VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

KNIHOVNA ALGORITMŮ PRO ŠIFROVÁNÍ TEXTU

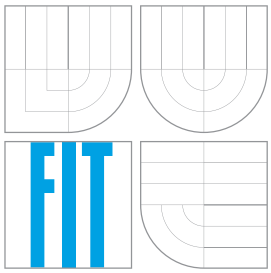
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JIŘÍ MIKULKA

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

KNIHOVNA ALGORITMŮ PRO ŠIFROVÁNÍ TEXTU

LIBRARY OF ALGORITHMS FOR TEXT CIPHERING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JIŘÍ MIKULKA
AUTHOR

VEDOUCÍ PRÁCE — Ing. RADEK KUBÍČEK
SUPERVISOR

BRNO 2011

Abstrakt

Tato práce podává přehled historických i moderních metod a postupů využívaných v kryptografii. Popisuje a zhodnocuje šifry, které byly používány od samotných počátků šifrování informací až po šifry používané v dnešní době. Tento přehled by měl čtenáři poskytnout informace, na jejichž základě by byl schopen rozlišovat mezi jednotlivými šiframi, znát jejich výhody a nevýhody a umět zvolit nejvhodnější šifru pro konkrétní účel. Studované šifry jsou implementovány v knihovně `CipherLib`, která nabízí ukázku použití jednotlivých šifer.

Abstract

This thesis brings an overview of historical and modern methods and approaches used in cryptography. It also describes and assesses ciphers, which have been used since the very beginning of encryption till modern ciphers. Based on information resulting from this overview the reader should be able to distinguish between ciphers, know their advantages and disadvantages, and be able to choose the best cipher for any purpose. Ciphers mentioned in this thesis are implemented in a library called `CipherLib`, which shows usage of every described cipher.

Klíčová slova

kryptografie, kryptoanalýza, bezpečnost, šifrování, šifra, klasická kryptografie, moderní kryptografie, knihovna

Keywords

cryptography, cryptoanalysis, security, encryption, cipher, classical cryptography, modern cryptography, library

Citace

Jiří Mikulka: Knihovna algoritmů pro šifrování textu, bakalářská práce, Brno, FIT VUT v Brně, 2011

Knihovna algoritmů pro šifrování textu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Radka Kubíčka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jiří Mikulka
10. května 2011

Poděkování

Rád bych poděkoval vedoucímu mé bakalářské práce Ing. Radku Kubíčkovi za odborné vedení, čas a cenné rady, které mi při tvorbě práce věnoval. Zároveň bych chtěl poděkovat všem, kteří mě při práci podporovali.

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2010/2011

Zadání bakalářské práce

Řešitel: **Mikulka Jiří**
Obor: Informační technologie
Téma: **Knihovna algoritmů pro šifrování textu**
Library of Algorithms for Text Ciphering
Kategorie: Bezpečnost

Pokyny:

1. Seznamte se s problematikou šifrování textových dat, její historií a postupným vývojem šifrovacích algoritmů.
2. Navrhněte multiplatformní knihovnu implementující nejpoužívanější šifrovací algoritmy od těch nejstarších až po nejnovější. Knihovna musí implementovat šifrování i dešifrování, zahrňte substituční i transpoziční metody.
3. Naimplementujte navrženou knihovnu.
4. Vytvořte aplikaci demonstrující schopnosti vytvořené knihovny.

Literatura:

- http://www.shaman.cz/sifrovani/
- Dle pokynů a doporučení vedoucího.

Při obhajobě semestrální části projektu je požadováno:

- Body 1, 2 a částečně 3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kubíček Radek, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2010
Datum odevzdání: 18. května 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Ochrana informací a zabezpečení komunikace jsou již po dlouhá staletí velmi diskutovanými oblastmi. Nejen v moderní době, ale i ve středověku či starověku si lidé uvědomovali, že klíčem k úspěchu v podnikání, politice či vojenském tažení nejsou jen znalosti či dovednosti, ale především ochrana těchto informací před nepovolanými osobami. Tato tajemství umožňovala obchodníkům prosadit se na trhu se zcela novým produktem nebo vojevůdcům použít nepředvídatelnou strategii. Osoby uchovávající informace v tajnosti si vždy plně uvědomovaly rizika, která byla spojena s odhalením informace. Proto byli povoláváni nejlepší matematikové, fyzikové, jazykovědci a jiní intelektuálové, aby vytvářeli systémy k dokonalému utajení informace. Takové systémy však lze vytvořit jen stěží, protože jsou limitovány aktuálními poznatky z různých vědních oborů a lze předpokládat, že s vývojem znalostí v daných vědních oborech budou i limity překonány a systémy se tak stanou nedokonalými. V opozici vůči tvůrcům těchto systémů vždy stála skupina luštitelů neboli kryptoanalytiků, kteří se snažili utajované informace odhalit. Neustálý boj na poli utajení informací se odehrává mezi *kryptografy* a *kryptoanalytiky*, podle jejichž znalostí a schopností je možné posuzovat úroveň bezpečnosti vytvořeného systému. Po dlouhá staletí to byli kryptoanalytici, kteří byli vždy krok před kryptografy, ale s nástupem moderní kryptografie se jejich role vyměnily.

Pro vytváření systémů pro utajování informací vznikala nová odvětví. Ať se jednalo o *steganografii*[1] (ukrytí zprávy) či *kryptografii*[2] (ukrytí významu zprávy, nikoli zprávy samotné), cílem těchto metod bylo co nejlepší zajištění bezpečnosti informace. Kryptografii lze rozdělit na dvě základní oblasti – *kódování* a *šifrování* (viz kapitola 3).

V dnešní době jsou informace uchovávány převážně v elektronické podobě. To s sebou přináší řadu výhod, ale také spoustu nevýhod, které jsou spojeny především s bezpečným uložením na datových médiích či přenosem mezi počítači. Pokud by např. zprávy z tajného výzkumu nebyly šifrovány, mohly by nedopatřením padnout do rukou nepovolané osoby, což by mohlo mít za následek nejen ukončení výzkumu, ale i ohrožení mnoha lidí. Proto je dnes stále důležitější data chránit a pečlivě vybírat osoby, které mají k datům přístup. Problematika kryptografie je tak velmi aktuální a v budoucnu tomu nebude jinak. Proto je na tuto oblast nahlíženo jako na velmi perspektivní, bohatství společností je totiž z velké části tvořeno i informacemi.

V této práci je čtenáři představen vývoj šifer počínaje starověkými postupy až po moderní metody. Šifry, které jsou v této práci studovány, jsou implementovány v knihovně

[1] Steganografie – z řeckých slov *steganos* (στεγανός, tj. schovaný) a *graphein* (γράφειν, tj. psát) [23].

[2] Kryptografie – z řeckých slov *kryptos* (κρυπτός, tj. skrytý) a *graphein* (γράφειν, tj. psát) [23].

algoritmů pro šifrování textu, jejíž vytvoření je cílem této práce. I nepoučený čtenář by měl být po přečtení této práce schopen rozlišovat mezi různými šiframi, znát jejich výhody a nevýhody a na základě těchto znalostí vybírat vhodné šifry pro konkrétní případy použití.

Práce je členěna do několika kapitol – v kapitole 2 je představen historický vývoj šifer v souvislosti s vyspělostí společnosti a vědy. V kapitole 3 je uveden pojem *kryptografie* spolu s používanou *terminologií* a provedena základní *klasifikace šifer*. Kapitola 4 podrobně popisuje implementované šifry. V následující kapitole 5 jsou kladeny požadavky na knihovnu a návrh této knihovny. Zde je také představen návrh aplikace, která demonstruje možnosti implementované knihovny. V kapitole 6 je pak podrobně popsána implementace knihovny i aplikace, včetně výběru implementačního jazyka a způsobu rozšíření knihovny. V poslední kapitole 7 je zhodnocen přínos této práce, implementované knihovny a aplikace. Jsou zde také představeny možnosti pokračování a rozšíření této práce v budoucnu.

Kapitola 2

Historie

Potřeba zabezpečení komunikace a informací existuje stejně dlouho jako komunikace sama. Lidé odjakživa uchovávají informace, které jsou určeny jen omezenému okruhu lidí. A proto historie kryptografie sahá tisíce let zpátky. Nejedná se o jedinou specializovanou oblast matematiky či informatiky, ale o syntézu více vědních oborů, které se vyvíjejí spolu s lidských věděním a poznáním. Díky této mezioborové provázanosti lze kryptografické algoritmy studovat z jiných úhlů pohledu a získat tak věrnější obraz tehdejší doby a společnosti. Paralelně s kryptografií se vyvíjí i kryptoanalýza studující naopak způsoby a metody prolomení kryptografických mechanismů. Tuto vlastnost kryptoanalýzy lze s úspěchem použít např. pro luštění nápisů v neznámých písmech, které z dnešního pohledu šifru tvoří.

Cílem této kapitoly je nastínění nejvýznamnějších událostí, jež ovlivnily vývoj kryptografie. Událostí v této kapitole probírané korespondují s vývojem kryptografie představeným v [7, 11, 29, 38].

2.1 Klasická kryptografie

Mezi jedny z nejstarších známých případů použití kryptografie patří **egyptské hieroglyfy**. Původ kamenných desek s hieroglyfy nalezených při Napoleonově tažení do Egypta v roce 1799 je podle doposud provedených výzkumů datován do 2. století př. n. l. Objev a zejména rozluštění hieroglyfů na známé *Rosettské desce* (viz [6]) odstartoval další řadu úspěšných rozluštění egyptských hieroglyfů pocházejících z období poloviny 2. tisíciletí až 2. století př. n. l. Metody a postupy, jež Jean-François Champollion, nejvýznamnější luštitel egyptských hieroglyfů, používal při jejich luštění, lze považovat za čistou kryptoanalýzu. Přestože nápisy v hieroglyfech nebyly původně zamýšleny jako šifrovaná zpráva, z dnešního hlediska je lze považovat za šifry, protože skutečný význam je ukrytý v nesrozumitelném textu či obrazu.

Podobným případem odhalení významu starodávných písem je i rozluštění **lineárního písma B**. Toto písmo bylo používáno v pozdní mykénské civilizaci v 14. – 12. století př. n. l. na Krétě; tedy přibližně v době, kdy se odehrávala na severozápadě dnešního Turecka trojská válka. Analýzou stovek destiček nalezených při archeologických vykopávkách na severu Kréty se vytvořila dostatečná databáze textů v tomto písmu, pomocí které bylo možné písmo rozluštit. Největší zásluhy na rozluštění tohoto písma má Michael Ventris, který systematickou prací dokázal odhalit fonetickou podobu téměř všech znaků, jež byly na destičkách nalezeny. Díky jeho práci mohlo po více než 3000 letech znovu ožít písmo a jazyk používaný vyspělou mykénskou civilizací koncem 2. tisíciletí př. n. l. [38]

Židovská kultura a náboženství obsahuje spoustu tajemství a mystiky. Aby proroctvím

zapsaným v Bibli bylo přidáno na významnosti, někteří bibličtí pisatelé používali šifry. Použití židovských šifer lze považovat za skutečně první metody se záměrem zakrytí obsahu zprávy. K vynálezu těchto šifer mohlo dojít právě u Židů, neboť židovská vzdělanost byla ve své době na velmi vysoké úrovni ve srovnání s okolními národy. Mezi známé židovské šifry patří např. **šifra Atbaš** (resp. *Atbš*, jelikož hebrejština nepoužívá samohlásky), jejíž princip podle [38] spočívá v nahrazení každého písmene textu písmenem vzdáleným od konce abecedy stejný počet míst, jako je nahrazované písmeno vzdáleno od začátku abecedy. Název této šifry pochází z písmen hebrejské abecedy – první písmeno (*alef*), poslední písmeno (*tav*), druhé písmeno (*bet*) a předposlední písmeno (*šin*). Použití této šifry je zřejmé v biblické knize Jeremiáš pocházející z 6. století př. n. l.: *„všechny krále severu, blízké i vzdálené, jednoho po druhém – všechna království země, co jich je na světě. A po nich se napije král Šešak!“* [13, Jr 25,26] a *„Ach, jak byl ten Šešak lapen, jak chloubu světa dobyli? Jak mohl být tak zpustošen Babylon mezi národy!“* [13, Jr 51,41]. V těchto ukázkách je namísto skutečného názvu města **Babel** (resp. Babylon) použito **Šešak**, protože město Babel (resp. Babylon) je v židovské kultuře vnímáno silně negativně (zmatení, otroctví, hřích, aj.). Mezi další biblické šifry patří celé prorocké knihy Daniel či Zjevení, kde je uvedeno *„číslo šelmy“* 666 nepochybně tvořící šifru.

Použití kryptografie je známo i v Řecku, zejména ve vojenském státě Sparta. Vojevůdci používali v 5. století př. n. l. pro vzájemnou *bezpečnou* komunikaci transpoziční **šifry Skytale**[1]. Princip této šifry spočívá v omotání pruhu papíru kolem n-stěnného hranolu; zpráva je zapsána postupně na stěny hranolu a odmotáním pruhu papíru z hranolu vznikne zašifrovaná zpráva.

Obrázek 2.1: Šifrovací váleček **Skytale** používaný ve Spartě (zdroj: [1]).

Římský císař a vojevůdce Julius Caesar v 1. století př. n. l. také používal pro komunikaci se svými generály různých šifer. Nejznámější a nejpoužívanější z nich je **Caesarova šifra**, jíž byl sám autorem, a zmiňuje se o použití této šifry v *Zápiscích o válce galské*. Princip této substituční šifry spočívá v nahrazení každého znaku písmenem nacházejícím se o 3 pozice od něj vpravo.

Začátek středověku patřil nejen v kryptografii, ale i jiných vědních odvětvích arabské vzdělanosti. Šíře znalostí arabských kultur byla mnohonásobně větší než ostatních kultur (odtud pochází základy dnešní medicíny, matematiky, aj.). Množství znalostí umožnilo vzniknout novému odvětví – **kryptoanalýze**. Prvními používanými metodami byla **frekvenční analýza**[2], díky které bylo možné všechny dosud známé šifry prolomit.

[1] *Skytale* (σκυτάλε, tyč či hůl používaná ve Spartě jako šifra pro psaní depeší; na proužek kůže šikmo omotaný kolem tyče byla podélně zapsána depeše, takže po odmotání byla nesrozumitelná: velitelé v zahraničí měli hůl stejné tloušťky, kolem které omotali proužek kůže, a tak byli schopní depeši přečíst) [23, s. 736].

[2] Frekvenční analýza zkoumá četnost znaků v prostém a šifrovaném textu. Porovnáním těchto četností jsou odvozeny jednotlivé substituce.

Ve středověké Evropě byla všechna věda, poznání, vzdělanost a moudrost soustředěna do klášterů, a tak gramotnost (a tudíž i potřeba zabezpečení psané korespondence) byla mezi lidmi na velmi nízké úrovni. To se však v renesanci změnilo – nastal rozkvět všech vědních oborů a oblastí. Významným přínosem pro kryptografii byl Leon Battista Alberti, který vůbec poprvé v historii navrhl použití polyalfabetické šifry. Tuto myšlenku sám podrobně nerozpracovával, ale navázal na něj Blaise de Vigenère, který navrhl polyalfabetickou **Vigenèrovu šifru** využívající 26 různých abeced. Tato šifra se stala dostupnými kryptoanalytickými metodami neprolomitelná, protože disponuje velkým množstvím klíčů[3]. Kryptografie v této době zažívala období velkého rozmachu, a tak se jí mnozí lidé věnovali. K nejznámějším patří Johannes Trithemius, který napsal první tištěnou knihu o kryptografii (1499 *Steganographia*) a také navrhl vlastní šifru, Giovanni Battista Porta, jenž se zaměřil na tvorbu digrafové šifry, či Sir Francis Bacon, který navrhl biliterální pětibitový kód. Kromě substitučních šifer vznikaly i transpoziční šifry vycházející z jednoduché sloupcové transpozice; mezi tyto šifry patří např. dvojitá sloupcová transpozice či šifra AMSCO.

Používání šifer a kódů s sebou přináší i určitá rizika – např. pokud komunikující strany naprosto důvěřují bezpečnosti šifry, mohou zapomenout na obezřetnost a další bezpečnostní opatření. V 16. století tak bylo odhaleno spiknutí proti anglické královně Alžbětě I. vedené skotskou královnou Marií Stuartovnou. Díky bystrosti, logickému a kombinačnímu myšlení kryptoanalytiků byly rozluštěny všechny zašifrované dopisy, které si Marie vyměňovala se svými stoupenci, a tak byli obviněni a popraveni všichni účastníci spiknutí.

Když v 19. století nastala vědeckotechnická revoluce, přišlo na světlo světa mnoho nápadů a vynálezů, z nichž mnohé se využívají dodnes. Pro vzdálenou komunikaci byl navržen a zkonstruován telegraf. Použití telegrafu však vyžadovalo speciální kód, který by umožnil přenášet rychle i dlouhé zprávy na velkou vzdálenost. Za tímto účelem byla vynalezena **Morseova abeceda**, která reflektuje četnost znaku v jeho kódovém přepisu (hodně frekventované znaky mají kratší kódovaný zápis, kdežto málo používané znaky mají zápis delší). Veřejná obliba šifer a kódů rostla. Šifry se už nepoužívali jen jako nástroj pro utajení informace, ale stále více plnily i funkci zábavnou. Mezi takové šifry patří např. malý a velký polský kříž, různé přesmyčky a modifikace Morseovy abecedy (uzly na provaze, aj.) a mnohé další transpoziční šifry (více viz [40]).

Jelikož ve viktoriánské Anglii si páry nemohly veřejně projevovat vzájemnou náklonnost, komunikovali pomocí šifrovaných zpráv v novinách. Používaly jednoduché šifry jako např. **podle plotu** pracující na podobném principu jako starověká šifra Skytale. Luštěním těchto šifer si krátil čas i Lord Playfair, jež je autorem stejnojmenné **šifry Playfair**. Ve stejné době probíhala na americkém západě zlatá horečka a mnoho Evropanů a Američanů tam utíkalo hledat štěstí. Mezi nimi i Thomas Beale, který se rozhodl svůj poklad ukrýt. Zanechal 3 dokumenty v tzv. **knižní šifře** přesně popisující místo, kde je poklad uschován. Podle [38] se však doposud podařilo rozluštit pouze jeden dokument. Toto období však v oblasti kryptografie (resp. kryptoanalýzy) přineslo zlomový okamžik – Charles Babbage našel způsob, jakým rozluštit polyalfabetickou substituční Vigenèrovu šifru. Tento objev znamenal největší přínos pro kryptografii a kryptoanalýzu od dob arabských učenců na začátku středověku, kteří položili základy frekvenční analýze. V stejné době dospěl ke stejnému způsobu prolomení Vigenèrovy šifry i pruský důstojník Friedrich Wilhelm Kasiski, který své závěry publikoval. Tato metoda je dnes známa jako **Kasiského test** (*Kasiski examination*).

[3] Celkový počet dostupných klíčů pro Vigenèrovu šifru je $26! = 403\,291\,461\,126\,605\,635\,584\,000\,000$.

2.2 Světové války

První polovina 20. století byla ve znamení světových válek. První světová válka bývá označována jako *válka chemiků* (bojové plyny), druhá světová válka jako *válka fyziků* (atomová energie) a předpokládá se, že pokud by vypukla třetí světová válka, pak by to byla *válka matematiků* (informace). S nejmodernější bojovou technikou bylo pro všechny bojující strany nutné nasadit vysokou úroveň zabezpečení komunikace.

V první světové válce německá armáda sázela na rychlý a překvapivý útok, proto na úroveň šifrování nekladli velké nároky. Jejich šifrované zprávy byly pro protivníky zcela průhledné. Za zmínku však stojí několik šifer – (1) **šifra ADFGVX** využívá náhodné abecedy spolu s číslicemi zapsané do Polybiova čtverce, znaky prostého textu jsou zašifrovány do dvojice znaků určující řádek a sloupec ve čtverci; (2) poměrně kuriózní metodu **mikrotečky** používali němečtí agenti v Latinské Americe, pomocí optických zařízení dokázali zmenšit celou stranu textu do velikosti tečky za větou, takto zašifrované zprávy posílali ve zdánlivě neškodném textu. Během první světové války se objevila dosud neprolomená **Vernamova šifra** známá také jako **jednorázová tabulková šifra** (*one-time pad*). Tato šifra využívá zcela náhodně generovaný klíč, jež je po zašifrování zprávy zničen. Dle [38] je dodnes tato šifra využívána pro šifrování přímé komunikace mezi prezidenty USA a Ruska.

Druhá světová válka již disponovala mnohem většími znalostmi z fyziky, matematiky a dalších odvětví včetně kryptografie. Všechny bojující strany si uvědomovaly vysokou důležitost bezpečné komunikace, proto vytvářely nejrůznější kryptografické systémy. Německo používalo rotující stroj **Enigma**. Rotující stroje jsou elektromechanická zařízení implementující polyalfabetické substituční šifry s mnoha doprovodnými nastaveními. Šifrovací stroj Enigma nejprve před válkou sloužil ve veřejném sektoru, pro válečné účely byla zvýšena její bezpečnost a byla nasazena ve všech armádních odvětvích. Rozluštění šifry Enigma započal polský matematik a kryptolog Marian Rajewski, v jeho práci pokračoval Alan Turing, který pro rozluštění šifry Enigma navrhl Turingův stroj (viz [38]) schopný řešit jakýkoliv řešitelný problém.

Na straně americké armády také probíhal výběr nejbezpečnější a přitom nejrychlejší šifry pro zajištění bezpečnosti přenášených zpráv. Pro šifrování zpráv používali přirozený jazyk indiánského kmene **Navajo** (jazyk tohoto malého kmene byl zcela odlišný od jazyka jiných indiánských kmenů a kromě členů tohoto kmene jejich jazyk nikdo neovládal). Jejich jazyk však nedisponoval všechny potřebnými termíny (např. letadlo, bomba, atd.), proto bylo nutné vytvořit slovní spojení ze slov v jazyce Navajo k popsání těchto pojmů. Přestože se jednalo o přirozený jazyk, nebyla tato šifra nikdy prolomena a řadí se tak mezi několik málo neprolomených šifer.

2.3 Moderní kryptografie

Již v průběhu války se začal rozvíjet počítačový průmysl. Příchod počítačů znamenal radikální změnu v pohledu na informace – dosud byly informace vnímány jako řetězce znaků, rovnice či obrázky, s příchodem počítačů bylo nutné informace nějakým způsobem uložit do paměti počítače. Od tohoto okamžiku všechny šifry pracují s informacemi coby posloupností bitů s určitým významem. Všechny doposud vytvořené šifry mají jednu zásadní nevýhodu – všechny komunikující strany musí vlastnit stejný klíč, vzniká tak problém distribuce klíčů. Již v roce 1883 položil Auguste Kerckhoffs von Nieuwenhof jeden ze základních principů kryptografie známý jako **Kerckhoffsův princip**: *„Bezpečnost šifrovacího systému nesmí záviset na utajení algoritmu, pouze na utajení klíče.“* [28]

Jakmile se situace po válce uklidnila, lidé opět začali podnikat a obchodovat. S tím byla spojena nutnost chránit firemní tajemství před nepovolanými osobami. Po několika letech vývoje vznikla **šifra Feistel** (později jako Lucifer), která informaci v binární reprezentaci šifrovala po stejně velkých blocích – stala se základem všech známých blokových šifer. Později z této šifry vznikla **šifra DES**, která byla i standardizovaná. Přestože ve své době tato šifra nabízela dostatečnou úroveň zabezpečení, v dnešní době ji neposkytuje a musela být nahrazena (využívá se např. šifra TDEA, což je 3x aplikovaná šifra DES, šifra AES, aj.). Stále však setrvává zásadní problém – všechny komunikující strany musí vlastnit totožný klíč. Šifra se poté stává natolik bezpečnou, nakolik je zajištěna bezpečnost distribuce klíče.

Řešením problému distribuce klíčů se zabýval Whitfield Diffie a Martin Hellman, kteří navrhli systém asymetrické distribuce klíčů. Tento systém je nazýván jako **Diffie-Hellman key exchange** (popis systému viz [9], text patentu viz [15]). Jejich přínos je pro kryptografii naprosto zlomový, protože po více než 2000 letech používání šifer navrhli systém, který umožňoval použití více klíčů pro zašifrování (resp. dešifrování) zprávy. Implementací tohoto schématu se zabývali Ronald Rivest, Adi Shamir a Leonard Adleman, kteří hledali vhodnou matematickou funkci splňující požadavky vyplývající ze systému navrženého Diffie a Hellmanem. Výsledkem jejich práce jsou funkce pracující s prvočísly v modulární aritmetice jednoduché na výpočet, ale náročné (až nemožné) na prolomení. Jejich funkce se opírají o předpoklad, že faktorizace (tj. rozklad čísla na prvočísla) velkých čísel je sice možná, ale časově přesahuje přijatelné meze. Soubor jimi navržených funkcí vytváří **RSA šifru** (text patentu viz [33]).

Přestože šifra RSA (popis algoritmu viz [18]) poskytuje vysokou úroveň zabezpečení, její použití je časově a paměťově velmi náročné kvůli výpočtům s vysokými čísly. Phil Zimmermann chtěl nabídnout bezpečnou komunikaci jak armádě, vládě a firmám, tak všem občanům. Snažil se proto vytvořit software **PGP** (*Pretty Good Privacy*), který by generování klíčů pro RSA šifru urychloval. Urychlení docílil spojením nového asymetrického šifrovaní s klasickým symetrickým šifrováním. Symetrické klíče je možné velmi rychle generovat, jejich distribuce je zajištěna principy asymetrické kryptografie. Celá komunikace se tím velmi urychlí (a také jsou kladeny nižší nároky na výpočetní výkon počítačů generujících klíče pro šifru RSA). PGP umožňuje komukoliv bezpečně komunikovat na internetu, což je vzhledem k rostoucímu trendu elektronické komunikace velmi žádoucí.

Ze současných kryptografických algoritmů stojí za zmínku šifra **AES** (***A****dvanced* ***E****ncrypytion* ***S****tandard*) – z kandidátů na tento standard byl vybrán algoritmus **Rijndael**, bloková šifra **Blowfish** silně závislá na klíči (viz [37]), **Serpent** či **Twofish** (obě tyto šifry patřily mezi kandidáty na šifru AES). Z proudových šifer je dnes hojně využívána šifra **RC4** (je např. použita v protokolu SSL). Kromě proudových a blokových šifer jsou dnes používány i mnohé jednocestné hashovací algoritmy (např. **MD5**, **SHA**, aj.) pro šifrování hesel v elektronické komunikaci.

2.4 Budoucnost

Přestože současné kryptografické algoritmy poskytují zabezpečenou komunikaci a šifrování dat, spousta lidí bezpečnost podceňuje (převážná většina komunikace, transakcí, atd. probíhá elektronicky, proto je jejich zabezpečení nutné). V současné době také existuje stále více kryptoanalytiků, kteří se snaží zabezpečení komunikace prolomit; je jen otázkou času, kdy se to podaří. Distribuovaný výpočetní výkon dnešních počítačů by postačoval na rozluštění i těch nejsložitějších šifer během několika hodin či dní. Tento způsob útoku však není nejvýhodnější, protože platí **Mooreův zákon**: *„Počet tranzistorů umístěných na čipu se*

přibližně zdvojnásobí každých 24 měsíců.“ (*„The number of transistors incorporated in a chip will approximately double every 24 months.“*) [17] Během několika let tak výrobní proces narazí na úroveň atomů a nutně se bude hledat zcela nový směr ve výrobě počítačových komponent.

Již několik let probíhají výzkumy, které si kladou za cíl vytvořit kvantový počítač. Výpočetní výkon takových počítačů by byl mnohonásobně vyšší než výkon současných počítačů. To by znamenalo zásadní změny i v kryptografii. Šifry, které jsou dnes považovány za neprolomitelné právě kvůli omezení výpočetním výkonem, by najednou ztratily svou bezpečnost – kvantové počítače by dokázaly spočítat klíč během okamžiku. Proto je nutné zavést zcela nový koncept kryptografie postavený na kvantové fyzice – kvantovou kryptografii.

Kvantová kryptografie využívá vlastností fotonů jako je jejich natočení a další. Díky tomu by bylo možné vytvořit naprosto nerozluštitelné kvantové šifry, s nimiž by si neporadily ani kvantové počítače. Při jakémkoliv zásahu do šifry by byla celá přenášená zašifrovaná informace zničena; tím by se zabránilo všem útokům. Proto by nebylo možné bez znalosti přesné konfigurace proudu fotonů zprávu ani přečíst. [38]

Kapitola 3

Kryptografie

Dle [38] je **kryptografie** neboli šifrování vědní nauka, která řeší problém, jakým způsobem převést zprávu do podoby, jež je srozumitelná pouze se speciální znalostí (většinou se jedná o znalost algoritmu a klíče). Kryptografie je spolu s **kryptoanalýzou** (tj. nauka o luštění šifrovaných zpráv) součástí oboru **kryptologie**, který zahrnuje cokoliv, co má spojitost se šiframi.

Kryptografie má jako každé jiné odvětví vlastní terminologii umožňující používat jednotná označení jevů, a tak předcházet nedorozuměním. Pojmy jsou zde vysvětleny podle [21]. Za českým pojmem je v závorce uveden používaný anglický ekvivalent.

3.1 Terminologie

Prostý text (*plaintext*) je původní zpráva určená k zabezpečení. Většinou se jedná o zprávu v některém běžně používaném jazyce, který je srozumitelný oběma komunikujícím stranám. Tato zpráva může být zabezpečena několika způsoby. Jedním způsobem je **steganografie** (*steganography*) zakrývající samotnou existenci zprávy. Mezi tyto metody patří neviditelné inkousty, mikrotečky, aj. Metody **kryptografie** (*cryptography*) nezakrývají existenci zabezpečené zprávy, ale zprávu vytváří nesrozumitelnou pro nepovolanou osobu transformací prostého textu. Existují dvě základní metody transformace prostého textu. Při **transpozici** (*transposition*) jsou písmena v prostém textu zpřeházena; původní pořadí písmen je porušeno. Slovo `šifra` po transpozici může být zapsáno jako `FŠRAI`. Pokud je použita **substituce** (*substitution*), písmena v prostém textu jsou nahrazena jinými písmeny, čísly nebo symboly. Slovo `šifra` pak může být zapsáno jako `19-9-6-18-1`. Důležitý rozdíl mezi transpozicí a substitucí spočívá v tom, že při použití transpozice jsou zachována všechna písmena ve změněném pořadí, kdežto u substituce jsou původní znaky nahrazeny jinými. Transformovaný prostý text se nazývá **šifrový text** (*ciphertext*).

Pro zápis prostého textu se používá **otevřená abeceda** (*plain alphabet*). Znaky této abecedy jsou zpravidla zapisovány `malými písmeny`. Pro zašifrování prostého textu se využívá **šifrová abeceda** (*cipher alphabet*) a znaky této abecedy jsou zapisovány `VELKÝMI PÍSMENY`. Při výskytu prostého a šifrovaného textu je pak jednoznačně rozlišeno, o jaký text se jedná. Slovo `šifra` je zapsána v otevřené abecedě, kdežto `FŠRAI` je zapsána v šifrové abecedě. U některých šifer se v šifrové abecedě vyskytuje více substitucí pro jedno písmeno otevřené abecedy. Tyto substituce jsou nazývány **homofony** (*homophones*) – např. písmeno `a` může být nahrazeno čísly `10`, `16` či `76`. U jiných šifer může šifrová abeceda obsahovat znaky, které nemají význam (nejsou spárovány s žádnými znaky otevřené abecedy).

Tyto znaky jsou nazývány **nuly** (*nulls*) a slouží pouze pro zkomplikování luštění zašifrované zprávy. Šifrovací systémy se také mohou lišit počtem šifrovaných abeced, které využívají. Pokud je používána pouze jedna abeceda, jsou tyto systémy nazývány **monoalfabetické** (*monoalphabetic*), při použití více abeced jsou systémy nazývány **polyalfabetické** (*polyalphabetic*).

Přestože slova *šifra* a *kód* bývají často považována za synonyma, existuje mezi nimi rozdíl. **Šifra** (*cipher*) je postup, kterým je srozumitelný prostý text převeden na nesrozumitelný šifrový text znak po znaku. Někdy může být šifrována skupina 2 znaků – tzv. **digrafy** (*digraph*) či **bigramy** (*bigram*); zřídka jsou šifrovány i skupiny více znaků tzv. **polygramy** (*polygrams*). Při šifrování se mohou také zavádět tzv. **nomenklátory** (*nomenclator*), což jsou slova či fráze, kterými je nahrazena část prostého textu. Seznam nomenklátorů však musí vlastnit obě komunikující strany, proto jejich zavedení přináší ohrožení bezpečnosti šifrovacího systému. **Kód** (*code*) je také postup, který převádí prostý text na šifrový text, ale převádí skupiny znaků (slabiky, slova, více slov aj.) na jiné skupiny znaků (čísla, slova, věty aj.). Kódy se skládají z mnoha slov, frází, písmen a slabik společně s **kódovými slovy** (*codewords*) nebo **kódovými čísly** (*codenumbers*), které nahrazují odpovídající části prostého textu.

Při převodu prostého textu do šifrovaného textu se často využívá **klíč** (*key*) sloužící jako jedinečný parametr šifrovacího algoritmu. Klíč může být ve formě **klíčového slova** (*keyword*), **klíčové fráze** (*keyphrase*) či **číselného klíče** (*keynumber*). Použitím klíče vzniká spousta šifrových abeced, a tak jsou kryptoanalytické útoky náročnější, resp. zvyšuje se úroveň bezpečnosti kryptografického systému.

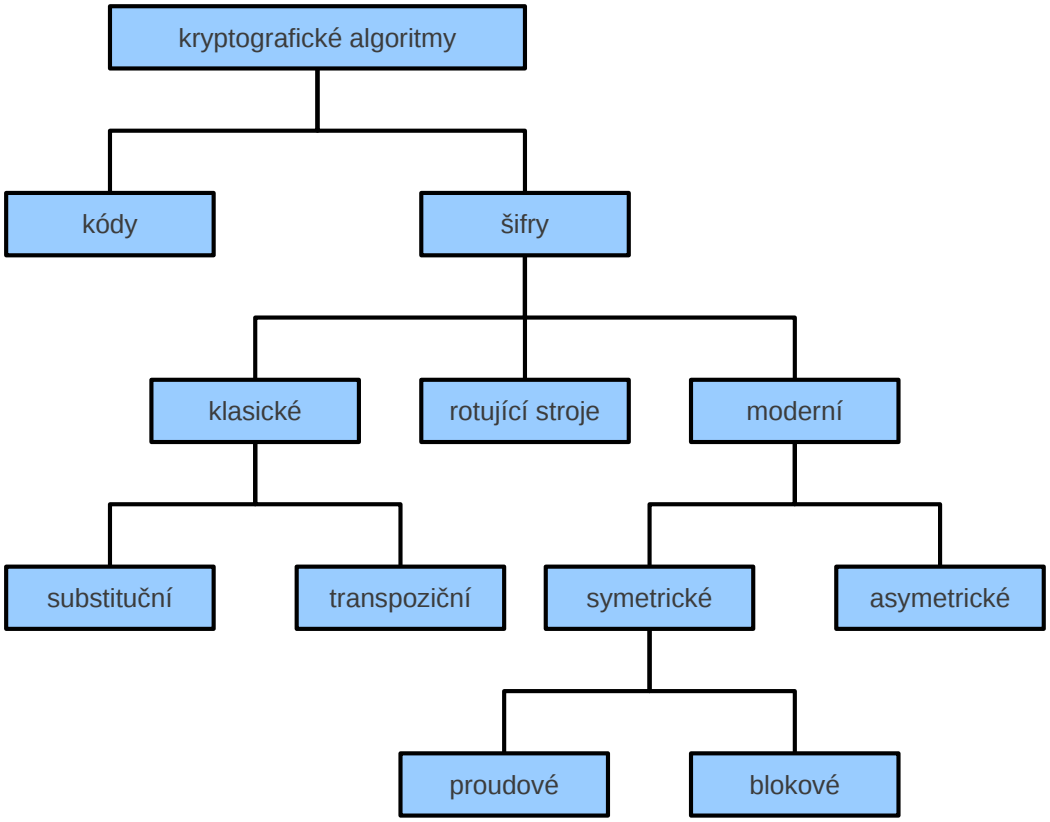
Transformace prostého textu do **šifrovaného textu** (*ciphertext*) se nazývá **šifrování** (*encipher*) či **kódování** (*encode*). Opačný proces (převod šifrovaného textu do prostého textu) se nazývá **dešifrování** (*decipher*) či **dekódování** (*decode*). Pokud dešifrování (resp. dekódování) provádí osoba, která nezná algoritmus nebo klíč, pak se tento proces označuje jako **kryptoanalýza** (*cryptanalysis*). **Kryptologie** (*cryptology*) je obor zahrnující jak **kryptografii**, tak **kryptoanalýzu**. [21]

3.2 Klasifikace šifer

Pro usnadnění orientace v šifrách (resp. kódech) je nutné zavést klasifikaci, podle které jsou šifry rozlišovány. V této práci autor vychází z klasifikace kryptografických algoritmů uvedené v [19]. Tato klasifikace (viz obrázek 3.1) rozlišuje kódy a šifry. Šifry dále rozděluje na klasické šifry, rotující stroje a moderní šifry. V následujícím textu budou vysvětleny principy jednotlivých skupin algoritmů a probírány jejich výhody a nevýhody. Zástupci jednotlivých kategorií budou podrobně probíráni v kapitole 4. Při popisu jednotlivých kategorií autor vychází z [21, 35, 36, 38].

3.2.1 Šifry

Šifra je kryptografický pojem popisující algoritmus pro šifrování (resp. dešifrování) prostého textu skládající se z posloupnosti pevně daných kroků. Provedením této posloupnosti lze prostý text zašifrovat a získat tak šifrový text. Šifrový text pak obsahuje všechny informace, které byly obsaženy v prostém textu. Bez znalosti klíče je velmi obtížné či téměř nemožné dešifrovat zašifrovanou zprávu do srozumitelné podoby.



Obrázek 3.1: Klasifikace kryptografických algoritmů (zdroj: vlastní tvorba dle [19]).

Klasické šifry

Klasické šifry zahrnují kryptografické algoritmy hojně používané v historii. Tyto algoritmy však v dnešní době nenabízejí tak vysokou úroveň zabezpečení, jaká se od šifer vyžaduje. Tyto šifry operují s otevřenou a šifrovou abecedou a prostý text převádějí znak po znaku (případně skupiny znaků) do šifrového textu. Náročnost těchto šifer není vysoká, k zašifrování (resp. dešifrování) textu není zapotřebí speciálních přístrojů či počítačů. Proto s příchodem počítačů tyto šifry nejsou nadále považovány za bezpečné, přestože v minulosti byly mnohé z nich označovány jako *nerozluštitelné*. Některé z těchto šifer však mají i dnes význam – např. Vernamova šifra se stala základem moderních proudových šifer a také hraje roli na poli kvantové kryptografie. Slabou stránkou těchto šifer je náchylnost na různé druhy útoků (např. **frekvenční analýza**[1]) bez jakékoli znalosti algoritmu šifry či klíče. Snahou o eliminaci tohoto nedostatku je zavedení více šifrových abeced (vznikají tak polyalfabetické šifry).

Mezi klasické šifry jsou zde zařazeny všechny, které pracují s prostým textem coby polem znaků. Znaky prostého textu jsou pak pomocí předem specifikovaného algoritmu

[1] **Frekvenční analýza** je kryptoanalytická metoda vycházející ze znalosti jazyka, ve kterém byla zpráva zapsána. V každém jazyce lze vytvořit histogram frekvencí výskytů jednotlivých znaků. Podobný histogram výskytů znaků lze vytvořit i v zašifrované zprávě. Porovnáním těchto histogramů lze odvodit páry písmen mezi otevřenou a šifrovanou abecedou. Tato je metoda je však použitelná pouze při použití jedné šifrované abecedy (tedy u **monoalfabetických šifer**), při použití více šifrovaných abeced je tato metoda neúčinná.

transformovány na šifrovaný text, šifrování tedy probíhá na úrovni znaků (či větších entit jako jsou slabiky a slova). Tyto šifry jsou v zásadě dvojího charakteru – buď znaky prostého textu nahrazují jinými (pozice v prostém i zašifrovaném textu zůstává stejná) nebo znak prostého textu zůstává zachován, ale jeho pozice se mění. Způsob dešifrování u těchto šifer je inverzní k šifrování; tzn. akce při dešifrování probíhá v opačném pořadí jako při šifrování.

Substituční šifry Substituční metody provádějí nahrazování znaků prostého textu jinými znaky z šifrové abecedy podle daného algoritmu. Pro zašifrování prostého textu existuje funkce $E(pt)$, jejímž vstupem je prostý text a výstupem je šifrovaný text. K dešifrování existuje funkce $D(ct)$, která je inverzní k šifrovací funkci. Pro tyto funkce platí:

$$E(pt) = E(D(ct)) = ct \tag{3.1}$$

$$D(ct) = D(E(pt)) = pt \tag{3.2}$$

$$D(ct) = E^{-1}(ct) \tag{3.3}$$

Vzhledem k tomu, že tyto šifry transformují prostý text znak po znaku, musí být předem specifikováno, které znaky prostého textu budou šifrovány a které budou ignorovány. Tento problém je u většiny šifer vyřešen převodem prostého textu do anglické abecedy (tzn. odstraní se všechny akcenty a znaky národních abeced se tak nahradí ekvivalenty v anglické abecedě). Některé šifry umožňují i převod číslic (resp. čísel) či jiných speciálních znaků. Ostatní znaky bývají zpravidla ignorovány.

Transpoziční šifry Transpoziční šifry zachovávají všechny znaky prostého textu, pouze mění jejich pořadí. Prostý text je znak po znaku transformován v šifrovaný text podle předem daného algoritmu. Dešifrovací funkce je dána reverzním sledem operací šifrovací funkce (pro každý znak šifrovaného textu lze určit umístění v prostém textu). Šifrovaný text je přesmyčkou textu prostého.

Kombinované šifry Tato skupina šifer využívá principů jak substitučních šifer, tak principů šifer transpozičních. Spojení těchto přístupů umožňuje vytvářet šifry poskytující vyšší úroveň zabezpečení než samostatné šifry substituční či transpoziční.

Rotující stroje

Rotující stroje jsou elektromechanická zařízení používaná pro šifrování a dešifrování tajných zpráv. Rotující stroje byly rozšířeny především v letech 1930 – 1945 a tehdy znamenaly vrchol kryptografie. Mezi nejznámější příklady patří německé šifry *Enigma* či *Lorenz* nebo šifra *Fialka* sovětské armády za 2. světové války (viz [30]).

Princip těchto strojů je založen na polyalfabetické substituční šifře. Tvoří však zvláštní skupinu šifer, protože šifrované abecedy jsou vytvářeny samotnou mechanikou stroje. Konstrukce jednotlivých strojů se může lišit, přesto jejich základní struktura je velmi podobná. Jak je uvedeno v [30], rotující stroje se skládají ze sady rotorů (neboli koleček či bubínků), což jsou rotující disky s polem elektrických kontaktů na obou stranách. Elektrické spoje mezi kontakty pak implementují substituce mezi písmeny. Pro zajištění vyšší bezpečnosti rotory mění svoje pozice, čímž způsobí substituce mezi jinými písmeny při zachování stejných elektrických spojů. Tyto změny pozic generují nové šifrové abecedy, což je důvod, proč je na rotující stroje pohlíženo jako na polyalfabetické substituční šifry.

Moderní šifry

Mezi moderní šifry patří všechny metody, které na prostý text nenahlížejí jako na posloupnost znaků, nýbrž **posloupnost bitů**. Ke vzniku těchto šifer tak mohlo dojít až v okamžiku, kdy se změnil pohled na informace. Před vznikem počítačů byly informace chápány jako řetězce znaků, rovnice, obrázky, aj. Se vznikem počítačů bylo nutné řešit otázku uložení a manipulace s informacemi – v této době vzniklo nové paradigma, které informaci chápe jako posloupnost bitů s určitým významem. Moderní šifry transformují text tímto způsobem – prostý text je převeden na binární posloupnost, které v průběhu šifrování není přidělován žádný význam (pro zvýšení rychlosti může být tato posloupnost převedena na číslo v různých číselných soustavách).

Moderní metody šifrování lze rozdělit na dvě základní skupiny podle toho, zda se pro zašifrování a dešifrování využívá stejný nebo různý klíč. Pokud je pro šifrování i dešifrování použit tentýž klíč, jsou tyto metody nazývány jako **symetrické šifry** (*private-key ciphers*). Pokud je pro dešifrování použit jiný klíč než pro zašifrování, pak jsou tyto metody nazývány jako **asymetrické šifry** (*public-key ciphers*).

Symetrické šifry Jak již bylo zmíněno výše, princip těchto šifer spočívá v tom, že prostý text je transformován na posloupnost bitů (ty mohou nabývat hodnot buď 0 nebo 1) a pro zašifrování i dešifrování je použit pouze jeden klíč. Velikost tohoto klíče je pak závislá na použité šifře. Tyto šifry pohlíží na klíče podobně jako na vstupní informaci, tedy jako na posloupnost bitů, velikost klíče je pak délka této binární posloupnosti.

Symetrické šifry mohou s posloupností bitů pracovat dvojím způsobem – buď je tato posloupnost zpracovávána bit po bitu (**proudové šifry**) nebo je rozdělena na bloky pevné velikosti (**blokové šifry**) a teprve s těmito bloky jsou prováděny akce popsané v algoritmu šifry.

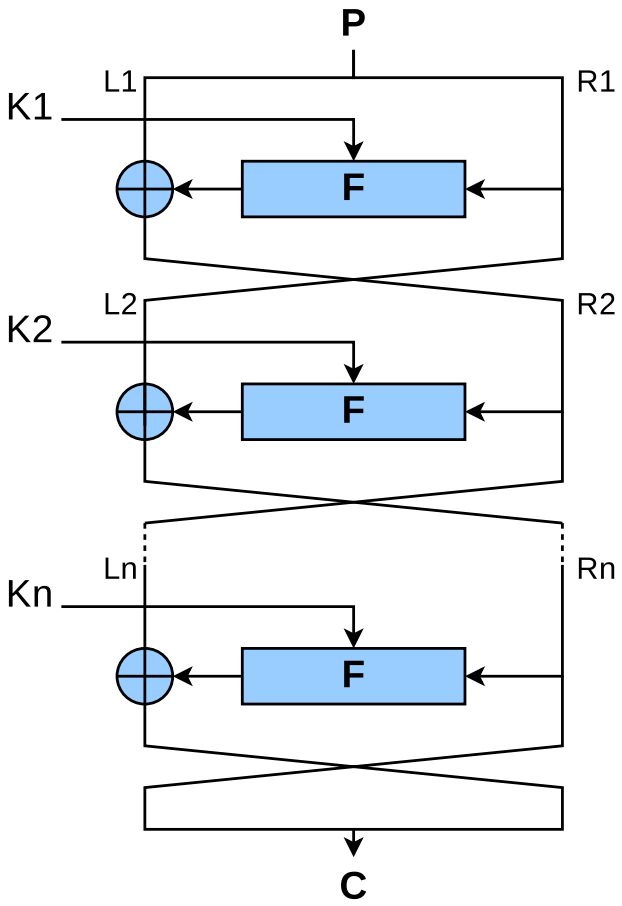
Proudové šifry Proudové šifry transformují vstupní posloupnost bitů ve výstupní posloupnost tak, že každý bit je pomocí předem specifikované funkce (většinou je používán logický exkluzívní součet `XOR`) kombinován s pseudonáhodným **proudem bitů** (*keystream*), ten je vytvářen ze znalosti šifrovacího klíče a algoritmu.

Tyto šifry jsou modifikacemi **Vernamovy šifry**, která jako první navrhuje transformaci prostého textu pomocí náhodně generovaného klíče. Vzhledem k vysoké výpočetní náročnosti generování zcela náhodného klíče práce přistupuje k vytváření pseudonáhodného klíče. Proudové šifry zajišťují vysokou úroveň zabezpečení právě díky generování (pseudo)náhodného klíče, ten je však nezbytně nutné bezpečně doručit všem komunikujícím stranám.

Blokové šifry Prostý text převedený na posloupnost bitů je v rámci blokové šifry převeden na bloky o pevné velikosti. Velikost těchto bloků je závislá na typu zvolené šifry. Po rozdělení prostého textu na bloky je každý blok za použití klíče a algoritmu specifického pro zvolenou šifru transformován ve výstupní blok, všechny šifrované bloky jsou spojeny a vytvoří šifrovaný text.

Tyto šifry využívají v algoritmu modifikovanou **Feistelovu síť**. Její princip spočívá v rozdělení vstupního bloku na poloviny a v n iteracích aplikace tzv. **Feistelovy funkce** na pravou polovinu bloku (viz obrázek 3.2). Po vypočtení nové hodnoty pravé poloviny bloku je levá a pravá polovina vyměněna. Feistelova funkce může být parametrizována klíčem závislým na iteraci.

Tyto šifry dlouhou dobu zajišťovaly velmi dobrou ochranu dat a informací, ale vzhledem ke konečnému počtu možných klíčů jsou tyto šifry náchylné k tzv. **útoku hrubou silou** (*brute-force attack*), kdy jsou postupně zkoušeny všechny klíče z prostoru možných klíčů. Se zvyšujícím se výpočetním výkonem počítačů tak tyto šifry přestávají zajišťovat potřebnou bezpečnost. Pro vyšší bezpečnost je potřeba zvětšit prostor možných klíčů (tzn. delší šifrovací klíče a delší pracovní bloky); toto opatření však klade vysoké nároky na výpočetní výkon počítačů, čímž se šifrování značně zpomalí a není efektivní.



Obrázek 3.2: Ukázka Feistelovy funkce (zdroj: vlastní tvorba dle [16]).

Asymetrické šifry Přestože nový pohled informaci jako posloupnost bitů byl zcela revoluční a zajišťoval velmi dobrou ochranu informací, neustále přetrvával problém, se kterým se lidé potýkali již od počátku – nutnost bezpečného předání tajného klíče všem komunikujícím stranám. S řešením tohoto problému přišli v roce 1976 Whitfield Diffie a Martin Hellman ve svém příspěvku [9] pro IEEE Transactions on Information Theory. Navrhovali **systém výměny klíčů** známý jako **Diffie–Hellman key exchange**.

Princip tohoto systému lze ilustrovat na poštovní zásilce a visacích zámcích. Nejprve odesílatel pověsí na zásilku svůj zámek a odešle zásilku příjemci. Ten na zásilku přidá svůj zámek a pošle ji zpět odesílateli. V tomto okamžiku jsou na zásilce zámky obou stran. Odesílatel ze zásilky sundá svůj zámek a naposledy ji odešle příjemci. Ten si sundá vlastní zámek a může si prohlížet obsah zásilky. [38]

Tento systém zaručil vysokou úroveň zabezpečení informace a zejména vyřešil problém bezpečného předání tajného klíče. V současné době hojně využívaná šifra **RSA** (text patentu [33]) využívá systému výměny klíčů Diffie-Hellman a patří mezi několik dosud neprolomených šifer.

3.2.2 Kódy

Termínem **kódy** jsou v kryptografii označovány metody, které se používají k transformaci zprávy do nesrozumitelné podoby. Cílem této transformace je zabránění porozumění obsahu zprávy osobám, které nevlastní nezbytnou informaci (např. kódovou knihu). Nejčastějším způsobem kódování je vytvoření a použití kódové knihy se seznamem běžně používaných slov a frází spolu s kódovým slovem. Zakódovaný **prostý text** bývá zpravidla označován jako **kódový text** (*codetext*).

Velmi často bývají pojmy *kódování* a *šifrování* zaměňovány a používány nesprávně. Proto je důležité tyto pojmy rozlišovat – kódy pracují s prostým textem na úrovni významu (tzn. slova, fráze či celé věty jsou transformovány v jiný text podle kódové knihy), kdežto šifry pracují na úrovni jednotlivých písmen (případně malých skupin písmen) či jednotlivých bitů u moderních šifer. Zde je zřejmá výhoda šifer oproti kódům – u šifer není nutné uchovávat rozsáhlý seznam kódových slov, což zvyšuje bezpečnost celého systému. Pokud šifra nebo kód vyžadují spoustu doprovodných informací, stává se méně bezpečná a více náchylná k prolomení.

Kapitola 4

Implementované šifry

V této kapitole jsou popsány všechny implementované šifry a kódy. Popis obsahuje **stručnou charakteristiku**, **algoritmus šifrování a dešifrování** (označen značkou f(x)), **pracovní klíče** ( ) a také očekávaný **vstupní** ( ) a **výstupní text** ( ). Pro popis formátu vstupního či výstupního textu a pracovních klíčů byla zvolena terminologie uvedená v tabulce 4.1.

termín	**popis**
Libovolné znaky.	Vstupem šifry mohou být všechny (ne)tisknutelné znaky.
Anglická abeceda [a bílé znaky] [a číslice] [bez {`znaky`}].	Ve textu jsou nejprve písmena národních abeced nahrazena ekvivalenty v anglické abecedě a poté jsou všechny nealfabetické znaky odstraněny. Pokud je uvedeno *bílé znaky*, všechny bílé znaky v textu jsou nahrazeny mezerami (více mezer za sebou je nahrazeno jednou mezerou). Pokud je uvedeno *číslice*, všechny číslice ve vstupním textu zůstávají. Pokud je uvedeno *bez* {***znaky***}, pak ze vstupního textu jsou odstraněny (případně nahrazeny jinými v závislosti na šifře) i znaky uvedené ve výčtu *znaky*.
Náhodná abeceda [a číslice].	Náhodně uspořádaná anglická abeceda. Pokud je uvedeno *číslice*, pak se při vytváření abecedy používají i číslice.
Klíčovaná abeceda [a číslice].	Podobně jako *náhodná abeceda*, ale nejprve jsou uvedeny všechny znaky klíčového slova (opakující se znaky jsou ignorovány) a pak následují všechna zbývající písmena v abecedním pořadí. Pokud je uvedeno *číslice*, pak se při vytváření abecedy používají i číslice, přičemž číslice jsou řazeny před písmeny.
Číselný klíč.	Celé číslo s libovolným počtem číslic, přičemž jednotlivé číslice se v číslu mohou vyskytovat vícekrát.
Unikátní číselný klíč.	Celé číslo, ve kterém je každá číslice nejvýše jedenkrát.
Hexadecimální klíč délky `délka`.	Klíče složený pouze z hexadecimálních číslic o délce `délka`.

Tabulka 4.1: Terminologie zvolená pro popis šifer.

Jelikož je tato práce zaměřena na studium kryptografických algoritmů řadících se mezi klasické metody (viz kapitola 3), převážná většina šifer má z dnešního pohledu jednoduchý

princip. Šifry jsou uvedeny v abecedním pořadí, avšak pokud je více šifer podobných či se jedná o varianty jedné šifry, jsou uvedeny pospolu (značka > před názvem šifry znamená, že princip popisované šifry vychází z výše uvedené šifry mající o jednu značku > méně).

Vzhledem k počtu popisovaných šifer je narušena klasifikace kryptografických algoritmů zavedená v kapitole 3. V této kapitole jsou proto algoritmy rozděleny do těchto kategorií – (1) klasické šifry, (2) rotující stroje, (3) moderní šifry a (4) kódy. Popis šifer v této kapitole vychází z [2, 3, 12, 26, 39].

4.1 Klasické šifry

Mezi klasické šifry se řadí všechny kryptografické algoritmy používané pro šifrování textu od samotných počátků kryptografie až po konec 2. světové války. Všechny tyto šifry pracují s prostým textem coby řetězcem znaků, kdežto moderní šifry (po 2. světové válce) pracují s prostým textem v binární podobě.

V následujícím textu budou rozlišeny základní skupiny klasických šifer – **substituční šifry** jsou označeny písmenem S, **transpoziční šifry** písmenem T a **kombinované šifry** jsou označeny oběma písmeny ST nebo TS.

Afinní šifra (*Affine cipher*) S

Afinní šifra patří mezi monoalfabetické substituční šifry, jelikož využívá pouze jedinou šifrovou abecedu (obecně délky m, při použití anglické abecedy je $m = 26$), která je určena dvěma číselnými parametry a a b. Speciálním případem této šifry je např. Caesarova šifra, šifra Atbaš nebo také šifra ROT-xx.

Každý znak prostého textu je převeden na odpovídající číselnou hodnotu ($a = 1, b = 2, ..., z = 26$). Tato hodnota je matematickou funkcí pro šifrování převedena na jinou hodnotu. Šifrovaný text vzniká nahrazením vypočtené hodnoty písmenem. Šifrovací funkce má tvar

$$E(x) = (ax + b) \mod m,$$

kde x je hodnota znaku, a je parametr udávající velikost kroku, b je parametr udávající posun abecedy vpravo a m je počet znaků v abecedě. Pro dešifrování platí inverzní funkce

$$D(x) = a^{-1}(x - b) \mod m,$$

kde a^{-1} je *multiplikativní inverze*[1] k a. Jelikož obě tyto funkce pracují v modulární aritmetice, není zaručeno, že obě rovnice jsou řešitelné pro libovolné parametry a a b. Aby byly obě rovnice řešitelné (tedy šifrování i dešifrování bylo proveditelné), je nutné, aby parametry a a m byla nesoudělná čísla[2].

Anglická abeceda.

[1] *„Multiplikativní inverze čísla x na tělese Z_p (p je prvočíslo) je takové číslo x^{-1}, pro které platí, že $xx^{-1} \equiv 1$. Inverze může existovat i pro čísla na Z_m (m je přirozené číslo), ale není to zaručeno. Pro zjištění její hodnoty se využívá dvou postupů. Prvním je hádání, případně zkoušení všech ostatních čísel. Tento postup je efektivní pro malá p, když je hádajícím člověk (je to často vidět). Druhým postupem je rozšířený Euklidův algoritmus.“* [25]

[2] Nesoudělná čísla mají právě jednoho společného dělitele – číslo 1.

Tato šifra má 3 číselné klíče, zadávají se však jen klíče a (krok) a b (posun vpravo), jelikož klíč m je dán počtem písmen v pracovní abecedě (při použití anglické abecedy je $m = 26$).

Anglická abeceda.

> Atbaš (*Atbash*) S

Šifra Atbaš patří mezi jedny z nejstarších kryptografických algoritmů, protože byla používána Židy již v 6. stol. př. n. l. Tuto šifru lze považovat za speciální případ afinní šifry s parametry $a = 25$ a $b = 25$.

Každý znak prostého textu je nahrazen znakem z anglické abecedy, které je od konce abecedy vzdáleno stejný počet znaků, jako je nahrazovaný znak vzdálen od počátku anglické abecedy (např. písmeno `a` je nahrazeno písmenem `z` či písmeno `c` je nahrazeno písmenem `x`).

Anglická abeceda.

Tato šifra nemá klíč. Pokud je uvažována jako speciální případ afinní šifry, pak má parametry $a = 25$, $b = 25$ a $m = 26$.

Anglická abeceda.

> Caesarova šifra S

Tato šifra také patří mezi jedny z nejstarších, protože ji vymyslel a s úspěchem používal Julius Caesar v 1. stol. př. n. l. Z pohledu novějších šifer ji lze považovat za speciální případ afinní šifry s parametry $a = 1$, $b = 3$ a $m = 26$.

Každé písmeno prostého textu je nahrazeno písmenem, které se nachází v abecedě o 3 písmena vpravo (např. písmeno `A` je nahrazeno písmenem `D`).

Anglická abeceda.

Tato šifra nemá klíč. Pokud je uvažována jako speciální případ afinní šifry, pak má parametry $a = 1$, $b = 3$ a $m = 26$.

Anglická abeceda.

>> Klíčovaná Caesarova šifra (*Keyed Caesar cipher*) S

Princip této šifry je zcela totožný s Caesarovou šifrou. Na rozdíl od Caesarovy šifry však pracuje s klíčovanou abecedou, která je vytvořena pomocí zadaného klíčového slova.

>> ROT-xx S

Princip této šifry je totožný s Caesarovou šifrou – jedná se o její obecnou podobu, která používá posun abecedy o xx písmen. Posun je zadán jako číselný klíč, který se také projeví v názvu (např. posun o 13 vytváří šifru ROT-13).

>>> Kámasútra S

Tato šifra patří mezi jedny ze starověkých šifer (4. až 6. stol. n. l.) a je uvedena v knize Kámasútra. Princip spočívá ve vytvoření náhodné abecedy, kde 1. písmeno je spárováno s 14. písmenem, 2. písmeno s 15. písmenem až 13. písmeno je spárováno s 26. písmenem.

Tuto šifru lze také považovat za variantu šifry ROT-13, na rozdíl od šifry ROT-13 používá tato šifra zcela náhodnou abecedu.

- Znaky prostého textu jsou nahrazeny písmenem v páru (např. písmeno `A` je nahrazeno písmenem `T` a naopak).
- Anglická abeceda.
- Náhodná abeceda (prvních 13 písmen je spárováno s následujícími 13 písmeny).
- Anglická abeceda.

Baconova šifra (*Baconian cipher*) S

Princip této šifry spočívá v substituci znaků prostého textu předem danými skupinami písmen `A` a `B`.

- Každé písmeno prostého textu je nahrazeno pětimístnou kombinací písmen `A` a `B`. Substituční kombinace písmen `A` a `B` jsou dány tímto schématem – písmenu `A` odpovídá kombinace `AAAAA`, písmenu `B` odpovídá `AAAAB`, písmenu `C` odpovídá `AAABA`, atd.
- Anglická abeceda.
- Tato šifra nemá klíč.
- Text délky dělitelné 5 složený z písmen z množiny $\{A, B\}$.

Čínská šifra (Chinese cipher) T

Čínská šifra patří mezi jednoduché transpoziční metody, protože zachovává všechny znaky prostého textu, pouze mění jejich pořadí. Princip spočívá v přepisu prostého textu do matice o n řádcích (viz tabulka 4.2), kde n je zadaný číselný klíč.

```
vstupní text:  UKÁZKA ČÍNSKÉ ŠIFRY
číselný klíč:  4
výstupní text: FIÍČURŠN KY SAÁ ÉKKZ
```

F	I	Í	Č	U
R	Š	N		K
Y		S	A	Á
	É	K	K	Z

Tabulka 4.2: Ukázka použití Čínské šifry.


Prostý text je přepsán do sloupců o n řádcích, kde n je zadaný číselný klíč. Každý nově vytvořený sloupec se připojí **před** naposledy vytvořený sloupec. Pořadí znaků ve sloupci se střídá – do prvního sloupce jsou znaky zapsány shora dolů, do druhého sloupce zdola nahoru, do třetího opět shora dolů, atd. Po přepisu celého prostého textu do těchto sloupců je šifrovaný text vytvořen složením jednotlivých řádků.


Libovolné znaky.


Číselný klíč udávající počet řádků.


Libovolné znaky.

> Podle plotu (*Rail fence*) T

Princip této transpoziční šifry je založen na přepsání prostého textu na více řádků. Zašifrovaný text vznikne spojením těchto řádků. [38]

Tato šifra je variantou čínské šifry, liší se pouze ve směru přepisu textu – šifra podle plotu text zapisuje směrem doprava, kdežto čínská šifra směrem doleva.


Prostý text je přepsán na n řádků, kde n je zadaný číselný klíč. Na každý řádek je zapsán pouze jeden znak, poté se přechází na další řádek. Začíná se na prvním řádku a pokračuje se směrem dolů, jakmile se zapisuje na poslední řádek, změní se směr a pokračuje se směrem k prvnímu řádku.


Libovolné znaky.


Číselný klíč udávající počet řádků.


Libovolné znaky.

Číslo písmene (*Letter number*) S

Jednoduchá substituční šifra nahrazující znaky prostého textu čísly.


Každé písmeno prostého textu je nahrazeno dvouciferným číslem udávajícím jeho pozici v uspořádané anglické abecedě. Písmeno `A` je nahrazeno číslem 01.


Anglická abeceda.


Tato šifra nemá klíč.


Čísla $\{01, 02, ..., 26\}$ spojená `-`.

Digraf (*Digraph*) S

Jak sám název napovídá, šifra pracuje s digrafy, což jsou skupiny 2 písmen (obecně různých znaků). Princip šifry spočívá v nahrazení digrafů v prostém textu jinými digrafy, jež jsou dány tabulkou. Jelikož anglická abeceda má 26 písmen, existuje celkem $26 \times 26 = 676$ různých digrafů.


Prostý text je rozdělen na skupiny po 2 písmenech – první písmeno udává sloupec a druhý znak udává řádek v tabulce digrafů. Digraf ležící na souřadnicích daných dvojicí písmen z prostého textu je zapsán do šifrovaného textu. Pokud je délka prostého textu lichá, pak je na konec doplněno písmeno `X`.


Anglická abeceda.


Dvě písmena určující digraf na pozici $[A, A]$.


Anglická abeceda.

Monoalfabetická šifra (*Monoalphabetic cipher*)

S

Základní typ substitučních šifer využívající náhodnou abecedu, která určuje způsob nahrazování znaků prostého textu.


Prostý text je znak po znaku nahrazen písmenem z klíčované abecedy na stejné pozici, jako je pozice nahrazovaného znaku v uspořádané anglické abecedě.


Anglická abeceda.


Náhodná abeceda.


Anglická abeceda.

> Klíčové slovo (*Keyword*)

S

Substituční šifra využívající klíčovanou abecedu.

Tuto šifru lze také považovat za variantu monoalfabetické šifry, na rozdíl od šifry monoalfabetické šifry používá tato šifra klíčovanou abecedu místo náhodné abecedy.


Prostý text je znak po znaku nahrazen písmenem z klíčované abecedy na stejné pozici, jako je pozice nahrazovaného znaku v uspořádané anglické abecedě.


Anglická abeceda.


Klíčové slovo, na jehož základě je vytvořena klíčovaná abeceda.


Anglická abeceda.

Napoleonova šifra

S

Napoleonova šifra se řadí mezi polyalfabetické substituční šifry, proto není možné použít frekvenční analýzu pro její rozluštění. Princip spočívá ve vyhledání šifrovaného znaku v tzv. *Napoleonově tabulce* na základě znaku prostého textu a znaku klíčového slova.

Při konstrukci Napoleonovy tabulky se nejprve vytvoří dvojice po sobě jdoucích písmen (`A - B, C - D, ..., Y - Z`) tvořící hlavičku tabulky; prvnímu písmenu ve dvojici je přiřazena první polovina uspořádané anglické abecedy a druhému písmenu je přiřazena druhá polovina anglické abecedy zrotována vlevo o tolik znaků, jaký je index dvojice v hlavičce tabulky (dvojice `A - B` má index 0, dvojice `C - D` má index 1, atd.).


Každý znak prostého textu je nahrazen znakem nalezeným v Napoleonově tabulce. Vyhledání šifrovaného znaku v tabulce probíhá tímto způsobem – (1) je vyhledána dvojice v hlavičce tabulky obsahující odpovídající znak klíčového slova, (2) v obou polovinách abecedy je vyhledán znak prostého textu a (3) znak prostého textu je nahrazen znakem v druhé polovině na stejném indexu.


Anglická abeceda.

Klíčové slovo.

Anglická abeceda.

Permutační šifra (*Permutation cipher*) T

Způsob přeskupování znaků je u této transpoziční šifry dán permutačním číselným klíčem. Klíč obsahuje každou číslici menší než délka klíče právě jedenkrát (platný klíč je např. 3012, kdežto neplatný klíč je např. 4012).

Prostý text je rozdělen na skupiny o n znacích, kde n je délka číselného permutačního klíče. V každé skupině jsou pak znaky přeskupeny podle schématu daného číselným klíčem (skupina `ŠIFRA` je podle klíče 20413 zašifrována jako `FŠAIR`).

Libovolné znaky.

Číselný klíč, v němž každá číslice menší než délka klíče je právě jedenkrát.

Libovolné znaky.

Polybiův čtverec (*Polybius square*) S

Ve 2. století před naším letopočtem vymyslel řecký historik a politik Polybios jednoduchý, ale velmi účinný (s ohledem na tehdejší metody) způsob šifrování zpráv. Šifra spočívá ve vytvoření klíčované abecedy bez $\{J\}$, která je vepsána do čtvercové tabulky (viz tabulka 4.3) o rozměrech 5×5. Každé písmeno v abecedě (resp. tabulce) je možné indexovat pomocí dvou čísel z množiny $\{1, 2, ..., 5\}$.

```
klíčové slovo:          polybiův čtverec
upravené klíčové slovo: polybiuvcter
šifrová abeceda:        POLYBIUVCTERADFGHKMNQSWXZ
```

	1	2	3	4	5
1	P	O	L	Y	B
2	I	U	V	C	T
3	E	R	A	D	F
4	G	H	K	M	N
5	Q	S	W	X	Z

Tabulka 4.3: Ukázka vytvoření Polybiova čtverce.

Každé písmeno prostého textu je nahrazeno dvojicí čísel z množiny $\{1, 2, ..., 5\}$ – první číslice určuje řádek a druhá číslice určuje sloupec v Polybiově čtverci. Na místě daném těmito souřadnicemi leží šifrovaný znak.

Anglická abeceda bez $\{J\}$.

Klíčové slovo bez $\{J\}$.

Dvojice čísel $\{1, 2, ..., 5\}$. Počet dvojic je stejný jako počet písmen v prostém textu.

> ADFGVX

ST

Tato šifra byla používána německou armádou v 1. světové válce. Vychází z principu Polybiova čtverce – do čtverce o rozměrech 6×6 je vepsána náhodná abeceda a číslice (případně klíčovaná abeceda a číslice). Řádky i sloupce tohoto čtverce nejsou indexovány čísly 1 – 6, ale postupně písmeny $\{A, D, F, G, V, X\}$. Existuje i varianta ADFGV využívající Polybiův čtverec o rozměrech 5×5 (pak je použita náhodná abeceda bez $\{J\}$ nebo $\{Q\}$). [38]

f(x) Každý znak prostého textu je převeden na dvojici písmen z množiny $\{A, D, F, G, V, X\}$. První písmeno v této dvojici určuje řádek v Polybiově čtverci a druhé písmeno určuje sloupec. Převedený prostý text má proto sudou délku a musí obsahovat jen tyto znaky.

Transpoziční část této šifry spočívá ve vytvoření matice s již upraveným textem. Tato matice má tolik sloupců, kolik je znaků klíčového slova. Sloupce jsou postupně indexovány znaky klíčového slova. Upravený text je vepsán do této matice a poté je matice přečtena po sloupcích. Pořadí sloupců je dáno jejich abecedním pořadí (pokud existuje více sloupců označených stejným znakem, pak rozhoduje jeho pozice).

Anglická abeceda a číslice.

Pracovním klíčem je náhodná abeceda nutná pro sestavení Polybiova čtverce (variantou může být pouze klíčové slovo pro vytvoření klíčované abecedy). Druhé klíčové slovo slouží pro transpoziční část šifry.

Text sudé délky složený z písmen z množiny $\{A, D, F, G, V, X\}$.

> Bifid

ST

Šifra Bifid využívá Polybiův čtverec, ve kterém je uložena náhodná abeceda bez $\{J\}$. Využívají se jak substituční, tak transpoziční principy šifrování.

f(x) Prostý text je znak po znaku převáděn na dvojice čísel. Tato dvojice má význam souřadnice v Polybiově čtverci – první číslice určuje řádek a druhá číslice sloupec v Polybiově čtverci. Po převedení všech znaků na dvojice čísel jsou nejprve spojeny všechny řádkové indexy (tj. první z dvojice) a poté sloupcové indexy (tj. druhé ve dvojici). Tato posloupnost je rozdělena na nové dvojice, které jsou převedeny na písmena podle používaného Polybiova čtverce.

Anglická abeceda bez $\{J\}$. Písmeno `J` je nahrazeno písmenem `I`.

Náhodná abeceda bez $\{J\}$.

Anglická abeceda bez $\{J\}$.

>> Trifid

ST

Princip této šifry je podobný šifře Bifid. Šifra Trifid však pracuje se třemi Polybiovými čtverci o rozměrech 3×3 – je použita celá anglická abeceda a znak `'.'`. Každý znak prostého textu je tak zakódován do trojmístného čísla, kde první číslice udává Polybiův čtverec, druhá udává sloupec a třetí řádek v tomto čtverci. Po převedení všech znaků prostého textu na tato trojčíslí jsou nejprve zapsány všechny indexy čtverců, poté následují indexy sloupců a nakonec indexy řádků. Tento řetězec číslic je pak rozdělen na nová trojčíslí, která jsou na základě čtverců převedena na písmena.

> Nihilist

S

Tato šifra byla používána odpůrci carského režimu ve 2. polovině 19. stol. v Rusku; název šifry je odvozen od názvu jejich hnutí. Šifra využívá Polybiův čtverec s vepsanou klíčovanou abecedou a klíčové slovo pro zašifrování textu. Jak klíčovaná abeceda, tak klíčové slovo neobsahují písmeno `J` (každý výskyt je nahrazen písmenem `I`).

f(x) Znaky prostého textu jsou převedeny na dvojici čísel – první číslice udává řádek a druhá číslice udává sloupec v Polybiově čtverci s klíčovanou abecedou. Klíčové slovo je také převedeno na tento číselný zápis. Znaky prostého textu jsou pak sčítány se znaky klíčového slova; vzniknou tak dvouciferná nebo tříciferná čísla.

Anglická abeceda bez $\{J\}$.

Klíčové slovo pro sestavení klíčované abecedy bez $\{J\}$ a klíčové slovo pro šifrování.

Posloupnost dvouciferných nebo tříciferných čísel.

> Playfair

S

Tato substituční šifra je pojmenovaná po svém autorovi Lordu Playfair, který ji vymyslel v roce 1854. Jedná se o vůbec první digrafovou šifru. Její princip využívá Polybiův čtverec s vepsanou klíčovanou abecedou.

f(x) Prostý text je rozdělen na dvojice písmen a tyto dvojice jsou šifrovány podle těchto pravidel:

1. Pokud jsou dvě následující písmena stejná, je mezi ně vloženo písmeno `X`.
2. Pokud obě písmena leží na stejném řádku v tabulce, každé z nich je nahrazeno písmenem ležícím o jednu pozici vpravo (cyklicky přes tentýž řádek tabulky).
3. Pokud obě písmena leží ve stejném sloupci v tabulce, každé z nich je nahrazeno písmenem ležícím o jednu pozici dolů (cyklicky přes tentýž sloupec tabulky).
4. Pokud písmena neleží ve stejném řádku či sloupci, je první písmeno nahrazeno písmenem, které leží na stejném řádku jako první písmeno a ve stejném sloupci jako druhé písmeno; druhé písmeno dvojice je nahrazeno písmenem, které leží ve stejném sloupci jako první písmeno a na stejném řádku jako druhé písmeno.

Anglická abeceda bez $\{J\}$.

Klíčové slovo bez $\{J\}$.

Anglická abeceda bez $\{J\}$.

>> Čtyři čtverce (*Four-square cipher*)

S

Princip této šifry je založen na digrafech (prostý text je rozdělen na dvojice znaků a tyto dvojice jsou pak zašifrovány). Šifra využívá 4 čtverců o rozměrech 5×5 (viz tabulka 4.4), přičemž ve čtvercích 1 a 4 jsou vepsány anglické abecedy bez $\{J\}$ a ve čtvercích 2 a 3 jsou vepsány klíčované abecedy bez $\{J\}$ (každá klíčovaná abeceda je sestrojena z jiného klíčového slova).

Výběr nové dvojice písmen je velmi podobný jako u šifry Playfair.

1	2
3	4

Tabulka 4.4: Uspořádání pracovních abeced pro šifru čtyř čtverců.

f(x) Každá dvojice prostého textu je nahrazena jinou dvojicí. Ve čtverci 1 se vyhledá první písmeno dvojice, ve čtverci 4 se vyhledá druhé písmeno. První znak zašifrované dvojice je nalezen ve čtverci 2 na stejném řádku, jako je řádek prvního písmene ve čtverci 1, a ve stejném sloupci, jako je sloupec druhého písmene ve čtverci 4. Druhý znak je nahrazen písmenem, které je ve čtverci 3 ve stejném sloupci, jako je první znak ve čtverci 1, a na stejném řádku, jako je druhý znak ve čtverci 4.

Anglická abeceda bez $\{J\}$.

Dvě klíčová slova.

Anglická abeceda bez $\{J\}$ sudé délky.

>> Dva čtverce (*Two square cipher*) S

Princip této šifry je velmi podobný šifře čtyři čtverce. Na rozdíl od šifry čtyři čtverce používá jen 2 Polybiovy čtverce (tyto čtverce se zapisují nad sebe).

Výběr nové dvojice písmen je velmi podobný jako u šifry Playfair.

f(x) Vstupní text je rozdělen na dvojice znaků. Každá dvojice je pak zašifrována na dvojici písmen – první znak je převeden na písmeno, které leží v horním čtverci na stejném řádku jako první znak a ve stejném sloupci jako druhý znak ve spodním čtverci, druhý znak je pak převeden na písmeno, které leží ve spodním čtverci na stejném řádku jako druhý znak a ve stejném sloupci jako první znak v horním čtverci.

Anglická abeceda bez $\{J\}$.

Dvě klíčová slova.

Anglická abeceda bez $\{J\}$ sudé délky.

Skoková šifra (*Skip cipher*) T

Tato transpoziční šifra provádí 2 transpozice za sebou – při první transpozici je prostý text zrotován o zadaný počet znaků a při druhé transpozici jsou postupně cyklicky vybírány znaky vzdálené o zadaný krok.

f(x) Prostý text je nejprve zrotován o n míst vpravo. Poté je proveden cyklický výběr písmen (vybraná písmena jsou zapsána do šifrovaného textu). Na velikost kroku je kladena podmínka, že nesmí být dělitelem délky prostého textu a ani nesmí být násobkem dělitelů délky prostého textu.

Libovolné znaky.

První číselný klíč udává počet znaků pro rotaci. Druhý číselný klíč udává velikost kroku pro výběr znaků.

Libovolné znaky.

Sloupcová transpozice (*Columnar transposition*) T

Jedná se základní typ transpozičních metod klasických šifer. Tato šifra zachovává všechny znaky prostého textu, jen mění jejich pořadí.

f(x) Prostý text je přepsán do tabulky, jejíž sloupce jsou pojmenovány postupně znaky klíčového slova. Počet sloupců této tabulky je tedy shodný s počtem znaků v klíčovém slovu. Po přepsání celého prostého textu do tabulky je tato tabulka přečtena sloupec po sloupci. Pořadí sloupců je určeno abecedním pořadím sloupců (pokud se v klíčovém slovu vyskytuje jedno písmeno vícekrát, rozhodující roli hraje pozice v klíčovém slovu).

Libovolné znaky.

Klíčové slovo, na jehož základě se sestaví tabulka.

Libovolné znaky.

> AMSCO T

Tato šifra patří mezi čistě transpoziční metody (tedy znaky prostého textu jsou zachovány, pouze se změní jejich pořadí).

f(x) Prostý text je zapsán do matice o tolika sloupcích, kolik cifer má číselný klíč, střídavě po 1 a 2 znacích. Po přepsání celého prostého textu do matice je tato matice přečtena sloupec po sloupci. Pořadí sloupců je určeno vzrůstající hodnotou indexu sloupce určeného číselným klíčem.

Libovolné znaky.

Unikátní číselný klíč.

Libovolné znaky.

> Dvojitá sloupcová transpozice (*Double columnar transposition*) T

Tato šifra pracuje jako dvě jednoduché sloupcové transpozice za sebou, přičemž každá využívá jiné klíčové slovo. Po provedení první sloupové transpozice s prvním klíčem je šifrovaný text znovu zašifrován sloupcovou transpozicí s druhým klíčem.

> Skytale T

Tuto transpoziční šifru používali již Sparťané ve 3. stol. p. n. l. pro ukrytí zpráv. Šifra využívá n-stěnný hranol, kolem kterého je omotán pruh jakéhokoliv popisovatelného materiálu. Prostý text je zapisován na stěny hranolu. Při odmotání pruhu vznikne zašifrovaná zpráva.

f(x) Prostý text je zapisován na stěny hranolu. Při odmotání pruhu popisovatelného materiálu jsou znaky prostého textu přeskupeny.

Libovolné znaky.

Číselný klíč udávající počet stěn hranolu.

Libovolné znaky.

> U̇bchi

T

Tato transpoziční šifra byla používána německou armádou v 1. světové válce. Její princip spočívá v několikanásobném zapsání textu do tabulky, jejíž sloupce jsou označeny písmeny klíčového slova.

- Do tabulky o tolika sloupcích, kolik znaků má klíčové slovo, je vepsán prostý text. V prvním kroku je vytvořen pomocný text přepisem sloupců tabulky v abecedním pořadí (v případě vícenásobného výskytu stejného znaku hraje rozhodující roli pozice v klíčovém slovu). Tento text je znovu přepsán do tabulky. V dalším kroku je na konec přidáno písmeno `Z` tolikrát, kolik slov má klíčová fráze. Šifrovaná zpráva vznikne přepisem sloupců tabulky v abecedním pořadí.
- Libovolné znaky.
- Klíčové slovo či fráze.
- Libovolné znaky.

Vigenèrova šifra

S

Jedná se o polyalfabetickou substituční šifru pracující s tzv. *Vigenèrovým čtvercem*. Tento čtverec (viz tabulka 4.5) obsahuje 26 sloupců a 26 řádků (na každém řádku je zapsána celá uspořádaná abeceda a abeceda na následujícím řádku je vždy zrotována o 1 písmeno vlevo). Díky použití více abeced je tato šifra odolná proti frekvenční analýze, protože stejný znak prostého textu může být zašifrován do různých písmen.

- Prostý text je znak po znaku zašifrován za použití klíčového slova. Nad prostý text je zapsáno klíčové slovo cyklicky se opakující nad celou délkou prostého textu. Zašifrování znaku spočívá v nalezení písmene ležícího na průsečíku sloupce, který je dán znakem prostého textu, a řádku daného příslušícím písmenem klíčového slova.
- Anglická abeceda.
- Klíčové slovo.
- Anglická abeceda.

> Autoklíč (*Autokey*)

S

Šifra Autoklíč je variantou Vigenèrovy šifry. Jako klíč je použito jak klíčové slovo, tak samotný prostý text.

- Princip šifrování je vysvětlen u Vigenèrovy šifry. Klíč je u této šifry tvořen zadaným klíčovým slovem a n znaky prostého textu, kde n je rozdíl mezi délkou prostého textu a délky klíčového slova.
- Anglická abeceda.
- Klíčové slovo.
- Anglická abeceda.

	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

Tabulka 4.5: Vigenèrův čtverec.

> Beaufortova šifra

S

Princip této šifry je založen na Vigenèrově šifře. Proces šifrování i dešifrování je totožný, liší se jen pracovní abecedy (resp. uspořádání Vigenèrova čtverce). Vigenèrův čtverec u této šifry je zkonstruován tak, že první sloupec je `A - Z`, druhý sloupec je `Z - Y` namísto `B - A`, třetí sloupec je `Y - X` namísto `C - B`, atd.

> Gronsfeld

S

Princip této šifry je založen na Vigenèrově šifře. Na rozdíl od Vigenèrovy šifry využívá šifra Gronsfeld klíčovanou abecedu. Také se nevyužívá klíčové slovo pro šifrování, ale číselný klíč. Číslice v tomto klíči odpovídají postupně písmenům anglické abecedy ($a = 0, b = 1, ..., j = 9$). Šifrování je pak totožné jako u Vigenèrově šifry.

>> Klíčovaná Vigenèrova šifra (*Keyed Vigenère cipher*)

S

Princip této šifry je zcela totožný s šifrou Gronsfeld. Na rozdíl od šifry Gronsfeld však pracuje s klíčovým slovem pro šifrování namísto číselného klíče.

> Larrabee

S

Substituční šifra vycházející z principu Vigenèrovy šifry. Proces šifrování i dešifrovaní je totožný, tato šifra přidává možnost šifrování čísel (viz [24]).

f(x) Nejprve jsou všechna čísla převedena na řetězec začínající písmenem `Q`, za tímto písmenem následuje jedno písmeno z množiny $\{A = 1, B = 2, ..., I = 9\}$ udávající počet číslic. Následují převedené číslice ($\{A = 1, B = 2, ..., J = 0\}$). Po převedení všech čísel v prostém textu je text zašifrován Vigenèrovou šifrou.

Anglická abeceda a číslice.

Klíčové slovo.

Anglická abeceda.

4.2 Rotující stroje

Šifry klasifikované coby rotující stroje jsou ve valné většině polyalfabetické (např. Enigma), proto je lze považovat modifikace Vigenerovy šifry se specifickými počátečními nastaveními. V implementované knihovně algoritmů pro šifrování textu z tohoto důvodu nebyla implementována žádná šifra spadající do této kategorie. Tyto šifry jsou velmi komplexní a rozsahem přesahují tuto práci.

4.3 Moderní šifry

Mezi moderní kryptografické metody jsou řazeny všechny šifry pracující s prostým textem coby posloupností jednotlivých bitů (na rozdíl od klasických metod, které s prostým textem pracují coby řetězcem). Podle práce s posloupností bitů se dělí na (1) blokové a (2) proudové (viz kapitola 3.2.1).

Z množství dnes používaných symetrických či asymetrických algoritmů bylo implementováno jen několik, aby byl ilustrován postupný vývoj šifer. Oblast moderní kryptografie však zasahuje mimo rozsah této práce. Vybrané šifry jsou (nebo byly) standardy pro šifrování, případně byly kandidáty na tyto standardy.

Blowfish

Šifra Blowfish (viz [37]) byla v roce 1993 navržena Bruce Schneierem jako volně dostupná alternativa k standardizované šifře DES. Jedná se o blokovou šifru s délkou bloku 64 bitů, ale s velikostí klíče od 1 do 448 bitů. S ohledem na velikost klíče a množinu možných klíčů nabízí šifra Blowfish mnohem vyšší úroveň zabezpečení než šifra DES. Vzhledem k velikosti prostoru klíčů zatím nebyl proveden žádný úspěšný kryptoanalytický útok na tuto šifru.

Šifra (popis algoritmu viz [5]) se skládá ze dvou částí – v první části je vytvořen rozsáhlý **plánovač klíčů** (*key schedule*), který je závislý na zadaném vstupním klíči, a v druhé části je zpráva zašifrována. Šifra je další variantou **Feistelovy šifry** (resp. využívá Feistelovu funkci) v 16 iteracích.

f(x) V první části je na základě klíče vytvořen rozsáhlý plánovač klíčů – tzv. pole `P` (18 hodnot) a pole `S-box` ($4 \times 256 = 1024$ hodnot). Plánovač celkem obsahuje $18 + 1024 =$

1032 hodnot. Výpočet těchto klíčů je časově značně náročný, proto je možné plánovač jednorázově vypočítat při prvním použití šifry a pak vždy používat stejný klíč.

V další části je vstupní text po blocích šifrován. Jakmile je každý blok zašifrován, jsou všechny tyto zašifrované bloky spojeny a vytvoří tak zašifrovanou zprávu.

1. Blok je rozdělen na levou (horních 32 bitů) a pravou část (spodních 32 bitů).
2. Nová levá část je vypočtena ze stávající levé části a odpovídajícího klíče z pole `P` (vybírá se podle indexu iterace) pomocí logické funkce `XOR` (v rovnici je tato funkce značena jako $\oplus$).

$$x_L = x_L \oplus P_i \tag{4.1}$$

3. Nová pravá část je vypočtena z hodnoty, kterou vrátí Feistelova funkce, jejíž argumentem je nově vypočtená levá strana, a ze stávající pravé strany. Feistelova funkce využívá předem vypočtené pole `S-box` a její výpočet probíhá následovně:
   (a) Nejprve je 32 bitů dlouhý vstup rozdělen na 4 stejně dlouhé části a, b, c, d, které jsou převedeny na decimální číslo sloužící jako index do pole `S-box`.
   (b) Poté je vypočtena nová hodnota levé strany za použití pole `S-box`:

$$F(x_L) = S_1[a] + S_2[b] \mod 2^{32} \tag{4.2}$$
$$F(x_L) = F(x_L) \oplus S_3[c] \tag{4.3}$$
$$F(x_L) = F(x_L) + S_4[d] \mod 2^{32} \tag{4.4}$$

4. Po obou těchto výpočtech jsou hodnoty pravé a levé strany vyměněny.
5. Krok 2 je prováděn v 16 iteracích. Poté je znovu vyměněna pravá a levá strana (tzn. zrušení výměny z poslední iterace). Pravá i levá strana jsou pak exkluzivně sečteny (tzn. logická operace `XOR`) se zbývajícími hodnotami v poli `P`. Poté jsou levá a pravá strana spojeny a je vytvořen nový zašifrovaný blok.

$$x_L, x_R = x_R, x_L \tag{4.5}$$
$$x_R = x_R \oplus P_{17} \tag{4.6}$$
$$x_L = x_L \oplus P_{18} \tag{4.7}$$


Libovolné znaky převedené do binární reprezentace.


Hexadecimální klíč délky 1 – 112.

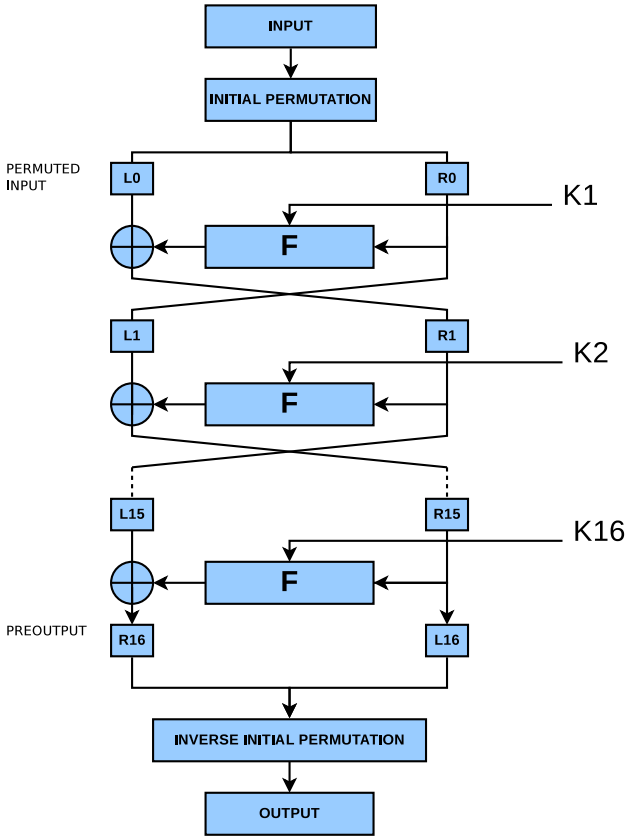

Hexadecimální zápis zašifrovaných binární dat.

DES

DES šifra (***D**ata **E**ncryption **S**tandard*) je bloková šifra, která byla v 70. letech vybrána NBS (National Bureau of Standards) jako standard pro šifrování dat v civilních státních organizacích v USA (viz [8]). Následně se rozšířila i do soukromého sektoru.

V dnešní době je tato šifra považována za nespolehlivou, protože využívá jen 64 bitový klíč, z něhož je pouze 56 efektivních. Proto je možné tuto šifru prolomit **útokem hrubou silou** (*brute-force attack*) v řádech hodin. Hlavním nedostatkem této šifry je nutná znalost tajného klíče oběma komunikujícími stranami (tedy její bezpečná distribuce) a také velikost množiny možných klíčů (doporučené velikosti klíčů viz [14]).

Šifra DES pracuje s 64 bitů dlouhými bloky (tzn. prostý text je převeden do binární podoby a rozdělen na bloky této velikosti; pokud poslední blok není 64 bitů dlouhý, je doplněn potřebným počtem binárních 0). Každý blok je zašifrován podle schématu na obrázku 4.1; všechny zašifrované bloky jsou pak spojeny a převedeny do hexadecimální podoby.



Obrázek 4.1: Schéma zašifrování jednoho bloku šifrou DES (zdroj: vlastní tvorba dle [8]).

Každý 64 bitů dlouhý blok je zašifrován v těchto krocích:

1. Nejprve je blok upraven tzv. **počáteční permutací** (*initial permutation*), což je standardem specifikovaná funkce, která v bloku zamění pořadí jednotlivých bitů (provede se jednoduchá permutační šifra).
2. Poté je blok rozdělen na levou část a pravou část. Levá část obsahuje 32 nejvyšších bitů, pravá část naopak 32 nejnižších bitů.
3. Poté je vytvořena nová levá část pouhým nahrazením stávající pravou. Nová pravá část je vypočtena ze stávající levé i pravé části podle vztahu:

$$L_{n+1} = R_n \tag{4.8}$$
$$R_{n+1} = L_n \oplus f(R_n, K_{n+1}) \tag{4.9}$$

V tomto vztahu figuruje funkce f, což je Feistelova funkce. Jejími argumenty je stávající pravá strana a klíč, který je vypočten v **plánovači klíče** (*key schedule*) z aktuální iterace (plánovač klíče pracuje se zadaným vstupním klíčem a pro

každou iteraci vygeneruje jiný klíč podle předem specifikovaného schématu). Feistelova funkce vytváří z aktuální pravé strany a klíče nový 32 bitů dlouhý blok v těchto krocích:

(a) **Expanze** (*expansion*) – 32 bitů dlouhý vstup pravé strany je rozšířen na 48 bitů podle funkce specifikované ve standardu.

(b) **Smíchání s klíčem** (*key mixing*) – nový 48 bitů dlouhý blok je exkluzivně sečten s klíčem, který je vypočten v plánovači klíčů podle aktuální iterace šifry DES.

(c) **Nahrazení** (*substitution*) – vzniklý 48 bitů dlouhý blok je rozdělen na 6 bitů dlouhé bloky, které jsou pomocí pole `S-box` převedeny na 4 bitové bloky. Funkce pole `S-box` je specifikována ve standardu. Poté jsou tyto bloky spojeny.

(d) **Permutace** (*permutation*) – v posledním kroku je 32 bitů dlouhý blok zamíchán, takže se změní podle předem specifikované funkce pořadí jednotlivých bitů.

4. Provede se 16 iterací kroku 3.
5. V závěrečném kroku se pravý a levý blok spojí a znovu se upraví tzv. inverzí permutační funkcí (*inverse initial permutation*), tedy pořadí bitů v bloku je změněno.


Libovolné znaky převedené do binární reprezentace.


Hexadecimální klíč délky 16.

Hexadecimální zápis zašifrovaných binární dat.

TDEA

Šifra TDEA (viz [4]) (***T**riple **D**ata **E**ncryption **A**lgorithm*), známá také jako 3DES, TDES či Triple-DES, aplikuje za sebou 3x šifru DES. Délka klíče se proto rozšíří až na $3 \times 64 = 192$ bitů. Existují tak varianty TDEA šifry podle zadaných klíčů:

1. Všechny tři klíče jsou nezávislé ($K_1 \neq K_2 \neq K_3$).
2. Klíče `K1` a `K2` jsou nezávislé ($K_1 \neq K_2$), ale `K1` a `K3` jsou stejné ($K_1 = K_3$).
3. Všechny tři klíče jsou identické ($K_1 = K_2 = K_3$) – tj. klasická šifra DES.

XOR

XOR šifru lze považovat za základ všech moderních šifer, protože interpretuje vstupní prostý text jako posloupnost 1 a 0 a tuto posloupnost exkluzivně sčítá s binárním klíčem.


Prostý text je převeden na binární reprezentaci. Tento binární řetězec je exkluzivně sečten (výstupem funkce `XOR` je logická 1 právě tehdy, když vstupní hodnoty jsou různé; jsou-li vstupní hodnoty stejné, výstupem je logická 0) s klíčem v binární podobě. Klíč je cyklicky opakován po celé délce prostého textu.


Libovolné znaky.


Klíčové slovo.

Binární posloupnost (pro vyšší přehlednost se používá hexadecimální podoba).

4.4 Kódy

ASCII

Toto kódování bylo přijato jako americký standardní kód pro výměnu informací (***A**merican **S**tandard **C**ode for **I**nformation **I**nterchange*). Původní definice tohoto standardu byla sedmibitová (tedy zahrnovala 128 znaků), později však byla rozšířena na osmibitovou verzi (256 znaků). Prvních 128 znaků je totožných pro všechna ASCII kódování, dalších 128 znaků je specifických pro danou zemi (resp. jazyk).

Prostý text je znak po znaku převáděn na číselnou ordinální hodnotu.


Libovolné znaky.


Čísla (nejvýše trojciferná) oddělená `-`.

Base64

Base64 kódování bylo vytvořeno pro potřeby reprezentace binárních dat pomocí tisknutelných ASCII znaků (viz [20]). Používá se především v případech, kdy je nutné uložit nebo poslat binární data pomocí medií, která jsou navržena pro práci s textovými daty. Base64 kódování je používáno v mnoha aplikacích jako např. email (MIME), ukládání komplexních dat v XML formátu, atd.

V této variantě se používá 64 tisknutelných ASCII znaků - velká a malá písmena, číslice a znaky `+` a `/`. Jako **zarovnání** (*padding*) se používá znak `=`.

Při kódování prostého textu do Base64 kódování je každý znak nahrazen jeho 8 bitovou binární ordinální hodnotou. Pokud tento řetězec není násobkem 6, pak je na konci doplněn tolika 0, aby byl dělitelný 6. Tento řetězec je rozdělen na skupiny po 6 bitech, které jsou pak převedeny na desítkovou hodnotu. Tato hodnota slouží jako index v převodní tabulce složené ze 64 znaků. Znak v tabulce je použit jako zakódovaná hodnota. Pokud prostý text nelze rozdělit na trojice, zakóduje se poslední jeden (resp. dva) znak a přidají se dvě (resp. jedno) rovnítka (viz obrázek 4.6).

```
prostý text:     Base64 code
zakódovaný text: QmFzZTY0IGNvZGU=
dekódovaný text: Base64 code
```

Tabulka 4.6: Příklad použití kódování Base64.


Libovolné znaky.


Řetězec složený z velkých či malých písmen, číslic a znaků `+`, `/` a `=`.

Dvorak

Toto kódování je založeno na rozdílném rozložení klávesnice `US QWERTY` a `US Dvorak`. Rozložení klávesnice `US Dvorak` bylo patentováno v roce 1936 a přinášelo menší pohyb prstů po klávesnici a tím vyšší rychlost psaní s nižší námahou. [10]

f(x) Znaky prostého textu napsaného na klávesnici s rozložením `US QWERTY` jsou kódovány na znaky ležící na stejných pozicích na klávesnici s rozložením `US Dvorak`. Při dekódování jsou zapisované znaky na klávesnici `US Dvorak` nahrazovány znaky na stejných pozicích na klávesnici s rozložením `US QWERTY`.

Znaky `US QWERTY` rozložení klávesnice, včetně znaků dostupných přes `AltGr` či `Shift`.

Znaky zahrnuté v `US Dvorak` rozložení klávesnice, včetně znaků dostupných přes `AltGr` či `Shift`.

Morseova abeceda (*Morse code*)

Toto kódování vzniklo v 1. polovině 19. stol. pro potřeby telegrafní komunikace, kdy bylo potřeba zajistit přenos obsahu zprávy na velkou vzdálenost. Přenášet zvuk (resp. hlas) ještě nebylo možné, proto Alfred Vail sestavil pro Samuele F. B. Morse kódování (viz tabulka 4.7), které bylo možné použít pro zakódování zprávy a její následný telegrafní přenos. Toto kódování respektovalo frekvenci znaků – nejpoužívanější znaky mají nejkratší kódovaný přepis (např. `e`, `t`, `a`, `i`, `m`, `n`), kdežto nejméně používané mají zápis nejdelší.

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	--..	0	-----	1	.----	2	..---	3	...--
4	....-	5	.....	6	-....	7	--...	8	---..	9	----.

Tabulka 4.7: Mezinárodně používaná Morseova abeceda.

f(x) Každý znak prostého textu je zakódován na posloupnost teček a čárek podle předem specifikované Morseovy abecedy.

Anglická abeceda a číslice.

Posloupnost teček a čárek – písmena (resp. číslice) jsou oddělena lomítkem, slova jsou oddělena dvěma lomítky.

Klepání (*Tap code*)

Toto kódování je určeno pro zakódování prostého textu znak po znaku na posloupnost dvojic klepnutí. Tento způsob kódování je používán např. vězni pro komunikaci s jinými celami. Princip je založen na tabulce o rozměrech 6×6, do které je zapsána nejprve celá anglická abeceda a pak číslice.

f(x) Každý znak prostého textu je zakódován na dvě posloupnosti klepnutí – počet klepnutí v první skupině určuje řádek a počet klepnutí ve druhé skupině sloupec, ve kterém se kódovaný znak nachází.

Anglická abeceda a číslice.

Posloupnost klepání (v grafické podobě se používají tečky) – písmena (resp. číslice) jsou oddělena lomítkem, slova jsou oddělena dvěma lomítky.

Kapitola 5

Návrh knihovny a aplikace

Tato kapitola je věnována problematice návrhu knihovny algoritmů pro šifrování textu a také návrhu aplikace demonstrující funkce a možnosti této knihovny. V následujícím textu jsou stanoveny požadavky na knihovnu (resp. aplikaci), je navrženo řešení těchto požadavků a také nabídnuto možné aplikační rozhraní (API[1]) knihovny.

5.1 Knihovna

Předmětem této práce je navržení a implementace **knihovny**[2] (*library*) nabízející možnosti pro šifrování a dešifrování textu.

5.1.1 Analýza požadavků

Studované algoritmy (viz kapitola 4) byly implementovány v rámci knihovny, jejíž pracovní název byl zvolen `CipherLib`. Tento název vystihuje obě esenciální myšlenky – knihovna šifer[3].

Mezi hlavní požadavky kladené na knihovnu `CipherLib` patří snadná **rozšiřitelnost** knihovny o další šifry (resp. kódy) a tedy její **znovupoužitelnost**, **jednotné rozhraní** pro práci s různými druhy šifer a kódů, zajistit tak **zapouzdřenost** (všechny akce probíhající uvnitř knihovny jsou neviditelné pro program, ve kterém je knihovna importována; knihovna vystupuje jako **černá skříňka**) a také **obsluha výjimek**[4] v rámci knihovny.

K dalším požadavkům patří **přehlednost** a snadná orientace v rámci knihovny či možnost vytváření funkcí využitelných ve více algoritmech.

Dále je také nutné, aby vytvoření nové šifry (resp. kódu) vyžadovalo *co nejméně úsilí* (tzn. pouhá deklarace by měla novou šifru zpřístupnit v dalších částech knihovny a také ji zpřístupnit v programu, v němž je knihovna importována).

[1] **A**pplication **P**rogramming **I**nterface (**API**) – tento termín označuje rozhraní pro programování aplikací, jedná se o sadu funkcí, tříd a metod nějaké knihovny, které mohou být při programování využity. API specifikuje způsob volání jednotlivých funkcí ve zdrojovém kódu programu.

[2] **Knihovna** označuje množinu funkcí, tříd (spolu s metodami těchto tříd), datových typů a hodnot, které jsou implementovány jako jeden modul, jež lze importovat do jiných programů. Knihovna pracuje jako tzv. **černá skříňka** (*black box*), protože vnitřní strukturu a chování nelze ovlivnit či měnit; knihovna nabízí jen aplikační rozhraní, jež určuje, jakým způsobem je možné knihovnu využívat v jiných programech.

[3] Zde je v pracovním názvu drobná chyba, protože knihovna neimplementuje pouze šifry, ale i kódy.

[4] **Obsluha výjimek** (*exception handling*) je programová konstrukce navržená k ošetření výskytu výjimečné situace, která by změnila normální běh programu.

5.1.2 Objektový návrh

Z výše uvedených požadavků přímo vyplývá i způsob řešení – použití **objektově orientovaného přístupu** při tvorbě knihovny. Tento přístup poskytne možnosti pro vytvoření jednotného rozhraní pro všechny implementované šifry a kódy. Objektově orientovaný přístup vyžaduje vytváření tříd, které jsou dostupné přes API knihovny. Pokud by knihovna obsahovala velké množství různých tříd, které by však implementovaly metody stejných názvů, umožnilo by to snadnou práci se všemi implementovanými šiframi (resp. kódy) za použití stejných metod, přestože algoritmus těchto metod by byl různý.

Objektově orientované programování (dále jen OOP) přináší nové koncepty programování, které jsou označovány jako základní stavební kameny objektově orientovaného paradigmatu. Definice těchto konceptů jsou bez úprav převzaty z [22]:

Objekty (*objects*) – „*spojují data a funkcionalitu společně do jednotek zvaných objekty, ze kterých se potom skládá výsledný objektově orientovaný program na rozdíl od strukturovaného složeného z procedur a funkcí. Objekty jsou tedy základní jednotkou modularity i struktury v objektově orientovaném programu, která umožňuje problém intuitivně rozdělit na přímo realitě korespondující podčásti a díky jejich vzájemné komunikaci i tento problém řešit.*“ [22]

Abstrakce (*abstraction*) – „*neboli schopnost programu ignorovat/zjednodušit/zanedbat některé aspekty informací či vlastnosti objektů, se kterými program pracuje. Abstrakce je pohled na vybraný problém reálného světa a jeho počítačové řešení. Při vytváření takovéto abstrakce je vhodné mít možnost skrývat detaily do jakési* ***černé skříňky*** *(black box), která je pro okolí definovaná pouze svým rozhraním, přes které komunikuje s okolím, a nikoli vnitřními detaily implementace, která může být podstatným zjednodušením reálného světa. Při konstrukci černé skříňky je dobré mít na paměti varianty a invarianty řešeného problému*[5]*.*

Důležitým rozhodnutím je potom míra abstrakce, tedy jak hodně je vzdálená funkčnost černé skříňky od reality, respektive jak detailně potřebujeme realitu pomocí této abstrakce modelovat. Zde je hlavním limitujícím faktorem složitost a náročnost implementace perfektní černé skříňky, která by byla jednoduše použitelná, určená pouze svou funkcionalitou velmi blízkou reálnému chování abstrahovaného objektu či problému.“ [22]

Zapouzdření (*encapsulation*) – „*zajišťuje již na úrovni definice sémantiky jazyka, že uživatel nemůže měnit interní stav objektů libovolným (tedy i neočekávaným) způsobem, ale musí k tomu využívat poskytované rozhraní (operace nad objektem). Každý objekt tedy nabízí protokol, který určuje, jak s ním mohou ostatní objekty interagovat (komunikovat). Ostatní objekty se tedy při komunikaci s tímto objektem spoléhají pouze na jeho externí (poskytované) rozhraní a skutečná implementace zůstává dokonale skryta. Právě tento koncept zásadně zjednodušuje vývoj nových vlastností a vylepšování stávající implementace našeho programu, protože nám stačí zachovat pouze zpětnou kompatibilitu rozhraní objektů a nikoli skrytých implementací*[6]*. Klíčo-*

[5] *Invariant* je taková část programu, že se hodnoty proměnných ani chování této části programu nemění při opakovaném provádění (průchodu) kódu této části programu. Zcela opačně je popsán *variant*, což je část programu, která hodnoty proměnných nebo svoje chování alespoň v některých průchodech mění.

[6] Některé OOJ (objektově orientované jazyky) umožňují přísné zapouzdření instančních proměnných porušovat pomocí tzv. modifikátorů viditelnosti.

vou roli hraje tato vlastnost také pro umožnění znovupoužitelnosti obecnějších objektů v různých projektech.“ [22]

Polymorfismus (*polymorphism*) – „*neboli mnohotvárnost využívá mechanismus zasílání zpráv. Namísto běžného volání podprogramů (procedur a funkcí) ve strukturovaném programování se v OOJ využívá mechanismu zasílání zpráv. Konkrétní použitá metoda reagující na zaslání zprávy závisí na konkrétním objektu, jemuž je tato zpráva zaslána. Například, pokud máme objekt orel a pošleme mu zprávu* `rychle_se_přemísti`*, tak implementace reagující metody bude pravděpodobně obsahovat příkazy pro roztažení křídel a vzlétnutí. Pokud ale budeme mít objekt gepard, tak implementace metody volané při obdržení téže zprávy bude obsahovat například příkaz pro rozběhnutí se. Obě reakce na příjem stejné zprávy byly uzpůsobeny potřebám konkrétního objektu, který zprávu obdržel, a tudíž byly plně v jeho vlastní režii. Takto získáme polymorfismus, kdy ta samá proměnná programu může během jeho běhu obsahovat či odkazovat různé objekty, a tak může zaslání stejné zprávy objektu ve stejné proměnné při provádění stejné části kódu vyvolat během různých kontextů (časových bodů, obsahů proměnných či instančních proměnných, hodnot parametrů) rozdílné reakce (invokovat různé metody).*“ [22]

Dědičnost (*inheritance*) – „*je způsob, jak implementovat sdílené chování. Nové objekty tak mohou sdílet a rozšiřovat chování těch již existujících bez nutnosti vše znovu reimplementovat*[7]*. Dědičnost se v praxi využívá ke dvěma účelům: (1) k indikaci, že nové chování specializuje jiné již existující chování*[8] *a (2) pro sdílení kódu (implementace metod). Tato vlastnost je tedy velmi důležitá pro udržovatelnost a rozšiřitelnost (robustnost) objektově orientovaných systémů.*“ [22]

Výše uvedené vlastnosti OOP přímo nabízí řešení požadavků specifikovaných v kapitole 5.1.1.

Zapouzdřenost – tento požadavek kladený na knihovnu vyžadoval, aby program, ve kterém je knihovna importována jako modul, neměl možnost přistupovat k atributům objektů přímo, pouze pomocí metod. Toho lze docílit použitím privátních atributů, jejichž hodnoty jsou získávány (resp. nastavovány) pomocí veřejných metod.

Rozšiřitelnost – aby bylo přidávání vlastních šifer (resp. kódů) uživatelsky přívětivé, lze využít dědičnosti. Každá implementovaná šifra (resp. kód) dědí od nadřazené třídy obecné šifry (resp. obecného kódu). Při dědění jsou převzaty všechny atributy i metody, které poskytoval nadřazený objekt; k těmto vlastnostem je možné přidat další nebo stávající upravit. Obecná šifra (resp. kód) opět dědí od obecné třídy, která charakterizuje část reality, která prostý text transformuje na šifrový text. Tímto vícenásobným děděním vzniká hierarchie tříd šifer a kódů, do níž lze snadno doplnit vlastní šifry či kódy.

Pokud nově implementovaná šifra či kód jsou velmi podobné již existující šifře či kódů, může být mnohem výhodnější dědit od těchto konkrétních tříd než od obecných tříd pro šifry či kódy. Díky tomu není nutné stejné algoritmy programovat vícekrát, ale lze používat pouze jeden zápis algoritmu a poté dědičnost.

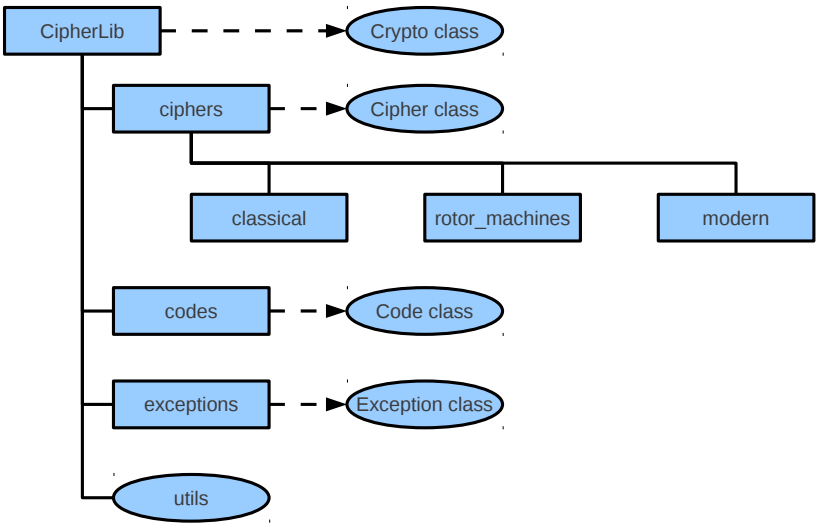
[7] Jazyky založené na objektech pracují místo dědičnosti s pojmem delegace.

[8] Ve staticky typovaných jazycích hraje vztah dědičnosti významnou roli při typové kontrole.

Znovupoužitelnost – knihovna je navržena tak, aby nebyla použitelná jen v rámci této práce, ale aby byla importovatelná jako modul. Z toho důvodu nemůže být knihovna `CipherLib` závislá na demonstrační aplikaci.

Jednotnost rozhraní – díky jednotnému rozhraní všech šifer a kódů je možné pracovat se všemi algoritmy voláním stejných metod pro nastavování klíčů, šifrování (resp. kódování) či dešifrování (reps. dekódování). Tato vlastnost je dána děděním metod od nejobecnější třídy, kdy konkrétní šifry (resp. kódy) pouze reimplementují používané metody podle algoritmu specifického pro šifru (resp. kód).

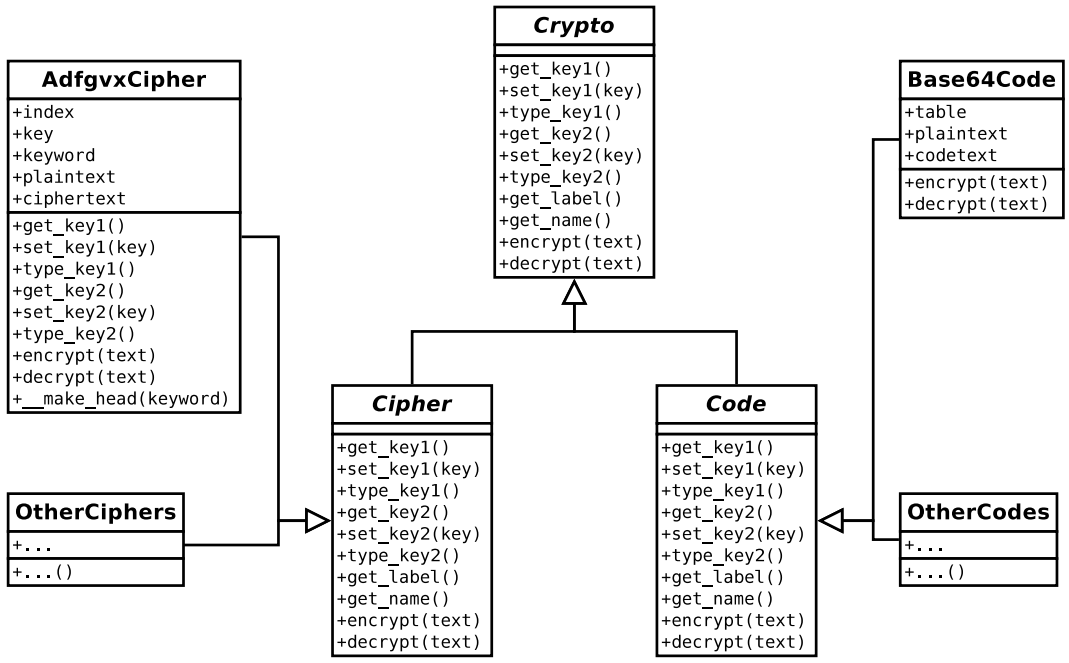
Snadná orientace v knihovně – pro snadnou orientaci v knihovně při vytváření instancí objektů šifer či kódů nebo při volání pomocných funkcí je nutné vytvořit takovou strukturu knihovny, která by tento požadavek splňovala. Proto by měly být všechny kódy uloženy v jednom adresáři, stejně tak klasické a moderní šifry a rotující stroje by měly být uloženy v speciálních adresářích. Tento způsob uložení souborů poskytuje přehlednou orientaci v knihovně (viz obrázek 5.1).



Obrázek 5.1: Struktura knihovny `CipherLib`.

Obsluha výjimek – přestože všechny šifry či kódy očekávají přesně daný formát klíčů a zprávy, nelze zaručit, že uživatel bude knihovní funkce či metody objektů volat vždy se správným formátem dat. Aby se předešlo pádu aplikace způsobeného nekonzistencí vstupních a očekávaných dat, je nutné tyto situace ošetřit. K tomu se využívá tzv. obsluha výjimek. Díky tomu je uživatel upozorněn na chybný formát dat spolu s popisem chyby a může ji napravit. Toto chování knihovny má za následek vyšší stabilitu a spolehlivost knihovny (resp. programů knihovnu využívajících). Mnoho objektově orientovaných jazyků definuje třídy obsluhující výjimky; lze tak snadno vytvářet vlastní obsluhu chyb.

Vyhodnocením požadavků a návrhu jejich řešení vznikl objektový návrh knihovny, který je uveden na obrázku 5.2. Uvedený **diagram tříd** (*class diagram*) není (a ani nemůže být kvůli rozšiřitelnosti) úplný, obsahuje jen základní třídy se zvýrazněním dědičnosti a vztahů mezi třídami.



Obrázek 5.2: Diagram tříd knihovny `CipherLib`.

5.1.3 Aplikační rozhraní knihovny

Všechny vytvořené třídy pro definici šifry nebo kódu, které dědí od jiné třídy (obecné nebo konkrétní šifry nebo kódu), přebírají všechny jejich vlastnosti (atributy a metody). Metody pak mohou reimplementovat, čímž způsobí jiné chování při volání metody stejného jména. V rámci celé knihovny je tak žádoucí používat stejné názvy metod a pouze měnit jejich implementaci.

Studiem různých šifer bylo zjištěno, že počet klíčů u šifer se pohybuje v rozmezí $0 - 2$, kódy ve většině případů žádné klíče nemají. Kvůli tomuto zjištění byl rovněž navržen počet klíčů 2 (resp. metod pracujících s klíči).

Z požadavků na knihovnu a závěrů ze studia různých druhů šifer bylo navrženo aplikační rozhraní knihovny, které je uvedeno v tabulce 5.1.

5.2 Aplikace

Pro demonstraci možností objektově orientované knihovny `CipherLib` byla navržena grafická aplikace. Vzhled aplikace a její obsluha by měla být maximálně prostá a zaměřená pouze na funkčnost aplikačního rozhraní knihovny. Tato aplikace by měla automaticky prohledávat dostupné šifry a kódy a nabízet je k šifrování (resp. kódování) prostého textu. Dále by měla aplikace využívat všechny metody, které jsou u šifer (resp. kódů) implementovány. V aplikaci by měla být možnost výběru šifry (resp. kódu), zadání potřebných klíčů včetně možnosti generování vhodného klíče (případně vlastní šifrovací abecedy) a zadání vstupního textu. Pro vyšší použitelnost aplikace by měla být možnost načtení souboru s prostým textem a uložení šifrovaného textu do souboru. Stručná nápověda k použití aplikace je samozřejmostí. Výše uvedené požadavky na aplikaci by měly efektivně využívat aplikačního rozhraní knihovny `CipherLib` (viz tabulka 5.2).

metoda	popis činnosti metody
`constructor`	Konstruktor třídy, ve kterém mohou být připraveny atributy nezávislé na klíči nebo vstupním textu.
`get_key1()`	Získání hodnoty prvního klíče.
`set_key1(key)`	Nastavení hodnoty prvního klíče hodnotou `key`.
`type_key1()`	Získání typu prvního klíče.
`get_key2()`	Získání hodnoty druhého klíče.
`set_key2(key)`	Nastavení hodnoty druhého klíče hodnotou `key`.
`type_key2()`	Získání typu druhého klíče.
`get_label()`	Získání popisu šifry (resp. kódu).
`get_name()`	Získání jména šifry (resp. kódu).
`encrypt(text)`	Zašifrování (resp. zakódování) vstupu `text`.
`decrypt(text)`	Dešifrování (resp. dekódování) vstupu `text`.

Tabulka 5.1: Aplikační rozhraní knihovny `CipherLib`.

metoda	výskyt volání metody v aplikaci
`constructor`	Výběr šifry (resp. kódu) ze seznamu šifer a vytvoření instance vybrané třídy.
`get_key1()`	Získání hodnoty výchozího prvního klíče; pokud je při šifrování klíč upravován, touto metodou je změna reflektována v aplikaci.
`set_key1(key)`	Nastavení hodnoty prvního klíče hodnotou `key` zadanou v aplikaci.
`type_key1()`	Získání typu prvního klíče kvůli možnosti generování vhodného klíče.
`get_key2()`	Získání hodnoty výchozího druhého klíče; pokud je při šifrování klíč upravován, touto metodou je změna reflektována v aplikaci.
`set_key2(key)`	Nastavení hodnoty druhého klíče hodnotou `key` zadanou v aplikaci.
`type_key2()`	Získání typu druhého klíče kvůli možnosti generování vhodného klíče.
`get_label()`	Získání popisu šifry (resp. kódu) a zobrazení popisu v aplikaci.
`get_name()`	Získání jména šifry (resp. kódu) a zobrazení jména v aplikaci.
`encrypt(text)`	Zašifrování (resp. zakódování) vstupu `text` zadaného v aplikaci.
`decrypt(text)`	Dešifrování (resp. dekódování) vstupu `text` zadaného v aplikaci.

Tabulka 5.2: Využití aplikačního rozhraní knihovny v demonstrační aplikaci.

Kapitola 6

Implementace knihovny a aplikace

V kapitole 5 byla navržena knihovna algoritmů pro šifrování textu a bylo také nastíněno teoretické řešení požadavků kladených na knihovnu. V této kapitole bude podrobně popsána samotná implementace knihovny, představen způsob, jakým lze knihovnu rozšířit, a v neposlední řadě také komentována implementace demonstrační aplikace.

6.1 Implementační jazyk

Objektově orientovaný návrh knihovny vyžaduje použití jazyka, který implementuje objektově orientovaná paradigmata. V dnešní době existuje spousta takových jazyků (např. Java, C++, C#, Smalltalk, Object Pascal, Visual Basic, .NET, Lisp, PHP, Python, Ruby, ...), z nichž pro účely této práce byl vybrán jazyk `Python 3.1.2`. Tento jazyk se řadí mezi tzv. *čistě objektově orientované*, protože výpočet probíhá výhradně pomocí vzájemného zasílání zpráv mezi objekty.

Mezi hlavní přednosti jazyka `Python 3.1.2` patří jeho standardní práce s řetězci v `UTF-8` kódování, což je v rámci knihovny `CipherLib` velmi žádoucí. Tento jazyk také umožňuje ošetření výjimek, ať už vestavěných nebo uživatelsky definovaných. Této vlastnosti je využito při implementaci ošetření výjimečných stavů, které by mohly vést k pádu knihovny. Mezi další přednosti tohoto jazyka patří především rozšířenost jazyka mezi uživateli, bezplatné používání, velmi kvalitní dokumentace dostupná on-line na [31], množství vestavěných funkcí a modulů. V neposlední řadě byla jedním z kritérií i autorova znalost daného jazyka a jeho schopnost jej využívat.

Vzhledem k tomu, že knihovna `CipherLib` je implementována v jazyce `Python 3.1.2`, jedná se podle konvencí tohoto jazyka spíše o **balíček** (*package*) nežli o knihovnu. Proto je při tvorbě nutné dodržet pravidla[1], která jsou pro tvorbu balíčků určena. Při psaní zdrojového kódu byly dodrženy rady a doporučení v *PEP 8 – Style Guide for Python Code* (viz [34]) a *Google Python Style Guide* (viz [27]).

6.2 Implementace tříd

Struktura knihovny je znázorněna již v kapitole 5 na obrázku 5.1. Tato struktura respektuje klasifikaci kryptografických algoritmů na kódy a šifry, které jsou dále děleny na klasické a moderní šifry a rotující stroje. Ve struktuře knihovny jsou také zahrnuty soubory s vlastní

[1] `http://docs.python.org/release/3.1.2/tutorial/modules.html#packages`

obsluhou výjimek a soubory s definovanými funkcemi, které mohou být využívány napříč knihovnou.

6.2.1 Třída `Crypto`

Pro všechny kryptografické algoritmy je vytvořena abstraktní třída `Crypto`, jež se definována v souboru `CipherLib/crypto.py`. Definice této třídy je uvedena v programu 6.1. Tato třída obsahuje deklaraci všech metod využitelných u různých druhů šifer. Tato třída je v hierarchii tříd v knihovně předkem všech šifer i kódů.

```
#!/usr/bin/env python3.1
# -*- coding: utf-8 -*-

class Crypto:
  """Abstraktni trida pro vsechny kryptograficke algoritmy"""
  def set_key1(self, key):
    """Nastaveni hodnoty prvniho klice"""
    pass

  def get_key1(self):
    """Ziskani hodnoty prvniho klice"""
    return None

  def type_key1(self):
    """Ziskani typu prvniho klice"""
    return None

  def set_key2(self, key):
    """Nastaveni hodnoty druheho klice"""
    pass

  def get_key2(self):
    """Ziskani hodnoty druheho klice"""
    return None

  def type_key2(self):
    """Ziskani typu druheho klice"""
    return None

  def get_name(self):
    """Ziskani nazvu sifry"""
    return None

  def get_label(self):
    """Ziskani popisu sifry"""
    return self.__doc__

  def encrypt(self, input):
    """Zasifrovani vstupniho textu"""
    return None

  def decrypt(self, input):
    """Desifrovani vstupniho textu"""
    return None
```

Program 6.1: Definice třídy `Crypto`.

Třída `Cipher`

Pro přehlednost knihovny byla vytvořena i třída `Cipher`, jež, jak již sám název napovídá, je obecným vzorem všech šifer. Tato třída dědí od třídy `Crypto`, takže disponuje všemi metodami, jež jsou v třídě `Crypto` definovány. Definice třídy `Cipher` (viz program 6.2) je uvedena v souboru `CipherLib/ciphers/cipher.py`. Při definici je pouze implementována metoda `get_name()` vracející název šifry.

Všechny šifry dědí od třídy `Cipher` a pouze reimplementují metody specifické pro konkrétní šifru. Podle navržené struktury knihovny `CipherLib` jsou všechny klasické šifry ukládány v adresáři `CipherLib/ciphers/classical`, rotující stroje jsou ukládány v `CipherLib/ciphers/rotor_machines` a moderní šifry jsou ukládány v `CipherLib/ciphers/modern`

```
#!/usr/bin/env python3.1
# -*- coding: utf-8 -*-

import re
from ..crypto import Crypto

class Cipher(Crypto):
  """Abstraktni trida pro vsechny sifry"""
  def get_name(self):
    """Ziskani nazvu sifry"""
    name = type(self).__name__
    name = re.sub("([a-z])([A-Z])([a-z])", "\\1 \\2\\3", name)
    return name
```

Program 6.2: Definice třídy `Cipher`.

Třída `Code`

Podobně jako byla pro všechny třídy vytvořena třída `Cipher`, je i pro všechny kódy vytvořena obecná třída `Code`. Všechny implementované kódy pak dědí od této třídy. Třída `Code` dědí od třídy `Crypto`, takže také disponuje všemi jejími metodami. V definici třídy `Code` (viz program 6.3) v souboru `CipherLib/codes/code.py` je pouze implementována metoda `get_name()` pro získání názvu kódu.

Všechny vytvořené kódy přímo dědí od třídy `Code` a jsou uloženy ve stejném adresáři (tj. `CipherLib/codes`).

6.2.2 Třída `Error`

Kromě vestavěných výjimek v jazyce `Python 3.1.2` je nutné v knihovně využívat i vlastní obsluhu výjimek. Tyto výjimky nastávají, kdy algoritmus šifry vyžaduje klíč i vstupní text daného formátu, ale získá klíč nebo text jiného formátu. Aby se předešlo pádu knihovny nebo aplikace knihovnu využívající, je implementováno několik výjimek, které zahrnují všechny mimořádné situace. V tabulce 6.1 jsou uvedeny implementované výjimky spolu s popisem situace, ve které je konkrétní výjimka ošetřována.

V dokumentaci k jazyku `Python 3.1.2` (viz [31]) je specifikovaný způsob vytváření vlastní obsluhy výjimek. Nejprve je nutné vytvořit abstraktní třídu (v případě knihovny je to `Error`), jež dědí od vestavěné třídy `Exception`. Od abstraktní třídy `Error` (definice této

```
#!/usr/bin/env python3.1
# -*- coding: utf-8 -*-

import re
from ..crypto import Crypto

class Code(Crypto):
  """Abstraktni trida pro vsechny kody"""
  def get_name(self):
    """Ziskani nazvu kodu"""
    name = type(self).__name__
    name = re.sub("([a-z])([A-Z])([a-z])", "\\1 \\2\\3", name)
    return name
```

Program 6.3: Definice třídy `Code`.

název výjimky	**popis výjimky**
`InputNumberError`	chybný číselný klíč
`LengthNumberError`	chybný délka číselného klíče
`InputKeyError`	chybné klíčové slovo či fráze
`LengthKeyError`	chybná délka klíčového slova či fráze
`InputAlphabetError`	chybná vstupní abeceda
`LengthAlphabetError`	chybná délka vstupní abecedy
`InputTextError`	chyba ve vstupním textu (nepovolené znaky, ...)
`LengthTextError`	chybná délka vstupního textu

Tabulka 6.1: Seznam implementovaných výjimek v knihovně `CipherLib`.

třídy je v souboru `CipherLib/exceptions/exceptions.py` a je uvedena i v programu 6.4) je možné děděním vytvářet vlastní třídy definující obsluhu výjimek.

```
#!/usr/bin/env python3.1
# -*- coding: utf-8 -*-

class Error(Exception):
  """Abstraktni tridy pro vsechny vlastni vyjimky"""
  def __str__(self):
    """Definice chovani pri vypisu chybove zpravy"""
    return str(self.value)
```

Program 6.4: Definice třídy `Error`.

Třída `InputNumberError`

K obsluze různých výjimek v knihovně lze vytvářet třídy, které konkrétní výjimky definují. Tyto výjimky jsou vytvářeny v souboru `CipherLib/exceptions/exceptions.py`. Ze všech implementovaných výjimek je v tomto textu ukázáno jen ošetření situace, kdy algoritmus

očekává číselný klíč, ale získá klíč jiného typu (resp. vstup nelze převést na číslo). Definice této výjimky je uvedena v programu 6.5.

```
#!/usr/bin/env python3.1
# -*- coding: utf-8 -*-

class InputNumberError(Error):
  """Chybny ciselny klic"""
  def __init__(self, value = None):
    """Vytvoreni chybove zpravy"""
    self.value = 'Wrong numeric key'
    self.value += ': ' + value if value else ''
```

Program 6.5: Definice třídy `InputNumberError`.

6.3 Rozšíření knihovny

Mezi požadavky kladenými na knihovnu byla i snadná rozšiřitelnost o další kryptografické algoritmy. Knihovna proto byla implementována tak, že pouhým vytvořením souboru s definicí třídy pro konkrétní šifru je tato šifra dostupná v rámci celé knihovny i v programu, v němž je knihovna importována. Této vlastnosti se docílilo tak, že v každém adresáři s definicí jednotlivých šifer existuje soubor `__init__.py` (jeho název je dán v pravidlech[2] pro vytváření balíčků v `Python 3.1.2`), při jehož spuštění jsou z aktuálního adresáře zpřístupněny všechny třídy v rámci celé knihovny.

Při vytváření nové třídy je nutné znát její zařazení v klasifikaci algoritmů – zda se jedná o klasickou šifru, moderní šifru či rotující stroj nebo se jedná o kód. V programu 6.6 je ukázán postup při vytváření nové šifry. Šifra se jmenuje `TEST`, patří mezi moderní šifry a pracuje pouze s jedním klíčovým slovem. Definice této šifry by byla uvedena v souboru `CipherLib/ciphers/modern/test.py`. Při vytváření nových algoritmů je nutné dodržet pravidlo, že název třídy dané šifry je ve tvaru `NazevCipher` (pro kódy je tento tvar `NazevCode`), kde `Nazev` je skutečný název šifry bez diakritických značek v `CamelCase` notaci. Kódy mohou být umístěny pouze v adresáři `CipherLib/codes` a šifry mohou být podle charakteru umístěny v `CipherLib/ciphers/classical`, `CipherLib/ciphers/rotor_machines` či `CipherLib/ciphers/modern`.

6.4 Implementace aplikace

Pro tvorbu demonstrační aplikace byl opět zvolen jazyk `Python 3.1.2`; knihovna `CipherLib` je napsána v tomto jazyce a pro `Python 3.1.2` existuje **binding**[3] na Qt framework. V následujícím textu budou představena pozitiva tohoto frameworku při vytváření demonstrační aplikace a také stručně popsána implementace aplikace.

6.4.1 Použitý framework

Firma Trolltech vydala v roce 1992 první verzi frameworku s názvem Qt. V roce 2008 koupila firma Nokia tento framework a neustále pokračuje v jeho vývoji. Jedná se velmi

[2] `http://docs.python.org/release/3.1.2/tutorial/modules.html#packages`

[3] **Binding** označuje knihovnu poskytující stejné rozhraní a funkčnost jako jiná knihovna napsaná v jiném programovacím jazyce.

```
#!/usr/bin/env python3.1
# -*- coding: utf-8 -*-

from ..cipher import Cipher
from ...exceptions import exceptions

class TestCipher(Cipher):
  """Moderni sifra TEST pracujici s jednim klicovym slovem"""
  def set_key1(self, key):
    """Nastaveni hodnoty prvniho klice"""
    if str(key).isalpha():
      self.key = str(key)
    else:
      raise exceptions.InputKeyError(str(key))

  def get_key1(self):
    """Ziskani hodnoty prvniho klice"""
    return self.key

  def type_key1(self():
    """Ziskani typu prvniho klice"""
    return "key"

  def encrypt(self, input):
    """Zasifrovani prosteho textu input"""
    ...
    return self.ciphertext

  def decrypt(self, input):
    """Odsifrovani sifroveho textu input"""
    ...
    return self.plaintext
```

Program 6.6: Definice šifry `TEST`.

rozsáhlý framework napsaný v jazyce C++ poskytující třídy a funkce pro práci s grafickým rozhraním, multimédii, vektorovou grafikou, síťovými záležitostmi, aj. Neustálý vývoj tohoto frameworku přináší stále nové možnosti programování a díky své komplexnosti se tento framework těší velké popularitě. Tomu nasvědčuje i množství bindingů do mnoha jiných jazyků. Binding Qt frameworku pro Python se nazývá PyQt (viz [32]) a je vyvíjen společností Riverbank.

6.4.2 Design aplikace

Při tvorbě demonstrační aplikace byl kladen důraz na jednoduché prostředí s minimálním počtem nastavovacích prvků, které by činily aplikaci méně přehlednou. Uživatelské rozhraní obsahuje seznam dostupných šifer, který je automaticky aktualizován při spuštění aplikace, název a krátký popis vybrané šifry, pole pro vstupní a výstupní text, pole pro zadání klíčů a tlačítka pro generování klíčů, tlačítka pro šifrování (resp. dešifrování) a tlačítko pro prohození obsahů textových polí. Další nejnutnější položky jsou zahrnuty v hlavní nabídce.

Kapitola 7

Závěr

Tato práce je zaměřena na studium kryptografických algoritmů od samotných počátků kryptografie až po současnost a jejich implementaci v rámci programové knihovny. Kryptografické algoritmy se rozlišují na *kódy* a *šifry* – vzhledem k množství šifer existuje podrobnější klasifikace na *klasické šifry* (substituční, transpoziční či jejich kombinace), *rotující stroje* (mechanická zařízení implementující polyalfabetické substituční šifry) a *moderní šifry* (symetrické a asymetrické). Algoritmy byly studovány jak z hlediska jejich matematické podstaty, tak z hlediska vlivu doby a prostředí, v němž byly navrženy. Na vzestupné složitosti kryptografických algoritmů lze pozorovat, jak se znalosti lidstva s časem rozšiřovaly a potřeba bezpečného utajení informací stoupala.

Studované algoritmy jsou řazeny jednak podle výše uvedené klasifikace, ale také podle podobnosti s jinými algoritmy. Toto řazení umožňuje sledovat vývoj jednotlivých skupin šifer s časem. Všechny probírané šifry a kódy jsou implementovány v rámci knihovny algoritmů pro šifrování textu `CipherLib`. Tato knihovna je implementována jako balíček (*package*) v jazyce `Python 3.1.2`, jež lze importovat v dalších programech v tomto jazyce. Struktura této knihovny nejen koresponduje s výše zmíněnou klasifikací algoritmů, ale také umožňuje libovolné rozšíření o vlastní algoritmy při zachování stejného rozhraní. Přestože je práce zaměřena na studium klasických šifer, je probíráno i několik moderních metod. Moderní metody jsou ale často velmi náročné, proto je jejich studiu vhodné věnovat více prostoru. Některé z klasických šifer není možné studovat, protože již nejsou žádným způsobem zdokumentovány. Pro demonstraci možností knihovny `CipherLib` a principu všech studovaných šifer je vytvořena grafická aplikace v jazyce `Python 3.1.2` za použití PyQt frameworku. Tu lze s užitkem využít také jako učební pomůcku.

Vzhledem k stále rostoucí nutnosti a oblibě elektronické komunikace a digitálního uchovávání dat je obor kryptografie velmi perspektivní a metody pro zabezpečení digitálních informací je nezbytné neustále zdokonalovat. Proto by se autor této práce i v budoucnu chtěl zaměřit na důkladné studium problematiky nových metod zabezpečení.

Literatura

[1] Skytale. [online], Naposledy navštíveno 4. 5. 2011.
URL http://upload.wikimedia.org/wikipedia/commons/5/51/Skytale.png

[2] Akins, T.: Cipher Tools. [online], Naposledy navštíveno 4. 5. 2011.
URL http://rumkin.com/tools/cipher/

[3] American Cryptogram Association: American Cryptogram Association Home. [online], Naposledy navštíveno 4. 5. 2011.
URL http://www.cryptogram.org/

[4] Barker, W. C.: *Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher*. U.S. Department of Commerce, National Institute of Standards and Technology, Naposledy navštíveno 2. 5. 2011.
URL http://csrc.nist.gov/publications/nistpubs/800-67/SP800-67.pdf

[5] Brabec, T.: Popis šifry Blowfish. [online], Naposledy navštíveno 4. 5. 2011.
URL http://moon.felk.cvut.cz/~pjv/Jak/_info/i677/html/blowfish.htm

[6] British Museum: The Rosetta Stone. [online], Naposledy navštíveno 3. 5. 2011.
URL http://www.britishmuseum.org/explore/highlights/highlight_objects/aes/t/the_rosetta_stone.aspx

[7] Bruff, D.: Cryptography Timeline. [online], Naposledy navštíveno 4. 5. 2011.
URL http://derekbruff.com/teaching/cryptotimeline.htm

[8] Daley, W. M.; Kammer, R. G.: *Data Encryption Standard (DES)*. U.S. Department of Commerce, National Institute of Standards and Technology, Naposledy navštíveno 2. 5. 2011.
URL http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf

[9] Diffie, B. W.; Hellman, M. E.: New Directions in Cryptography. In *IEEE Transactions on Information Theory*, ročník 22, IEEE Information Theory Society, Listopad 1976, ISSN 0018-9448, s. 644–654.
URL http://www-ee.stanford.edu/~hellman/publications/24.pdf

[10] Dvorak, A.; Dealey, W. L.: Typewriter Keyboard. [online], Naposledy navštíveno 5. 5. 2011.
URL http://www.google.com/patents/about?vid=2040248

[11] Ellison, C.: Cryptography Timeline. [online], Naposledy navštíveno 4. 5. 2011.
URL http://world.std.com/~cme/html/timeline.html

[12] Esslinger, B.; Südmeyer, P.; Wacker, A.: CrypTool-Online. [online], Naposledy navštíveno 4. 5. 2011.
URL http://www.cryptool-online.org/

[13] Flek, A.; Hedánek, J.; Hoffman, P.; aj.: *Bible : překlad 21. století*. Praha: Biblion, 2009, ISBN 978-80-87282-00-7.

[14] Giry, D.: Keylength – Cryptographic Key Length Recommendation. [online], Naposledy navštíveno 4. 5. 2011.
URL http://www.keylength.com/

[15] Hellman, M. E.; Diffie, B. W.; Merkle, R. C.: Cryptographic apparatus and method. [online], Naposledy navštíveno 25. 4. 2011.
URL http://www.google.com/patents?vid=4200770

[16] Ikeda, Y.: Figure of Feistel Structure of Block Ciphers. [online], Naposledy navštíveno 25. 4. 2011.
URL http://upload.wikimedia.org/wikipedia/commons/d/d2/Feistel.png

[17] Intel: Moore's Law and Intel Innovation. [online], Naposledy navštíveno 5. 5. 2011.
URL http://www.intel.com/about/companyinfo/museum/exhibits/moore.htm

[18] Ireland, D.: RSA Algorithm. [online], Naposledy navštíveno 4. 5. 2011.
URL http://www.di-mgt.com.au/rsa_alg.html

[19] Jarosz, Q.: Cipher taxonomy. [online], Naposledy navštíveno 4. 5. 2011.
URL http://upload.wikimedia.org/wikipedia/en/8/85/Cipher-taxonomy.svg

[20] Josefsson, S.: RFC 4648 : The Base16, Base32, and Base64 Data Encodings. [online], Naposledy navštíveno 5. 5. 2011.
URL http://tools.ietf.org/html/rfc4648

[21] Kahn, D.: *The codebreakers*. London: Sphere, 1977, ISBN 0-7221-5149-7.

[22] Křivka, Z.; Kolář, D.: *Principy programovacích jazyků a objektově orientovaného programování : Studijní opora*. Brno: FIT VUT v Brně, první vydání, 2008.

[23] Liddell, H. G.; Scott, R.: *A Greek-English Lexicon*. Oxford: Oxford University Press, 1996, ISBN 978-0198642268.

[24] Lord, B.: Bob Lord's Home Page. [online], Naposledy navštíveno 4. 5. 2011.
URL http://www.ilord.com/

[25] Mička, P.: Multiplikativní inverze. [online], Naposledy navštíveno 12. 4. 2011.
URL http://www.algoritmy.net/article/46/Multiplikativni-inverze

[26] Olefsky, J.: Braingle: Codes, Ciphers, Encryption and Cryptography. [online], Naposledy navštíveno 4. 5. 2011.
URL http://www.braingle.com/brainteasers/codes/index.php

[27] Patel, A.; Picard, A.; Jhong, E.; aj.: Google Python Style Guide. [online], Naposledy navštíveno 4. 5. 2011.
URL http://google-styleguide.googlecode.com/svn/trunk/pyguide.html

[28] Petitcolas, F.: La Cryptographie Militaire. [online], Naposledy navštíveno 27. 1. 2011.
URL http://petitcolas.net/fabien/kerckhoffs/

[29] Piper, F.; Murphy, S.: *Kryptografie*. Praha: Dokořán, 2006, ISBN 80-7363-074-5.

[30] Proc, J.: Crypto Machines. [online], Naposledy navštíveno 25. 4. 2011.
URL http://www.jproc.ca/crypto/

[31] Python Software Foundation: Python 3.1.2 documentation. [online], Naposledy navštíveno 27. 4. 2011.
URL http://docs.python.org/release/3.1.2/

[32] Riverbank Computing Limited: Riverbank. [online], Naposledy navštíveno 4. 5. 2011.
URL http://www.riverbankcomputing.com/

[33] Rivest, R. L.; Shamir, A.; Adleman, L. M.: Cryptographic communications system and method. [online], Naposledy navštíveno 25. 4. 2011.
URL http://www.google.com/patents?vid=4405829

[34] van Rossum, G.; Warsaw, B.: PEP 8 – Style Guide for Python Code. [online], Naposledy navštíveno 4. 5. 2011.
URL http://www.python.org/dev/peps/pep-0008/

[35] Salomaa, A.: *Computation and automata*, kapitola 7. Cambridge New York: Cambridge University Press, první vydání, 1985, ISBN 0-521-30245-5.

[36] Salomaa, A.: *Public-key cryptography*. Berlin Heidelberg New York: Springer-Verlag, druhé vydání, 1996, ISBN 3-540-61356-0.

[37] Schneier, B.: Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish). [online], Naposledy navštíveno 2. 5. 2011.
URL http://www.schneier.com/paper-blowfish-fse.html

[38] Singh, S.: *Kniha kódů a šifer*. Praha: Dokořán, 2003, ISBN 80-86569-18-7.

[39] Willemse, J.: RuffNekk's Crypto Pages. [online], Naposledy navštíveno 4. 5. 2011.
URL http://ruffnekk.stormloader.com/

[40] Štráfelda, J.: Šifrování a signály. [online], Naposledy navštíveno 4. 5. 2011.
URL http://www.shaman.cz/sifrovani/

Příloha A

Obsah CD

Na přiloženém CD jsou uloženy zdrojové soubory této práce spolu se zdrojovými soubory knihovny a aplikace, které byly v rámci této práce vytvořena. Kvůli testování knihovny a aplikace na operačním systému Windows (verze XP a vyšší) a Linux/Unix jsou na CD také uloženy potřebné podpůrné programy.

A.1 Struktura souborů uložených na CD

prg/ Podpůrné programy potřebné pro testování aplikace.

- **Linux/** Zdrojové soubory pro instalaci podpůrných nástrojů pro Linux/Unix. Je zde také umístěn soubor `README` s podrobným postupem při instalaci těchto podpůrných nástrojů.
- **Windows/** Zdrojové soubory nebo binární soubory pro instalaci podpůrných nástrojů pro Windows (XP a vyšší). Je zde také umístěn soubor `README.txt` s podrobným postupem při instalaci těchto podpůrných nástrojů.

src/ Zdrojové soubory knihovny a aplikace.

- **CipherLib/** Zdrojové soubory knihovny `CipherLib`.
- **resources/** Soubory potřebné pro aplikaci (ikony, nápověda, ...).

txt/ Zdrojové soubory této práce.

- **fig/** Grafické soubory použité v této práci.
- **style/** Šablona pro LaTeX určující styl této práce a pravidla pro citace.

A.2 Instalace podpůrných nástrojů

Pro správnou funkčnost knihovny `CipherLib` je nutné nainstalovat interpret jazyka `Python 3.1.2`. Zdrojové soubory jsou pro Linux uloženy v `prg/Linux/Python-3.1.2.tar.bz2` a pro Windows jsou uloženy v `prg/Windows/python-3.1.2.msi`.

Demonstrační aplikace využívá PyQt framework, jehož knihovny je nutné doinstalovat do operačního systému. Při instalaci PyQt na systém Windows je potřeba postupovat v tomto pořadí:

krok	**program**	**zdrojové soubory**
1.	`Python 3.1.2`	(`prg/Windows/python-3.1.2.msi`)
2.	`PyQt 4.8.3`	(`prg/Windows/PyQt-Py3.1-x86-gpl-4.8.3-1.exe`)

Při instalaci PyQt na systém Linux/Unix je nutné doinstalovat Qt framework, SIP a PyQt v tomto pořadí:

krok	**program**	**zdrojové soubory**
1.	`Python 3.1.2`	(`prg/Linux/Python-3.1.2.tar.bz2`)
2.	`Qt 4.7.2`	(`prg/Linux/qt-everywhere-opensource-src-4.7.2.tar.gz`)
3.	`SIP 4.12.1`	(`prg/Linux/sip-4.12.1.tar.gz`)
4.	`PyQt 4.8.3`	(`prg/Linux/PyQt-x11-gpl-4.8.3.tar.gz`)

Podpůrné programy jsou také dostupné on-line na stránkách jejich tvůrců:

program	**odkaz**
`Python 3.1.2`	`http://www.python.org/download/releases/3.1.2/`
`Qt 4.7.2`	`http://qt.nokia.com/downloads`
`SIP 4.12.1`	`http://www.riverbankcomputing.co.uk/software/sip/download`
`PyQt 4.8.3`	`http://www.riverbankcomputing.co.uk/software/pyqt/download`

Příloha B

Manuál

Zde je uveden stručný návod k použití implementované knihovny `CipherLib` a demonstrační aplikace.

B.1 Návod k použití knihovny `CipherLib`

Knihovnu lze využívat dvojím způsobem – buď jen využívat již implementované šifry (resp. kódy) a pomocné funkce, nebo ji rozšiřovat (viz kapitola 6). Každý typ používání má svá specifická omezení, která je nutné dodržet pro správnou funkčnost knihovny a aplikace knihovnu využívající.

Knihovna `CipherLib` byla navržena a implementována tak, aby ji bylo možné jako modul importovat v programech v jazyce `Python 3.1.x`. Zde je ukázka použití knihovny pro zašifrování zprávy šifrou Blowfish. Ukázkový program B.1 odráží strukturu knihovny a klasifikaci šifer a kódů. V ukázce je také použita funkce pro generování klíče vhodného pro vybranou šifru (v případě šifry Blowfish je to hexadecimální klíč o délce 1 až 112).

```
#!/usr/bin/env python3.1
# -*- coding: utf-8 -*-
# pouziti knihovny CipherLib pro zasifrovani zpravy sifrou Blowfish

import CipherLib

# generovani klice (1 - 112 hexadecimalnich znaku)
key = CipherLib.utils.generate_key("1-112HEX")
data = "Testovaci zprava" # zprava pro zasifrovani

# vytvoreni instance tridy sifry Blowfish
cipher = CipherLib.ciphers.modern.BlowfishCipher()
cipher.set_key1(key) # nastaveni pracovniho klice

encrypt = cipher.encrypt(data) # zasifrovani zpravy
decrypt = cipher.decrypt(encrypt) # odsifrovani zpravy

# tisk originalniho textu, zasifrovaneho a odsifrovaneho textu
print("data:\t\t{0}\nkey:\t\t{1}\nencrypt:\t{2}\ndecrypt:\t{3}".
      format(data, key, encrypt, decrypt))
```

Program B.1: Ukázka použití knihovny `CipherLib` a šifry `Blowfish`.

Možný výstup programu je uveden v programu B.2 (zašifrovaná zpráva se bude lišit v závislosti na vygenerovaném klíči):

```
data:    Testovaci zprava
key:     E133DDC2DBA26753
encrypt: 812A262A84798A6DE1076D25947335BE
decrypt: Testovaci zprava
```

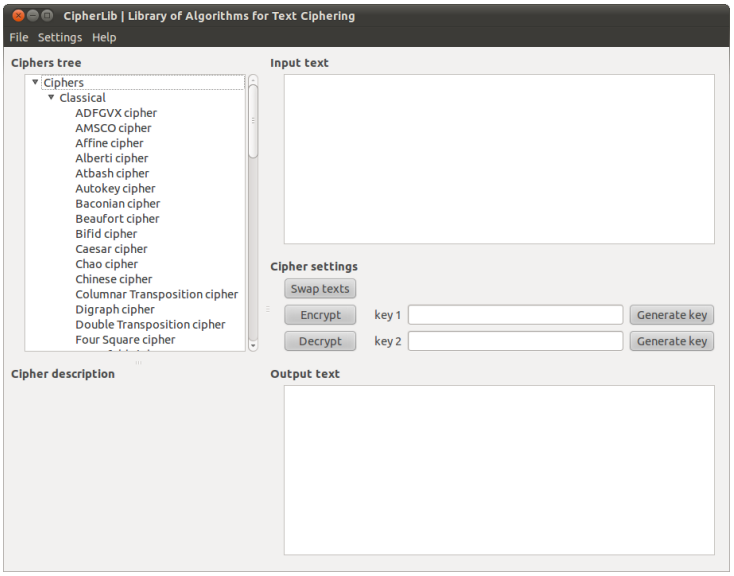
Program B.2: Výstup ukázkového programu B.1.

Některé šifry používají dva klíče, proto je nutné nastavit i druhý klíč voláním metody `set_key2(key)`. Samozřejmě je také možné zjišťovat nastavené klíče pomocí metod `get_key1, get_key2` a jejich typ `type_key1, type_key2`. Pro získání jména šifry je možné použít metodu `get_name` a pro získání jejího popisu `get_label`. Pro zašifrování (resp. dešifrování) se používá metoda `encrypt` (resp. `decrypt`).

B.2 Návod k použití aplikace

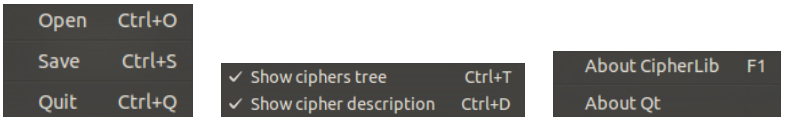
Pro zašifrování (resp. dešifrování) textu lze postupovat tímto způsobem:

1. Spusťte aplikaci (viz obrázek B.1). Aplikace nabízí jednoduchou nabídku se základními operacemi (viz obrázek B.2).



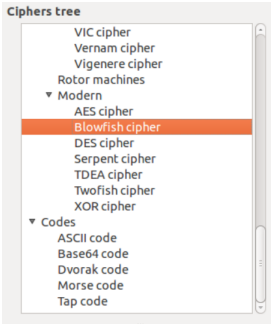
Obrázek B.1: Aplikace po spuštění.

2. Nejprve je nutné v seznamu šifer a kódů (v aplikaci označeno jako **Ciphers tree**) vybrat šifru, která bude použita pro zašifrování nebo dešifrování textu (viz obrázek B.3). Po vybrání šifry se zobrazí její stručný popis v **Cipher description**. Kromě toho se také pod **Input text** zobrazí název vybrané šifry (resp. kódu). Jelikož každá šifra vyžaduje různý počet klíčů, které se mohou lišit typy, jsou tato pole automaticky změněna v závislosti na typu šifry.



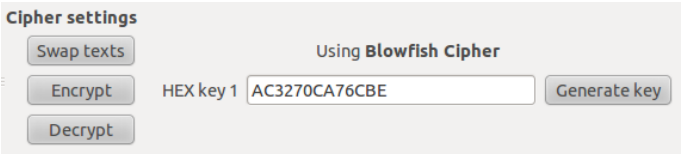
(a) Nabídka `File`. (b) Nabídka `Settings`. (c) Nabídka `Help`.

Obrázek B.2: Hlavní nabídka v demonstrační aplikaci.



Obrázek B.3: Výběr šifry.

3. Pokud šifra potřebuje nějaký klíč, je nutné jej zadat v dalším kroku. Lze použít tlačítko **Generate key**, které vygeneruje vhodný klíč pro konkrétní šifru (viz obrázek B.4). Šifry, které nevyžadují žádné klíče, automaticky skryjí políčka pro vkládání klíčů.



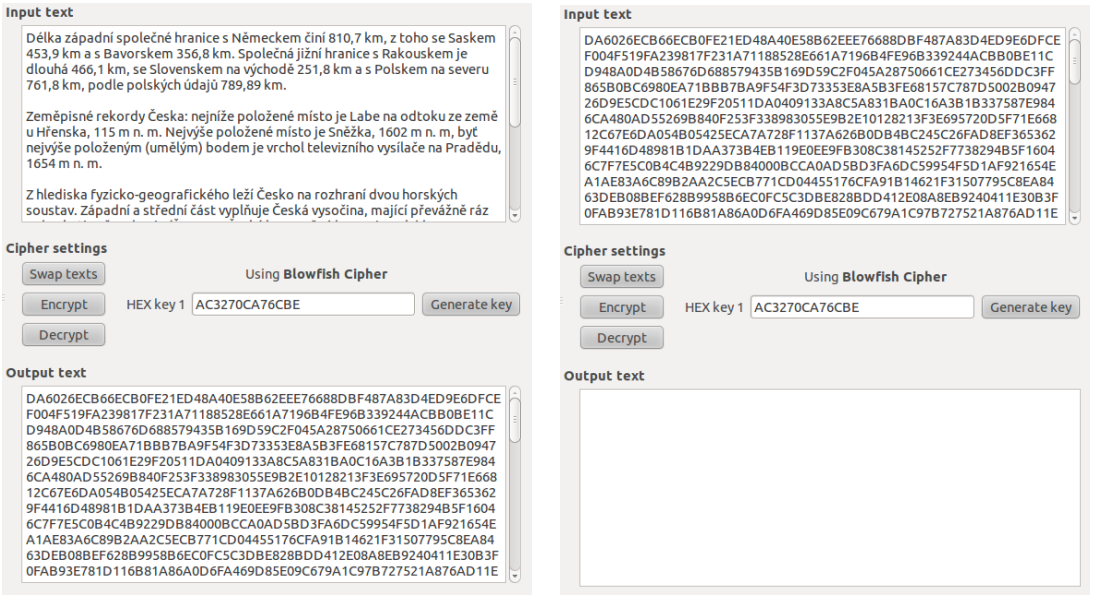
Obrázek B.4: Generování klíče.

4. Do pole **Input text** se zadává text určený k zašifrování nebo dešifrování a po stisku tlačítka **Encrypt** nebo **Decrypt** se spustí požadovaný proces. Pokud jsou klíče i vstupní text správně zadány, v poli **Output text** se vypíše zašifrovaný (resp. dešifrovaný) text (viz obrázek B.5(a)). Pokud je vstupní text uložen v textovém souboru s příponou `txt`, pak jej lze načíst do pole **Input text** pomocí klávesové zkratky `Ctrl+O`, případně pomocí nabídky `File / Open`. Stejně tak výstupní text je možné uložit do textového souboru pomocí klávesové zkratky `Ctrl+S`, případně pomocí nabídky `File / Save` (viz obrázek B.2(a)).

5. Pro testování jiné šifry lze pokračovat znovu krokem 2. Pro ukončení aplikace lze použít klávesovou zkratku `Ctrl+Q`, případně nabídku `File / Quit` (viz obrázek B.2(a)).

Aplikace nabízí i další možnosti:

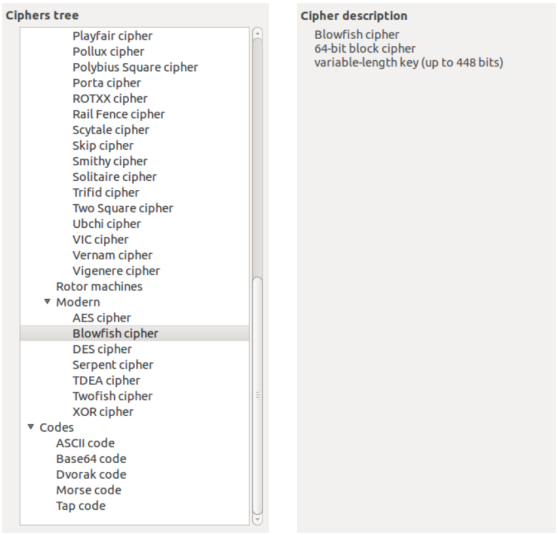
- Pro ověření správnosti šifrování a dešifrování je možné použít tlačítko **Swap texts**, které výstupní text vloží do pole vstupního textu a výstupní text smaže (viz obrázek B.5(b)).



(a) Šifrování vstupního textu. (b) Prohození textových polí.

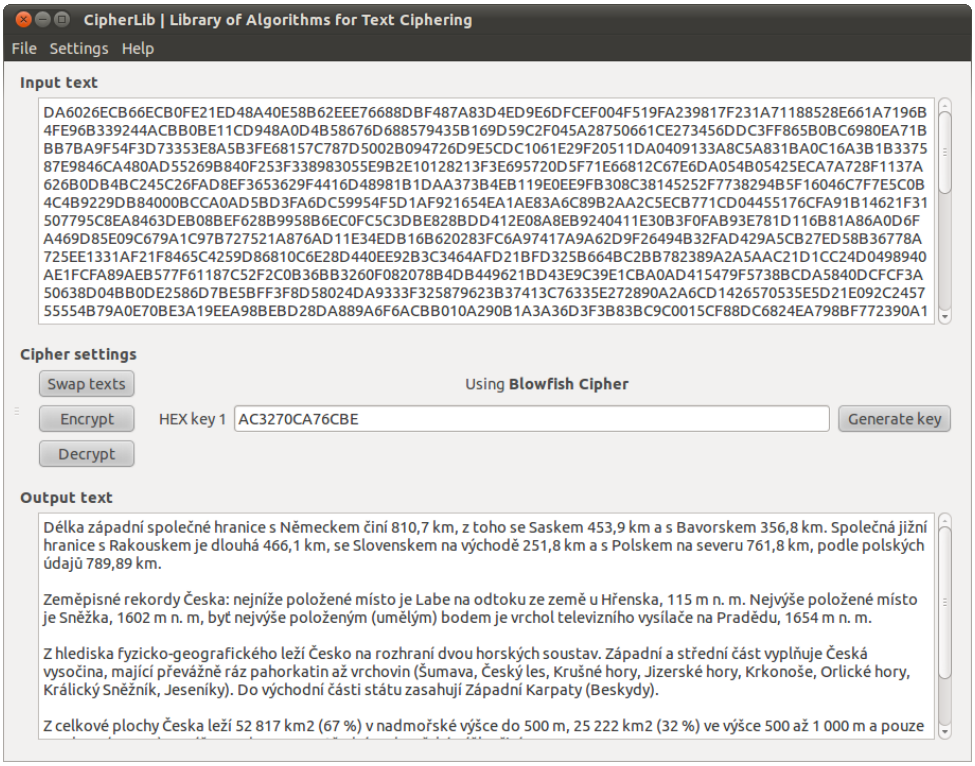
Obrázek B.5: Práce s textovými poli.

- Pole s informacemi o šifře **Cipher description** je možné skrýt (viz obrázek B.6(a)) tažením myší, klávesovou zkratkou `Ctrl+D` nebo pomocí nabídky `Settings / Show cipher description` (viz obrázek B.2(b)). Stejně tak seznam šifer – toto pole lze také skrýt (viz obrázek B.6(b)) tažením myši, klávesovou zkratkou `Ctrl+T` nebo pomocí nabídky `Settings / Show ciphers tree` (viz obrázek B.2(b)). Samozřejmě lze skrýt obě tato pole najednou (viz obrázek B.7).



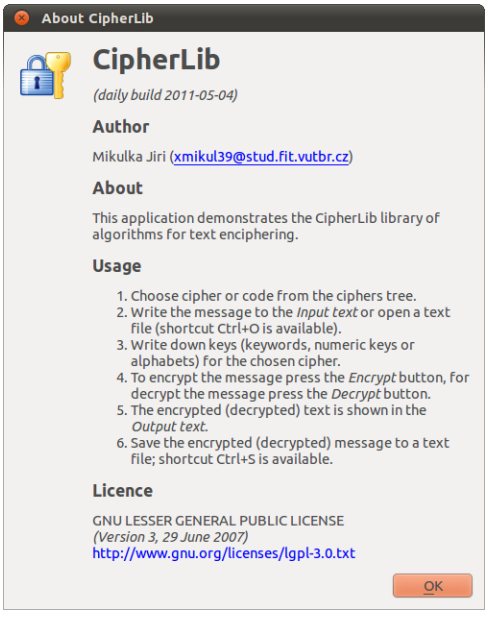
(a) Skrytí popisu šifry. (b) Skrytí seznamu šifer.

Obrázek B.6: Skrývání seznamu šifer a popisu šifry.



Obrázek B.7: Skrytí popisu šifry i seznamu šifer.

- Nápovědu k aplikaci (viz obrázek B.8) lze otevřít klávesou `F1` nebo pomocí nabídky `Help / About CipherLib` (viz obrázek B.2(c)).



Obrázek B.8: Nápověda k aplikaci.