

Aplikace symetrické kryptografie

Application of Symmetric Cryptography

Bc. Tomáš Novosád

Diplomová práce
2014



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně

Fakulta aplikované informatiky

akademický rok: 2013/2014

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Tomáš Novosád**
Osobní číslo: **A12392**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Informační technologie**
Forma studia: **prezenční**

Téma práce: **Aplikace symetrické kryptografie**

Zásady pro vypracování:

1. Vypracujte literární rešerši na dané téma.
2. Seznamte se s různými dostupnými metodami symetrické kryptografie.
3. Vytvořte ve zvoleném prostředí (C#, JAVA, Mathematica s vlastním GUI, atd.) aplikaci zahrnující komplexní systém konvenčních i moderních symetrických šifrovacích algoritmů.
4. Vytvořte v rámci aplikace možnosti vícenásobného šifrování různými systémy a základní práce se soubory.
5. Otestujte aplikaci na zvolené sadě příkladů šifrových textů a souborů.
6. Prostudujte možnosti budoucího modulárního rozšíření, exportu a importu dat.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. PAAR, Christof. Understanding cryptography: a textbook for students and practitioners. Heidelberg: Springer, 2010, xviii, 372 s. ISBN 978-3-642-04100-6.
2. MOLLIN, Richard A. An introduction to cryptography. 2nd ed. Boca Raton: Chapman, 2007, x, 413 s. ISBN 15-848-8618-8.
3. KATZ, Jonathan a Yehuda LINDELL. Introduction to modern cryptography. Boca Raton: Chapman, 2008, xviii, 534 s. ISBN 978-1-58488-551-1.
4. PACHGHARE, V.K. a Yehuda LINDELL. Cryptography and information security. New Delhi: Chapman, 2009, xviii, 534 s. ISBN 978-812-0335-219.
5. ZELENKA, Josef. Ochrana dat: kryptologie. Vyd. 1. Hradec Králové: Gaudeamus, 2003, 198 s. ISBN 80-704-1737-4.
6. SWENSON, Christopher. Modern cryptanalysis: techniques for advanced code breaking. Indianapolis: Wiley, 2008, xxviii, 236 s. ISBN 978-0-470-13593-8.
7. PIPER, F a Sean MURPHY. Kryptografie. 1. vyd. v českém jazyce. Překlad Pavel Mondschein. Praha: Dokořán, 2006, 157 s. ISBN 80-736-3074-5.
8. BITTO, Ondřej. Šifrování a biometrika aneb tajemné bity a dotyky. Vyd. 1. Kralice na Hané: Computer Media, 2005, 168 s. ISBN 80-866-8648-5.

Vedoucí diplomové práce:

doc. Ing. Roman Šenkeřík, Ph.D.

Ústav informatiky a umělé inteligence

Datum zadání diplomové práce:

21. února 2014

Termín odevzdání diplomové práce:

20. května 2014

Ve Zlíně dne 21. února 2014

prof. Ing. Vladimír Vašek, CSc.
děkan



doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

ABSTRAKT

Cílem této diplomové práce bylo vytvořit komplexní systém pro šifrování a dešifrování textů a souborů pomocí symetrických šifer. Aplikace byla následně otestována na několika textech a souborech. V teoretické části jsou podrobně popsány principy některých symetrických šifrovacích algoritmů, zejména těch, které jsou použity ve vytvořené aplikaci. V praktické části je popsána vytvořená aplikace, její funkce a možnost budoucího rozšíření aplikace o další šifry.

Klíčová slova: kryptografie, symetrická kryptografie, šifrování, dešifrování

ABSTRACT

The objective of this work was to create complex system for encryption and decryption of texts and files with symmetric ciphers. Application was tested with several texts and files. Theoretical part describes principles of some symmetric ciphers, especially ciphers, which are used in created application. Practical part describes created application, its functions and options for application expansion with new ciphers.

Keywords: cryptography, symmetric cryptography, encryption, decryption

Rád bych poděkoval vedoucímu mé práce doc. Ing. Romanu Šenkeříkovi ,Ph.D. za cenné názory a připomínky, které mi poskytl k vypracování této práce a své rodině za veškerou jejich podporu.

Prohlašuji, že

- беру на вѣдомі, же оdevзданіем дипломовѣ/бакалѣрскѣ прѣце souhlasím se zveřejněním své прѣце podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- беру на вѣдомі, же дипломовѣ/бакалѣрскѣ прѣце bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk дипломовѣ/бакалѣрскѣ прѣце bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího прѣце;
- был/а jsem seznámen/a s tím, že na moji дипломовую/бакалѣрskou прѣцу se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- беру на вѣдомі, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- беру на вѣдомі, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – дипломовую/бакалѣрskou прѣцу nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- беру на вѣдомі, že pokud bylo k vypracování дипломовѣ/бакалѣрскѣ прѣце využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky дипломовѣ/бакалѣрскѣ прѣце využít ke komerčním účelům;
- беру на вѣдомі, že pokud je výstupem дипломовѣ/бакалѣрскѣ прѣце jakýkoliv softwarový produkt, považují se za součást прѣце rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení прѣце.

Prohlašuji,

- že jsem na дипломовѣ прѣцу pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze дипломовѣ прѣце a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně

.....
podpis diplomanta

OBSAH

ÚVOD.....	10
I TEORETICKÁ ČÁST	11
1 KRYPTOLOGIE.....	12
1.1 ZÁKLADNÍ POJMY.....	12
1.2 KRYPTOGRAFIE	13
1.2.1 Symetrická kryptografie	13
1.2.2 Asymetrická kryptografie.....	14
1.2.3 Hybridní kryptografie.....	14
1.3 KRYPTOANALÝZA	16
1.3.1 Ciphertext-only attack	16
1.3.2 Known-plaintext attack	16
1.3.3 Chosen-plaintext attack.....	16
1.3.4 Chosen-ciphertext attack.....	16
1.3.5 Brute force attack	16
1.3.6 Dictionary attack	17
1.3.7 Birthday attack	17
2 SYMETRICKÁ KRYPTOGRAFIE	18
2.1 HISTORICKÉ ŠIFRY.....	18
2.1.1 Caesarova šifra	18
2.1.2 Vigenérova šifra	18
2.1.3 Autokláv	19
2.2 PROUDOVÉ ŠIFRY	20
2.2.1 Vernamova šifra	20
2.2.2 RC4	20
2.2.2.1 Key scheduling algorithm(KSA).....	21
2.2.2.2 Pseudo-random generation algorithm(PRGA).....	21
2.3 BLOKOVÉ ŠIFRY	22
2.3.1 Padding Modes.....	22
2.3.1.1 ANSI X.923	22
2.3.1.2 ISO 10126	22
2.3.1.3 PKCS7.....	22
2.3.1.4 Zero byte padding	23
2.3.2 Operační módy	23
2.3.2.1 ECB.....	23
2.3.2.2 CBC.....	24
2.3.2.3 CFB	25
2.3.2.4 OFB.....	26
2.3.2.5 CTR.....	27
2.3.3 DES	28
2.3.3.1 Odvození podklíčů	28
2.3.3.2 f – funkce	29
2.3.3.3 Šifrování.....	31

2.3.3.4	Dešifrování.....	32
2.3.4	Triple DES	32
2.3.4.1	Šifrování.....	33
2.3.4.2	Dešifrování.....	33
2.3.5	AES	33
2.3.5.1	Odvození podklíčů	34
2.3.5.2	Šifrování.....	36
2.3.5.3	Dešifrování.....	37
2.3.6	Blowfish	39
2.3.6.1	Odvození podklíčů	39
2.3.6.2	f – funkce	40
2.3.6.3	Šifrování.....	41
2.3.6.4	Dešifrování.....	42
2.3.7	Twofish	42
2.3.7.1	h – funkce.....	43
2.3.7.2	Odvození podklíčů	44
2.3.7.3	g – funkce.....	45
2.3.7.4	F – funkce	45
2.3.7.5	Šifrování.....	46
2.3.7.6	Dešifrování.....	47
2.3.8	Serpent.....	47
2.3.8.1	Odvození podklíčů	48
2.3.8.2	Šifrování.....	48
2.3.8.3	Dešifrování.....	49
II	PRAKTICKÁ ČÁST	50
3	POPIS APLIKACE	51
3.1	GRAFICKÉ ROZHRANÍ	52
3.2	VSTUPY A VÝSTUPY APLIKACE.....	54
3.2.1	Texty.....	54
3.2.2	Šifrovací kód	55
3.2.3	Soubory	56
3.3	VAROVNÉ ZPRÁVY	57
3.4	ČASOVÁ NÁROČNOST ŠIFROVÁNÍ.....	60
3.5	FUNKCE APLIKACE	61
3.5.1	Šifrování textu.....	61
3.5.2	Dešifrování textu.....	61
3.5.3	Vícenásobné šifrování textu.....	62
3.5.4	Vícenásobné dešifrování textu	64
3.5.5	Vícenásobné dešifrování textu pomocí šifrovacího kódu	65
3.5.6	Šifrování souboru	65
3.5.7	Dešifrování souboru	66
3.5.8	Vícenásobné šifrování souboru	67
3.5.9	Vícenásobné dešifrování souboru	68
3.5.10	Vícenásobné dešifrování souboru pomocí šifrovacího kódu	70
4	TESTOVÁNÍ APLIKACE.....	71

5	MOŽNOSTI MODULÁRNÍHO ROZŠÍŘENÍ APLIKACE	72
6	ANALÝZA EXISTUJÍCÍCH MOBILNÍCH APLIKACÍ.....	73
	ZÁVĚR.....	75
	ZÁVĚR V ANGLIČTINĚ	76
	SEZNAM POUŽITÉ LITERATURY	77
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	79
	SEZNAM OBRÁZKŮ	80
	SEZNAM TABULEK.....	83
	SEZNAM PŘÍLOH.....	84

ÚVOD

Využití kryptografie k šifrování zpráv se datuje dále, než by se běžnému člověku mohlo na první pohled zdát. První použití šifer k zašifrování zpráv se datuje ještě do doby před naším letopočtem. Nejznámějším případem je použití šifry, u které jsou všechny znaky abecedy posunuty o 3 znaky doprava. Tuto šifru využíval Julius Caesar pro komunikaci se svými vojsky. Podle něj dostala tato šifra název Caesarova šifra. Od té doby se kryptografie dále vyvíjela a byly postupně vynalezeny nové šifrovací algoritmy. Největšího vývoje se ovšem kryptografie dočkala až ve druhé polovině 20. století. V dnešní době je kryptografie využívána v mnoha oblastech lidského života.

Tato práce se zaměřuje na symetrickou kryptografii, která je dnes hojně používána. S neustále se zrychlujícím nárůstem výpočetního výkonu se však postupně některé šifry, nebo některé varianty délky klíče pro danou šifru stávají prolomitelnými útokem hrubou silou v dostatečně krátkém čase. Z tohoto důvodu jsou používány varianty šifer s delšími klíči. V dnešní době se také stále více objevují tzv. hybridní šifry, které jsou kombinací symetrické a asymetrické kryptografie a využívají jejich výhod jako je rychlost symetrické kryptografie a bezpečnost asymetrické kryptografie.

V této práci jsou na úvod vysvětleny některé kryptologické pojmy, které je třeba znát v další části práce, která se podrobněji zabývá symetrickou kryptografií. Podrobně vysvětlené principy jsou zde zejména u šifer, které jsou následně použity v naprogramované aplikaci.

Praktická část popisuje vytvořenou aplikaci, která je komplexním systémem pro šifrování a dešifrování textů a souborů pomocí symetrických šifer. Tato aplikace je naprogramována v programovacím jazyce C#.

I. TEORETICKÁ ČÁST

1 KRYPTOLOGIE

1.1 Základní pojmy

Kryptologie je věda zabývající se šifrováním, dešifrováním, prolamováním šifer. Dělí se na dva podobory a to kryptografii a kryptoanalýzu.

Kryptografie (z řeckého *kryptós* - „skrytý“ a *gráphein* - „psát“) je věda zabývající se utajováním obsahu zpráv, tak že je obsah zprávy převeden do nečitelné podoby a jen osoba se speciální znalostí (tj. znalost použitého šifrovacího algoritmu a klíče) je schopná z této nečitelné podoby získat původní zprávu. Podstatou kryptografie je navrhování a vytváření šifrovacích algoritmů.

Steganografie (z řeckého *steganós* - „schovaný“ a *gráphein* - „psát“) je věda zabývající se utajováním zpráv, tak aby případný útočník nepoznal, že mezi účastníky komunikace probíhá výměna nějakých tajných zpráv. Příkladem je např. použití neviditelného inkoustu, ukrytování dat do obrázku.

Kryptoanalýza (z řeckého *kryptós* – „skrytý“ a *analýein* – „uvolnit“ či „rozvázat“) je věda zabývající se prolamováním šifrovacích algoritmů, tak že daná osoba může získat obsah šifrované zprávy i bez znalosti klíče. Často je kryptoanalýza používána na testování odolnosti a zranitelnosti šifrovacích algoritmů.

Kryptoanalytik je člověk zabývající se kryptoanalýzou.

Šifrovací algoritmus (šifra) udává postup použití matematických operací na otevřeném textu k jeho převedení do nečitelné podoby.

Šifrování je použití šifrovacího algoritmu a tedy převedení otevřeného textu na šifrovaný text.

Dešifrování je postup opačný k šifrování a tedy převádí šifrovaný text na čitelnou podobu původní zprávy.

Otevřený text je zpráva (řetězec znaků), která bude použita k zašifrování.

Klíč je utajená informace sloužící k šifrování a dešifrování zprávy. Bezpečnost šifrovacího algoritmu zpravidla závisí právě na utajení klíče, tak aby ho znali jen osoby mezi kterými probíhá šifrovaná komunikace.

Šifrovaný text je výsledkem použití šifrovacího algoritmu na otevřený text.

1.2 Kryptografie

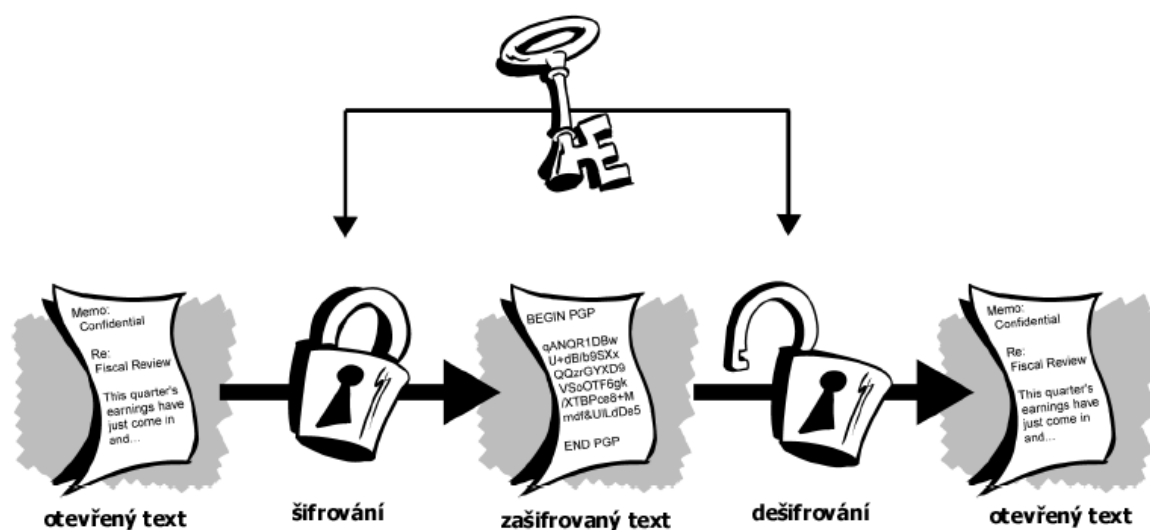
Kryptografie je věda, která se zabývá navrhováním nových šifrovacích algoritmů. Můžeme ji rozdělit na symetrickou, asymetrickou a hybridní.



Obr. 1 Základní princip šifrování a dešifrování [8]

1.2.1 Symetrická kryptografie

Šifrovací algoritmus je označován jako symetrický, pokud lze dešifrovací klíč snadno odvodit z klíče šifrovacího. V praxi bývají oba klíče u symetrických šifrovacích algoritmů identické. Z tohoto důvodu bývají takovéto šifrovací algoritmy označovány za algoritmy s tajným klíčem nebo také jako algoritmy jednoklíčové. Na rozdíl od asymetrických šifrovacích algoritmů je tedy nutné držet šifrovací i dešifrovací klíč v tajnosti. [2]

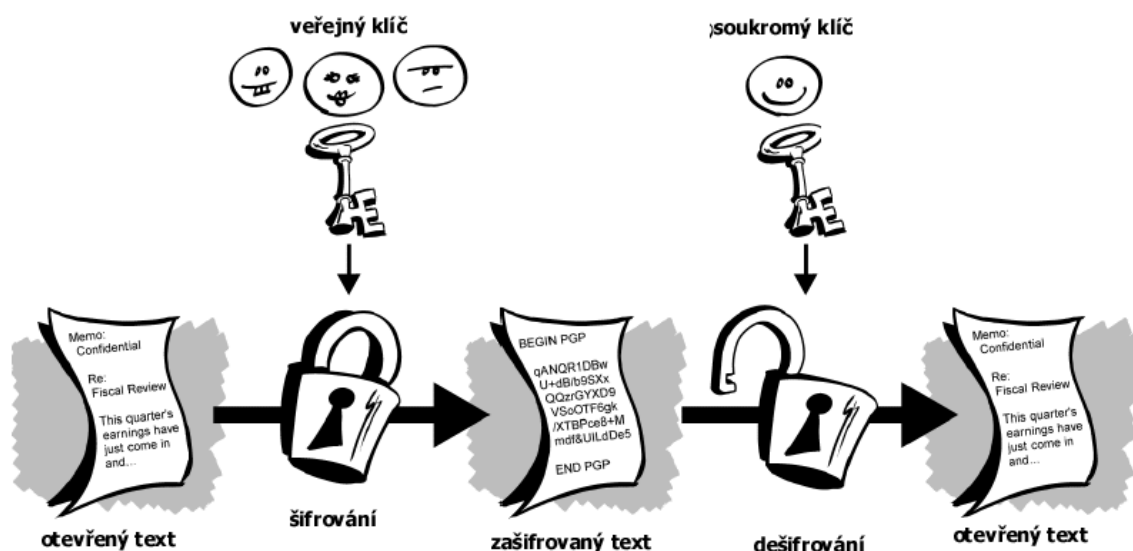


Obr. 2 Princip šifrování a dešifrování pomocí symetrického šifrovacího algoritmu [8]

Mezi výhody symetrické kryptografie patří jednoduchost šifrovacího i dešifrovacího algoritmu a rychlost. Naopak mezi nevýhody patří nutnost přenosu klíče mezi účastníky komunikace a tedy i nutnost uchování tohoto klíče v tajnosti. Další nevýhodou je velký počet klíčů při komunikaci s více osobami. [2]

1.2.2 Asymetrická kryptografie

Šifrovací algoritmus je označován jako asymetrický, pokud dešifrovací klíč nelze odvodit z klíče šifrovacího. Asymetrické šifrovací algoritmy bývají také často nazývány jako šifrovací algoritmy s veřejným klíčem. Šifrovací klíč může být, a často také bývá, veřejný. Z toho tedy vyplývá, že stačí držet v tajnosti pouze dešifrovací klíč. [2]



Obr. 3 Princip šifrování a dešifrování pomocí asymetrického šifrovacího algoritmu [8]

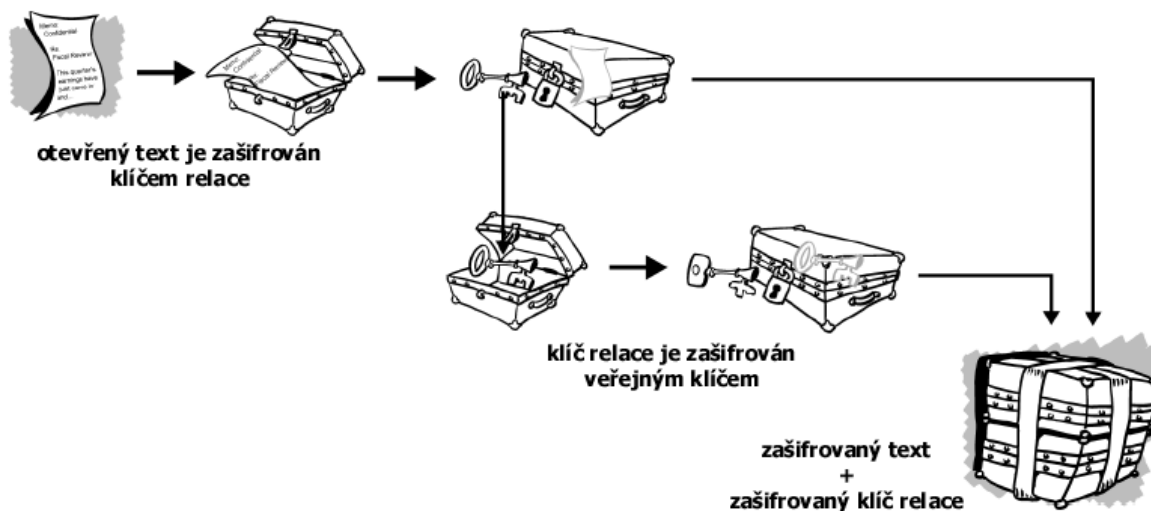
Hlavní výhodou asymetrické kryptografie je, že si účastníci komunikace nemusí mezi sebou poslat dešifrovací klíč. Mezi nevýhody patří vysoké výpočetní nároky a nízká rychlost. [2]

Nejznámějšími asymetrickými šiframi jsou RSA a El Gamal.

1.2.3 Hybridní kryptografie

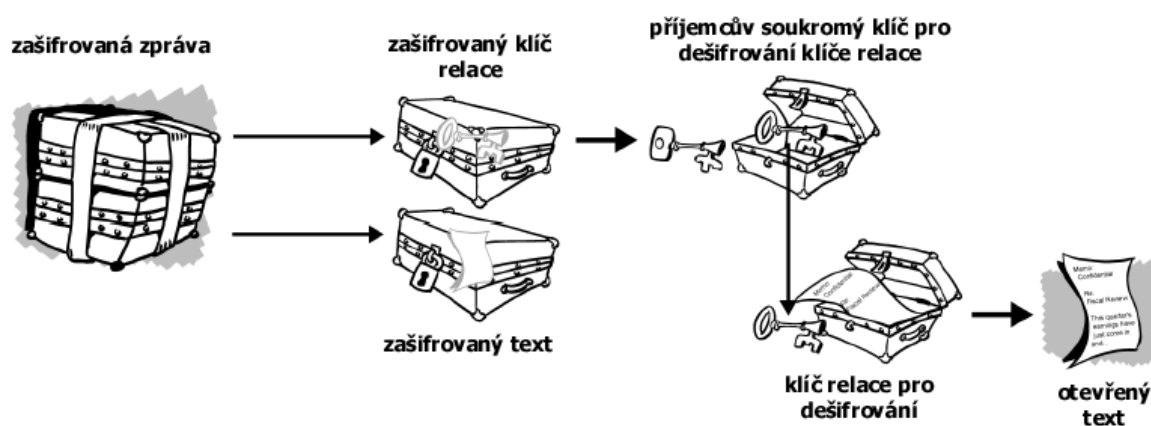
Hybridní kryptografie využívá výhod symetrické i asymetrické kryptografie. Používá se zejména kvůli vysokému výpočetnímu nároku asymetrické kryptografie.

Princip šifrování je takový, že text nebo data se zašifrují symetrickou šifrou, symetrický klíč se zašifruje veřejným klíčem z klíčového páru asymetrické šifry. Takto zašifrovaný klíč se posílá spolu se zašifrovanou zprávou příjemci. [8]



Obr. 4 Princip šifrování hybridní kryptografie [8]

Princip dešifrování je takový, že příjemce po přijetí zašifrované zprávy dešifruje zašifrovaný symetrický klíč pomocí soukromého klíče z klíčového páru asymetrické šifry. Příjemce poté zašifrovanou zprávu dešifruje pomocí dešifrovaného symetrického klíče. [8]



Obr. 5 Princip dešifrování hybridní kryptografie [8]

Zástupcem hybridní kryptografie je šifra PGP.

1.3 Kryptoanalýza

Kryptoanalýza se zabývá prolamováním šifrovacích algoritmů a teda snaží se získat otevřený text ze zašifrovaného textu bez znalosti použitého šifrovacího algoritmu a klíče, nebo jen klíče. Kryptoanalytických útoků existuje celá řada.

1.3.1 Ciphertext-only attack

Toto je typ útoku, kdy útočník zná pouze šifrovaný text. V tomto případě je často používána frekvenční analýza jednotlivých znaků šifrovaného textu. Většina moderních šifrovacích algoritmů je odolná vůči těmto útokům. [3]

1.3.2 Known-plaintext attack

Toto je typ útoku, kdy útočník zná část otevřeného textu a k němu jeho zašifrovanou formu. Útočník se pomocí těchto informací snaží dešifrovat zbytek šifrovaného textu, čehož může dosáhnout, když ze známých informací uhodne klíč. [3]

1.3.3 Chosen-plaintext attack

Toto je typ útoku, kdy je útočník schopný mít jakýkoliv otevřený text a k němu příslušný šifrovaný text. Hlavním úkolem je zjistit klíč z těchto informací. [3]

1.3.4 Chosen-ciphertext attack

Toto je typ útoku, kdy je útočník schopný mít jakýkoliv šifrovaný text. Útočník je schopný zavést do systému jakýkoliv šifrovaný text a z nasbíraných informací se snaží uhádnout tajný dešifrovací klíč. Tento typ útoku se používá zejména na asymetrické šifrovací algoritmy. [3]

1.3.5 Brute force attack

Toto je typ útoku, kdy útočník zkouší všechny možné kombinace klíče, dokud ze šifrovaného textu nezíská otevřený text. [3]

1.3.6 Dictionary attack

Toto je typ útoku, kdy útočník zkouší k dešifrování předem dané kombinace klíčů. V podstatě se jedná o Brute force attack, ale útočník nezkouší všechny možné kombinace klíčů [3]

1.3.7 Birthday attack

Toto je typ útoku, kdy se útočník snaží narušit komunikaci a najít kolizi. Používá se zejména u hašovacích funkcí. Tento útok je založen na tzv. Narozeninovém problému z teorie pravděpodobnosti, který říká, že pro skupinu náhodně vybraných 23 (či více) lidí, je více než 50% pravděpodobnost, že nějací dva lidé budou mít narozeniny ve stejný den.[3]

2 SYMETRICKÁ KRYPTOGRAFIE

2.1 Historické šifry

2.1.1 Caesarova šifra

Caesarova šifra, neboli také šifra s pevným posunem je speciálním případem substitučních šifer. Implementace této šifry je velice jednoduchá: každý znak otevřeného textu je posunut o pevný počet míst v abecedě. Např. pokud bude posun o tři pozice, tak A bude zašifrováno jako D, B bude zašifrováno jako E. Pokud se dostaneme na konec abecedy, tak se šifra chová tak, že konec abecedy navazuje na začátek abecedy, tzn. X bude zašifrováno jako A, Y bude zašifrováno jako B a Z bude zašifrováno jako C. [4]

$$\text{Šifrování : } e_k(x) = x + k \bmod 26$$

$$\text{Dešifrování : } d_k(y) = y - k \bmod 26$$

Obr. 6 Šifrování a dešifrování pomocí Caesarovy šifry [4]

Šifra pracuje s čísly pozic jednotlivých znaků otevřeného textu, šifrovaného textu a klíče. Na obr. 6 tedy znak x reprezentuje pozici znaku otevřeného textu v abecedě, znak k reprezentuje pozici znaku klíče v abecedě a znak y reprezentuje pozici znaku šifrovaného textu v abecedě.

Na tuto šifru lze použít dva druhy kryptoanalytického útoku a to útok hrubou silou, který je v tomto případě účinný, jelikož klíč může nabývat pouze 26 hodnot. Druhým útokem je frekvenční analýza jednotlivých znaků šifrovaného textu. [4]

2.1.2 Vigenérova šifra

Vigenérova šifra patří mezi polyalfabetické substituční šifry. Princip této šifry je podobný jako u Caesarovy šifry s tím rozdílem, že Vigenérova šifra pracuje s 26 posunutými abecedami. K šifrování a dešifrování se používá tzv. Vigenérova tabulka, nazývaná též jako Vigenerův čtverec nebo Tabula recta. [7]

	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

Obr. 7 Vigeněrův čtverec [9]

Šifrování probíhá tak, že pokud klíč zadaný uživatelem nemá délku otevřeného textu, je doplněn tak, že se daný klíč opakuje za sebou, než je dosažena délka otevřeného textu. Poté se šifruje pomocí Vigeněrova čtverce tak, že se vezme znak otevřeného textu, který je ve Vigeněrově čtverci reprezentován na vertikální ose a znak klíče, který je ve Vigeněrově čtverci reprezentován na horizontální ose. Provede se průsečík těchto dvou písmen v tabulce, který udává zašifrovaný znak. Při dešifrování se postupuje tak, že na horizontální ose najdeme znak klíče a v tomto sloupci najdeme zašifrovaný znak a od tohoto znaku si vyneseme přímkou doleva a tak zjistíme znak otevřeného textu. [7]

Příklad doplnění klíče:

Otevřený text: SEJDEMESEUMOSTU (délka 15)

Klíč: KCLICKCLICKLI (délka 15)

Při zadání klíče o délce 1 se jedná o Caesarovu šifru. Obecně platí, že čím delší klíč, tím je šifra bezpečnější. [7]

2.1.3 Autokláv

Autokláv funguje stejně jako Vigeněrova šifra, jen s tím rozdílem, že klíč není v tomto případě doplňován opět stejným klíčem, ale je doplňován otevřeným textem. [1]

Příklad doplnění klíče:

Otevřený text: SEJDEMESEUMOSTU (délka 15)

Klíč: KLICSEJDEMESEUM (délka 15)

2.2 Proudové šifry

Proudové šifry pracují tak, že zpracovávají otevřený text znak po znaku, nebo také bit po bitu. Proudové šifry jsou rychlejší a jednodušší na implementaci než šifry blokové. Jejich nevýhodou je nižší odolnost vůči kryptoanalytickým útokům. [1]

Klíč je u těchto šifer zpravidla generován generátorem pseudonáhodných čísel. Proudové šifry se dají rozdělit na dvě skupiny a to synchronní, které generují jednotlivé prvky klíče bez závislosti na šifrovaném textu a asynchronní, které generují jednotlivé prvky klíče v závislosti na šifrovaném textu. [4]

Mezi zástupce této skupiny patří Vernamova šifra, FISH a RC4.

2.2.1 Vernamova šifra

V roce 1917 Gilbert Vernam patentoval tuto šifru. Šifra pracuje tak, že se vygeneruje náhodný či pseudonáhodný klíč stejné délky jako je délka otevřeného textu. Šifrování probíhá tak, že se provede operace XOR mezi otevřeným textem a vygenerovaným klíčem. Dešifrování funguje stejně, jen je operace XOR použita na šifrovaný text a klíč. Tato šifra se považuje za neprolomitelnou. Klíč nesmí být použitý pro šifrování vícekrát, jinak se šifra stává náchylnější kryptoanalytickým útokům, díky čemuž šifrovaný text neposkytuje žádné informace o otevřeném textu. To znamená, že šifrovaný text může být se stejnou pravděpodobností jakýkoliv otevřený text stejné délky. V roce 1949 publikoval Claude Shannon článek, ve kterém dokazuje, že při správném použití je Vernamova šifra neprolomitelná. [5]

2.2.2 RC4

Šifru RC4 vynalezl v roce 1987 Ron Rivest. Zdrojové kódy a popis šifry nebyly přístupné veřejnosti až do roku 1994, kdy byl popis algoritmu zveřejněn na internetu. Je to jedna z nejpoužívanějších proudových šifer díky své jednoduchosti, rychlosti a jednoduché implementaci. Používá se k šifrování internetové komunikace v síťových protokolech SSL, TLS, WEP, WPA atd. Tato šifra je velmi podobná Vernamově šifře. Rozdíl je v tom, jak je

generován klíč. Klíč je u této šifry generován pomocí dvou algoritmů, které jsou provedeny za sebou, a to KSA(Key scheduling algorithm) a PRGA(Pseudo-random generation algorithm). Šifrování probíhá stejně jako u Vernamovy šifry, a to tedy tak, že je provedena operace XOR na otevřený text a vygenerovaný klíč. [6]

2.2.2.1 Key scheduling algorithm(KSA)

Na začátku algoritmu je inicializováno pole S hodnotami 0 až 255 ($N = 256$). Klíč K zadává uživatel, který má obvykle délku 5 až 16 B. To znamená, že např. pro klíč délky 5 je vytvořeno pole $K[0] \dots K[4]$. Tento klíč je následně použit k promíchání prvků pole S . Po promíchání je toto pole použito pro generování klíče pomocí algoritmu PRGA. [6]

```

Inicializace :
for  $i = 0, \dots, N - 1$  do
     $S[i] = i$ ;
     $j = 0$ ;
end
Zamíchání prvků :
for  $i = 0, \dots, N - 1$  do
     $j = (j + S[i] + K[i])$ ;
    prohození prvků( $S[i], S[j]$ );
end

```

Obr. 8 Pseudokód algoritmu KSA [6]

2.2.2.2 Pseudo-random generation algorithm(PRGA)

Tento algoritmus používá promíchané pole S z předchozího algoritmu KSA ke generování jednotlivých bajtů klíče pro zašifrování zprávy. [6]

```

 $i = j = 0$ ;
Generování streamu klíčl :
 $i = i + 1$ ;
 $j = j + S[i]$ ;
prohození prvků( $S[i], S[j]$ );
 $t = S[i] + S[j]$ ;
Výstup :  $z = S[t]$ ;

```

Obr. 9 Pseudokód algoritmu PRGA [6]

2.3 Blokové šifry

Blokové šifry pracují tak, že zpracovávají otevřený text v blocích pevné délky, která se liší v závislosti na použitém šifrovacím algoritmu. Nejčastěji používané délky bloků jsou 64 bitů a 128 bitů. [4]

Pokud otevřený text nemá délku násobku délky bloku použitého šifrovacího algoritmu, tak je třeba otevřený text doplnit, tak aby i poslední blok měl danou délku. Toto doplnění lze realizovat několika metodami, kterým se říká metody doplnění neboli „Padding modes“.

Blokové šifry mohou pracovat v několika módech a to ECB, CBC, CFB, OFB, CTR.

2.3.1 Padding Modes

Všechny metody si názorně ukážeme na příkladu, kdy budeme uvažovat bloky o délce 64 bitů a otevřený text bude obsahovat 80 bitů a to FF FF FF FF FF FF FF FF FF FF. Z toho vyplývá, že otevřený text musí být doplněn na 128 bitů neboli 16 bajtů.

2.3.1.1 ANSI X.923

U této metody je otevřený text doplňován bajty s hodnotou 00 a poslední bajt udává, kolik bylo doplněno bajtů. [10]

Otevřený text: FF FF FF FF FF FF FF FF FF FF

Otevřený text po doplnění: FF FF FF FF FF FF FF FF FF FF 00 00 00 00 00 06

2.3.1.2 ISO 10126

U této metody je otevřený text doplňován bajty s náhodnou hodnotou a poslední bajt udává, kolik bylo doplněno bajtů. [10]

Otevřený text: FF FF FF FF FF FF FF FF FF FF

Otevřený text po doplnění: FF FF FF FF FF FF FF FF FF FF 5A B1 59 8F 02 06

2.3.1.3 PKCS7

U této metody je otevřený text doplňován bajty s hodnotou udávající, kolik bylo doplněno bajtů. [10]

Otevřený text: FF FF FF FF FF FF FF FF FF FF

Otevřený text po doplnění: FF FF FF FF FF FF FF FF FF FF 06 06 06 06 06 06

2.3.1.4 Zero byte padding

U této metody je otevřený text doplňován bajty s hodnotou 00.

Otevřený text: FF FF FF FF FF FF FF FF FF FF

Otevřený text po doplnění: FF FF FF FF FF FF FF FF FF FF 00 00 00 00 00 00

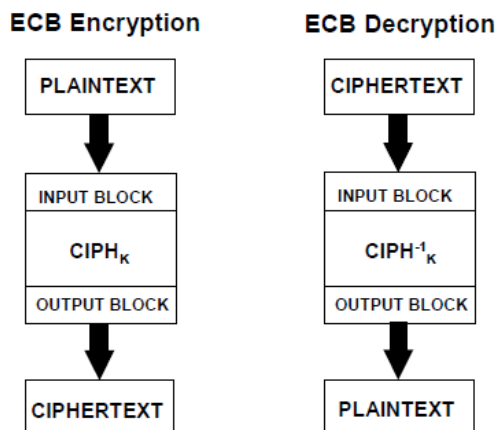
Tato metoda se nedoporučuje používat, protože při dešifrování a případném následném odstranění doplněných bajtů algoritmus odstraní všechny bajty s hodnotou 00 na konci dešifrovaného textu (souboru), i když na konci textu mohlo být i před doplněním a zašifrováním několik bajtů s hodnotou 00. Může se tedy stát, že se odstraní i bajty, které byly součástí textu (souboru). [10]

2.3.2 Operační módy

Operačních módů pro symetrické blokové šifrovací algoritmy existuje pět, které jsou doporučeny organizací NIST – Národní institut standardů a technologie. Zajímavostí je, že módy OFB a CTR používají blokovou šifru jako proudovou. [1]

2.3.2.1 ECB

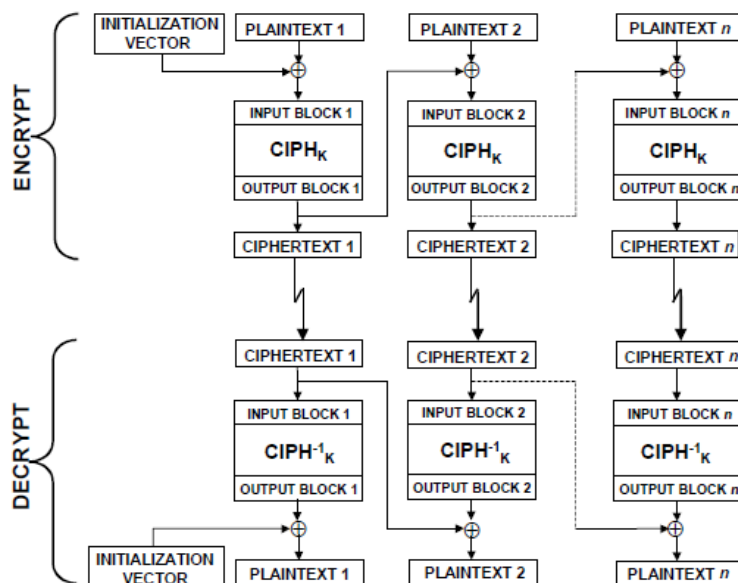
ECB (Electronic codebook) je nejjednodušším z operačních módů, ale zároveň nejzranitelnějším. Na obr. 10 je naznačen princip šifrování a dešifrování blokové šifry v módu ECB (Plaintext – otevřený text, Input block – vstupní blok dat, Ciph – šifrovací algoritmus, Output block – výstupní blok z šifrovacího algoritmu, Cipher text – šifrovaný text). Pracuje tak, že všechny bloky textu jsou šifrovány stejným klíčem. Slabinou tohoto módu je, že pokud šifrujeme otevřený text stejným klíčem, dostaneme vždy stejný šifrovaný text. Na základě tohoto můžeme říci, že je tento mód deterministický. Kvůli této vlastností se tento mód nedoporučuje používat pro šifrování velkého množství dat. Nejčastěji se tento mód používá pro zašifrování klíče. Na rozdíl od ostatních módů, se u ECB neprojevuje šíření chyby. [4]



Obr. 10 Princip šifrování a dešifrování módu ECB [11]

2.3.2.2 CBC

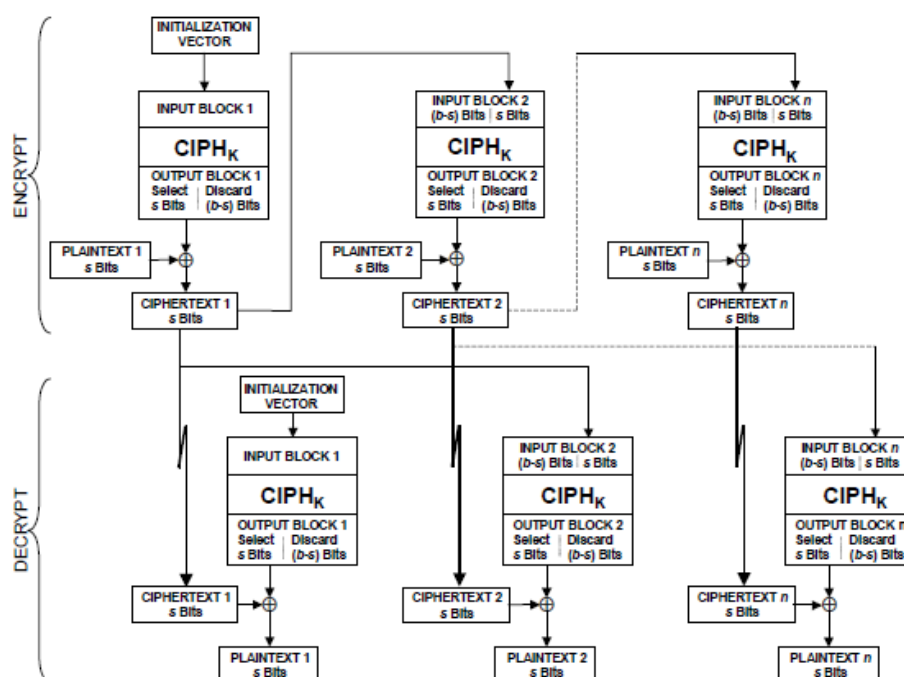
Na obr. 11 je naznačen princip šifrování a dešifrování blokové šifry v módu CBC (Encrypt – zašifrovat, Decrypt – dešifrovat, Plaintext – otevřený text, Initialization vector – inicializační vektor, Input block – vstupní blok dat, Ciph – šifrovací algoritmus, Output block – výstupní blok z šifrovacího algoritmu, Cipher text – šifrovaný text). CBC (Cipher block chaining) neboli „řetězení šifrovaných bloků“ je mód, který pracuje tak, že na začátku algoritmu je použita operace XOR (exclusive OR) na blok otevřeného textu a inicializační vektor, který je náhodně generován. Poté je takto zpracovaný blok zašifrován. Na další blok je před zašifrováním také použita operace XOR, ale ta je použita na blok otevřeného textu a zašifrovaný předchozí blok. Takto zpracovaný blok je zašifrován. Stejným způsobem, jakým byl zašifrován druhý blok, se zašifrují všechny ostatní bloky. Z tohoto vyplývá, že u tohoto módu dochází k řetězení chyby. Na rozdíl od módu ECB, zde nedojde k případu, že by zašifrování otevřeného textu stejným klíčem vedlo ke stejnému výstupu. Toto zajišťuje náhodně generovaný inicializační vektor a následné řetězení v průběhu šifrování jednotlivých bloků. Lze tedy říci, že je tento mód nedeterministický. [4]



Obr. 11 Princip šifrování a dešifrování módu CBC [11]

2.3.2.3 CFB

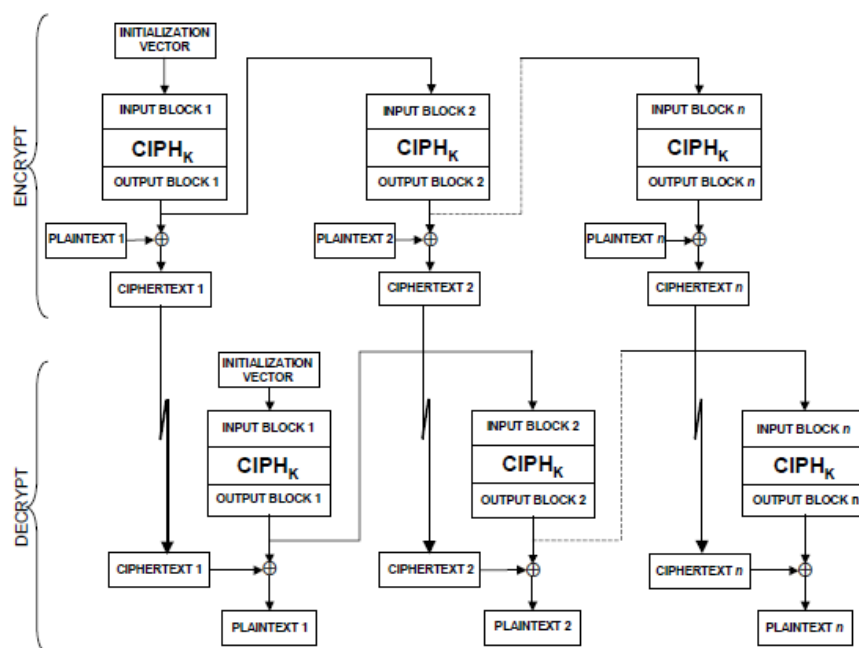
Na obr. 12 je naznačen princip šifrování a dešifrování blokové šifry v módu CFB (Encrypt – zašifrovat, Decrypt – dešifrovat, Plaintext – otevřený text, Initialization vector – inicializační vektor, Input block – vstupní blok dat, Ciph – šifrovací algoritmus, Output block – výstupní blok z šifrovacího algoritmu, Cipher text – šifrovaný text). CFB (Cipher feedback) je mód, který pracuje tak, že na začátku algoritmu je zašifrován inicializační vektor. Po jeho zašifrování je použita operace XOR na blok otevřeného textu a zašifrovaný inicializační vektor. Takto vytvořený zašifrovaný blok otevřeného textu je poté zašifrován a pomocí operace XOR je sloučen s dalším blokem otevřeného textu, čímž získáme zašifrovaný druhý blok otevřeného textu. Stejným způsobem, jakým byl zašifrován druhý blok, se zašifrují všechny ostatní bloky. Stejně jako mód CBC je i mód CFB nedeterministický. [4]



Obr. 12 Princip šifrování a dešifrování módu CFB [11]

2.3.2.4 OFB

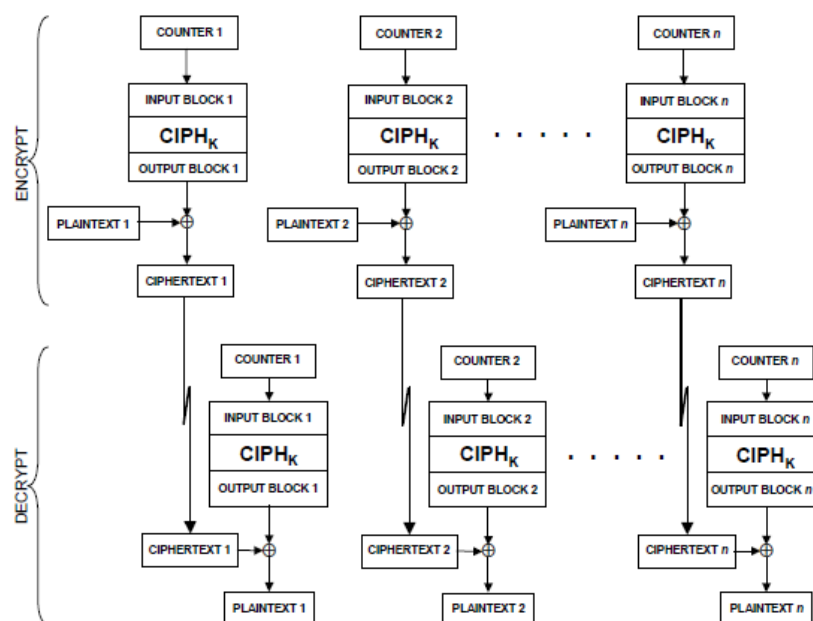
Na obr. 13 je naznačen princip šifrování a dešifrování blokové šifry v módu OFB (Encrypt – zašifrovat, Decrypt – dešifrovat, Plaintext – otevřený text, Initialization vector – inicializační vektor, Input block – vstupní blok dat, Ciph – šifrovací algoritmus, Output block – výstupní blok z šifrovacího algoritmu, Cipher text – šifrovaný text). OFB (Output feedback) mód je téměř stejný jako mód CFB. OFB má oproti CFB jedinou změnu a to místo zdroje zpětné vazby. Mód OFB tedy pracuje tak, že na začátku algoritmu je zašifrován inicializační vektor. Po jeho zašifrování je použita operace XOR na blok otevřeného textu a zašifrovaný inicializační vektor, čímž získáváme zašifrovaný první blok otevřeného textu. Zde na rozdíl od módu CFB je jako vstup použit zašifrovaný inicializační vektor, který je zašifrován a pomocí operace XOR je sloučen s dalším blokem otevřeného textu, čímž získáme zašifrovaný druhý blok otevřeného textu. Stejným způsobem, jakým byl zašifrován druhý blok, se zašifrují všechny ostatní bloky. Tento mód je taktéž nedeterministický. Výhodou OFB módu je, že výpočty blokové šifry nejsou závislé na otevřeném textu, s proto lze si předpřipravit zašifrování vstupů a následně je jen sloučit pomocí operace XOR s bloky otevřeného textu. [4]



Obr. 13 Princip šifrování a dešifrování módu OFB [11]

2.3.2.5 CTR

Na obr. 14 je naznačen princip šifrování a dešifrování blokové šifry v módu CTR (Encrypt – zašifrovat, Decrypt – dešifrovat, Plaintext – otevřený text, Counter – čítač, Input block – vstupní blok dat, Ciph – šifrovací algoritmus, Output block – výstupní blok z šifrovacího algoritmu, Cipher text – šifrovaný text). CTR (Counter mode) je mód, který k docílení nedeterministického chování využívá čítač. Existují dvě varianty, jak je čítač vytvářen. První variantou je, že je na začátku algoritmu určena hodnota čítače a s jednotlivými kroky je tato hodnota inkrementována. Druhou variantou je, že je hodnota čítače určena náhodně pro každý krok algoritmu. Šifrování probíhá tak, že je zašifrován čítač. Poté je použita operace XOR na tento zašifrovaný čítač a blok otevřeného textu. Tímto vznikne zašifrovaný blok otevřeného textu. Tento proces se opakuje pro všechny bloky otevřeného textu. CTR mód je považován za nejbezpečnější operační mód. Další výhodou je, že pro šifrování i dešifrování stačí stejný algoritmus. Jedinou změnou v dešifrování oproti šifrování je to, že do operace XOR vstupuje blok zašifrovaného textu namísto bloku otevřeného textu.[4]



Obr. 14 Princip šifrování a dešifrování módu CTR [11]

2.3.3 DES

DES (Data Encryption Standard) je symetrická bloková šifra, která byla zveřejněna roku 1977. Původně byla tato šifra vyvinuta v 60. letech firmou IBM a měla název Lucifer.

Tato šifra pracuje s bloky dat o velikosti 64 bitů, k šifrování a dešifrování využívá klíč o délce 64 bitů, z čehož je každý osmý bit paritní, takže je použito jen 56 bitů tohoto klíče. Jedná se o iterativní algoritmus, kde je každý blok otevřeného textu zašifrován v 16 rundových funkcích. Algoritmus využívá dvou kryptografických technik a to permutace a substituce.

Na začátku algoritmu je třeba z klíče odvodit 16 podklíčů, které jsou následně využity v rundových funkcích. [4]

2.3.3.1 Odvození podklíčů

Na začátku se odstraní paritní bity neboli každý osmý bit klíče a navíc se změní pořadí zbylých 56 bitů podle počáteční permutace zobrazené na obr. 15.

<i>PC – 1</i>							
57	49	41	33	25	17	9	1
58	50	42	34	26	18	10	2
59	51	43	35	27	19	11	3
60	52	44	36	63	55	47	39
31	23	15	7	62	54	46	38
30	22	14	6	61	53	45	37
29	21	13	5	28	20	12	4

Obr. 15 Počáteční permutace klíče šifry DES [4]

Po této permutaci je klíč rozdělen na dvě poloviny o délce 28 bitů. Poté jsou bity každé poloviny posunuty o jednu nebo dvě pozice doleva, podle toho o kolikátou rundu (klíč) se jedná:

- V rundách 1,2,9,16 jsou bity obou polovin klíče posunuty o jednu pozici doleva
- V ostatních rundách jsou bity obou polovin klíče posunuty o dvě pozice doleva

V každé rundě se po provedení posunu bitů, obě poloviny spojí a provede se permutace podle obr. 16, čímž získáme 48 bitový klíč, který je poté použit pro zašifrování bloku v dané rundě. [4]

<i>PC – 2</i>							
14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

Obr. 16 Permutace klíče šifry DES [4]

2.3.3.2 *f* – funkce

Tato funkce je jádrem šifry DES a zajišťuje šifrování dat. Vstupem do této funkce je pravá polovina bloku dat z předchozí rundy a podklíč pro danou rundu. Nejprve se provede expanze (Expansion) bloku dat z 32 bitů na 48 bitů podle obr. 17. S takto upraveným blokem dat je dále provedena operace XOR s podklíčem. Výsledný 48 bitový blok se následně rozdělí na 8 částí po 6 bitech. Každá 6 bitová část je následně substituována za 4 bitovou část podle substituční tabulky pro danou část, jak je zobrazeno např. na obr. 19, který představuje substituční tabulku pro první část. Číslo řádku získáme spojením prvního a šestého bitu dané části a číslo sloupce získáme spojením druhého až pátého bitu dané části. Takto získáme 4 bitovou část, která nahradí tu 6bitovou část. Toto provedeme pro

všech osm částí, s tím, že každá část má vlastní substituční tabulku. Poté získáme osm 4 bitových částí, které po spojení vytvoří 32 bitový blok. Poté provedeme permutaci (Permutation) bitů takto vytvořeného bloku dat podle obr. 18. [4]

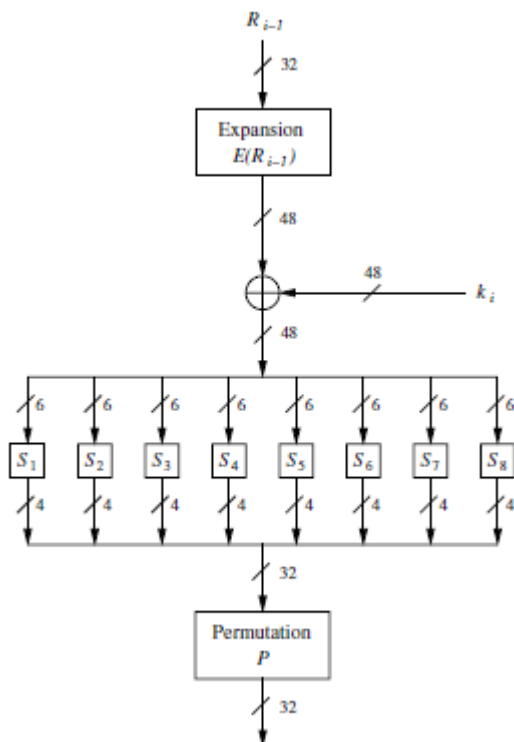
E												
32	1	2	3	4	5							
4	5	6	7	8	9							
8	9	10	11	12	13							
12	13	14	15	16	17							
16	17	18	19	20	21							
20	21	22	23	24	25							
24	25	26	27	28	29							
28	29	30	31	32	1							

P												
16	7	20	21	29	12	28	17					
1	15	23	26	5	18	31	10					
2	8	24	14	32	27	3	9					
19	13	30	6	22	11	4	25					

Obr. 17 Expanze bloku dat [4]

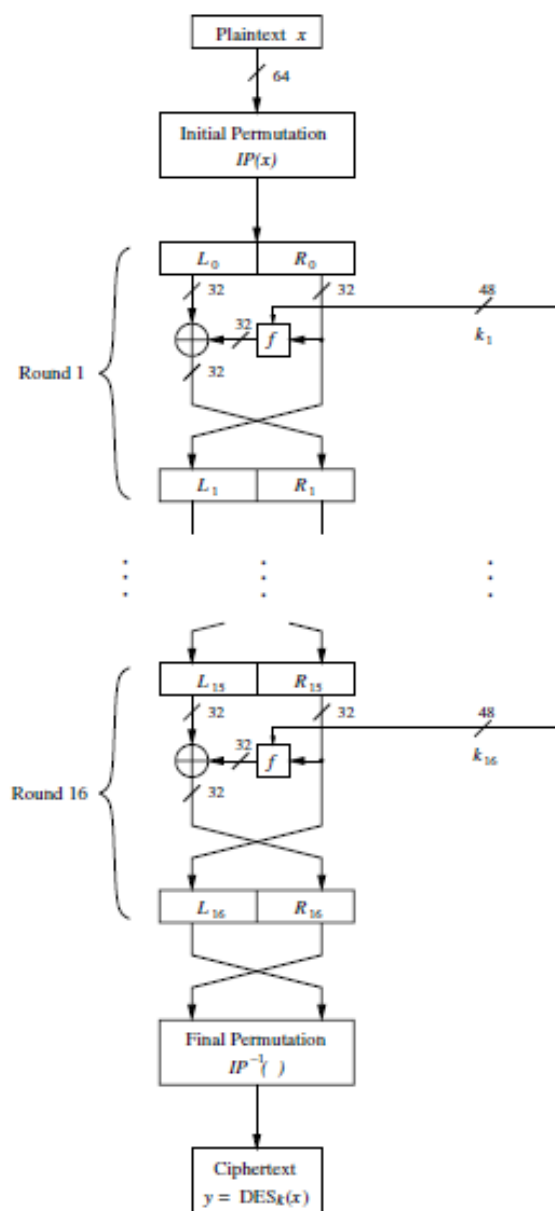
Obr. 18 Permutace bloku dat v f -funkci [4]

S_1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	01	10	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

Obr. 19 Příklad substituční tabulky S_1 [4]Obr. 20 Struktura f -funkce [4]

2.3.3.3 Šifrování

Šifra DES využívá tzv. Feistelovu síť, která je použita u mnoha symetrických blokových šifer. Výhodou tohoto řešení je, že algoritmy pro šifrování a dešifrování jsou téměř identické. [4]



Obr. 21 Feistelova struktura šifry DES [4]

Z obr. 21 je vidět, že jako první operace se provede počáteční permutace (Initial Permutation) 64bitového bloku otevřeného textu. Jednotlivé bity otevřeného textu jsou promíchány podle obr. 22.

<i>IP</i>															
58	50	42	34	26	18	10	2								
60	52	44	36	28	20	12	4								
62	54	46	38	30	22	14	6								
64	56	48	40	32	24	16	8								
57	49	41	33	25	17	9	1								
59	51	43	35	27	19	11	3								
61	53	45	37	29	21	13	5								
63	55	47	39	31	23	15	7								

Obr. 22 Počáteční permutace bloku otevřeného textu šifry DES [4]

Poté je takto upravený blok otevřeného textu rozdělen na levou a pravou polovinu. Pravá polovina je zašifrována v f - funkci pomocí klíče pro danou rundu. Výstup této funkce je poté pomocí funkce XOR sloučen s levou polovinou a výsledek je poté pravou polovinou pro následující rundu. A neupravená pravá polovina se stává levou polovinou pro následující rundu. Takto se provede celkem 16 rund, po kterých se následně prohodí a spojí poloviny z výstupu 16. rundy. Na takto vytvořeném bloku se provede konečná permutace (Final Permutation) podle obr. 23. [4]

<i>IP⁻¹</i>															
40	8	48	16	56	24	64	32								
39	7	47	15	55	23	63	31								
38	6	46	14	54	22	62	30								
37	5	45	13	53	21	61	29								
36	4	44	12	52	20	60	28								
35	3	43	11	51	19	59	27								
34	2	42	10	50	18	58	26								
33	1	41	9	49	17	57	25								

Obr. 23 Konečná permutace bloku dat šifry DES [4]

Po provedení této permutace získáváme zašifrovaný 64 bitový blok dat. Takto se zašifrují všechny bloky dat a spojí se ve výsledný zašifrovaný text (data).

2.3.3.4 Dešifrování

Dešifrování probíhá stejně jako šifrování, pouze se obrátí pořadí podklíčů. [4]

2.3.4 Triple DES

Šifra Triple DES vznikla v roce 1999, jako nástupce šifry DES, která se díky zvyšujícímu se výkonu výpočetní techniky stala zranitelnou, jelikož bylo možno použít útok hrubou silou k prolomení šifry DES. Tato šifra byla později nahrazena šifrou AES. [5]

2.3.4.1 Šifrování

Princip této šifry je jednoduchý a využívá již existující šifru DES. Pro zašifrování otevřeného textu je třikrát použita šifra DES a to tak, že se nejdříve otevřený text zašifruje prvním klíčem, poté se dešifruje druhým klíčem a nakonec se zašifruje třetím klíčem. Z toho vyplývá, že mohou nastat tři varianty délky klíče: [4]

- Všechny klíče jsou rozdílné $k_1 \neq k_2 \neq k_3$ - délka klíče 192bitů
- Dva klíče ze tří jsou stejné $k_1 \neq k_2$ a $k_1 = k_3$ - délka klíče 128bitů
- Všechny klíče jsou stejné $k_1 = k_2 = k_3$ - délka klíče 64bitů - Tato možnost se ovšem nepoužívá, neboť se ve výsledku jedná o klasickou šifru DES

2.3.4.2 Dešifrování

Dešifrování probíhá tak, že zašifrovaný text je nejdříve dešifrován třetím klíčem, poté je zašifrován druhým klíčem a nakonec je dešifrován prvním klíčem. [4]

2.3.5 AES

V roce 1997 vyhlásil NIST veřejnou soutěž na novou šifru AES (Advanced Encryption Standard). Přihlášeno bylo celkem 15 algoritmů. Ve třech hodnotících kolech vyhodnocoval NIST a mezinárodní vědecká komunita výhody a nevýhody jednotlivých přihlášených algoritmů, aby zúžili počet kandidátů na novou šifru. V roce 2001 NIST oznámil, že novou šifrou AES bude algoritmus Rijndael, který navrhli dva belgičtí kryptografové Joan Daemen a Vincent Rijmen. [4]

Původně přihlášené algoritmus Rijndael má volitelnou délku bloku dat i klíče 128, 192 nebo 256 bitů. Jelikož byl ale požadavek na AES na délku bloku dat 128 bitů, tak se u šifry AES používá varianta algoritmu Rijndael s délkou bloku dat 128 bitů a s délkou klíče 128, 192 nebo 256 bitů. Podle délky klíče se odvíjí počet rundových funkcí algoritmu a to pro 128 bitový klíč algoritmus používá 10 rundových funkcí, pro 192 bitový klíč algoritmus používá 12 rundových funkcí a pro 256 bitový klíč algoritmus používá 14 rundových funkcí. Na rozdíl od šifry DES, se u šifry AES nepoužívá Feistelova síť. [4]

2.3.5.1 Odvození podklíčů

Pro jednotlivé varianty délky klíčů je odvozen jiný počet podklíčů a to vždy počet rundových funkcí plus jedna, tedy ze 128 bitového klíče je odvozeno 11 podklíčů, ze 192 bitového klíče je odvozeno 13 podklíčů a z 256 bitového klíče je odvozeno 15 podklíčů. Jednotlivé podklíče se odvozují rekurzivně, tzn. pro odvození podklíče k_i je třeba znát podklíč k_{i-1} . [4]

Klíč je rozdělen na 32 bitové bloky a jednotlivé bloky jsou označovány jako jednotlivé prvky pole W .

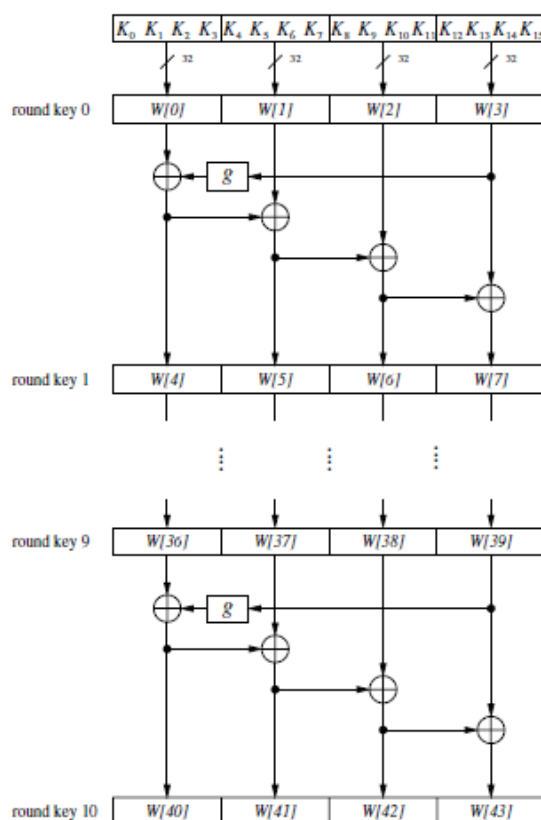
Např. podklíče ze 128 bitového klíče jsou vypočítány následujícím způsobem:

Levý krajní blok podklíče pro jednotlivé rundy získáme použitím následujícího vztahu:

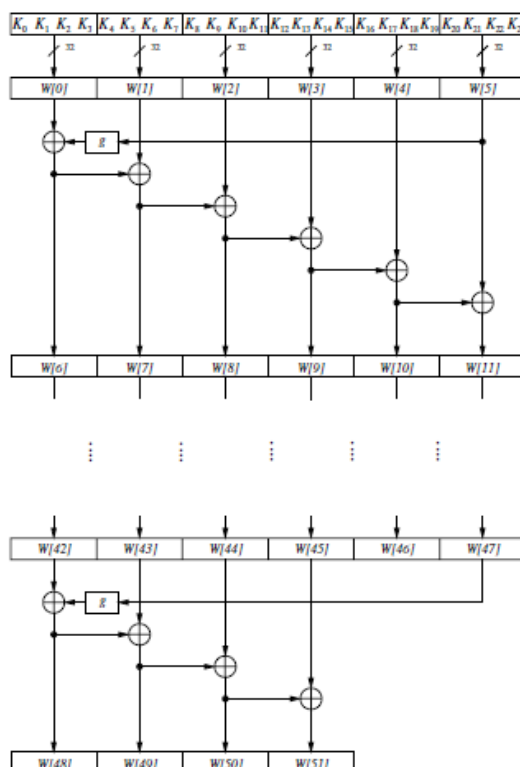
$$W[4i] = W[4(i-1)] + g(W[4i-1])$$

Zbylé bloky podklíče pro jednotlivé rundy získáme použitím následujícího vztahu:

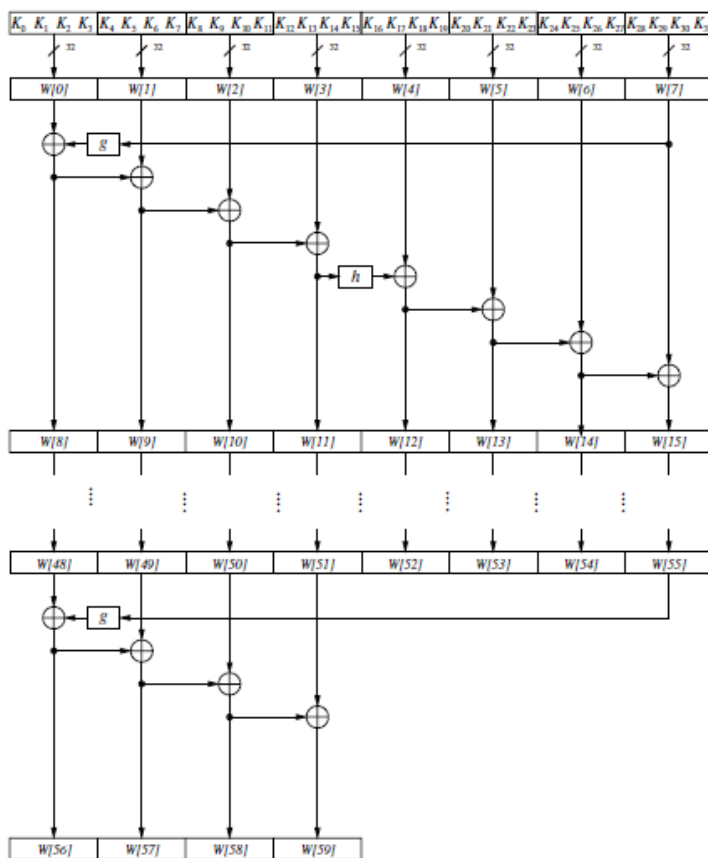
$$W[4i+j] = W[4i+j-1] + W[4(i-1)+j]$$



Obr. 24 Odvození podklíčů ze 128 bitového klíče [4]



Obr. 25 Odvození podklíčů ze 192 bitového klíče [4]



Obr. 26 Odvození podklíčů ze 256 bitového klíče [4]

2.3.5.2 Šifrování

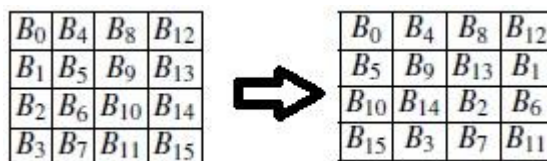
Na začátku algoritmu je provedena operace XOR otevřeného textu (Plain Text) a klíče. Poté dle délky klíče následuje 10, 12 nebo 14 rundových funkcí, které se skládají z následujících operací:

- Byte Substitution – Substituce nebo-li nahrazení hodnot jednotlivých bajtů jinými hodnotami, podle obr. 27, kde první čtyři bity bajtu udávají pozici na svislé ose a pátý až osmý bit bajtu udává pozici na vodorovné ose. Na průsečíku těchto os najdeme novou hodnotu pro daný bajt. [4]

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Obr. 27 Substituční tabulka šifry AES [4]

- Shift Rows – Tato operace provádí posun druhého řádku matice dat o jednu pozici doleva, posun třetího řádku matice dat o dvě pozice doleva a posun čtvrtého řádku matice dat o tři pozice doleva, jak je znázorněno na obr. 28. [4]



Obr. 28 Posun řádků matice dat [4]

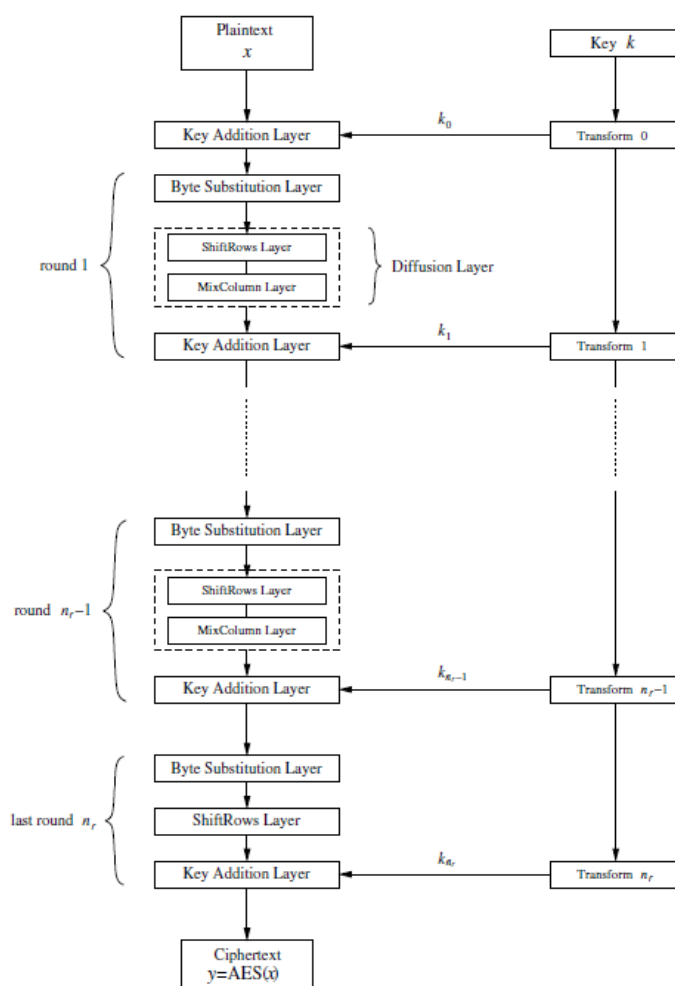
- Mix Columns – Vstup této operace je tedy výstup minulé operace, kdy došlo k posunu řádků matice dat. Takto upravená matice dat B je dále použita k výpočtu matice C tak, že každý sloupec matice B je vynásoben stejnou maticí, jak je znázorněno na obr. 29. [4]

$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} B_0 \\ B_5 \\ B_{10} \\ B_{15} \end{pmatrix}$$

Obr. 29 Příklad operace Mix Columns [4]

- Key Addition – Vstupem této operace je 16ti bajtová matice dat, která je výstupem z minulé operace, a 16ti bajtový podklíč. Tyto dva vstupy jsou poté sečteny pomocí operace XOR. [4]

V poslední rundové funkci není provedena operace Mix Columns. [4]



Obr. 30 Struktura šifrování AES [4]

2.3.5.3 Dešifrování

Jelikož nemá šifra AES Feistelovu strukturu, tak není dešifrování prováděno stejným algoritmem jako šifrování. Podklíče jsou u dešifrování použity v obráceném pořadí a

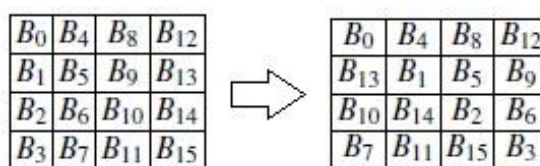
jednotlivé operace uvnitř rundové funkce, kromě operace Key Addition, jsou inverzními k těm u šifrování. Operace rundové funkce jsou tedy následující:

- Key Addition – Stejně jako u šifrování, dva 16 bajtové vstupy jsou sečteny pomocí operace XOR. [4]
- Inverse Mix Column - Matice dat C je použita k výpočtu matice B tak, že každý sloupec matice C je vynásoben stejnou maticí, jak je znázorněno na obr. 31. [4]

$$\begin{pmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{pmatrix} = \begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} \begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix}$$

Obr. 31 Příklad operace Inverse Mix Columns [4]

- Inverse Shift Rows - Tato operace provádí posun druhého řádku matice dat o jednu pozici doprava, posun třetího řádku matice dat o dvě pozice doprava a posun čtvrtého řádku matice dat o tři pozice doprava, jak je znázorněno na obr. 32. [4]



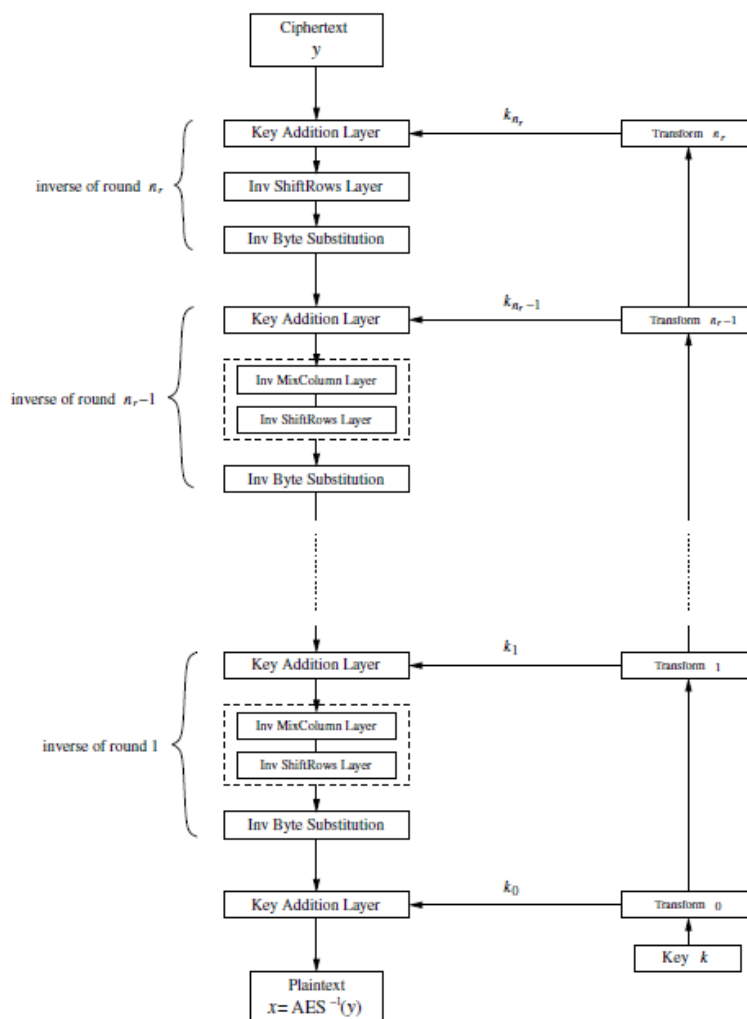
Obr. 32 Inverzní posun řádků matice dat [4]

- Inverse Byte Substitution - Substituce nebo-li nahrazení hodnot jednotlivých bajtů jinými hodnotami, podle obr. 33, kde první čtyři bity bajtu udávají pozici na svislé ose a pátý až osmý bit bajtu udává pozici na vodorovné ose. Na průsečíku těchto os najdeme novou hodnotu pro daný bajt. [4]

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Obr. 33 Inverzní substituční tabulka šifry AES [4]

V první rundové funkci není použita operace Inverse Mix Column. [4]



Obr. 34 Struktura dešifrování AES [4]

2.3.6 Blowfish

Blowfish je symetrická bloková šifra, kterou v roce 1993 navrhl Bruce Schneier. Tato šifra používá Feistelovu síť. Blowfish pracuje s bloky o velikosti 64 bitů a volitelnou délkou klíče až do velikosti 448 bitů. Hlavní výhodou této šifry je rychlost. [12]

2.3.6.1 Odvození podklíčů

U šifry Blowfish se z původního klíče odvozuje celkem 18 podklíčů o velikosti 32 bitů. Tyto podklíče jsou označovány a uloženy v poli P, které obsahuje 18 prvků o velikosti 32 bitů. Dále jsou pro výpočet pakliče třeba 4 S-boxy, kde každý obsahuje 256 prvků o velikosti 32 bitů. [12]

Algoritmus pro výpočet podklíčů:

1. Inicializace polí v pořadí P, S1 ... S4 pomocí pevného řetězce, který se skládá z desetinných míst čísla π , menších než počáteční tři, reprezentovaných hexadecimálními znaky. Např. P1 = 0x243f6a88

P2 = 0x85a308d3

P3 = 0x13198a2e

P4 = 0x03707344

2. XOR mezi P1 a prvními 32 bity klíče, XOR mezi P2 a druhými 32 bity klíče atd. Pokud se narazí na konec klíče, tak následuje opět ten samý klíč, takže se zřetězí klíč tolikrát za sebou, aby pokryl celou délku pole P.

3. Zašifrování řetězce samých nul pomocí podklíčů vypočítaných v předchozím kroku (2).

4. Nahrazení P1 a P2 výstupem z předchozího kroku (3).

5. Zašifrování výstupu z kroku 3 pomocí modifikovaných podklíčů.

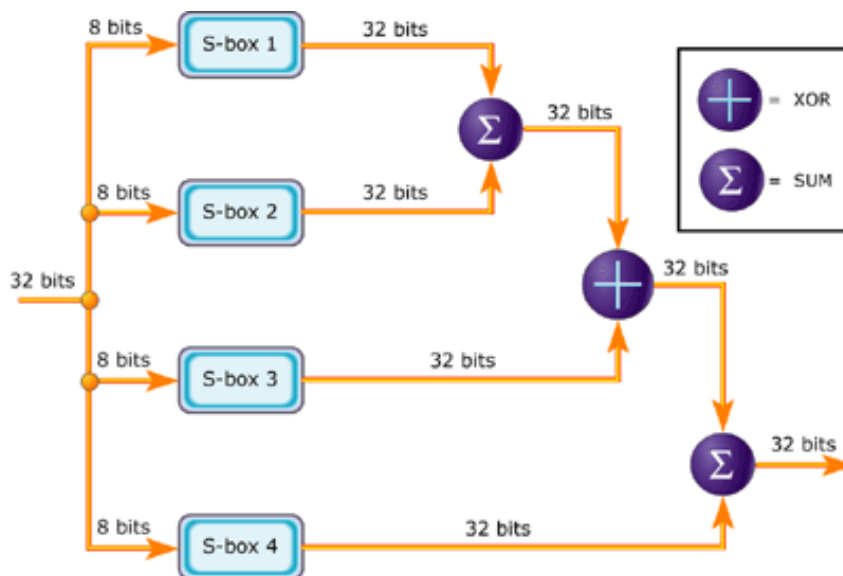
6. Nahrazení P3 a P4 výstupem z předchozího kroku (5).

7. Stejným způsobem pokračujeme do té doby, dokud nejsou nahrazeny všechny prvky pole P a všechny prvky S-boxů.

Celkem je třeba 521 iterací k vygenerování všech prvků pole P a S-boxů. [12]

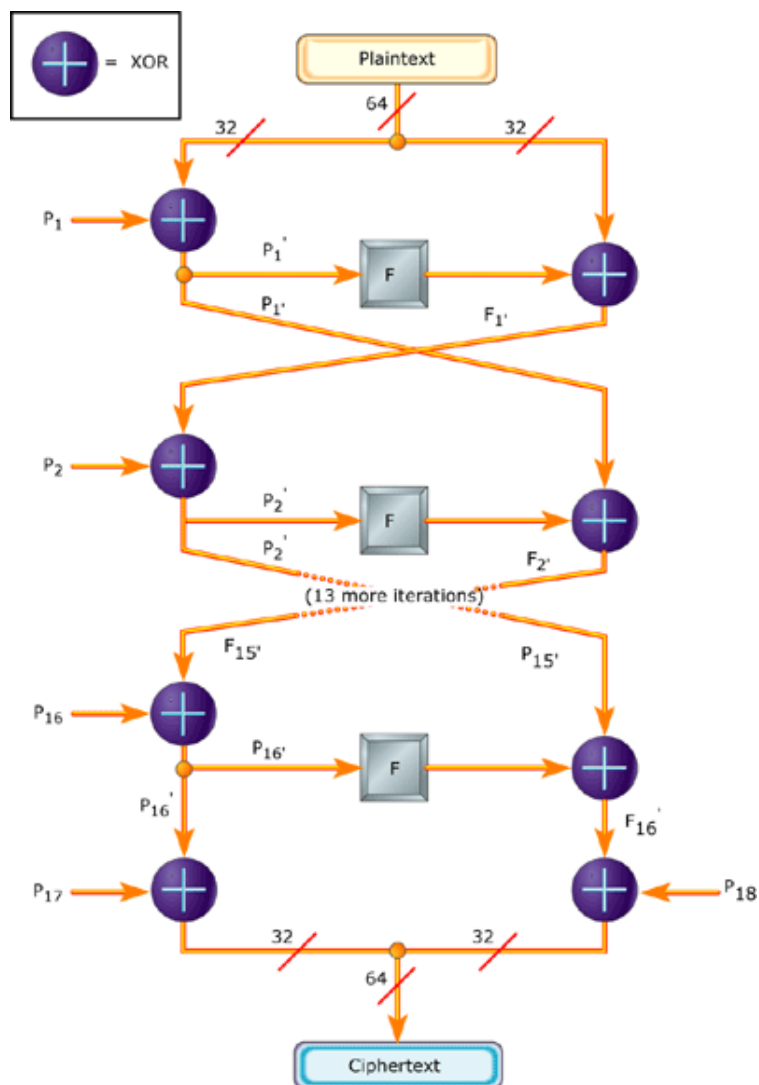
2.3.6.2 *f*–funkce

Vstupem do této funkce je 32 bitový blok dat, který je rozdělen na čtyři 8 bitové části. Dále je provedena substituce všech čtyř 8 bitových bloků, díky které získáme čtyři 32 bitové bloky dat. Poté je provedena operace XOR třetího bloku se součtem prvního a druhého bloku. Výsledek této operace je poté sečten se čtvrtým blokem, čímž dostaneme výstup *f*– funkce, který má délku 32 bitů. [13]

Obr. 35 f – funkce šifry Blowfish [13]

2.3.6.3 Šifrování

Šifrovací algoritmus obsahuje 16 rundových funkcí a používá 18 podklíčů. Struktura algoritmu je zobrazena na obr. 36, ze kterého vidíme, že je vstupní 64 bitový blok (Plain Text) rozdělen na dvě poloviny o délce 32 bitů. Levá polovina je pomocí operace XOR sloučena s podklíčem pro danou rundu. Takto upravený 32 bitový blok se stává vstupem pro pravou polovinu následující rundy. Vstup pro levou polovinu následující rundy vznikne tak, že se provede f - funkce na levou polovinu současné rundy. Takto upravená levá polovina je poté pomocí operace XOR sloučena s pravou polovinou a výsledek této operace je tedy vstupem pro levou polovinu následující rundy. Po provedení 16 rundových funkcí, je výstup dále zpracován tak, že je provedena operace XOR s levou polovinou a sedmnáctým podklíčem a operace XOR s pravou polovinou a osmnáctým podklíčem. Tyto dva bloky jsou poté spojeny a tvoří 64 bitový zašifrovaný výstup. [13]



Obr. 36 Struktura šifry Blowfish [13]

2.3.6.4 Dešifrování

Jelikož tato šifra pracuje na principu Feistelovy sítě, tak se dešifrování provádí stejným způsobem jako šifrování, jen klíče $P_1 \dots P_{18}$ jsou v obráceném pořadí. [12]

2.3.7 Twofish

Twofish je symetrická bloková šifra, kterou v roce 1998 navrhli Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall a Niels Ferguson. Tato šifra používá Feistelovu síť. Twofish pracuje s bloky o velikosti 128 bitů a volitelnou délkou klíče 128, 192 a 256 bitů. Tato šifra byla jedním z kandidátů pro šifru AES. [14]

2.3.7.1 *h – funkce*

Vstupem této funkce je 32 bitový blok X a seznam $L = (L_0, \dots, L_{k-1})$, který obsahuje k (délka klíče / 64) prvků o velikosti 32 bitů. Tato funkce pracuje v k krocích. V každém kroku jsou 4 bajty substituovány v S-boxu a následně jsou sloučeny pomocí operace XOR s bajtem odvozeným ze seznamu L . Nakonec jsou tyto 4 bajty opět substituovány pomocí S-boxu a poté vynásobeny maticí MDS. [15]

Detailní postup je následující:

$$l_{i,j} = \lfloor L_i / 2^{8j} \rfloor \bmod 2^8$$

$$x_j = \lfloor X / 2^{8j} \rfloor \bmod 2^8$$

kde $i = 0, \dots, k-1$ $j = 0, \dots, 3$

$$y_{k,j} = x_j \quad j = 0, \dots, 3$$

Pokud je $k = 4$

$$y_{3,0} = q_1[y_{4,0}] \oplus l_{3,0}$$

$$y_{3,1} = q_0[y_{4,1}] \oplus l_{3,1}$$

$$y_{3,2} = q_0[y_{4,2}] \oplus l_{3,2}$$

$$y_{3,3} = q_1[y_{4,3}] \oplus l_{3,3}$$

Pokud je $k \geq 3$

$$y_{2,0} = q_1[y_{3,0}] \oplus l_{2,0}$$

$$y_{2,1} = q_1[y_{3,1}] \oplus l_{2,1}$$

$$y_{2,2} = q_0[y_{3,2}] \oplus l_{2,2}$$

$$y_{2,3} = q_0[y_{3,3}] \oplus l_{2,3}$$

Pro všechny případy a hodnoty k

$$y_0 = q_1[q_0[q_0[y_{2,0}] \oplus l_{1,0}] \oplus l_{0,0}]$$

$$y_1 = q_0[q_0[q_1[y_{2,1}] \oplus l_{1,1}] \oplus l_{0,1}]$$

$$y_2 = q_1[q_1[q_0[y_{2,2}] \oplus l_{1,2}] \oplus l_{0,2}]$$

$$y_3 = q_0[q_1[q_1[y_{2,3}] \oplus l_{1,3}] \oplus l_{0,3}]$$

Nakonec je tedy provedeno násobení maticí MDS, která je naplněna stejnými hodnotami jako v g – funkci.

$$\begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{pmatrix} = \begin{pmatrix} \cdot & \dots & \cdot \\ \vdots & \text{MDS} & \vdots \\ \cdot & \dots & \cdot \end{pmatrix} \cdot \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

$$Z = \sum_{i=0}^3 z_i \cdot 2^{8i}$$

Z je poté výstupem h – funkce. [15]

2.3.7.2 Odvození podklíčů

Algoritmus musí odvodit 40 podklíčů $K_0 \dots K_{39}$ a 4 na klíči závislé S-boxy, které jsou dále použity v g funkci při šifrování a dešifrování. Twofish je navržen pro tři délky klíčů $N = 128$, $N = 192$ a $N = 256$. [15]

Vypočítá se $k = N/64$. Klíč M potom obsahuje $8k$ bajtů $m_0 \dots m_{8k-1}$. Klíč je poté rozdělen do $2k$ bloků o velikosti 32 bitů.

$$M_i = \sum_{j=0}^3 m_{(4i+j)} \cdot 2^{8j} \quad i = 0, \dots, 2k - 1$$

Poté se klíče M rozdělí do dvou vektorů o velikosti k .

$$M_e = (M_0, M_2, \dots, M_{2k-2})$$

$$M_o = (M_1, M_3, \dots, M_{2k-1})$$

Další vektor délky k je též odvozen z klíče. Klíč je rozdělen na části po 8 bajtech a tyto části jsou poté reprezentovány jak vektor. Jednotlivé vektory jsou poté vynásobeny s maticí o rozměrech 4×8 odvozenou z Reed-Solomonova kódu. Výsledek je reprezentován jako 32 bitový blok. [15]

$$\begin{pmatrix} s_{i,0} \\ s_{i,1} \\ s_{i,2} \\ s_{i,3} \end{pmatrix} = \begin{pmatrix} \cdot & \dots & \cdot \\ \vdots & \text{RS} & \vdots \\ \cdot & \dots & \cdot \end{pmatrix} \cdot \begin{pmatrix} m_{8i} \\ m_{8i+1} \\ m_{8i+2} \\ m_{8i+3} \\ m_{8i+4} \\ m_{8i+5} \\ m_{8i+6} \\ m_{8i+7} \end{pmatrix}$$

$$S_i = \sum_{j=0}^3 s_{i,j} \cdot 2^{8j}$$

kde $i = 0, \dots, k-1$

Poté tedy vznikne třetí vektor, odvozený z klíče, v následujícím tvaru:

$$S = (S_{k-1}, S_{k-2}, \dots, S_0)$$

Postup odvození 40 podklíčů je následující (ROL je rotace bitů doleva):

$$\rho = 2^{24} + 2^{16} + 2^8 + 2^0$$

$$A_i = h(2i\rho, M_e)$$

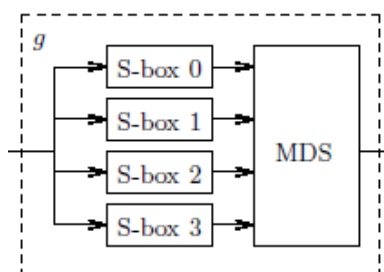
$$B_i = \text{ROL}(h((2i+1)\rho, M_o), 8)$$

$$K_{2i} = (A_i + B_i) \bmod 2^{32}$$

$$K_{2i+1} = \text{ROL}((A_i + 2B_i) \bmod 2^{32}, 9)$$

2.3.7.3 g – funkce

Vstupem g – funkce je 32 bitový blok dat, který je rozdělen na čtyři části o velikosti 8 bitů (1B). Následně je hodnota každého bajtu substituována za jinou hodnotu pomocí S-boxů. Výstupem z S-boxů jsou opět čtyři bloky o velikosti 8 bitů. Tyto čtyři bloky jsou poté ve formě vektoru vynásobeny MDS maticí, jejíž hodnoty jsou znázorněny na obr. 38. [14]



Obr. 37 g – funkce šifry Twofish [14]

$$\text{MDS} = \begin{pmatrix} 01 & \text{EF} & 5\text{B} & 5\text{B} \\ 5\text{B} & \text{EF} & \text{EF} & 01 \\ \text{EF} & 5\text{B} & 01 & \text{EF} \\ \text{EF} & 01 & \text{EF} & 5\text{B} \end{pmatrix}$$

Obr. 38 MDS matice [14]

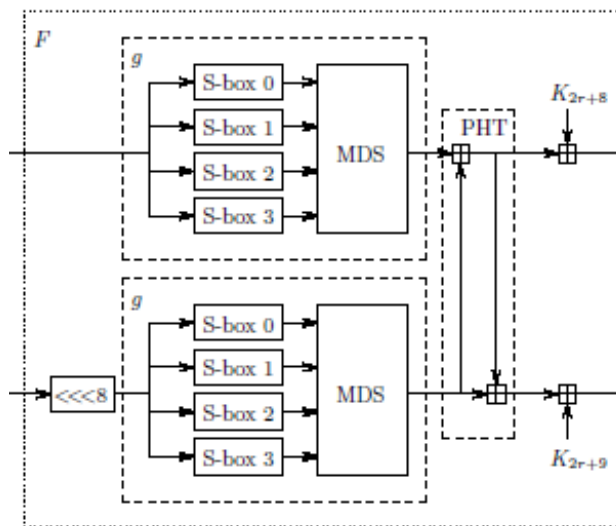
2.3.7.4 F – funkce

Vstupem F – funkce jsou dva 32 bitové bloky dat, z nichž u jednoho jsou na vstupu bity posunuty o osm doleva. Oba tyto bloky jsou poté zpracovány v g – funkci. Výstupy z g –

funkcí jsou poté zpracovány pomocí PHT (Pseudo-Hadamard transformace) následujícím způsobem, kde a a b značí výstupy z g - funkcí:

$$a' = a + b \bmod 2^{32} \qquad b' = a + 2b \bmod 2^{32}$$

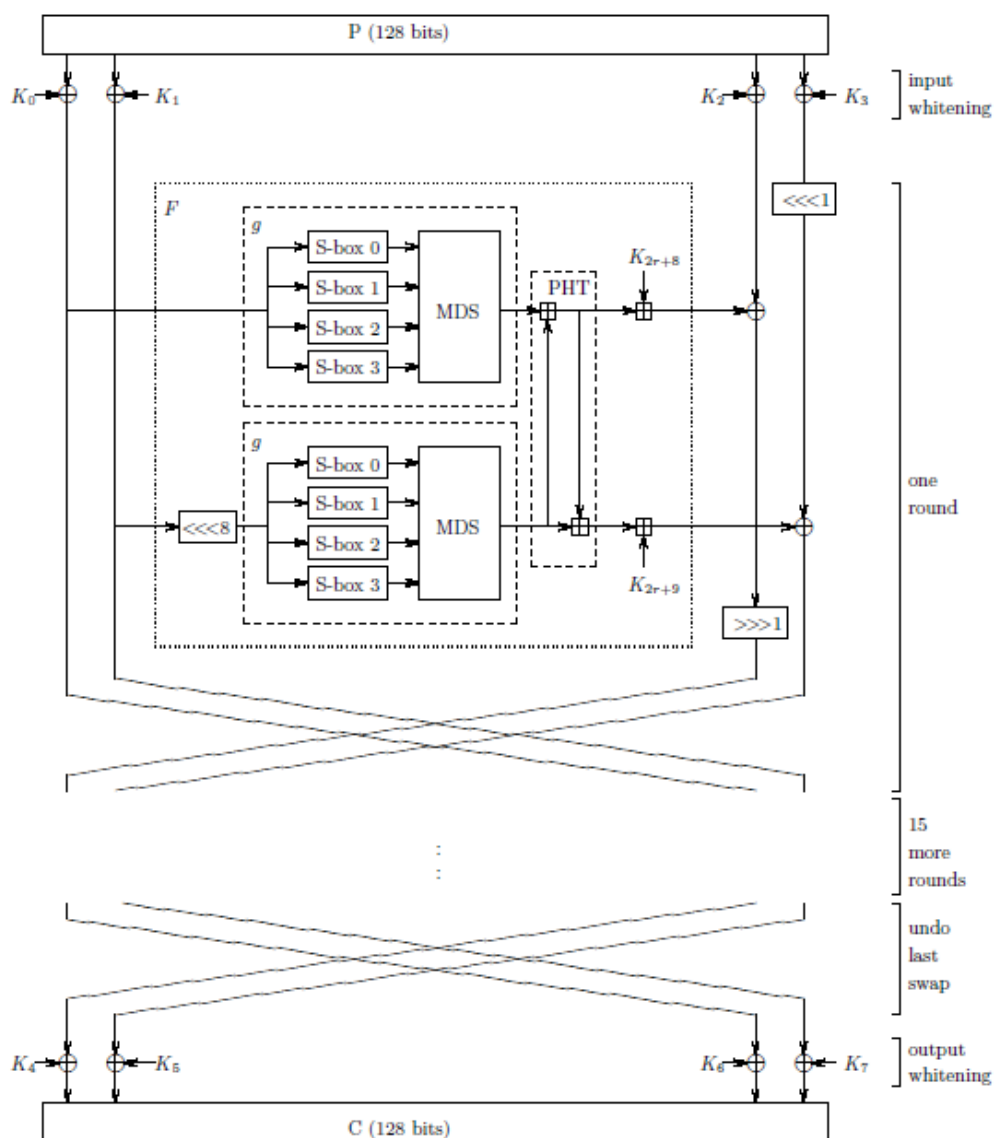
Po provedení této operace jsou výstupy sečteny s podklíčem pro danou rundu. [14]



Obr. 39 F – funkce šifry Twofish [14]

2.3.7.5 Šifrování

Šifrovací algoritmus obsahuje 16 rundových funkcí. Struktura algoritmu je zobrazena na obr. 40, ze kterého vidíme, že je vstupní 128 bitový blok rozdělen na dvě poloviny, které jsou dále rozděleny na další dvě poloviny, čímž vzniknou čtyři bloky dat o velikosti 32 bitů. Na začátku algoritmu je provedena operace XOR každého bloku s příslušným podklíčem. Takto zpracované levé dva bloky dat jsou poté vstupem pro pravé dva bloky dat následující rundy. Výpočet vstupu levých dvou bloků pro následující rundu je složitější a provádí se tímto způsobem: levé dva bloky daty jsou zpracovány v F funkci a jsou poté pomocí operace XOR sloučeny s pravými dvěma bloky dat. Po provedení této operace jsou bity prvního pravého bloku dat posunuty o jeden bit doprava a bity druhého pravého bloku dat jsou posunuty o jeden bit doleva. Takto zpracované bloky dat jsou poté vstupem pro levé dva bloky následující rundy. Tímto způsobem je provedeno 16 rundových funkcí a na konci je opět provedena operace XOR každého bloku s příslušným podklíčem. Nakonec tyto čtyři bloky spojíme a získáme zašifrovaný 128 bitový blok dat. [14]



Obr. 40 Struktura šifry Twofish [14]

2.3.7.6 Dešifrování

Přesto, že má tato šifra Feistelovu strukturu, tak pro dešifrování se nepoužívá stejný algoritmus jako pro šifrování. Za tuto vlastnost mohou rotace ve struktuře této šifry. [14]

2.3.8 Serpent

Serpent je symetrická bloková šifra, kterou v roce 1998 navrhli Ross Anderson, Eli Biham a Lars Knudsen. Serpent pracuje s bloky dat o velikosti 128 bitů a délkou klíče 256 bitů. Tato šifra byla jedním z kandidátů pro šifru AES. [16]

2.3.8.1 Odvození podklíčů

Pro tuto šifru je třeba ze zadaného 256 bitového klíče odvodit celkem 33 podklíčů velikosti 128 bitů. Na začátku je zadaný 256 bitový klíč K rozdělen na 8 částí o velikosti 32 bitů označených $w_{-8} \dots w_{-1}$. Z těchto osmi částí je poté odvozeno celkem 132 částí o velikosti 32 bitů značených $w_0 \dots w_{131}$ následujícím afinním rekurentním vztahem: [16]

$$w_i = (w_{i-8} \oplus w_{i-5} \oplus w_{i-3} \oplus w_{i-1} \oplus \phi \oplus i) \lll 11$$

kde ϕ je hodnota zlatého řezu $(\sqrt{5} + 1)/2$, v hexadecimální formě 0x9e3779b9

Hodnoty těchto 132 částí jsou následně substituovány pomocí S-boxů. 132 částí si rozdělíme na 4 segmenty o velikosti 33 částí. Poté i -tý prvek každého segmentu transformujeme pomocí S-boxu daného vztahem $S_{(r+3-i) \bmod r}$, kde $r = 32$. První a poslední dva kroky substituce jsou: [16]

$$\begin{aligned} \{k_0, k_{33}, k_{66}, k_{99}\} &= S_3(w_0, w_{33}, w_{66}, w_{99}) \\ \{k_1, k_{34}, k_{67}, k_{100}\} &= S_2(w_1, w_{34}, w_{67}, w_{100}) \\ &\dots \\ \{k_{31}, k_{64}, k_{97}, k_{130}\} &= S_4(w_{31}, w_{64}, w_{97}, w_{130}) \\ \{k_{32}, k_{65}, k_{98}, k_{131}\} &= S_3(w_{32}, w_{65}, w_{98}, w_{131}) \end{aligned}$$

Posledním krokem je vytvořit z prvků $k_0 \dots k_{131}$ 128 bitové podklíče K_i , kde $i \in \{0, \dots, r\}$

$$K_i = \{k_{4i}, k_{4i+1}, k_{4i+2}, k_{4i+3}\}$$

2.3.8.2 Šifrování

Šifrovací algoritmus se skládá ze tří částí a to:

- Počáteční permutace IP – tato operace prohází bity bloku dat podle obr. 41 [16]

0	32	64	96	1	33	65	97	2	34	66	98	3	35	67	99
4	36	68	100	5	37	69	101	6	38	70	102	7	39	71	103
8	40	72	104	9	41	73	105	10	42	74	106	11	43	75	107
12	44	76	108	13	45	77	109	14	46	78	110	15	47	79	111
16	48	80	112	17	49	81	113	18	50	82	114	19	51	83	115
20	52	84	116	21	53	85	117	22	54	86	118	23	55	87	119
24	56	88	120	25	57	89	121	26	58	90	122	27	59	91	123
28	60	92	124	29	61	93	125	30	62	94	126	31	63	95	127

Obr. 41 Počáteční permutace šifry Serpent [16]

- 32 rundových funkcí – Uvnitř rundové funkce jsou provedeny tři operace. Nejdříve se provede operace XOR se vstupním blokem dat B_i a podklíčem K_i o velikosti 128 bitů. Po této operaci se provede substituce pomocí S-boxu se 4 částmi bloku dat z předchozí operace, tedy s částmi o velikosti 32 bitů. Poslední operací v rundové funkci je lineární transformace, která zpracuje jednotlivé 32 bitové bloky. [16]

$$\begin{aligned}
 X_0, X_1, X_2, X_3 &= S_i(B_i \oplus K_i) \\
 X_0 &= X_0 \lll 13 \\
 X_2 &= X_2 \lll 3 \\
 X_1 &= X_1 \oplus X_0 \oplus X_2 \\
 X_3 &= X_3 \oplus X_2 \oplus (X_0 \ll 3) \\
 X_1 &= X_1 \lll 1 \\
 X_3 &= X_3 \lll 7 \\
 X_0 &= X_0 \oplus X_1 \oplus X_3 \\
 X_2 &= X_2 \oplus X_3 \oplus (X_1 \ll 7) \\
 X_0 &= X_0 \lll 5 \\
 X_2 &= X_2 \lll 22 \\
 B_{i+1} &= X_0, X_1, X_2, X_3
 \end{aligned}$$

kde \lll znázorňuje rotaci a \ll znázorňuje posun

V poslední rundě je Lineární transformace nahrazena následující operací:

$$B_r = S_{r-1}(B_{r-1} \oplus K_{r-1}) \oplus K_r$$

- Konečná permutace FP - tato operace prohází bity bloku dat podle obr. 42 [16]

0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
64	68	72	76	80	84	88	92	96	100	104	108	112	116	120	124
1	5	9	13	17	21	25	29	33	37	41	45	49	53	57	61
65	69	73	77	81	85	89	93	97	101	105	109	113	117	121	125
2	6	10	14	18	22	26	30	34	38	42	46	50	54	58	62
66	70	74	78	82	86	90	94	98	102	106	110	114	118	122	126
3	7	11	15	19	23	27	31	35	39	43	47	51	55	59	63
67	71	75	79	83	87	91	95	99	103	107	111	115	119	123	127

Obr. 42 Konečná permutace šifry Serpent [16]

2.3.8.3 Dešifrování

Dešifrování se liší od šifrování tím, že jsou použity inverzní S-boxy, inverzní lineární transformace a podklíče jsou v obráceném pořadí. [16]

II. PRAKTICKÁ ČÁST

3 POPIS APLIKACE

Vytvořená aplikace je komplexním systémem pro šifrování a dešifrování textů a souborů. Aplikace umožňuje zašifrovat data 10 různými šiframi, z nichž 4 jsou konvenční šifry, a 6 je moderních blokových šifer. Aplikace tedy implementuje následujících 10 šifer:

- Šifra s pevným posunem ROT-XX
- Vigenérova šifra
- Vernamova šifra
- Autokláv
- DES
- Triple DES
- AES
- Blowfish
- Twofish
- Serpent

Moderní blokové šifry můžeme dále použít v následujících operačních módech blokových šifer:

- ECB
- CBC
- CFB

Dále je možné vybrat mezi čtyřmi módy doplňování bloku dat u blokových šifer:

- ANSI X.923
- ISO 10126
- PKCS7
- Zero padding

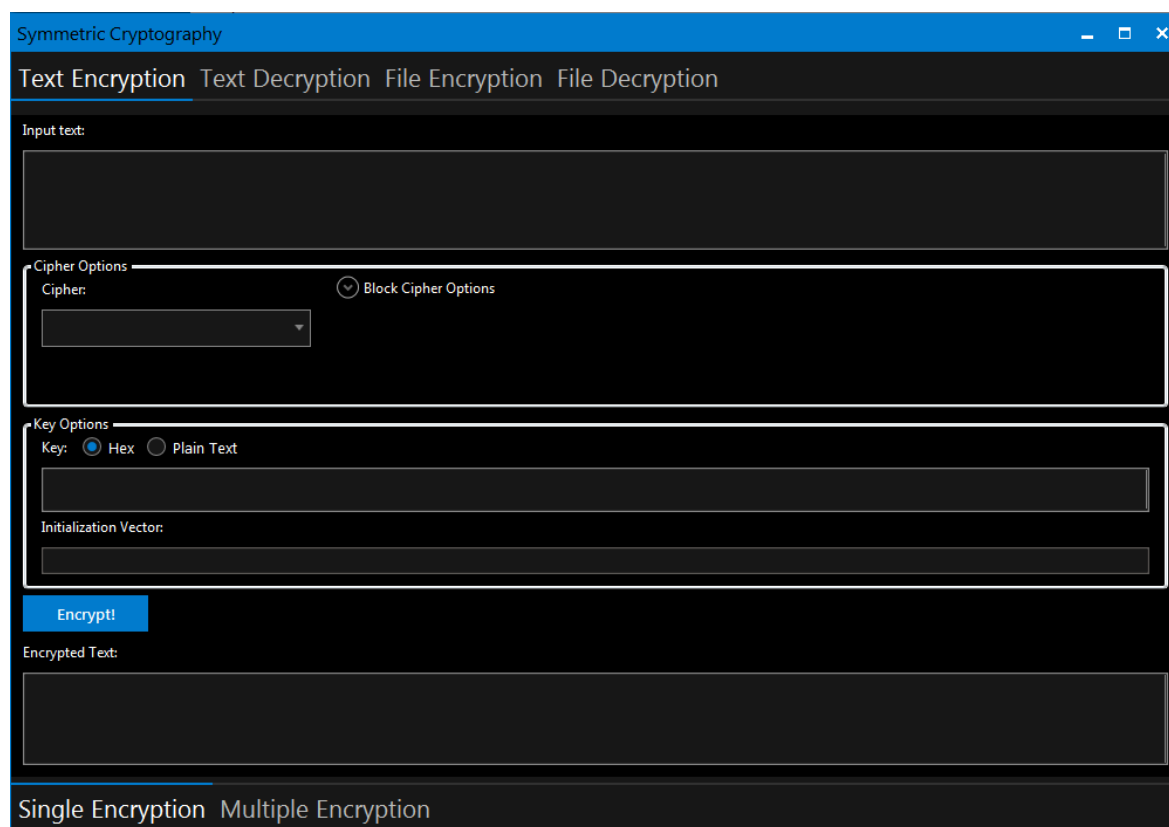
Aplikace taktéž implementuje vícenásobné šifrování a dešifrování. U vícenásobného šifrování aplikace navíc generuje šifrovací kód, který obsahuje nastavení použitých šifer a

klíčů, takže si uživatel nemusí pamatovat přesné nastavení několika použitých šifer, ale stačí mu uchovat tento šifrovací kód stejně jako by uchovával klíč při zašifrování dat jen jednou šifrou.

Celá aplikace se všemi popisky a varovnými zprávami je v anglickém jazyce.

Aplikace je naprogramována v programovacím jazyce C# s použitím technologie WPF. Pro tvorbu grafického rozhraní aplikace byla použita knihovna Elysium, která obsahuje prvky pro grafické rozhraní ve stylu Metro aplikací. Algoritmy moderních symetrických šifer jsou použity ze dvou existujících kryptografických knihoven, přičemž jedna je přímo knihovnou jazyka C# a to System.Security.Cryptography a druhá je knihovna Bouncy Castle, která je volně k použití.

3.1 Grafické rozhraní



Obr. 43 Hlavní okno aplikace

Aplikace je rozčleněná tak, že ve vrchní části jsou 4 záložky a to pro:

- Šifrování textu
- Dešifrování textu

- Šifrování souboru
- Dešifrování souboru

Navíc na záložkách pro šifrování textu a souboru jsou další 2 záložky umístěné ve spodní části aplikace:

- Jednoduché šifrování
- Vícenásobné šifrování

Na záložkách pro dešifrování textu a souboru jsou další 3 záložky umístěné ve spodní části aplikace:

- Jednoduché dešifrování
- Vícenásobné dešifrování
- Dešifrování s šifrovacím kódem

Záložky pro jednoduché neboli taky jednonásobné šifrování a dešifrování textu a souboru mají stejné tyto části:

- Nastavení šifry – Na obr. 44 je nastavení šifry po spuštění aplikace, kde je rolovací seznam pro výběr šifry. Pokud uživatel vybere jednu z blokových šifer, které jsou od konvenčních oddělené vodorovnou čarou v seznamu, tak se vedle výběru šifry objeví nastavení blokové šifry, kde si uživatel zvolí operační mód blokové šifry a mód doplnění bloku dat, jak je znázorněno na obr. 45



Obr. 44 Nastavení šifry



Obr. 45 Nastavení blokové šifry

- Nastavení klíče – V části pro nastavení klíče se nachází dva radio buttony pro výběr toho, zda uživatel zadává klíč v hexadecimální formě nebo ve formě normálního

textu a znaků. Dále se v nastavení klíče nachází pole pro zadání samotného klíče a pole pro inicializační vektor. V poli pro inicializační vektor se zobrazí inicializační vektor po zašifrování dat, pokud uživatel použil k zašifrování blokovou šifru v módech CBC nebo CFB.

Obr. 46 Nastavení klíče

Záložky pro vícenásobné šifrování a dešifrování obsahují nastavení pro vícenásobné šifrování/dešifrování, které je zobrazeno na obr. 47. Uživatel si v tomto nastavení vybere šifru, kterou chce použít, formát klíče a klíč samotný. Jakmile takto uživatel nastaví šifru, tak toto nastavení tlačítkem Add To List přidá do seznamu, který slouží pro vložení několika nastavených šifer, pomocí kterých se poté data zašifrují v pořadí od vrchu dolů, jak jsou šifry umístěny v seznamu. Pro manipulaci s nastavenými šiframi v seznamu slouží celkem 4 tlačítka, která umožňují smazat vybranou šifru, smazat všechny šifry a posun nahoru a dolů v seznamu.

Obr. 47 Nastavení vícenásobného šifrování

3.2 Vstupy a výstupy aplikace

3.2.1 Texty

Aplikace pracuje se znakovou sadou Windows-1250, která je používána pro střední Evropu, což znamená, že prvních 128 znaků ASCII tabulky je shodných s jakoukoli jinou znakovou sadou, ale znaky z rozšířené ASCII tabulky na pozicích 128 – 255 jsou rozdílné

mezi jednotlivými znakovými sadami. Znak ze znakové sady Windows-1250 jsou zobrazeny na obr. 48.

Codepage 1250 - Latin 2 Windows

	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-		0001	0002	0003	0004	0005	0006	0007	0008	0009	000A	000B	000C	000D	000E	000F
1-	0010	0011	0012	0013	0014	0015	0016	0017	0018	0019	001A	001B	001C	001D	001E	001F
2-	0020	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3-	0030	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>
4-	0040	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N
5-	0050	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^
6-	0060	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n
7-	0070	p	q	r	s	t	u	v	w	x	y	z	{		}	~
8-	0080	€	,	„	…	†	‡		%	Š	<	Ś	Ť	Ž	Ž	
9-	0090	‘	’	“	”	•	—		™	š	>	ś	ť	ž	ž	
A-	00A0	˘	˘	Ł	▣	Ą	Ł	§	©	Š	«	¬	-	®	Ž	
B-	00B0	±	ˆ	ı	ˆ	µ	¶	·	ˆ	ş	»	Ł	ˆ	İ	ı	
C-	00C0	Ŕ	Á	Â	Ã	Ä	Å	Ĉ	Ç	Č	É	Ê	Ë	Ė	İ	Ď
D-	00D0	Đ	Ň	Ń	Ó	Ô	Õ	Ö	×	Ř	Ů	Ú	Ů	Ů	Ý	ß
E-	00E0	í	á	â	ã	ä	å	Ĭ	ć	ç	č	é	ę	ě	ė	đ
F-	00F0	đ	ń	ñ	ó	ô	õ	ö	÷	ř	ů	ú	ű	ü	ý	ÿ

Obr. 48 Znaková sada Windows-1250 [17]

Vstupem při šifrování textu je otevřený text. Výstup je po zašifrování textu reprezentován řetězcem hexadecimálních znaků. Při dešifrování je vstupem řetězec hexadecimálních znaků a výstupem je otevřený text.

Klíč může uživatel zadat buďto hexadecimálně nebo znaky ASCII tabulky podle toho, kterou možnost si vybere. Inicializační vektor je reprezentován řetězcem hexadecimálních znaků.

3.2.2 Šifrovací kód

Speciálním typem výstupu je šifrovací kód, který je výstupem vícenásobného šifrování textu i souborů. Tento kód obsahuje informace o nastavení šifrovacího procesu při vícenásobném šifrování. Šifrovací kód se tvoří následujícím způsobem:

První bajt obsahuje počet šifer, který byl použit při vícenásobném šifrování. Poté je k tomuto řetězci postupně přidán počet bloků odpovídající počtu použitých šifer. Blok se skládá z následujících částí a s následující délkou:

- Použitá šifra – 1 B
- Operační mód – 1 B (00 pro konvenční šifry)
- Mód pro doplnění bloku dat – 1 B (00 pro konvenční šifry)
- Délka klíče – 3 B
- Klíč – proměnlivá délka v závislosti na použité šifře nebo klíči zadaném uživatelem
- Délka inicializačního vektoru – 1 B
- Inicializační vektor – proměnlivá délka v závislosti na použité blokové šifře

Poslední dvě části týkající se inicializačního vektoru jsou přítomny pouze, pokud se jedná o blokovou šifru s použitým operačním módem CBC nebo CFB.

Tento šifrovací kód ovšem přináší zranitelnost v tom smyslu, že jsou v něm obsaženy všechny potřebné informace k dešifrování. Tedy pokud by se dostal do rukou útočníkovi, tak by mohl být zneužit k dešifrování zprávy. Ovšem musíme uvážit fakt, že místo toho aby uživatel udržoval v tajnosti několik klíčů a k tomu si navíc pamatoval jednotlivé nastavení použitých šifer, tak mu stačí držet v tajnosti tento jeden šifrovací kód, stejně jako by držel v tajnosti šifrovací klíče.

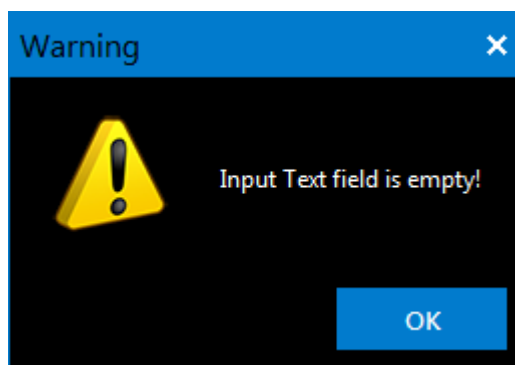
3.2.3 Soubory

Zašifrovat lze jakýkoliv soubor. Zašifrované soubory jsou ukládány s koncovkou .encd. Před zašifrováním souboru jsou datové bajty souboru ve formátu řetězce, ke kterému je přidán řetězec „5C2E2A2E2F“ a za tento řetězec je ještě přidán celý název původního souboru i s příponou v hexadecimálním tvaru. Tento řetězec je poté zašifrován a uložen jako soubor s koncovkou .encd. Tím je zajištěno, že je poté soubor dešifrován na soubor se stejným jménem a se stejným typem souboru jako byl původní soubor.

3.3 Varovné zprávy

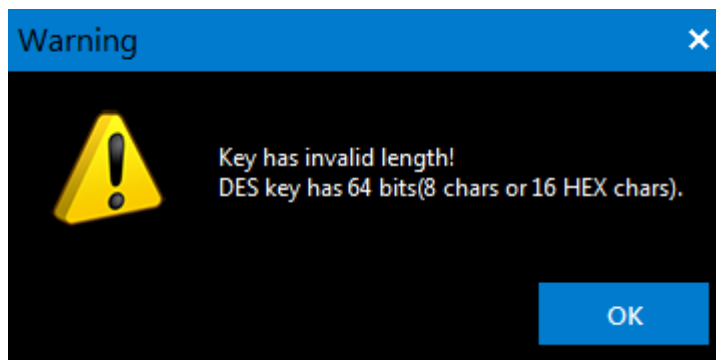
Aplikace obsahuje několik varovných zpráv ve formě dialogu, který je zobrazen za některých určitých okolností.

- Vstupní text k zašifrování/dešifrování nebyl zadán



Obr. 49 Varovná zpráva: Vstup nebyl zadán

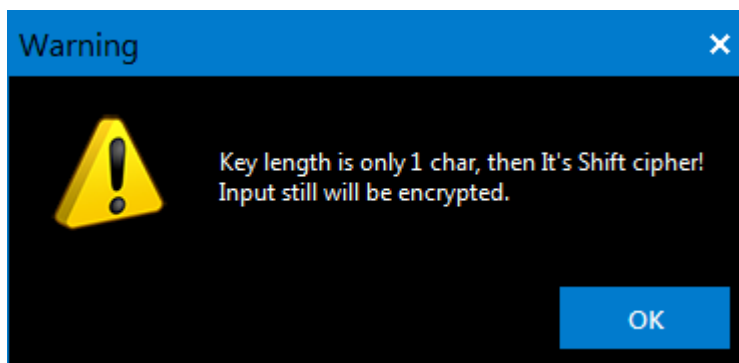
- Vstup k dešifrování textu není v hexadecimální formě - zobrazí se v případě, že uživatel zadal vstup k dešifrování textu, který není v hexadecimální formě a neobsahuje pouze znaky 0123456789abcdefABCDEF
- Cesta k souboru k zašifrování/dešifrování nebyla zadána - zobrazí se v případě, kdy uživatel nevybere cestu k souboru, který má být zašifrován/dešifrován
- Klíč nebyl zadán – zobrazí se v případě, že uživatel nezadal žádný klíč. Jedinou výjimkou u tohoto případu je šifrování pomocí Vernamovy šifry, kde uživatel nezadává klíč, protože ten je generován náhodně.
- Klíč není v hexadecimální formě – zobrazí se v případě, že uživatel vybral zadání hexadecimálního klíče a zadaný klíč není v hexadecimální formě a neobsahuje pouze znaky 0123456789abcdefABCDEF
- Klíč nemá správnou délku – zobrazí se v případě, že zadaný klíč nemá potřebnou délku. Význam má zejména u blokových šifer, kde je pevně daná délka klíčů pro jednotlivé šifry. U dešifrování musí mít klíč Vernamovy šifry stejnou velikost jako vstup k dešifrování.



Obr. 50 Varovná zpráva: Klíč nemá správnou délku

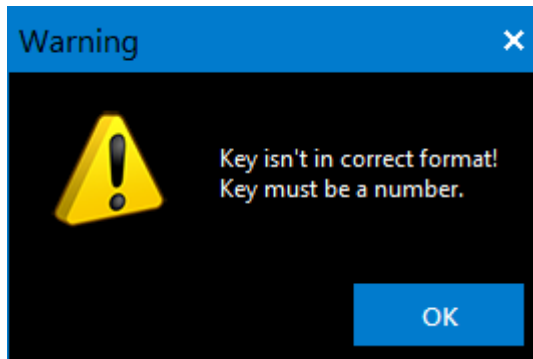
- Inicializační vektor nebyl zadán - zobrazí se v případě, že uživatel nezadal inicializační vektor. Tento případ může nastat pouze u dešifrování a to při použití blokové šifry v operačním módu CBC nebo CFB.
- Inicializační vektor nemá správnou délku - zobrazí se v případě, že zadaný inicializační vektor nemá potřebnou délku. Význam má zejména u blokových šifer, kde je pevně daná délka inicializačního vektoru pro jednotlivé šifry. Tento případ může nastat pouze u dešifrování a to při použití blokové šifry v operačním módu CBC nebo CFB.
- Inicializační vektor není v hexadecimální formě - zobrazí se v případě, že uživatelem zadaný inicializační vektor není v hexadecimální formě a neobsahuje pouze znaky 0123456789abcdefABCDEF
- Výstupní soubor nebyl vybrán – zobrazí se v případě, kdy uživatel nevybere cestu k souboru, kde má být uložen zašifrovaný soubor
- Složka pro uložení dešifrovaného souboru nebyla vybrána - zobrazí se v případě, kdy uživatel nevybere složku pro uložení dešifrovaného souboru
- Soubor není ve správném formátu – tato situace nastává při dešifrování souboru. Po dešifrování souboru aplikace hledá v dešifrovaném řetězci řetězec, který odděluje datové bajty od názvu původního souboru. Pokud tento řetězec nenajde, aplikace přeruší dešifrování a zobrazí danou varovnou hlášku. Tato situace může tedy nastat, pokud někdo pozmění zašifrovaný soubor nebo použije k dešifrování jiné nastavení šifry než při šifrování.

- Délka klíče pro Vigeněrovu šifru je jeden znak – zobrazí se v případě, že uživatel zadá klíč pouze o délce jednoho znaku, nebo 2 znaky v případě hexadecimálního klíče. V tomto případě s klíčem o délce 1 se totiž jedná pouze o šifru s pevným posunem.



Obr. 51 Varovná zpráva: Délka klíče pro Vigeněrovu šifru je 1

- Klíč není ve správné formě. Klíč musí být číslo - zobrazí se v případě použití šifry s pevným posunem, a to když uživatel zadá klíč, který není číslem a neobsahuje tedy pouze číslice 0123456789



Obr. 52 Varovná zpráva: Klíč není ve správném formátu

- Šifra nebyla vybrána - zobrazí se v případě, že uživatel nevybere z nabídky šifru, kterou chce použít k šifrování/dešifrování
- Operační mód nebyl vybrán - zobrazí se v případě, že uživatel nevybere z nabídky operační mód blokové šifry, který chce pro danou šifru použít k šifrování/dešifrování
- Chybný operační mód - zobrazí se v případě, že uživatel při dešifrování vybere operační mód odlišný od toho, který byl použit při šifrování

3.4 Časová náročnost šifrování

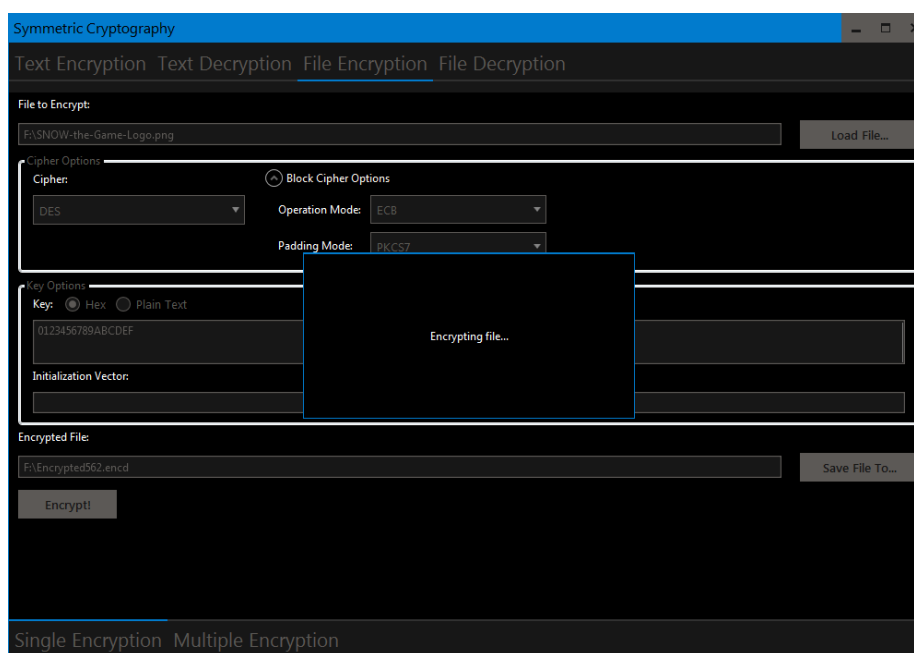
Se vzrůstající velikostí textu a souboru, také roste čas potřebný k zašifrování těchto dat. Nejpatrnější je to ovšem na souborech. Rychlost šifrování je závislá na výkonu počítače, zejména na procesoru.

V následující tabulce jsou naměřené časy šifrování pro několik souborů různých velikostí. Časy byly měřeny při běžné práci na počítači, při spuštěném MS Wordu, Google Chrome a MS Visual Studiu a na notebooku s procesorem Intel Core i5 s frekvencí 2,3 GHz. Všechny soubory byly šifrovány šifrou DES v operačním módu ECB. Časy v tabulce jsou pouze orientační a závisí na výkonu počítače a na použité šifře.

Velikost souboru [kB]	Čas šifrování [s]	Rychlost šifrování [kB/s]
20	1	20,00
95	21	4,52
133	42	3,17
170	69	2,46
252	156	1,62
306	225	1,36
352	302	1,17
409	412	0,99

Tabulka 1 Čas a rychlost šifrování souborů pomocí šifry DES

Pokud se tedy jedná o tyto časově náročnější operace, je deaktivováno GUI aplikace a v popředí je zobrazeno okno, které informuje o probíhající operaci.

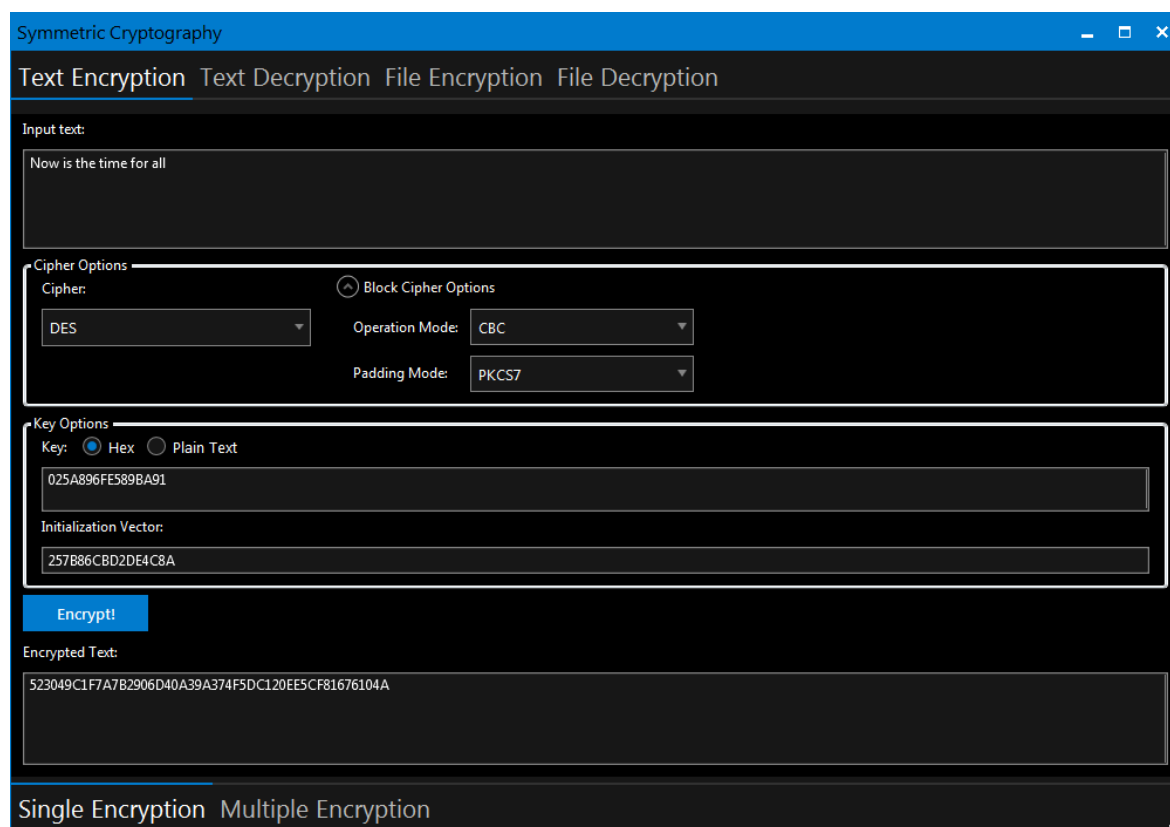


Obr. 53 Vzhled aplikace při časově náročnější operaci

3.5 Funkce aplikace

3.5.1 Šifrování textu

Šifrování textu najdeme na záložce Text Encryption a na této záložce vybereme ve spodní části aplikace záložku Single Encryption. Příklad zašifrování textu je na obr. 54. Uživatel nejdříve ve vrchní části aplikace zadá do pole text, který chce zašifrovat, poté provede nastavení šifry a zadání klíče. Jakmile uživatel zadá všechna potřebná data k zašifrování, tak kliknutím na tlačítko Encrypt! provede zašifrování zadaného textu. Zašifrovaný text se poté zobrazí v poli ve spodní části aplikace.



Obr. 54 Šifrování textu

3.5.2 Dešifrování textu

Dešifrování textu najdeme na záložce Text Decryption a na této záložce vybereme ve spodní části aplikace záložku Single Decryption. Příklad dešifrování zašifrovaného textu je na obr. 55. Uživatel nejdříve ve vrchní části aplikace zadá do pole zašifrovaný text, který chce dešifrovat, poté provede nastavení šifry a zadání klíče, popřípadě ještě zadání inicializačního vektoru, pokud uživatel použije u blokových šifer operační mód CBC nebo

CFB. Jakmile uživatel zadá všechna potřebná data k zašifrování, tak kliknutím na tlačítko Decrypt! provede dešifrování zadaného zašifrovaného textu. Výsledek dešifrování se poté zobrazí v poli ve spodní části aplikace.

Obr. 55 Dešifrování textu

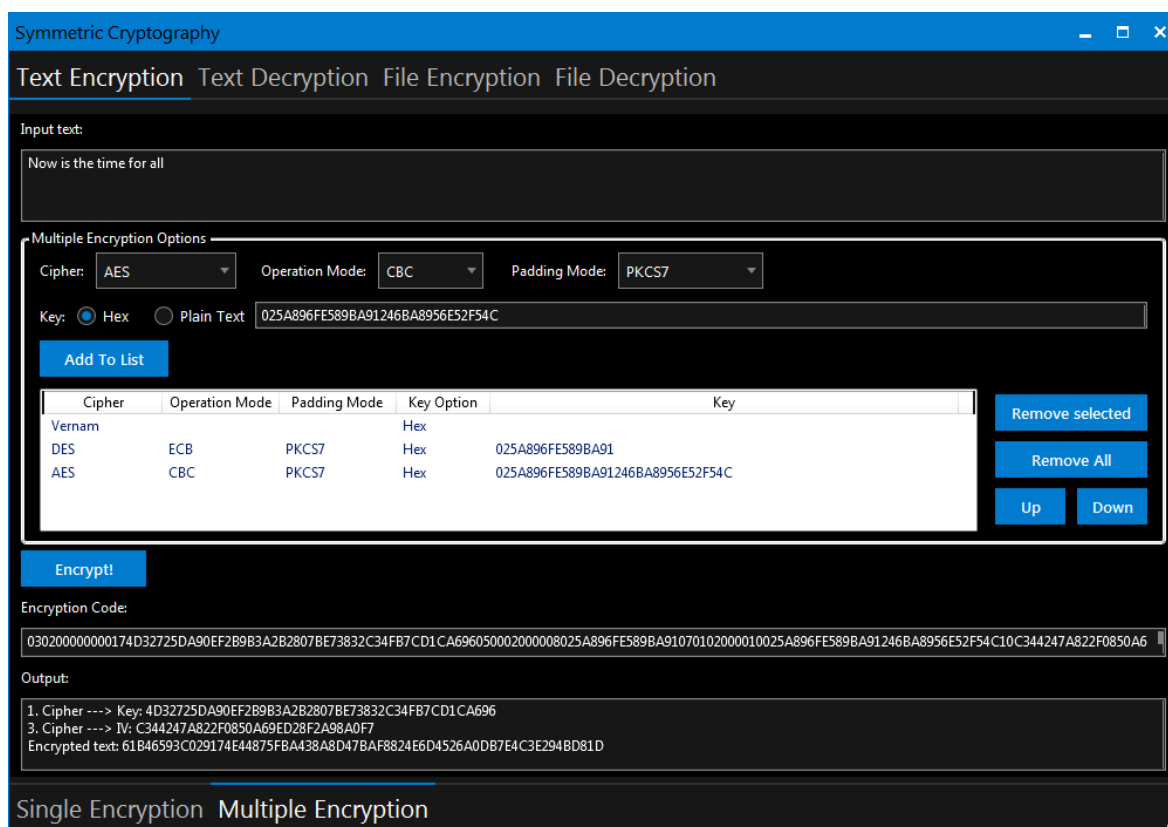
3.5.3 Vícenásobné šifrování textu

Vícenásobné šifrování textu najdeme na záložce Text Encryption a na této záložce vybereme ve spodní části aplikace záložku Multiple Encryption. Příklad vícenásobného zašifrování textu je na obr. 57. Uživatel nejdříve ve vrchní části aplikace zadá do pole text, který chce zašifrovat, poté uživatel nastaví jednotlivé šifry, které chce použít a přidá je do seznamu. Příklad zadání tří šifer k zašifrování textu a jejich zobrazení v seznamu je na obr. 56.

Cipher	Operation Mode	Padding Mode	Key Option	Key
Vernam			Hex	
DES	ECB	PKCS7	Hex	025A896FE589BA91
AES	CBC	PKCS7	Hex	025A896FE589BA91246BA8956E52F54C

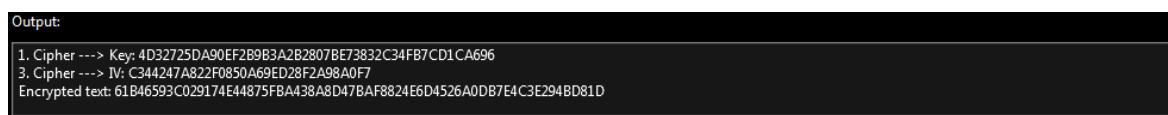
Obr. 56 Nastavení šifer pro vícenásobné šifrování textu

Až má uživatel v seznamu přidány všechny šifry, které chce použít, tak klikne na tlačítko Encrypt! a zadaný text bude zašifrován zadanými šiframi s daným nastavením.



Obr. 57 Vícenásobné šifrování textu

Výstupem jsou 2 pole ve spodní části aplikace, kde v prvním poli je šifrovací kód, který poté může být použit k dešifrování zašifrovaného textu, bez nutnosti zadávat všechny šifry s jejich nastaveními. Ve druhém poli je výstup po šifrování, který obsahuje samotný zašifrovaný text a také dodatečné informace, které jsou náhodně generovány v případě použití některých šifer, k jednotlivým použitým šifrům, jako je v našem případě klíč k použité Vernamově šifře a inicializační vektor k použité šifře AES v operačním módu CBC. Tento výstup je znázorněn na obr. 58.



Obr. 58 Výstup z vícenásobného šifrování textu

3.5.4 Vícenásobné dešifrování textu

Vícenásobné dešifrování textu najdeme na záložce Text Decryption a na této záložce vybereme ve spodní části aplikace záložku Multiple Decryption. Příklad vícenásobného dešifrování textu je na obr. 60. Uživatel nejdříve ve vrchní části aplikace zadá do pole zašifrovaný text, který chce dešifrovat, poté uživatel nastaví jednotlivé šifry, které byly použity k zašifrování, ale musí je do seznamu zadat v obráceném pořadí, než tomu bylo u vícenásobného šifrování textu, např. šifra, kterou jsme při zašifrování zadávali jako poslední, bude nyní jako první v seznamu. Příklad zadání tří šifer k dešifrování textu v obráceném pořadí než tomu bylo u vícenásobného šifrování textu a jejich zobrazení v seznamu je na obr. 59.

Cipher	Operation Mode	Padding Mode	Key Option	Key	Initialization Vector
AES	CBC	PKCS7	Hex	025A896FE589BA91246BA8956E52F5	C344247A822F0850A69ED28F2A98A0
DES	ECB	PKCS7	Hex	025A896FE589BA91	
Vernam			Hex	4D32725DA90EF2B9B3A2B2807BE73	

Obr. 59 Nastavení šifer pro vícenásobné dešifrování textu

Po nastavení všech šifer uživatel klikne na tlačítko Decrypt! a provede se dešifrování.

Symmetric Cryptography

Text Encryption Text Decryption File Encryption File Decryption

Input encrypted text:

61B46593C029174E44875FBA438A8D47BAF8824E6D4526A0DB7E4C3E2948D81D

Multiple Decryption Options

Cipher: Vernam Operation Mode: ECB Padding Mode: PKCS7

Key: ☒ Hex ☐ Plain Text 4D32725DA90EF2B9B3A2B2807BE73832C34FB7CD1CA696

Initialization Vector:

Add To List

Cipher	Operation Mode	Padding Mode	Key Option	Key	Initialization Vector
AES	CBC	PKCS7	Hex	025A896FE589BA91246BA8956E52F5	C344247A822F0850A69ED28F2A98A0
DES	ECB	PKCS7	Hex	025A896FE589BA91	
Vernam			Hex	4D32725DA90EF2B9B3A2B2807BE73	

Remove selected Remove All Up Down

Decrypt!

Decrypted Text:

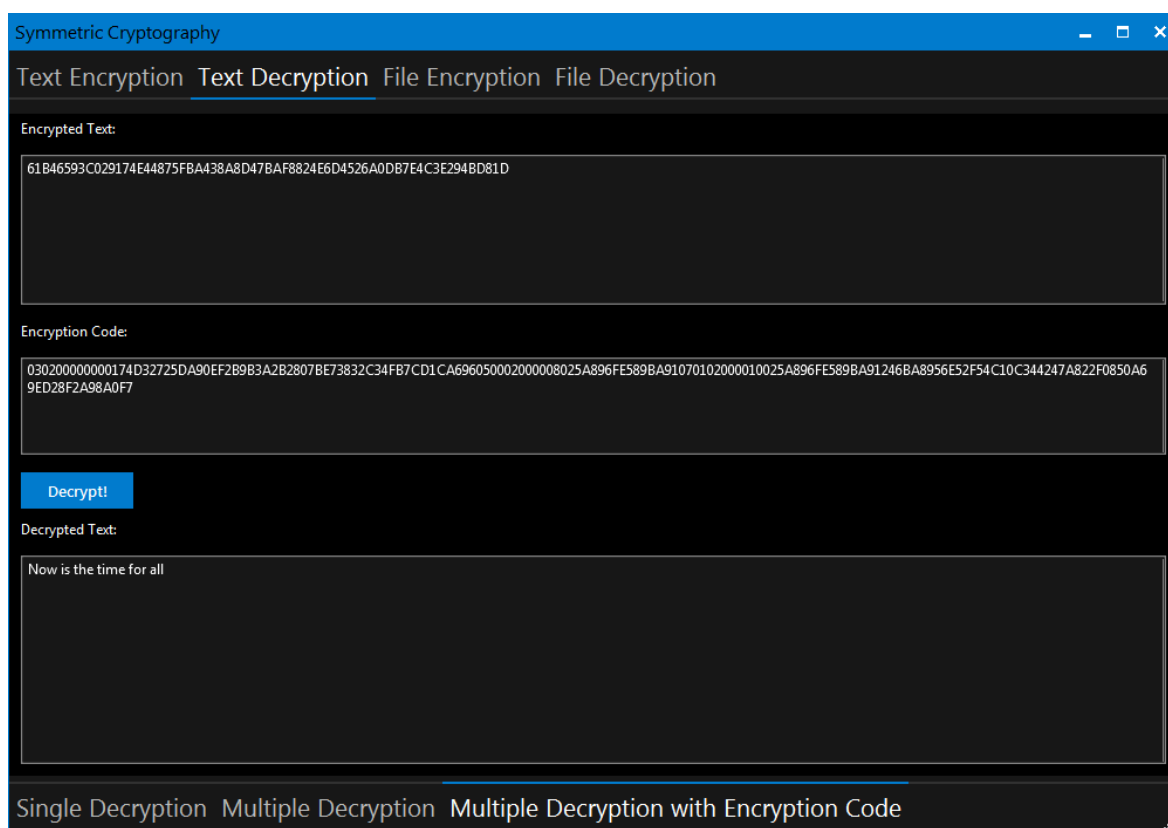
Now is the time for all

Single Decryption Multiple Decryption Multiple Decryption with Encryption Code

Obr. 60 Vícenásobné dešifrování textu

3.5.5 Vícenásobné dešifrování textu pomocí šifrovacího kódu

Vícenásobné dešifrování textu pomocí šifrovacího kódu najdeme na záložce Text Decryption a na této záložce vybereme ve spodní části aplikace záložku Multiple Decryption with Encryption Code. Příklad vícenásobného dešifrování textu pomocí šifrovacího kódu je na obr. 61. Uživatel nejdříve ve vrchní části aplikace zadá do pole zašifrovaný text, který chce dešifrovat. Poté do pole pro šifrovací kód zadá šifrovací kód, který byl generován aplikací při vícenásobném šifrování textu. Po vyplnění těchto dvou polí uživatel klikne na tlačítko Decrypt! a zašifrovaný text se dešifruje.

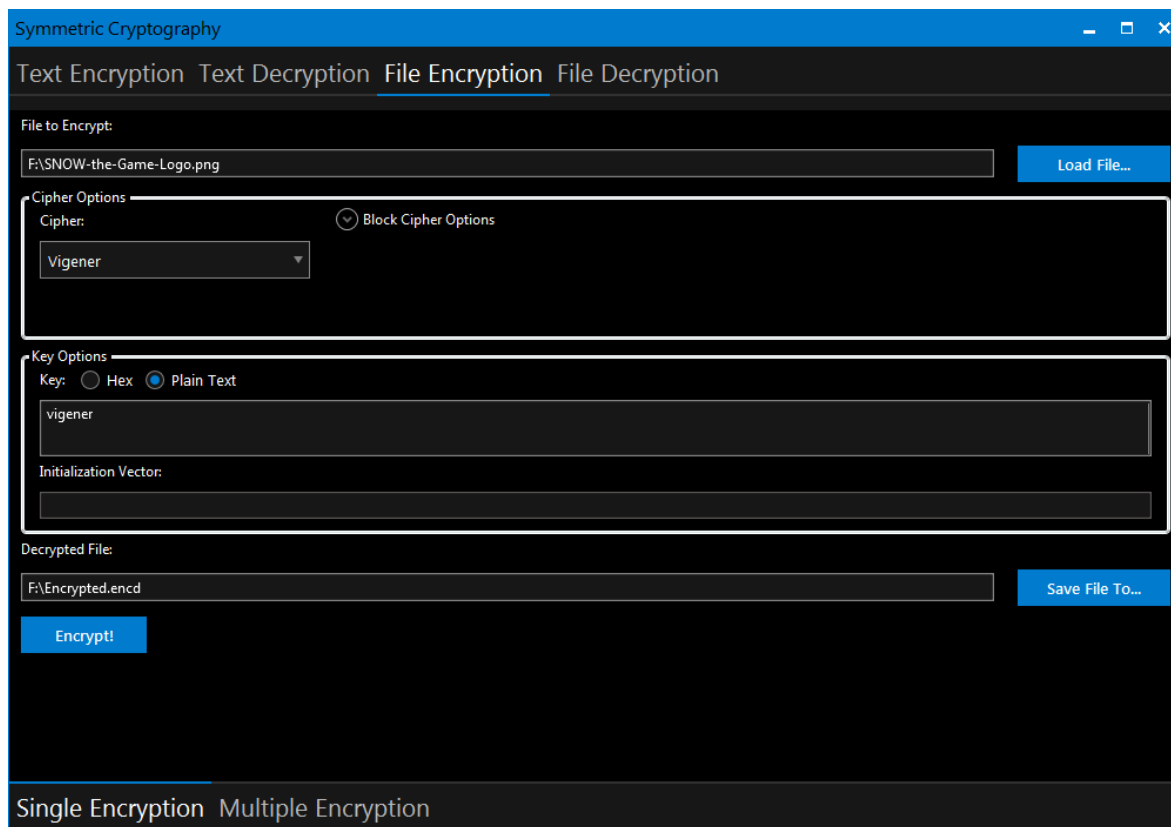


Obr. 61 Vícenásobné dešifrování textu pomocí šifrovacího kódu

3.5.6 Šifrování souboru

Šifrování souboru najdeme na záložce File Encryption a na této záložce vybereme ve spodní části aplikace záložku Single Encryption. Příklad šifrování textu je na obr. 62. Uživatel nejdříve vybere soubor, který chce zašifrovat. Vybere ho tak, že v horní části aplikace klikne na tlačítko Load File..., které otevře dialog pro výběr souboru. Poté uživatel nastaví šifru, zadá klíč a zvolí umístění a název zašifrovaného souboru. Umístění a název zašifrovaného souboru uživatel volí pomocí dialogu pro uložení souboru, který

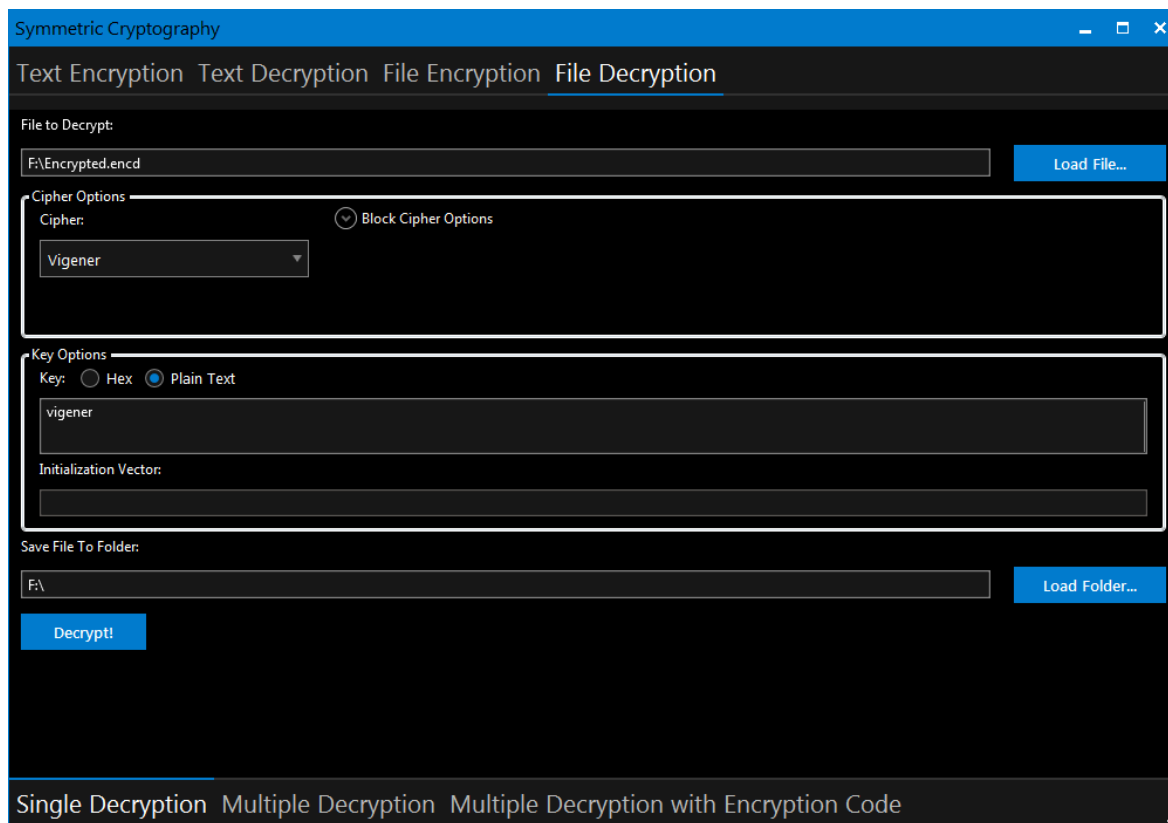
zobrazí kliknutím na tlačítko Save File To.... Po zadání a nastavení všech těchto potřebných informací k šifrování uživatel klikne na tlačítko Encrypt! a provede se zašifrování vybraného souboru.



Obr. 62 Šifrování souboru

3.5.7 Dešifrování souboru

Dešifrování souboru najdeme na záložce File Decryption a na této záložce vybereme ve spodní části aplikace záložku Single Decryption. Příklad dešifrování zašifrovaného souboru je na obr. 63. Uživatel nejdříve vybere soubor, který chce dešifrovat. Vybere ho tak, že v horní části aplikace klikne na tlačítko Load File..., které otevře dialog pro výběr souboru, který umožňuje vybrat pouze soubor s příponou .ncd. Poté uživatel nastaví šifru, zadá klíč, popřípadě ještě zadá inicializační vektor, pokud uživatel použije u blokových šifer operační mód CBC nebo CFB. Poté uživatel vybere umístění dešifrovaného souboru pomocí výběru složky, který zobrazí kliknutím na tlačítko Load Folder.... Po zadání a nastavení všech těchto potřebných informací k dešifrování uživatel klikne na tlačítko Decrypt! a provede se dešifrování vybraného souboru.



Obr. 63 Dešifrování souboru

3.5.8 Vícenásobné šifrování souboru

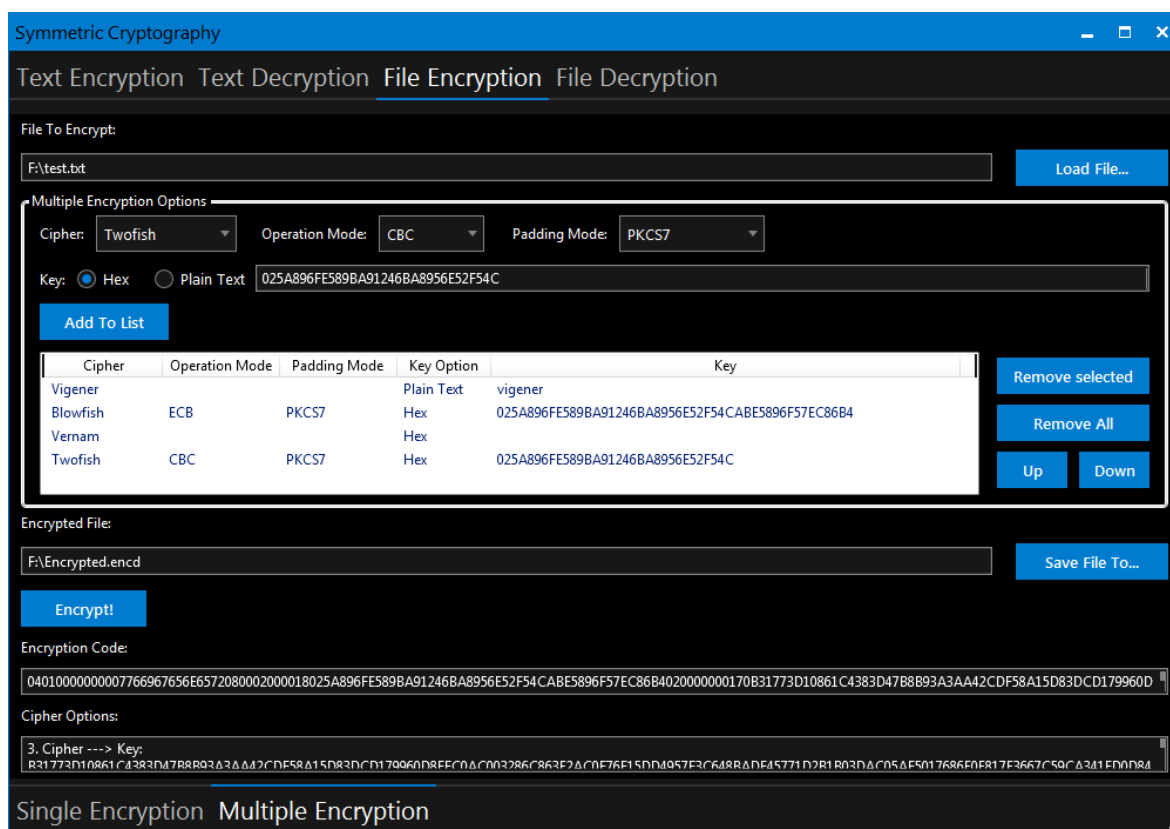
Vícenásobné šifrování souboru najdeme na záložce File Encryption a na této záložce vybereme ve spodní části aplikace záložku Multiple Encryption. Příklad vícenásobného zašifrování souboru je na obr. 65. Uživatel nejdříve vybere soubor, který chce zašifrovat. Vybere ho tak, že v horní části aplikace klikne na tlačítko Load File..., které otevře dialog pro výběr souboru. Poté uživatel nastaví jednotlivé šifry, které chce použít a přidá je do seznamu. Příklad zadání čtyř šifer k zašifrování souboru a jejich zobrazení v seznamu je na obr. 64.

Cipher	Operation Mode	Padding Mode	Key Option	Key
Vigener			Plain Text	vigener
Blowfish	ECB	PKCS7	Hex	025A896FE589BA91246BA8956E52F54CABE5896F57EC86B4
Vernam			Hex	
Twofish	CBC	PKCS7	Hex	025A896FE589BA91246BA8956E52F54C

Obr. 64 Nastavení šifer pro vícenásobné šifrování souboru

Po nastavení šifer musí uživatel ještě vybrat umístění a název zašifrovaného souboru a to tak, že klikne na Save File To..., čímž se zobrazí dialog pro uložení souboru s příponou

.encd. Po zadání a nastavení všech těchto potřebných informací k šifrování uživatel klikne na tlačítko Encrypt! a provede se šifrování vybraného souboru.



Obr. 65 Vícenásobné šifrování souboru

Výstupem je tedy zašifrovaný soubor v uživatelem zadaném umístění a 2 pole ve spodní části aplikace, kde v prvním poli je šifrovací kód, který poté může být použit k dešifrování zašifrovaného souboru, bez nutnosti zadávat všechny šifry s jejich nastaveními. Ve druhém poli je výstup po šifrování, dodatečné informace k šifrování, které jsou náhodně generovány v případě použití některých šifer, k jednotlivým použitým šifrům, jako je v našem případě klíč k použité Vernamově šifře a inicializační vektor k použité šifře Twofish v operačním módu CBC.

3.5.9 Vícenásobné dešifrování souboru

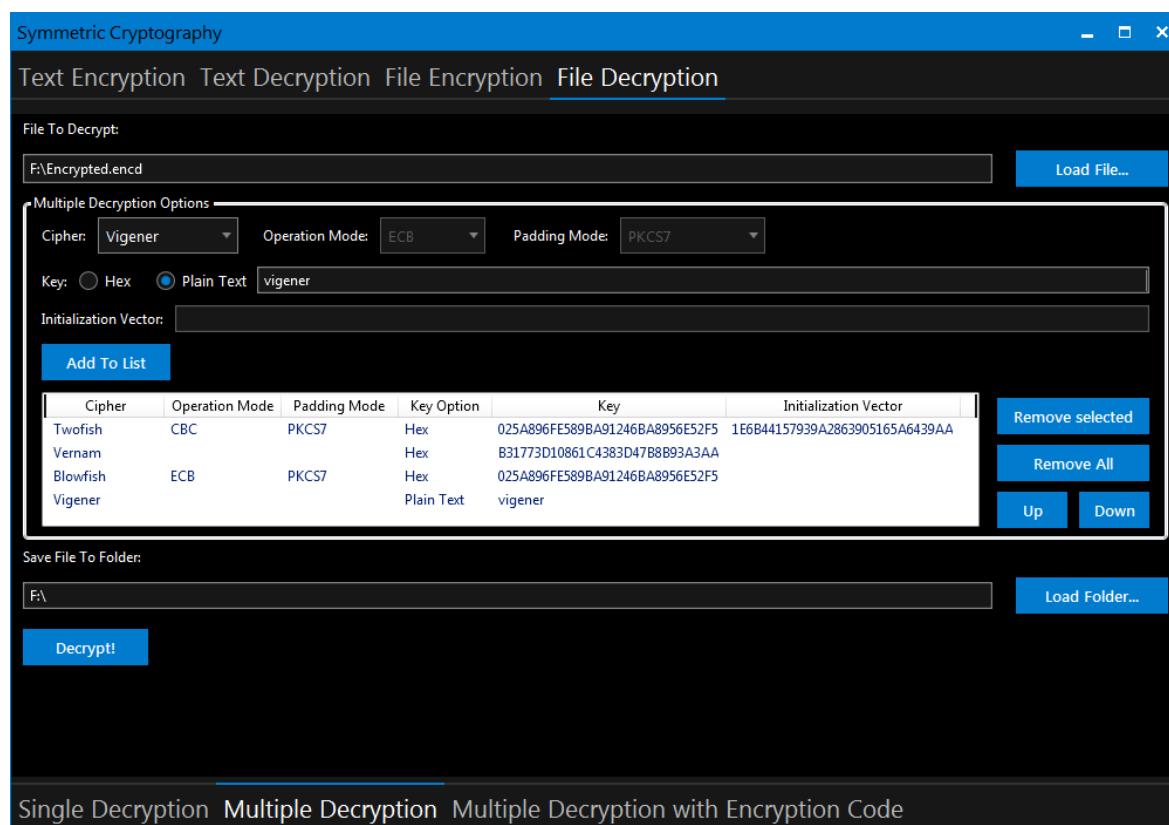
Vícenásobné dešifrování souboru najdeme na záložce File Decryption a na této záložce vybereme ve spodní části aplikace záložku Multiple Decryption. Příklad vícenásobného dešifrování souboru je na obr. 67. Uživatel nejdříve vybere soubor, který chce dešifrovat. Vybere ho tak, že v horní části aplikace klikne na tlačítko Load File..., které otevře dialog pro výběr souboru s koncovkou .encd. Poté uživatel nastaví jednotlivé šifry, které byly

použity k zašifrování, ale musí je do seznamu zadat v obráceném pořadí, než tomu bylo u vícenásobného šifrování souboru, např. šifra, kterou jsme při zašifrování zadávali jako poslední, bude nyní jako první v seznamu. Příklad zadání čtyř šifer k dešifrování souboru v obráceném pořadí než tomu bylo u vícenásobného šifrování textu a jejich zobrazení v seznamu je na obr. 66.

Cipher	Operation Mode	Padding Mode	Key Option	Key	Initialization Vector
Twofish	CBC	PKCS7	Hex	025A896FE589BA91246BA8956E52F5	1E6B44157939A2863905165A6439AA
Vernam			Hex	B31773D10861C4383D47B8B93A3AA	
Blowfish	ECB	PKCS7	Hex	025A896FE589BA91246BA8956E52F5	
Vigener			Plain Text	vigener	

Obr. 66 Nastavení šifer pro vícenásobné dešifrování souboru

Po nastavení šifer musí ještě uživatel vybrat složku do které bude uložen dešifrovaný soubor. To provede tak, že klikne na tlačítko Load Folder..., které zobrazí výběr složky. Po zadání a nastavení všech těchto potřebných informací k dešifrování uživatel klikne na tlačítko Decrypt! a provede se dešifrování vybraného souboru.

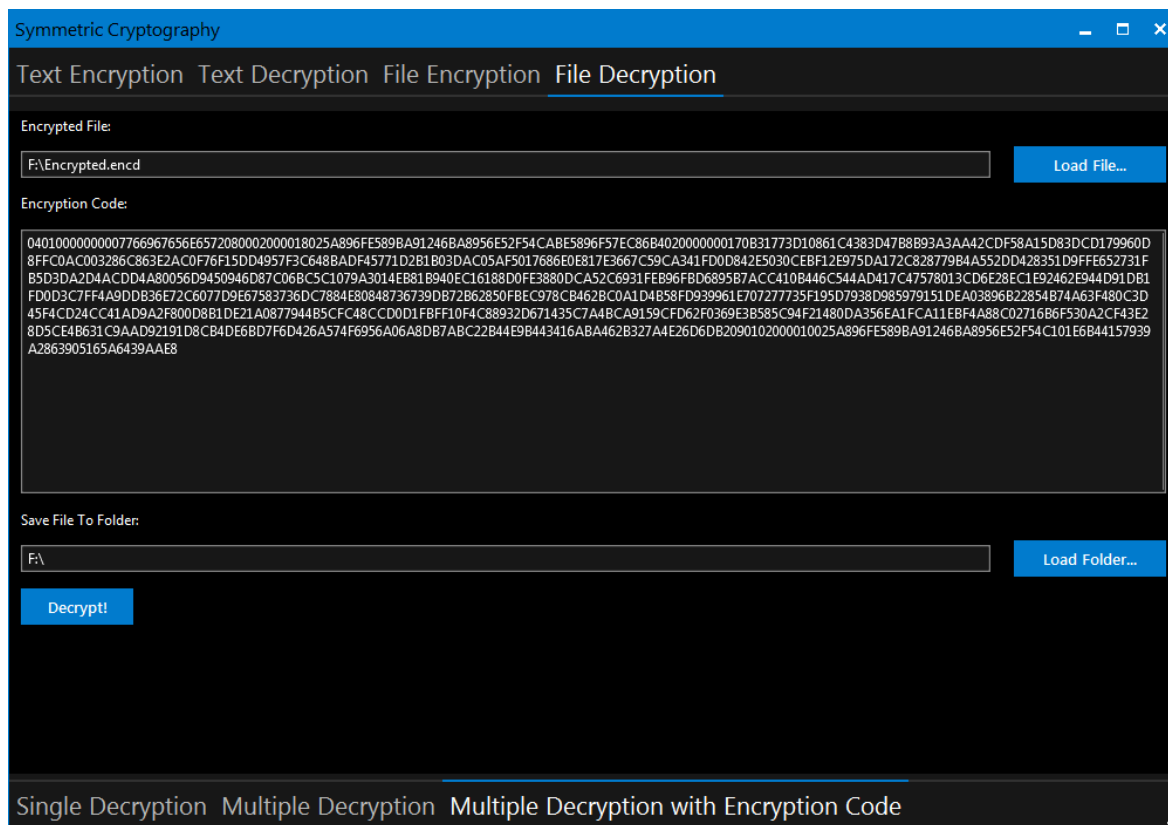


Obr. 67 Vícenásobné dešifrování souboru

Výstupem je dešifrovaný soubor s původním jménem uložený na uživatelem zadaném umístění.

3.5.10 Vícenásobné dešifrování souboru pomocí šifrovacího kódu

Vícenásobné dešifrování souboru pomocí šifrovacího kódu najdeme na záložce File Decryption a na této záložce vybereme ve spodní části aplikace záložku Multiple Decryption with Encryption Code. Příklad vícenásobného dešifrování souboru pomocí šifrovacího kódu je na obr. 68. Uživatel nejdříve vybere soubor, který chce dešifrovat. Vybere ho tak, že v horní části aplikace klikne na tlačítko Load File..., které otevře dialog pro výběr souboru s koncovkou .encd. Poté do pole pro šifrovací kód zadá šifrovací kód, který byl generován aplikací při vícenásobném šifrování textu. Poté uživatel vybere složku do které bude uložen dešifrovaný soubor. To provede tak, že klikne na tlačítko Load Folder..., které zobrazí výběr složky. Po vyplnění všech těchto polí uživatel klikne na tlačítko Decrypt! a zašifrovaný soubor se dešifruje.



Obr. 68 Vícenásobné dešifrování souboru pomocí šifrovacího kódu

4 TESTOVÁNÍ APLIKACE

Aplikace byla testována na několika různých textech, které jsou složeny ze znaků abecedy, znaků s diakritikou a dalších znaků. Testována byla každá šifra s jednotlivými možnostmi nastavení. Pro každou šifru bylo i otestováno několik různých klíčů, v některých případech i různé délky klíčů, zejména u některých blokových šifer, které akceptují např. dvě nebo tři délky klíče. Stejný postup byl použit i při testování šifrování a dešifrování souboru. Pro testování bylo použito několik souborů různých formátů jako např. textové dokumenty, obrázky, pdf soubory. Postupně tak byly otestovány všechny funkce aplikace. Testované texty a soubory jsou přiloženy na CD.

5 MOŽNOSTI MODULÁRNÍHO ROZŠÍŘENÍ APLIKACE

Aplikaci je možno rozšířit o další šifry, např. z kryptografické knihovny Bouncy Castle, ze které jsou v aplikaci použity tři šifry a to Blowfish, Twofish a Serpent. Tato knihovna obsahuje spoustu dalších šifer.

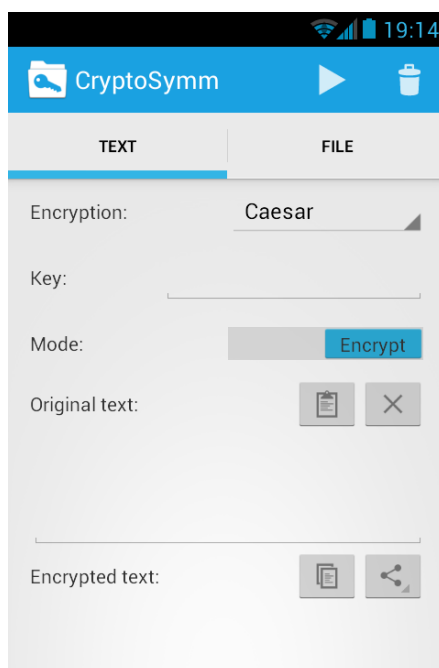
Aplikace je rozvržena tak, že každá šifra je v samostatném .cs souboru. Všechny tyto soubory jsou ve stejném jmenném prostoru SymmetricCiphers. Např. soubor pro výše zmíněné tři šifry obsahuje třídu s názvem šifry (BlowfishCipher, TwofishCipher, SerpentCipher). Tato třída obsahuje tři privátní proměnné, které slouží pro nastavení operačního módu blokové šifry, módu pro doplnění bloku dat a inicializačního vektoru. Třída dále obsahuje dva konstruktory, které slouží pro nastavení výše zmíněných proměnných. Třída dále obsahuje funkce pro generování inicializačního vektoru, šifrování, dešifrování, šifrování textu, dešifrování textu, šifrování souboru a dešifrování souboru.

Pokud bychom tedy chtěli do aplikace přidat další šifru a byla by to např. bloková šifra z knihovny Bouncy Castle stejně jako výše zmíněné šifry, tak bychom si zkopírovali obsah souboru některé existující šifry, kde bychom museli samozřejmě změnit název třídy a poté ve funkci na generování inicializačního vektoru případně změnit velikost generovaného inicializačního vektoru, která je v bajtech. Hlavním krokem by ovšem bylo změnit použitou třídu šifry z knihovny Bouncy Castle ve funkcích pro šifrování a dešifrování. Např. bychom vyměnili BlowfishEngine() za IDEAEngine(). Tím by ovšem naše práce nebyla ještě hotová, dále bychom museli v souboru MainWindow.xaml, což je soubor definující vzhled a vlastnosti grafického rozhraní, přidat naši šifru na první záložku do prvku ComboBox k ostatním šifrám. Poté bychom museli v souboru MainWindow.xaml.cs přidat do konstrukce switch pro první záložku další možnost, která by reprezentovala naši nově přidanou šifru v ComboBoxu, kde bychom provedli ošetření délky klíče a šifrování podobně jak v ostatních možnostech konstrukce switch. Tento postup bychom poté provedli pro všechny ostatní záložky aplikace, které umožňují vybírat šifru pomocí grafického prvku ComboBox.

6 ANALÝZA EXISTUJÍCÍCH MOBILNÍCH APLIKACÍ

Pro mobilní platformu Android existuje mnoho aplikací, které umožňují šifrování SMS zpráv, textů a souborů v telefonu atd. Na ukázkou jsou popsány následující 2 aplikace.

Aplikace **CryptoSymm (Cipher)** je aplikace, která umožňuje šifrování a dešifrování textů a souborů a jejich následné sdílení přímo z aplikace. Tato aplikace implementuje několik symetrických šifrovacích algoritmů. Některé algoritmy je možno použít i pro šifrování textů a souborů, některé však lze použít pouze pro šifrování textů. Šifry, které jsou dostupné pouze pro šifrování textů: Caesarova šifra a Vigenérova šifra. Šifry, které jsou dostupné jak pro šifrování textů tak i souborů jsou DES, Triple DES, AES, Blowfish a RC4.

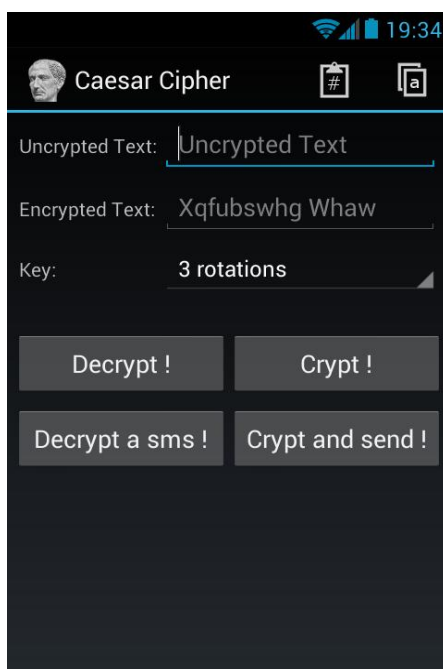


Obr. 69 Hlavní okno mobilní aplikace CryptoSymm

Tato aplikace je pomocí 2 záložek rozdělená na část pro šifrování textů a část pro šifrování souborů. Aplikace je velice intuitivní a jednoduchá na používání.

Další jednoduchou aplikací je aplikace **Caesar Cipher**, která implementuje šifru s pevným posunem, kde lze zvolit posun 0 až 25, tedy posun o libovolný počet míst v abecedě. Výhodou této aplikace je, že poskytuje možnost pro zašifrování textu a jeho poslání např. přes email nebo SMS pomocí nativních aplikací Androidu. Např. pokud uživatel zvolí

odeslání textu jako SMS, tak poté stačí, aby uživatel zadal číslo na které se má zašifrovaná zpráva poslat.



Obr. 70 Hlavní okno mobilní aplikace Caesar Cipher

Aplikace je velice jednoduchá na používání, obsahuje pouze pole pro vstup k šifrování/dešifrování, výběr posunu a tlačítka pro šifrování/dešifrování.

Pro mobilní platformu iOS je zřejmě nejzajímavější aplikace iMessage. Podle mnoha článků z amerických serverů z dubna 2013, sami americké bezpečnostní agentury FBI a DEA přiznaly, že je pro ně téměř nemožné zachytit komunikaci mezi dvěma uživateli s Apple zařízeními. Tohoto je docíleno tím, že je aplikace vytvořena na tzv. principu end-to-end šifrování, což znamená, že šifrování i dešifrování probíhá na zařízeních uživatele a nejsou tedy žádná data odesílána někam na server, což teoreticky znamená, že žádná třetí strana se nedostane ke zprávám uživatele. [18]

ZÁVĚR

Cílem této práce bylo vytvořit rešerši na téma Symetrická kryptografie a vytvořit aplikaci implementující konvenční i moderní symetrické šifry pro šifrování textů a souborů.

Na začátku práce je stručný úvod do kryptologie a některých pojmů z kryptologie. Je zde uvedeno rozdělení kryptologie, kde je okrajově zmíněno něco o kryptografii obecně a o jejím dělení. Jako součástí kryptologie je uvedeno i několik metod kryptoanalýzy.

Další část práce se už podrobně zabývá symetrickou kryptografií a jsou postupně popsány principy funkce historických, proudových a blokových šifer. Nejpodrobněji jsou popsány blokové šifry, u kterých je také možno použít mnoho různých možností. Před tím než jsou popsány samotné algoritmy jednotlivých blokových šifer, tak jsou vysvětleny jednotlivé módy pro doplnění bloku dat, které zajišťují doplnění bloku vstupních dat na délku, která je požadována pro danou šifru. Poté jsou popsány operační módy blokových šifer, které určují způsob jakým je bloková šifra použita k šifrování/dešifrování. Po seznámení se s těmito možnostmi blokových šifer, jsou již popsány principy funkce nejčastěji používaných blokových šifer.

Praktickou částí této práce bylo naprogramování komplexního systému pro šifrování a dešifrování textů a souborů pomocí konvenčních a moderních šifer. Aplikace je naprogramována v programovacím jazyce C#. Pro tvorbu moderního vzhledu aplikace byla použita knihovna Elysium, která obsahuje grafické prvky pro tvorbu aplikace ve stylu Metro aplikací. Moderní blokové šifry použité v aplikaci jsou implementovány ze dvou kryptografických knihoven a to z knihovny System.Security.Cryptography, která je knihovnou jazyka C# a druhou knihovnou je knihovna Bouncy Castle. Aplikace obsahuje 4 konvenční šifry a 6 moderních blokových šifer. Pro moderní blokové šifry aplikace implementuje 3 operační módy těchto šifer a 4 módy pro doplnění bloku dat. Aplikace kromě běžného šifrování a dešifrování obsahuje také vícenásobné dešifrování, které uživateli umožňuje nadefinovat nastavení několika šifer, kterými je poté vstupní text nebo soubor zašifrován. U vícenásobného šifrování, aplikace vygeneruje šifrovací kód, který obsahuje nastavení všech šifer i s klíči a je možné s tímto kódem poté dešifrovat zašifrovaný text nebo soubor aniž by uživatel musel definovat všechny šifry, které byly použity k zašifrování.

ZÁVĚR V ANGLIČTINĚ

The objective of this work was to create research about symmetric cryptography and create application, which implements encryption and decryption of texts and files with conventional and modern symmetric ciphers.

At the beginning of the work is a brief introduction to cryptology and some of the terms of cryptology. There is a breakdown of cryptology, where is marginally mentioned something about cryptography in general and its division. There is described several methods of cryptanalysis as part of cryptology.

The next part of the work describes symmetric cryptography in detail and there are described principles of historical, stream and block ciphers. Block ciphers are described in the most detail, they can be used with many different options. There are described different padding modes, which provide replenishment of the block of input data to a length that is required for the cipher. Then there are described operation modes of block ciphers that determine the method, how is block cipher used to encryption/decryption. After description of these options of block ciphers, there are described principles of the most common used block ciphers.

The practical part of this work was to program complex system for encryption and decryption texts and files with conventional and modern ciphers. Application is programmed in C# language. For modern look of the graphic user interface was used Elysium library, which contains Metro style graphic elements. Modern block ciphers in the application are implemented from two cryptographic libraries, the first one is System.Security.Cryptography, which is library from C# language, and the second one is Bouncy Castle library. Application implements 4 conventional ciphers and 6 modern block ciphers. Application implements 3 operating modes and 4 padding modes as options for block ciphers. Application also contains multiple encryption and decryption, which allows the user to define several ciphers, which are then used for encryption. For multiple encryption, application generates encryption code, which contains cipher options and keys and can be then used to decrypt text or file without setting all ciphers and keys, which have been used in encryption.

SEZNAM POUŽITÉ LITERATURY

- [1] MOLLIN, Richard A. An introduction to cryptography. 2nd ed. Boca Raton: Chapman, c2007, x, 413 s. ISBN 15-848-8618-8.
- [2] PIPER, F a Sean MURPHY. Kryptografie. 1. vyd. v českém jazyce. Překlad Pavel Mondschein. Praha: Dokořán, 2006, 157 s. ISBN 80-736-3074-5.
- [3] Types of attacks on cryptosystem. [online]. [cit. 2014-03-22]. Dostupné z: http://www.encryptionanddecryption.com/encryption/types_of_attacks.html
- [4] PAAR, Christof. Understanding cryptography: a textbook for students and practitioners. Heidelberg: Springer, c2010, xviii, 372 s. ISBN 978-3-642-04100-6.
- [5] KATZ, Jonathan a Yehuda LINDELL. Introduction to modern cryptography. Boca Raton: Chapman, 2008, xviii, 534 s. ISBN 978-1-58488-551-1.
- [6] PAUL, Goutam a Subhamoy MAITRA. *RC4 stream cipher and its variants*. Boca Raton, FL: CRC Press, c2012, xxv, 285 p. Discrete mathematics and its applications. ISBN 14-398-3135-1.
- [7] Vigenère Cipher. [online]. [cit. 2014-04-09]. Dostupné z: https://www.kettering.edu/sites/default/files/resource-file-download/Vigenere%20Cipher_1.pdf
- [8] Bezpečná komunikace uživatelů. [online]. [cit. 2014-04-10]. Dostupné z: <http://www.elektrorevue.cz/clanky/02021/index.html>
- [9] Encrypting Passwords With an Old-School Tabula Recta. [online]. [cit. 2014-04-10]. Dostupné z: <http://math.wonderhowto.com/inspiration/encrypting-passwords-with-old-school-tabula-recta-0123346/>
- [10] PaddingMode Enumeration (System.Security.Cryptography). [online]. [cit. 2014-04-10]. Dostupné z: [http://msdn.microsoft.com/en-us/library/system.security.cryptography.paddingmode\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/system.security.cryptography.paddingmode(v=vs.110).aspx)
- [11] Recommendation for Block Cipher Modes of Operation. [online]. [cit. 2014-04-10]. Dostupné z: <http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf>
- [12] Blowfish Encryption Algorithm. [online]. [cit. 2014-04-16]. Dostupné z: <http://pocketbrief.net/related/BlowfishEncryption.pdf>
- [13] Encrypting data with the Blowfish algorithm. [online]. [cit. 2014-04-16]. Dostupné z: <http://www.design-reuse.com/articles/5922/encrypting-data-with-the-blowfish-algorithm.html>
- [14] Twofish: A 128-Bit Block Cipher. [online]. [cit. 2014-04-20]. Dostupné z: <https://www.schneier.com/paper-twofish-paper.pdf>
- [15] On the Twofish Key Schedule. [online]. [cit. 2014-04-20]. Dostupné z: <https://www.schneier.com/paper-twofish-keysched.pdf>

[16] Serpent: A New Block Cipher Proposal. [online]. [cit. 2014-04-20]. Dostupné z: <http://www.cl.cam.ac.uk/~rja14/Papers/serpent0.pdf>

[17] Ascii Table - Codepage 1250 - Latin 2 Windows. [online]. [cit. 2014-05-03]. Dostupné z: <http://ascii-table.com/codepage.php?1250>

[18] Apple's iMessage encryption foils law enforcement, Justice Department complains - The Washington Post. [online]. [cit. 2014-05-13]. Dostupné z: http://www.washingtonpost.com/business/technology/apples-imessage-encryption-foils-law-enforcement-justice-department-complains/2013/04/05/f4a6b66e-9d68-11e2-a2db-efc5298a95e1_story.html

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

RSA	Rivest Shamir Adleman – návrháři šifry RSA
PGP	Pretty Good Privacy
FISH	Fibonacci Shrinking
XOR	Exclusive OR
SSL	Secure Socket Layers
TSL	Transport Layer Security
WEP	Wired Equivalent Privacy
WPA	Wi-Fi Protected Access
KSA	Key Scheduling Algorithm
PRGA	Pseudo-Random Generation Algorithm
ECB	Electronic Codebook
CBC	Cipher Block Chaining
CFB	Cipher Feedback
OFB	Output Feedback
CTR	Counter
NIST	National Institute of Standards and Technology
DES	Data Encryption Standard
IBM	International Bussiness Machines
AES	Advanced Encryption Standard
MDS	Maximum Distance Separable
PHT	Pseudo-Hadamard Transformation
IP	Initial Permutation
FP	Final Permutation

SEZNAM OBRÁZKŮ

Obr. 1 Základní princip šifrování a dešifrování [8]	13
Obr. 2 Princip šifrování a dešifrování pomocí symetrického šifrovacího algoritmu [8].....	13
Obr. 3 Princip šifrování a dešifrování pomocí asymetrického šifrovacího algoritmu [8]	14
Obr. 4 Princip šifrování hybridní kryptografie [8].....	15
Obr. 5 Princip dešifrování hybridní kryptografie [8].....	15
Obr. 6 Šifrování a dešifrování pomocí Caesarovy šifry [4]	18
Obr. 7 Vigenèrov čtverec [9]	19
Obr. 8 Pseudokód algoritmu KSA [6].....	21
Obr. 9 Pseudokód algoritmu PRGA [6]	21
Obr. 10 Princip šifrování a dešifrování módu ECB [11]	24
Obr. 11 Princip šifrování a dešifrování módu CBC [11]	25
Obr. 12 Princip šifrování a dešifrování módu CFB [11].....	26
Obr. 13 Princip šifrování a dešifrování módu OFB [11]	27
Obr. 14 Princip šifrování a dešifrování módu CTR [11]	28
Obr. 15 Počáteční permutace klíče šifry DES [4].....	29
Obr. 16 Permutace klíče šifry DES [4]	29
Obr. 17 Expanze bloku dat [4] Obr. 18 Permutace bloku dat v f – funkci [4].....	30
Obr. 19 Příklad substituční tabulky S_1 [4]	30
Obr. 20 Struktura f – funkce [4].....	30
Obr. 21 Feistelova struktura šifry DES [4]	31
Obr. 22 Počáteční permutace bloku otevřeného textu šifry DES [4]	32
Obr. 23 Konečná permutace bloku dat šifry DES [4]	32
Obr. 24 Odvození podklíčů ze 128 bitového klíče [4].....	34
Obr. 25 Odvození podklíčů ze 192 bitového klíče [4].....	35
Obr. 26 Odvození podklíčů ze 256 bitového klíče [4].....	35
Obr. 27 Substituční tabulka šifry AES [4]	36
Obr. 28 Posun řádků matice dat [4]	36
Obr. 29 Příklad operace Mix Columns [4]	37
Obr. 30 Struktura šifrování AES [4]	37
Obr. 31 Příklad operace Inverse Mix Columns [4].....	38

Obr. 32 Inversní posun řádků matice dat [4]	38
Obr. 33 Inversní substituční tabulka šifry AES [4]	38
Obr. 34 Struktura dešifrování AES [4]	39
Obr. 35 f – funkce šifry Blowfish [13]	41
Obr. 36 Struktura šifry Blowfish [13]	42
Obr. 37 g – funkce šifry Twofish [14]	45
Obr. 38 MDS matice [14]	45
Obr. 39 F – funkce šifry Twofish [14]	46
Obr. 40 Struktura šifry Twofish [14]	47
Obr. 41 Počáteční permutace šifry Serpent [16]	48
Obr. 42 Konečná permutace šifry Serpent [16]	49
Obr. 43 Hlavní okno aplikace	52
Obr. 44 Nastavení šifry	53
Obr. 45 Nastavení blokové šifry	53
Obr. 46 Nastavení klíče	54
Obr. 47 Nastavení vícenásobného šifrování	54
Obr. 48 Znaková sada Windows-1250 [17]	55
Obr. 49 Varovná zpráva: Vstup nebyl zadán	57
Obr. 50 Varovná zpráva: Klíč nemá správnou délku	58
Obr. 51 Varovná zpráva: Délka klíče pro Vigeněrovu šifru je 1	59
Obr. 52 Varovná zpráva: Klíč není ve správném formátu	59
Obr. 53 Vzhled aplikace při časově náročnější operaci	60
Obr. 54 Šifrování textu	61
Obr. 55 Dešifrování textu	62
Obr. 56 Nastavení šifer pro vícenásobné šifrování textu	62
Obr. 57 Vícenásobné šifrování textu	63
Obr. 58 Výstup z vícenásobného šifrování textu	63
Obr. 59 Nastavení šifer pro vícenásobné dešifrování textu	64
Obr. 60 Vícenásobné dešifrování textu	64
Obr. 61 Vícenásobné dešifrování textu pomocí šifrovacího kódu	65
Obr. 62 Šifrování souboru	66
Obr. 63 Dešifrování souboru	67

Obr. 64 Nastavení šifer pro vícenásobné šifrování souboru	67
Obr. 65 Vícenásobné šifrování souboru	68
Obr. 66 Nastavení šifer pro vícenásobné dešifrování souboru	69
Obr. 67 Vícenásobné dešifrování souboru.....	69
Obr. 68 Vícenásobné dešifrování souboru pomocí šifrovacího kódu.....	70
Obr. 69 Hlavní okno mobilní aplikace CryptoSymm	73
Obr. 70 Hlavní okno mobilní aplikace Caesar Cipher.....	74

SEZNAM TABULEK

Tabulka 1 Čas a rychlost šifrování souborů pomocí šifry DES	60
---	----

SEZNAM PŘÍLOH

Příloha P I - CD na kterém jsou zdrojové soubory aplikace, spustitelná aplikace a tato práce ve formátu pdf