Na schématu je vidět, že tabulka *dictionary* se zaměřuje na vyhledávání podle omezujících podmínek. Jak již bylo řečeno dříve, podmínky jsou tři: písmeno, jeho pozice ve slově a délka slova. Ačkoli je zde délka slova redundantním údajem (logicky by patřila do tabulky *words*), její umístění ve vyhledávací tabulce značně zrychluje vyhledávání slov. Tabulka *words* obsahuje samotné slovo a četnost jeho výskytů v jazyce. Četnost je využita pro řazení výsledků při vyhledávání, ale také jako klíčový atribut pro hodnocení výsledků.



Obrázek 5.1: Tabulkové schéma relační databáze slovníku

Pro rychlejší vyhledávání obsahuje tabulka *dictionary* také indexy. Ty jsou celkem čtyři: první tři jsou na každém ze sloupců *length*, *letter* a *position*, čtvrtý je na těchto třech sloupcích dohromady.

V každém stavu stavového prostoru je sestaven dotaz na databázi, který vyhledá všechny

[1] DBMS = *database management system*. V českém jazyce označován jako *systém řízení báze dat* (SŘBD).

vhodné varianty pro další postup. Takový dotaz může být poměrně složitý, zvláště ve dvou případech:

1. jedná-li se o stav, ve kterém je známá již větší část klíče,
2. obsahuje-li hledané slovo šifrového textu mnoho duplicitních znaků.

Databáze byla pro účely testování vytvořena ve dvou variantách. První, základní, obsahuje přibližně 27 000 slov, druhá, zmenšená, potom asi 3 000 slov. Uživatel může v konfiguračním souboru zvolit, zda bude pro vyhledávání použita základní, nebo zmenšená verze.

5.1.1 Obsah slovníku

Jako základ obsahu slovníku byl použit slovník vytvořený Výzkumnou skupinou zpracování přirozeného jazyka [1], který obsahoval velké množství slov získané z automatizovaného zpracování mnoha webových stránek. Z tohoto slovníku byly nejprve odstraněny řetězce očividně nepatřící do českého jazyka jako HTML tagy, řídicí nezobrazitelné znaky apod. Dále byl slovník redukován o řetězce obsahující číslice, neboť takový řetězec je pro substituční šifru bezcenný (nejsme schopni stanovit její zobrazení z ŠT na OT). Nakonec byla ze slovníku odstraněna slova s četností vyšší než 10 000 výskytů. Aby si čtenář mohl udělat představu, uveďme, že nejčastěji se vyskytující slovo – *v* – má 12 231 739 výskytů. Po této úpravě zůstalo ve slovníku cca 30 000 slov.

Dalším krokem bylo odstranění diakritiky ze slovníku a duplicitních položek, které po odstranění diakritiky zůstaly. Po provedení těchto operací zůstalo ve slovníku přibližně 27 000 slov.

Posledním krokem bylo vytvoření vyhledávací tabulky. Zde se již jednalo o operaci, kterou nešlo provést pouze pomocí jazyka SQL. Pro napsání pomocného převodního skriptu byl proto použit jazyk PHP a databázová vrstva *dibi* od Davida Grudla a jeho skupiny Nette Foundation [11].

Pro účely testování byla následně vytvořena ještě zredukovaná verze slovníku, obsahující pouze slova s četností větší než 20 000 výskytů. Tím byla velikost slovníku zmenšena přibližně na 3 000 hesel.

5.2 Uživatelské rozhraní

Program je psán s použitím knihovny Qt. Ta umožňuje snadné psaní programů s GUI *(graphical user interface* = grafické uživatelské rozhraní). Rozhraní je použito velmi jednoduché a, zpětně hodnoceno, program by se bez něj obešel, neboť se jedná o program na zpracování textu.

Rozhraní umožňuje načítat a ukládat zdrojový text. Ten může mít charakter jak otevřeného, tak i šifrového textu, v závislosti na přepínači nad polem zdrojového textu. Poznamenejme zde, že při vyplňování pole zdrojového textu je text automaticky upravován, aby odpovídal konvencím používaným v této práci. Vždy je odstraněna diakritika a otevřený text je převeden na malá písmena, šifrový text na velká.

Otevřený text je možné šifrovat podle zadaného klíče, šifrový dešifrovat nebo luštit. Poslední funkce je pomocná a slouží pro ruční luštění – frekvenční analýza. Tu je možné spustit na šifrovém textu pomocí položky v menu.

Kapitola 6

Testování

Aby bylo možné posoudit, jak program pracuje, byla na něm prováděna testování. Pro testování byl používán identický klíč, tzn. že každý znak otevřeného textu odpovídá stejnému znaku šifrového textu:

$$k = \{x \mapsto x \ : \ \forall x \in \Sigma(\mathcal{P})\}$$

Pro praktické šifrování to je klíč naprosto nepoužitelný, pro počítač jde však o klíč jako každý jiný.

Pro každý řetězec byla počítána doba výpočtu, měřeno opakovaně a doba výpočtu zprůměrována. Pro testování byl použit redukovaný slovník. Výsledky nejsou zcela přesné, neboť při krátkých slovech se mohou výsledky dotazů vyskytovat ve vyrovnávací paměti *(cache)* databáze. Vypnutí *cachování* by však způsobilo značné prodloužení doby běhu programu, stejný dotaz může být prováděn opakovaně v různých kontextech.

Slovo	**Výsledky**	**Čas (s)**
SVOBODA	alejeho vsakale alenebo alemezi protomu	0,236
PROC BYCHOM	byloproale projenjsou projakjsou diloproale mirealepro	15,570
PROC BYCHOM SE NETESILI	sveabyaleprotohorici	2 977,930

Tabulka 6.1: Přehled výsledků testovaných řetězců

Tabulka 6.1 ukazuje výsledky algoritmu pro řetězce ŠT SVOBODA A PROCBYCHOM při použití výše zmíněného klíče. Všimněme si, že do čela výsledků se tlačí slova složená z krátkých, ale často používaných slov. To je způsobeno nevhodně zvolenou ohodnocovací funkcí. Jako námět na další zlepšení se jeví použití jiným způsobem rankingu výsledků, např. tím, který použil ve své práci Olson [12]. Ten používá řazení na základě četnosti trigramů v jazyce.

Závěr

V práci byl popsán možný způsob algoritmického luštění monoalfabetické substituce vycházející z dříve uveřejněných algoritmů. Oproti těmto algoritmům přináší možnost luštit i šifry bez oddělených slov. Daní za to však je faktoriálová složitost, která se projevuje zejména u delších šifrových textů.

Další vývoj by se mohl vyvíjet směrem k lepšímu hodnocení výsledků. Z toho vyplývá optimalizace na vyhledávání s tímto hodnocením a ukončení algoritmu po nalezení zvoleného počtu výsledků. To nyní není možné, neboť nelze předem přesně odhadnout, který z výsledků dosáhne nejvyššího ohodnocení.

Dále by bylo vhodné změnit organizaci databáze rovněž na Olsonův model, neboť tu lze uložit do vyhledávací (hashovací) tabulky v paměti počítače a není nutné dělat dotazy na databázi, které jsou, zejména při použití velkého slovníku, nejpomalejší částí aplikace.

Literatura

[1] Výzkumná skupina zpracování přirozeného jazyka na FIT VUT.
URL http://www.fit.vutbr.cz/research/groups/nlp/

[2] GROŠEK, M., O.; VEJVODA; ZAJAC, P.: *Klasické šifry*. Slovenská technická univerzita v Bratislave, první vydání, 2007, ISBN 978-80-227-2653-5.

[3] HANČAR, P.: Frekvence písmen, bigramů, trigramů, délka slov. [online], Naposledy editováno 23. 3. 2008.
URL
http://nlp.fi.muni.cz/cs/Frekvence_pismen_bigramu_trigramu_delka_slov

[4] HANŽL, T.; PELÁNEK, R.; VÝBORNÝ, O.: *Šifry a hry s nimi*. Portál, první vydání, 2007, ISBN 978-80-7367-196-9, 198 s.

[5] HART, G. W.: To decode short cryptograms. *Communications of the ACM*, ročník 37, September 1994: s. 102–108, ISSN 0001-0782.
URL http://doi.acm.org/10.1145/182987.184078

[6] HÖNIGOVÁ, A.; MATYÁŠ, V. m.: *Anglicko-česká terminologie bezpečnosti informačních technologií*. Praha: Computer Press, 1996, ISBN 80-85896-44-3.

[7] JANEČEK, J.: *Odhalená tajemství šifrovacích klíčů minulosti*. Naše vojsko, první vydání, 1994, ISBN 80-206-0462-6, 183 s.

[8] JANEČEK, J.: *Gentlemani (ne)čtou cizí dopisy*. Books, první vydání, 1998, ISBN 80-85914-90-5, 175 s.

[9] KERCKHOFFS, A.: La cryptographie militaire. *Journal des sciences militaires*, ročník IX, č. Jan 1883, 1883: s. 5–38.
URL http://www.petitcolas.net/fabien/kerckhoffs/crypto_militaire_1.pdf

[10] LUCKS, M.: A constraint satisfaction algorithm for the automated decryption of simple substitution ciphers. In *Proceedings on Advances in cryptology*, CRYPTO '88, New York, NY, USA: Springer-Verlag New York, Inc., 1990, ISBN 0-387-97196-3, s. 132–144.
URL http://dl.acm.org/citation.cfm?id=88314.88360

[11] Nette Foundation: *dibi: tiny 'n' smart database layer*. [Online; cit. 2012-05-05].
URL http://dibiphp.com/cs/dokumentace

[12] OLSON, E.: Robust Dictionary Attack of Short Simple Substitution Ciphers. *Cryptologia*, ročník 31, č. 4, Říjen 2007: s. 332–342, ISSN 0161-1194.
URL http://dx.doi.org/10.1080/01611190701272369

[13] PELEG, S.; ROSENFELD, A.: Breaking substitution ciphers using a relaxation algorithm. *Communications of the ACM*, ročník 22, November 1979: s. 598–605, ISSN 0001-0782.
URL http://doi.acm.org/10.1145/359168.359174

[14] STINSON, D.: *Cryptography: Theory and Practice*. CRC/C&H, druhé vydání, 2002, ISBN 1584882069.

[15] Wikipedia: Tabula recta — Wikipedia, The Free Encyclopedia. 2012, [Online; cit. 2012-05-03].
URL
http://en.wikipedia.org/w/index.php?title=Tabula_recta&oldid=484545578

[16] Wikipedie: Flektivní jazyk — Wikipedie: Otevřená encyklopedie. 2012, [Online; cit. 2012-04-29].
URL http:
//cs.wikipedia.org/w/index.php?title=Flektivn%C3%AD_jazyk&oldid=8425102

[17] ZBOŘIL, F. V.; ZBOŘIL, F.: Základy umělé inteligence: Studijní opora, 2007.

Příloha A

Řešení šifer

A.1 Steganografická šifra (příklad 1.4)

Nápovědu k řešení dává samotná šifra svými verši „najdi slova mezi pěnou“ a „olemuj si veršů konce“. Při dalším zkoumání zjistíme, že vždy druhý a čtvrtý rým ve sloce mají společných několik prvních písmen, stejně tak třetí rým a první rým následující sloky. Přidržíme-li se nyní nápovědy „olemuj si veršů konce“, zjistíme, že podobnou pravidelnost mají i první se třetím rýmem a druhý se čtvrtým rýmem v každé sloce. Spojením těchto společných písmen dostaneme z každého rýmu dvě slova.

Tato šifra není ma**so**
Klíč k ní jistě najd**ete**
Stačí nápad, co jak la**so**
Klid Vám dodá, začn**ete**

Stále nic, nebo snad **ano**?
Další verš je potvo**ra**
Roh má čert, rým je tvé l**ano**
Další sloka opo**ra**

Rohlík dej si, co se st**alo**?
Balvan v srdci, skřeky v**ran**
Dech se krátí, pomačk**alo**
Balík mozku mnoho h**ran**

Debil nebuď ani p**ako**
Rdí se ten, kdo chytá vz**duch**
Naše šifra nosí s**ako**
Rdousí Tě jen malý **duch**

Najdi slova mezi pě**nou**
Zezelenals? To chce kl**id**
Oltář šifry je za stě**nou**
Ze slov jako epox**id**

Olemuj si veršů kon**ce**
Stanov, proč je zvláštní š**ev**
A to stačí, zazní zvon**ce**
Stabilní je tento j**ev**
A vyluštění zklidní krev

Tímto způsobem dostaneme slova: sokl, test, anoda, raroh, alobal, rande, akord, duchna, nouze, idol, cesta, Eva. Po přečtení prvních písmen dostáváme výsledek `STARARADNICE`.

Je zřejmé, že tato šifra není k praktickému šifrování použitelná. V šifrovacích hrách však není cílem, aby šifru nikdo nevyluštil. Spíše se snaží o originalitu a nápadnost.

A.2 Playfair (příklad 2.10)

Řešením je severské rčení: *„Neexistuje špatné počasí, to jen někteří lidé jsou špatné oblečení.“*

A.3 Monoalfabetická substituce (příklad 3.1)

Řešením je citát z románu *1984* George Orwella: *„Svoboda znamená svobodně prohlásit, že dvě a dvě jsou čtyři. Jestliže toto je dáno, všechno ostatní z toho vyplyne.“*

Příloha B

Obsah CD

Na CD se nachází v kořenovém adresáři několik podadresářů a souborů, které si nyní společně projdeme.

```
bin/
doc/
INSTALL
Makefile
README
sql/
 +- subcipher.sql
 +- subcipher_small.sql
src/
 +- Makefile
 +- SubCipher/
 |   +- <source files of SubCipher>
 +- SubCipher-setup/
     +- <source files of SubCipher-setup>
tex/
 +- Makefile
 +- <LaTeX source files>
```

Výpis B.1: Obsah CD

Kořenový adresář obsahuje soubory `INSTALL` a `README`, které popisují postup instalace, resp. uvádí základní informace o programu. Dále se zde nachází konfigurační soubor automatizovaného překladu, `Makefile`.

Adresáře `bin/` a `doc/` jsou prázdné. Do nich se při překladu vytváří spustitelné soubory, resp. při spuštění `$ make doc` dokumentace programu.

V adresáři `sql/` se nachází inicializační skripty databázových tabulek. Ty využijete při instalaci, viz příloha C.

V adresáři `src/` potom naleznete zdrojové soubory programu, v adresáři `tex/` zdrojové soubory této práce.

B.1 Automatizovaný překlad

Na CD se vyskytují zdrojové soubory k částem, které se na CD nemusí vyskytovat. Pro využití těchto částí je potřeba obsah CD zkopírovat na disk a vytvořit je pomocí automatického překladu. K tomu účelu je využit program `make`.

Pro překlad programu spusťte v kořenové adresáři příkaz `$ make build`. Ten spustí překlad ze zdrojových kódu do binárních souborů programu. Po skončení překladu budou tyto binární spustitelné soubory k nalezení v adresáři `bin/`. Pro vygenerování dokumentace k programu spusťte příkaz `$ make doc`. Dokumentaci poté naleznete v adresáři `doc/`, a to ve formátech HTML a zdrojových souborů pro LaTeX.

Pro vytvoření této technické zprávy ve formátu PDF spusťte příkaz `$ make text`. Práce bude vytvořena v adresáři `tex/` a v kořenovém adresáři bude vytvořen symbolický odkaz `bp_xkulic01.pdf`.

Příloha C

Instalace

C.1 Závislosti

- databáze MySQL ve verzi 5.0 nebo vyšší
- knihovna Qt ve verzi 4.7 nebo vyšší a program `qmake`
- překladač jazyka C++ (`GCC`, `MinGW` apod.)
- program `make` pro automatizovaný překlad

C.2 Postup instalace

Instalace sestává z několika málo kroků. Postup instalace je popsán pro operační systém GNU/Linux, pro systém MS Windows je velmi podobný a zvládne si jej odvodit každý zkušenější uživatel.

1. Vytvoření databáze. Program SubCipher využívá buď úplnou nebo redukovanou verzi sloníku. Chcete-li mít možnost využívat obě dvě, proveďte tento krok dvakrát, pokaždé s jiným jménem databáze.

```
$ mysql -u <username> -p
Enter password: <not displayed>

mysql> CREATE DATABASE <dbname> CHARACTER SET utf8 COLLATE
    utf8_general_ci;
Query OK, 1 row affected (0.02 sec)

mysql> exit
Bye
```

2. Naplnění databáze slovníkovými daty. Soubory s daty jsou uloženy v adresáři `sql/`. Můžete si vybrat mezi plnou a redukovanou verzí slovníku.

```
$ mysql -u <username> -p < sql/subcipher[_small].sql
Enter password: <not displayed>
```

3. Přeložte program.

```
$ make build
```

4. Vytvořte konfigurační soubor. Více se dozvíte v oddílu C.2.1.

5. Spusťte program.

```
$ bin/SubCipher
```

C.2.1 Konfigurační soubor

Konfigurace databázového spojení je uložena v konfiguračním souboru, který se nachází ve standardním uživatelském úložišti konfiguračních souborů. Umístění je závislé na operačním systému a nachází se v těchto umístěních:

Windows:	`%APPDATA%\SubCipher.ini`
Unix, Mac OS:	`$HOME/.config/SubCipher.ini`

Soubor je ve formátu `.ini` a jeho obsah je velmi jednoduchý:

```
1 [db]
2 hostname = <domenove-jmeno-nebo-ip-adresa-mysql-serveru>
3 databasename = <jmeno-databaze-na-mysql-serveru>
4 username = <prihlasovaci-jmeno>
5 password = <prihlasovaci-heslo>
```

Výpis C.1: Obsah souboru SubCipher.ini

Konfigurační soubor při rozbalení a přeložení programu neexistuje. Je však vytvořena utilitka `SubCipher-setup`, která umožňuje měnit (nebo vytvořit) obsah tohoto souboru nezávisle na operačním systému.