Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informačních technologií

Analýza vybraných kryptografických algoritmů

Bakalářská práce

Autor: Daniel Havrda
Studijní obor: Aplikovaná Informatika

Vedoucí práce: doc. Ing. Vladimír Soběslav, Ph.D.

Hradec Králové 2019

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 16.8.2019 Daniel Havrda

Poděkování:

Děkuji panu doc. Ing. Vladimíru Soběslavovi, Ph.D. za odborné vedení práce, poskytnutí konzultací a cenných rad.

Anotace

Tato bakalářská práce se zabývá rozborem kryptografických algoritmů a metod využívaných pro zabezpečení komunikace. V teoretické části práce jsou stručně popsány základní pojmy kryptografie. Dále jsou uvedeny některé příklady znázorňující historický vývoj. Největší pozornost je věnována detailnějšímu popisu moderní kryptografie s důrazem na požadavky a přehledný popis symetrických a asymetrických algoritmů. V poslední kapitole teoretické části jsou zmíněna další využití kryptografie, jako je ověření integrity dat a digitální podpisy. Účelem práce není nahradit učební materiály, ale spíše posloužit jako úvod do problematiky. Cílem praktické části je navrhnout a provést testy pro analýzu vybraných šifer s důrazem na výpočetní výkon a porovnat výsledky. V této časti je provedeno měření rychlosti symetrických šifer v závislosti na různých vstupních parametrech. Z hlediska bezpečnosti je u blokových šifer hodnocen rozptyl informací v závislosti na změně v klíči a otevřeném textu. Práce také znázorňuje rozdíl výkonu šifer v závislosti na použité knihovně.

Annotation

Title: Analysis of selected cryptographic algorithms

This bachelor thesis deals with the analysis of cryptographic algorithms and methods used for securing communication. In the theoretical part, the basic terms of cryptography are briefly described. In the next part some examples illustrating historical development are mentioned. The biggest attention is paid to more detailed description of modern cryptography with emphasis on requirements and a clear description of symmetric and asymmetric algorithms. The last chapter of the theoretical part mentions other uses of cryptography, such as data integrity verification and digital signatures. The purpose of this work is not to replace teaching materials, but rather to serve as an introduction to the topic. The aim of the practical part is to design and perform tests for the analysis of selected ciphers with an emphasis on computational performance and compare results. In that section, the throughput of symmetric ciphers based on changes of various parameters is measured. From the security point of view, block ciphers are evaluated for avalanche effect depending on the change in the key and plaintext. The work also shows the difference in cipher performance depending on the library used.

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Pod pojmem kryptografie rozumíme vědu, jejímž účelem je zamaskovat obsah zprávy a zajistit bezpečný přenos informací za pomoci matematických algoritmů. Spolu s kryptografií úzce souvisí pojem kryptoanalýza. Jedná se o přesný opak kryptografie, tedy snahu zjistit, jak bez znalosti klíče odvodit otevřený text ze zašifrované zprávy. Algoritmus pak není nic jiného než postup, kterým lze dosáhnout definovaného cíle.

Ačkoliv se počátky kryptografie datují do dávné historie, největší uplatnění nachází především v moderní době, kdy se objevila potřeba bezpečně předávat velké množství informací. Protože ve světě informatiky dochází k neustálým změnám, mění se také nároky na šifrování. Algoritmy musí být dostatečně výkonné, aby zvládly zpracovat velké množství dat, aniž by výrazně zpomalovaly systém. Stejně tak se mění požadavky na bezpečnost šifer. Díky rostoucímu výkonu počítačů nelze říci, že algoritmus, který je bezpečný dnes, bude také bezpečný za pět let.

Cíle této bakalářské práce lze rozdělit do dvou částí. Prvním cílem je poskytnutí uceleného přehledu, který může posloužit jako základní úvod do problematiky kryptografie. Z důvodu velkého množství informací v rámci tohoto tématu byla pozornost věnována převážně šifrám symetrické kryptografie. Druhý cíl se týká porovnání vybraných algoritmů s důrazem na výpočetní výkon. Kryptografické algoritmy mohou být použity v různých režimech na šifrování různých dat. Z této části by měl být patrný vliv některých vstupních parametrů na výkon šifer.

Pro zpracování této práce byly použity metody analýzy. Jednotlivé algoritmy byly implementovány v programu Python za pomocí volně dostupných knihoven a na základě navržených testů byly naměřeny potřebné údaje. Tyto hodnoty byly následně zpracovány a vyhodnoceny tak, aby z nich byl vliv zkoumaných parametrů na výkon dobře patrný. Dále byla využita srovnávací metoda v rámci vybraných knihoven, na jejímž základě byla předvedena efektivita konkrétní implementace pro dané úlohy.

V první kapitole této práce jsou popsány základní pojmy a historie kryptografie. Zmíněny jsou nejznámější historické algoritmy a popsán rozdíl mezi klasickou a moderní kryptografií.

Druhá kapitola je zaměřena na vývoj moderní kryptografie. První část této kapitoly se zabývá požadavky na vhodný šifrovací systém. V druhé části jsou pak popsány jednotlivé algoritmy. Pozornost je věnována hlavně symetrickým šifrám, které jsou dále rozděleny na blokové a proudové.

Třetí kapitola se věnuje dalším využitím kryptografických algoritmů. Jmenovitě se zabývá využitím šifrování pro ověření integrity dat a autentizaci entit.

Čtvrtá kapitola práce je zaměřena na testování vybraných algoritmů symetrického šifrování. V první části jsou představena různá kritéria pro hodnocení šifer a provedeny některé teoretické analýzy. Druhá část je zaměřena na praktické měření výkonu šifer. Důraz je kladen na analýzu vlivu různých nastavení na výpočetní výkon šifer.

V závěru této práce jsou shrnuty výsledky naměřených údajů a na jejich základě jsou uvedena doporučení.

1.1 Základní pojmy

Pro lepší pochopení této práce je potřeba vysvětlit nadcházející pojmy:

- **Otevřený text**: Zpráva v původní podobě, před zašifrováním. V matematických notacích označujeme M.
- **Šifrový text**: Zpráva po zašifrování. Standardně označujeme písmenem C.
- **Šifrování**: proces přeměny otevřeného textu na zašifrovaný text. Pro označení šifrovací funkce využíváme písmeno E a platí, že:

$$E(M) = C$$

- **Dešifrování**: proces přeměny zašifrovaného textu na původní zprávu. Pro dešifrovací funkci D, aplikovanou na šifrovaný text C platí:

$$D(C) = M$$

- **Šifra**: Systém, jehož významem je ukrytí smyslu zprávy a její následné dešifrování [3]. Zahrnuje veškerá pravidla a postupy. Pro zajištění správného znovuzískání šifrované zprávy musí platit následující rovnice:

$$D\big(E(M)\big) = M$$

- **Klíč**: Element určující výslednou podobu šifrované zprávy. Mění obecný postup na specifický.
- **Symetrická šifra**: Využívá stejný klíč pro šifrování i dešifrování zpráv. Pro zachování bezpečnosti zprávy musí tento klíč zůstat tajný.
- **Asymetrická šifra**: Využívá dva různé klíče. Veřejný klíč je používaný odesílatelem k zašifrování zprávy a je veřejně dostupný, soukromý klíč slouží příjemcem k dešifrování a pro bezpečnost zprávy musí zůstat utajený
- **Útok hrubou silou**: Snaha o prolomení šifry pomocí vyzkoušení všech možností klíče. Časová náročnost útoku roste s délkou klíče a klesá s rostoucím výpočetním výkonem.
- **Frekvenční analýza**: Metoda, která zjišťuje frekvenci použitých znaků a porovnává je s výskytem znaků v daném jazyce.
- **Útok se znalostí šifrového textu**: Útočník má k dispozici pouze několik zašifrovaných zpráv bez znalosti otevřeného textu nebo klíče. Jejich analýzou se pokouší zjistit podobu otevřeného textu nebo klíče [17].
- **Útok se znalostí otevřeného textu**: Útočník disponuje jak zašifrovaným, tak otevřeným textem. Jejich porovnáváním se pokouší odvodit klíč a algoritmus [17].
- **Útok s možností volby otevřeného textu**: Útočník si sám vybere podobu otevřeného textu a je schopen získat odpovídající zašifrovaný text.

1.2 Klasická kryptografie

Kryptografie, tedy nauka o utajování smyslu informace tak, aby ji byli schopni pochopit pouze lidé, pro které je zpráva určená, má svoje místo v lidské historii již více jak 2000 let [1]. Stejně jako nyní, i v dávných dobách byla schopnost zabezpečit informaci pro případ, že by se dostala do nepovolaných rukou velmi důležitá. Například vyzrazení klíčových plánů mohlo vést k prohře důležité bitvy či

únik citlivé informace mohl mít za následek skandál. Není tedy žádným překvapením, že se v průběhu historie objevovaly nejen stále dokonalejší šifry, ale také důmyslnější způsoby jak tyto šifry prolomit.

Mezi nejstarší známé šifry můžeme zařadit například šifru Atbash [2]. Jednalo se o jednoduchou substituční šifru, jejíž vznik datujeme přibližně do roku 600 př. n. l. Princip této šifry spočívá v určení vzdálenosti písmene od začátku abecedy a následným nahrazením písmenem se stejnou vzdáleností od konce [2]. První znak abecedy bude tedy nahrazen znakem posledním, druhý znak předposledním atd.

Další velmi známou šifrou je takzvaná Caesarova šifra. Jak již název napovídá, jedná se o šifru, která byla údajně popsána a používána Juliem Caesarem. Stejně jako v předchozím případě, se jedná o velmi jednoduchou substituční šifru, která využívá nahrazování znaků otevřeného textu znaky o předem určený počet míst dále v abecedě. Caesar sám údajně používal posun o tři místa, ale lze využít i posunu o jiný počet znaků [1].

Za zmínku také stojí Vigenèrova šifra. Tato šifra je pojmenována po francouzi Blaisovi de Vigenère a datuje se do konce 16. století [2]. Při použití této šifry je každý znak posunut o jiný počet znaků v abecedě v závislosti na klíči. Tato šifra využívá tzv. Vigenèrův čtverec. Jde o pomůcku, ve které jsou vypsány všechny možné varianty posunu a umožňuje jednodušší a rychlejší nalezení správného znaku.

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

Obrázek 1.1: Vigenèrův čtverec [20]

1.3 Moderní kryptografie

Kryptografie v dnešní podobě se začala rozvíjet v první polovině 20. století [2]. V roce 1917 si nechal americký inženýr Gilbert Vernam patentovat tzv. Vernamovu šifru. Jedná se o posun každého znaku otevřeného textu o zcela náhodný počet míst v abecedě. Jestliže je při zašifrování splněno několik podmínek, pak o této šifře platí, že je teoreticky neprolomitelná.

Za druhé světové války využívala německá armáda šifrovací přístroj Enigma. Toto revoluční zařízení, které sloužilo jako náhrada za zastaralé šifrovací systémy mělo až 10^{20} možných klíčů, tedy více než některé moderní algoritmy [1]. Prolomení tohoto systému bylo umožněno především kvůli nesprávnému použití a chybám při správě klíčů [1]. Úspěšné dešifrování takto zakódovaných zpráv pak znamenalo velikou výhodu pro Spojené národy.

V posledních letech se díky rychlému rozvoji počítačové techniky a nástupu internetu rozvíjí obor kryptografie a také jeho význam rychleji než kdy dříve. S rostoucím výpočetním výkonem se otvírají nové možnosti prolomení šifrování, a proto jsou neustále vyvíjeny silnější algoritmy. Tyto algoritmy bývají také otevřeně diskutovány, což často vede k odhalení jejich chyb.

2 Algoritmy moderní kryptografie

Dnes používané algoritmy se liší od dřívě používaných systému především využitím složitých matematických metod za účelem utajení obsahu zprávy. Moderní algoritmy zpravidla dělíme podle toho zda odesílatel a příjemce sdílejí tajemství nutné k dešifrování zprávy - klíč. Mluvíme tedy o šifrách symetrických a asymetrických.

2.1 Požadavky na šifrovací algoritmy

Na konci 19. století definoval a publikoval holandský profesor Auguste Kerckhoffs šest základních pravidel ve svých článcích o vojenské kryptografii. Tato pravidla položila základ požadavkům na účinnou moderní šifru a jsou následující [18]:

1) Šifra musí být prakticky nerozluštitelná i v případě, že její neprolomitelnost není matematicky podložená
2) Nesmí být požadavkem, aby princip šifry zůstal utajen a musí zůstat spolehlivý i v případě, že padne do rukou nepříteli.
3) Šifrovací klíč by měl být lehce sdělitelný a uchovatelný bez pomoci psaných poznámek. Tento klíč by mělo být možné kdykoliv změnit na přání komunikujících stran.
4) Šifra musí být použitelná pro telegrafickou komunikaci.
5) Šifrovací přístroj a dokumenty musí být přenosné, jejich použití a funkce nesmí vyžadovat součinnost více osob.
6) Vzhledem k účelu musí být systém jednoduchý na obsluhu, nesmí vystavovat uživatele zvýšenému psychickému napětí a celý proces šifrování či dešifrování musí být co nejjednodušší.

Ačkoliv některá z těchto pravidel již nejsou aktuální, druhé pravidlo, také známé jako Kerckhoffsův princip, je základem ochrany dat v dnešním světě. Tento princip byl později formulován také Američanem Claude E. Shannonem, který ho formuloval jako "Nepřítel zná systém" [18]. Oba tyto výroky nám říkají, že bezpečnost šifrovaných informací by měla zajistit existence utajeného klíče i v případě, že šifra samotná je dobře známá.

Shannon také definoval několik dalších pravidel, které by měly šifry splňovat. Například by šifrování nemělo mít za výsledek nárůst dat [3]. Myšlenkou tohoto pravidla je, že takováto šifra by poskytla kryptoanalytikům více dat aniž by obsahovala více užitečných informací. Další Shannonovo pravidlo říká, že chyby v procesu šifrování by se neměly nepřiměřeně šířit a způsobovat poškození informací dále ve zprávě [17]. Ostatní pravidla zmiňují, že složitost a rychlost implementace šifry by měla být úměrně složitá našim požadavkům na bezpečí dat a šifra by neměla být omezena podobou dat, na které bude aplikována [17].

2.1.1 Zmatení a rozptýlení informace

Dvě základní techniky, pomáhající zajistit bezpečnost šifry, se nazývají zmatení a rozptýlení. Obě tyto vlastnosti mají za účel zamezit prolomení šifry pomocí útoků zaměřených na redundantní informace [3].

Zmatení označuje postup, kterým se snažíme co nejvíce utajit vztah mezi otevřenou a šifrovanou zprávou [3]. Například Caesarova šifra je z hlediska požadavků na zmatení informace nevyhovující, protože stejný znak otevřeného textu je vždy šifrován na stejný znak šifrovaného textu.

Rozptýlení informace je statistická vlastnost, kdy každá redundance otevřeného textu je rovnoměrně rozptýlena v šifrovaném textu [1]. Aby šifra splňovala požadavek na bezpečný rozptyl informací, pak by mělo platit, že změna jednoho bitu v otevřeném textu vyústí ve změnu každého bitu zašifrovaného textu s 50% pravděpodobností.

Moderní šifry dosahují zmatení informací pomocí substitučních kroků a rozptýlení pomocí transpozice [3].

2.2 Symetrické šifrování

Symetrické algoritmy, někdy také nazývané konvenční, jsou algoritmy, které k šifrování i dešifrování zprávy využívají jediný klíč [17]. Oproti asymetrickým algoritmům bývají podstatně rychlejší, ovšem jejich nevýhoda spočívá v nutnosti

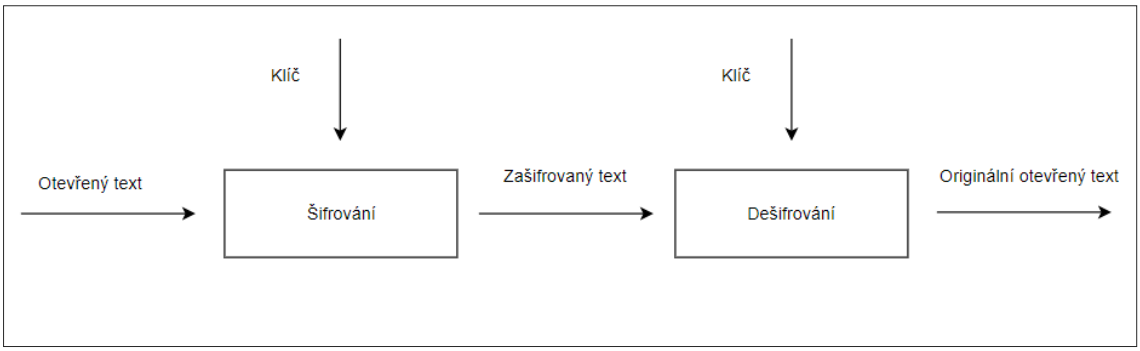
bezpečně předat klíč mezi odesílatelem a příjemcem. Pokud klíč nezůstane utajen, kdokoliv s jeho znalostí může předávanou zprávu dešifrovat.

Algoritmy symetrického šifrování můžeme dále rozdělit na proudové a blokové. Zatímco proudové šifry operují na zprávě po jednom bitu, blokové šifry jsou schopny zpracovávat zprávu po skupině bitů, které nazýváme bloky. Typická velikost takového bloku v dnešní době je 128 bitů neboli 16 bytů [4]. Takto velké bloky jsou dostatečně bezpečné a zároveň se s nimi dobře pracuje.

Pro symetrické šifrování s klíčem K lze tedy zapsat:

$$E_k(M) = C$$

$$D_k(C) = M$$



Obrázek 2.1 Symetrické šifrování [vlastní zpracování]

2.2.1 Blokové šifry

U blokových šifer je zpráva rozdělena do bloků o dané velikosti, na které je následně aplikován algoritmus. Platí, že velikost bloku otevřeného textu je totožná s velikostí bloku zašifrovaného textu. Tedy rozdělíme-li zprávu otevřeného textu na bloky o velikosti 128 bitů a aplikujeme šifrovací funkci, dostaneme zašifrované bloky také o velikosti 128 bitů. Tento proces je inverzní a po aplikaci dešifrovací funkce na 128 bitový blok zašifrovaného textu dostaneme opět 128 bitový blok původní zprávy. U blokových šifer se můžeme rozhodnout na použití takzvaného

operačního módu šifry, který určí, jakým způsobem je šifra využívána. Mezi hlavní módy patří:

- ECB: Režim kódové knihy (Electronic codebook mode) je nejjednodušším způsobem použití blokové šifry. Původní zpráva je rozdělena do bloků a každý blok je šifrován nezávisle na ostatních blocích [1]. Důležitou vlastností ECB je, že šifrování nemusí probíhat lineárně. Toho lze využít například, šifrujeme-li záznamy, na které se odkazujeme přímým přístupem. V takovém případě můžeme zpracovávat jednotlivé záznamy bez ohledu na ostatní [3]. Dále platí, že při použití stejného klíče budou shodné bloky originálního textu vždy zašifrovány do shodného bloku zašifrovaného textu. Tato skutečnost může mít negativní vliv na bezpečnost šifry. V reálných situacích se útržky zpráv často opakují, a pokud útočník získá přístup ke dvojicím zašifrovaného a otevřeného textu, může začít budovat slovník, který mu umožní tyto části zpráv dešifrovat. Pokud útočník získá přístup k dostatečnému počtu takovýchto zpráv, může se také pokusit například o útok za pomocí frekvenční analýzy.

- CBC: Řetězení šifrových bloků (Cipher Block Chaining) implementuje závislost každého bloku na předchozích blocích zprávy. Je toho docíleno tím, že každý blok otevřeného textu, kromě prvního, je před šifrováním XORován předchozím blokem šifrované zprávy. První blok obvykle bývá XORován inicializačním vektorem zpravidla označovaným *IV*. Ve výsledku je každý blok šifrovaného textu závislý na všech předchozích blocích otevřeného textu. Tato závislost podstatně ztěžuje frekvenční analýzu zprávy, ale také prakticky zabraňuje podvodné manipulaci se šifrovým textem [1]. Nevýhodou je, že kvůli závislosti na předchozích blocích nelze proces šifrování paralelizovat - tedy zpracovávat více bloků najednou. Dále se tu objevuje problém propagace chyb. Pokud se v šifrovaném textu změní hodnota jediného bitu, dojde k poškození celého bloku odpovídajícího otevřeného textu a jednoho bitu v následujícím bloku [3]. Synchronizační

chyby, tedy ztráta či přidání některých bitů znemožní správné dešifrování všech následujících bitů

- CFB: Režim šifrové zpětné vazby (Cipher FeedBack) vytváří podobně jako CBC závislost jednotlivých bloků zprávy. Tento režim převádí blokovou šifru na samo-synchronní proudovou šifru, čímž umožňuje šifrování dat po menších částech než celých blocích [3], například po jednotlivých znacích. Obdobně jako u CBC platí, že nelze šifrování paralelizovat.

2.2.1.1 DES

Za vznikem šifry DES - celým názvem Data Encryption Standard, také známé pod jmény Data Encryption Algorithm (DEA) a DEA-1 stojí americká firma IBM. Šifra vznikla v 70. letech a na konci desetiletí byla přijata jako federální šifrovací standard [3].

Již od své publikace byla šifra terčem kritiky. Mezi nejčastější obavy patřila účast NSA (National Security Agency) na vývoji algoritmu. Mnozí se obávali existence skrytých zadních vrátek a také kritizovali změnu původně plánované délky klíče 128 bitů na 56 bitů [3]. Navzdory všem obavám však nebyla existence zadních vrátek nikdy prokázána.

DES zpracovává otevřený text o blocích délky 64 bitů, které převádí na bloky zašifrovaného textu o stejné délce. Klíč používaný k zašifrování má také délku 64 bitů, ale každý osmý bit je využíván ke kontrole parity. Reálná délka klíče je tedy pouze 56 bitů.

Prvním krokem šifry je vstupní permutace IP. Tato permutace nemá praktický vliv na bezpečnost šifry a sloužila převážně k usnadnění práce s daty šifrovacímu hardwaru. Z důvodu obtížně softwarové implementace, jsou vstupní i výstupní transformace občas vynechávány [3]. Poté je blok zprávy rozdělen na levý a pravý, každý o délce 32 bitů. Následuje 16 kol identických operací nazývaných F-funkce. Samotná funkce se skládá ze čtyř kroků:

1.) V prvním kroku F-funkce jsou bity v klíči posunuty a je vybrán 48 bitový podklíč. Pravý blok dat je dále rozšířen na velikost 48 bitů pomocí expanzní permutace.

2.) Rozšířený pravý blok je zkombinován s podklíčem pomocí operace XOR.

3.) Po zkombinování dat s podklíčem je blok rozdělen na 8 bloků, každý o délce 6 bitů. Tyto bloky jsou následně použity ve vyhledávací tabulce zvané S-box za účelem transformace 6 vstupních bitů na 4 bity výstupní. Toho je dosaženo následovně: Z každého 6 bitového bloku vybereme první a poslední bit, které spolu reprezentují v desítkové soustavě čísla *<0..3>* a označíme je *m*, prostřední 4 bity, nabývajících hodnot *<0..15>* označíme *n* . Tyto hodnoty spolu odkazují na 4-bitové číslo v tabulce na souřadnicích S[*m,n*].

Tato část je extrémně důležitá pro bezpečnost šifry, neboť transformace je jediným nelineárním krokem šifry. Bez substituce by bylo tedy možné šifru řešit jako soustavu lineárních rovnic.

S-box 1:

14,	4,	13,	1,	2,	15,	11,	8,	3,	10,	6,	12,	5,	9,	0,	7,
0,	15,	7,	4,	14,	2,	13,	1,	10,	6,	12,	11,	9,	5,	3,	8,
4,	1,	14,	8,	13,	6,	2,	11,	15,	12,	9,	7,	3,	10,	5,	0,
15,	12,	8,	2,	4,	9,	1,	7,	5,	11,	3,	14,	10,	0,	6,	13,

Obrázek 2.2 První S-box [3]

4.) Na 32 bitový výstup S-boxů je aplikována P-box permutace, která mapuje vstupní pozici bitu na výstupní pozici. Po tomto kroku je výsledek permutace opět XORován, tentokrát s levou polovinou původního 64 bitového bloku, čímž dostaneme novou pravou polovinu bloku a původní pravá polovina se stane levou. Po tomto kroku pokračuje další runda algoritmu.

Po provedení poslední 16. rundy dojde k prohození levé poloviny bloku L16 s pravou polovinou R16 a jejich sloučení. Tím dostáváme výsledný blok R16L16 na který aplikujeme finální permutaci FP, která je inverzní k IP a také nemá vliv na účinnost algoritmu.

Dešifrování algoritmu probíhá identicky za pomoci stejné funkce, s jediným rozdílem, že klíče jsou zadávány v opačném pořadí.

Bezpečnost algoritmu DES

V průběhu let došlo k intenzivnímu zkoumání DES algoritmu a bylo poukázáno na kritické nedostatky v návrhu. Délka klíče 56 bitů je v dnešní době příliš krátká a umožňuje útoky hrubou silou. Dále byla prokázána existence slabých a poloslabých klíčů. Ačkoliv tvoří pouze malou množinu všech možných klíčů, je nutné je kontrolovat při implementaci algoritmu [17]. Algoritmus je také teoreticky náchylný vůči útokům se znalostí otevřeného textu a útokům s možností volby otevřeného textu. Například server crack.sh nabízí specializovaný systém schopný prolomení DES za pomocí útoku hrubou silou nejpozději do 26 hodin.

Z těchto důvodů není šifra DES nadále považována za bezpečnou.

2.2.1.2 3DES

Triple Data Encryption Standard je variantou šifry DES. Jedná se o trojnásobnou aplikaci DES za použití různých klíčů. Tím bylo dosaženo zvýšení odolnosti DES bez nutnosti přecházet na rozdílný šifrovací algoritmus. V dnešní době rozlišujeme dvě varianty 3DES podle počtu rozdílných klíčů:

1.) K1 != K2 != K3. Výsledkem je klíč o délce 168 bitů

2.) K1 !=K2, K3 = K1 s klíčem o délce 112 bitů.

Postup šifrování lze obecně zapsat pomocí rovnice:

$$C = E_{K3}(D_{K2}(E_{K1}(M)))$$

Tedy otevřený text M je zašifrován pomocí klíče K1, výsledek je dešifrován klíčem K2 a nakonec je znovu provedeno šifrování pomocí klíče K3.

Dešifrování pak probíhá inverzně:

$$M = D_{K1}(E_{K2}(D_{K3}(C)))$$

Nejdřív je šifrovaný text C dešifrován pomocí klíče K3, výsledek je šifrován pomocí klíče K2 a poté je provedeno finální dešifrování klíčem K1.

Velikou nevýhodou algoritmu 3DES je jeho malá rychlost. Dále je i přes zvětšení délky klíče považován za zastaralý a není doporučeno jej využívat [5].

2.2.1.3 Blowfish

Jedná se o blokovou šifru, která byla vytvořena americkým kryptologem Brucem Schneirem. Schneier se rozhodl vytvořit alternativu k DES, která nebude patentována a bude dostupná k použití pro všechny [3]. Cílem bylo vytvořit takový algoritmus, který bude rychlý, jednoduchý na implementaci a bude nabízet různé úrovně bezpečnosti [3]. Podobně jako DES zpracovává data o délce 64 bitů, ale k zašifrování využívá 32-448 bitový klíč.

Samotný proces šifrování probíhá ve dvou krocích. Prvním krokem je expanze klíče, kdy je klíč transformován do několika polí podklíčů o celkové délce 4168 bytů. Šifrovací funkce je tvořena 16 rundami. Každé kolo se skládá z klíčově závislé permutace, klíčově a datově závislé substituce a XOR operace [3]. Jediné operace, s kterými algoritmus funguje, jsou součet, XOR a odkazování se do vyhledávací tabulky, což má za následek vysokou rychlost algoritmu [3]. Na druhou stranu je šifra zpomalována tvorbou velkého množství subklíčů, které jsou nutné pro vytváření S-boxů.

Ačkoliv byly nalezeny některé slabé klíče a provedena kryptoanalýza ukazující na možné slabiny algoritmu se sníženým počtem rund [3], není v dnešní době známá žádná úspěšná kryptoanalýza ohrožující bezpečnost algoritmu s plným počtem rund.

2.2.1.4 AES

Autory tohoto algoritmu jsou J. Daemen a V. Rijmen. V roce 1997 byl přihlášen pod původním názvem Rijndael do soutěže o algoritmus AES(Advanced Encryption Standard), který měl řešit problémy dříve používané šifry DES. Převážně se jednalo o nadále nepostačující bezpečnost DES a malou rychlost varianty 3DES. V roce 2002 byl Rijndael schválen jako federální šifrovací standard [19]. Stejně jako v případě DES se jedná o symetrickou blokovou šifru, která ale využívá délku bloku

128 bitů, jenž lze rozšířit až na velikost 256 bitů a délku klíče 128, 192 nebo 256 bitů. Počet rund je 10 až 14 a záleží na velikosti klíče.

Na začátku algoritmus vygeneruje subklíče a provede fázi zvanou AddRoundKey, při které jsou byty stavu kombinovány pomocí operace XOR s vygenerovaným subklíčem. Každá runda algoritmu, s výjimkou poslední, probíhá ve 4 krocích.

1.) SubBytes - nelineární substituční proces, při kterém je každý byte nahrazen jiným, za pomoci vyhledávací tabulky AES S-box.
2.) ShiftRows - Transpozice prvků. Každý řádek stavu je posunut doleva o počet kroků odpovídající pořadí řádku v matici.
3.) MixColumn - Proces, při kterém je každý sloupec stavu interpretován jako vektor o velikosti 4 a nahrazen produktem násobení s polynomem.
4.) AddRoundKey

Závěrečná runda probíhá ve třech krocích
1.) SubBytes
2.) ShiftRows
3.) AddRoundKey

Bezpečnost AES

Přes velké množství pokusů o prolomení šifry, není v dnešní době znám žádný útok proveditelný v rozumném čase. Mezi nejznámější pokusy o prolomení AES patří útoky postranními kanály. Jedná se o typ útoku, při kterém se útočník nesnaží nalézt slabinu v matematickém popisu algoritmu, ale zaměřuje se na samotnou fyzickou implementaci. AES je také odolný vůči útokům hrubou silou, a to i za použití 128 bitového klíče.

Tento algoritmus lze tedy považovat v současné době za bezpečný.

2.2.2 Proudové šifry

Proudové šifry zpracovávají vstupní data jako proud bitů, na které jsou aplikovány změny. Jelikož v informatice bit může nabývat pouze dvou hodnot - 1 a 0, můžeme proudové šifry definovat pomocí dvou operací: ponechání a změny hodnoty bitu [1]. Protože se jedná o velmi jednoduché operace, je nutné zajistit, aby tyto změny nebyly předvídatelné pro osoby bez znalosti klíče. Toho je obvykle dosaženo vygenerováním dalšího proudu bitů za pomocí algoritmu a klíče, který je následně kombinován s proudem bitů otevřeného textu. Příkladem proudových šifer je například RC4, Salsa20, ale i Vernamova šifra.

2.2.2.1 Vernamova šifra

Tato jednoduchá proudová šifra využívá posun každého znaku otevřeného textu o náhodný počet míst v abecedě. K zašifrování využívá zcela náhodně vygenerovaný jednorázový klíč o stejné, nebo větší délce než je otevřený text [2]. Nezašifrovaná zpráva i klíč se převede na posloupnost bitů a je mezi nimi provedena operace XOR.

Bez využití počítačů by pak každé písmeno klíče představovalo posun v abecedě. Při zašifrování by byl každý znak otevřeného textu posunut právě o n míst, v závislosti na právě jednom znaku klíče.
V případě splnění tří následujících podmínek při volbě klíče je šifra považována za neprolomitelnou [3].
1.) Klíč má délku minimálně stejnou jako je nezašifrovaná zpráva.
2.) Klíč je vygenerován zcela náhodně. Není při něm využito například generátorů pseudonáhodných čísel.
3.) Klíč je využitý pouze jednou pro zašifrování jediné zprávy. Po zašifrování je klíč odesílatelem zničen a je vygenerován zcela nový náhodný klíč. Stejně tak je příjemcem klíč zničen po dešifrování.

Příklad:

Otevřený text: 01111010 01101001
Klíč: 10100010 00101101
XOR: ----------
Zašifrovaný text: 11011000 01000100

V případě dešifrování aplikujeme operaci XOR mezi zašifrovaným textem a klíčem. Tím dostaneme otevřený text.

Nebo bez počítačů, při využití anglické abecedy o 26 znacích:
Otevřený text: ZitraBude
Klíč: fckdjasdi

$$Z+f \bmod 26 = E$$
$$i+c \bmod 26 = k$$
$$t+k \bmod 26 = d$$

...
Po aplikaci všech posunů dostaneme zašifrovaný text ve tvaru
EkdujBmgm.

Pokud je zpráva zachycena a útočník nezná klíč, pak platí, že tuto zprávu nelze reálně prolomit hrubou silou. I v případě, že by někdo úspěšně vyzkoušel všechny klíče, odhalí každou myslitelnou zprávu a nebude moci určit, která z nich je ta pravá [2].
Jako ukázka může posloužit námi zašifrovaný text:

EkdujBmgm.

Pokusíme-li se zprávu dešifrovat pomocí klíče

rcbhfasdi

dostaneme otevřený text:

NicNebude

Klíč

jckdenswm

nám pak vrací otevřený text:

VitrFouka.

Hlavní nevýhody této šifry jsou spojeny právě s generováním a distribucí klíče [3]. Jelikož pro zachování absolutní bezpečnosti šifry nelze využít pseudonáhodných generátoru, už samotná tvorba klíče může být obtížná. Dalším problémem pak je délka klíče. Pro zprávy o délce několik tisíc znaků musíme vygenerovat stejně dlouhý klíč. Tento klíč je následně nutno bezpečně předat tak, aby k němu nikdo jiný neměl přístup. Pokud se při předávání ztratí jediný bit, pak celý zbytek zprávy nebude dávat smysl. Ačkoliv je kvůli kombinaci těchto faktorů praktické využití Vernamovy šifry minimální, byla tato šifra často používána například sovětskými agenty a údajně dokonce sloužila k zabezpečení horké linky mezi Kremlem a Pentagonem [3].

2.2.2.2 RC4

Tento zástupce symetrických, proudových šifer byl vynalezen v roce 1987 americkým kryptologem Ronem Rivestem, po němž je také pojmenován [3]. Šifra byla majetkem firmy RSA Data Security a nikdy nebyla oficiálně publikována. V polovině 90. let neznámý zdroj zveřejnil krátký zdrojový kód popisující algoritmus RC4 a ten se následně začal šířit po internetu. Jedná se o jednoduše fungující a snadno implementovatelnou šifru, která je také velmi rychlá i při softwarové implementaci. Rychlost šifrování RC4 je až 10x vyšší, jak u blokové šifry DES [3].

Algoritmus využívá dvě pole: pole S[], které obsahuje permutaci všech hodnot bytu 0,1,...,255, 256-bytové pole K[] a dvě počítadla *i,j*. V prvním kroku jsou inicializována pole S_0=0, S_1=1,S_2=2,...,S_{255}=255 a pole K pomocí N bytů klíče, který je v případě nutnosti opakován až dokud se nenaplní celé pole K_0,K_1,...,K_{255}.

Indexy *i* a *j* jsou inicializovaný na 0. V dalším kroku dojde k permutaci pole S pomocí

```
for i = 0 to 255
    j= (j+Si+Ki) mod 256;
    swap (S[i], S[j]);
```

Následně je generován keystream pseudonáhodných bytů K a prohodí se elementy v poli S.

```
i = (i + 1) mod 256
j = (j + S[i]) mod 256
swap (S[i],S[j])
t = (S[i]+S[j]) mod 256
K = S[t]
```

Tento byte je XORován s dalším bytem zprávy otevřeného textu a výsledkem je šifrovaný text. Dešifrování zprávy probíhá inverzně. Aby se zamezilo útokům s příbuznými klíči, je doporučeno zahodit prvních 256 bytů keystreamu. Ačkoliv byla dříve RC4 využívána například v internetových protokolech SSL, TLS či pro zabezpečení bezdrátových sítí pomocí WEP a WPA, není doporučeno tuto šifru z hlediska bezpečnosti nadále používat.

2.2.2.3 ChaCha20

Jedná se o 256 bitovou proudovou šifru, která vznikla modifikací algoritmu Salsa20. Existuje několik variant obou algoritmů, např. Salsa20/12, Salsa20/20 a jejich protějšky ChaCha12 a ChaCha20, kdy číslo za názvem algoritmu udává počet rund. Jelikož šifra využívá pouze jednoduché matematické operace, dosahuje velmi vysokých rychlostí, dokonce vyšších než u algoritmu AES [7]. Šifra využívá klíč o délce 256 bitů, ačkoliv varianta s klíčem o délce 128 bitů existuje také. Dalším vstupním parametrem je jednorázová hodnota o velikosti 64 bitů nonce

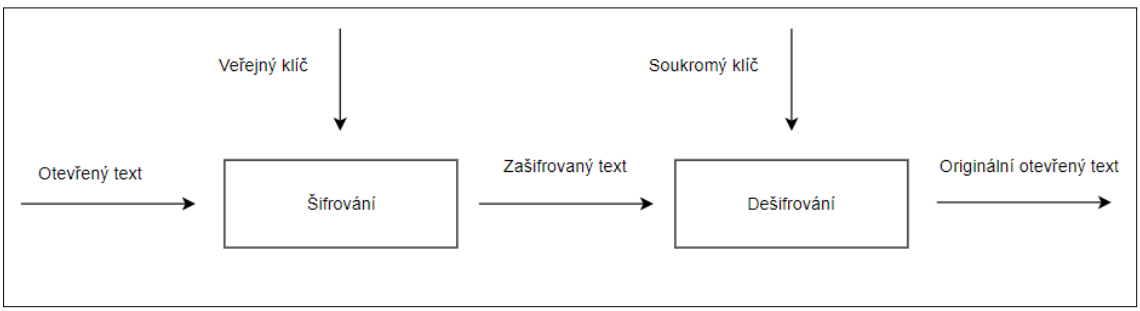
(number used only once). Vnitřní rundy fungují na stejném principu jako blokové šifry v CTR módu, k čemuž je využit čítač o délce 64bitů [8]. ChaCha20 v kombinaci s MAC funkcí Poly1305 byla vybrána společností google jako náhrada RC4 pro HTTPS komunikaci využívající TLS [9]. Tato iniciativa vyvolala stejnou změnu i v programu OpenSSH [10].

2.3 Asymetrické šifrování

Asymetrické šifrování, neboli šifrování s veřejným klíčem se od symetrických šifer liší v tom, že využívá dva různé klíče pro šifrování a dešifrování zprávy. První klíč nazýváme **veřejný klíč** a kdokoliv s jeho znalostí může šifrovat zprávy. Takto zašifrované zprávy už ovšem nemůže dešifrovat nikdo bez znalosti druhého klíče, který se označuje jako **soukromý klíč**. Dále platí, že soukromý klíč nemůže být vypočítán v rozumné době ze znalosti veřejného klíče [3].

Máme-li veřejný klíč Kv a soukromý klíč Ks pak platí:

$$E_{Kv}(M) = C$$
$$D_{Ks}(C) = M$$



Obrázek 2.3 Asymetrické šifrování [vlastní zpracování]

2.3.1 RSA

Tento algoritmus jenž byl pojmenován po svých třech autorech (Rivest, Shamir, Adleman) patří mezi nejrozšířenější asymetrické šifry. Velikou výhodou tohoto algoritmu je jeho zaručená bezpečnost při použití dostatečně velikého klíče a malá náročnost implementace.

Bezpečnost RSA je založena na obtížnosti matematického problému faktorizace, tedy rozkladu velikých čísel na prvočinitele. Zvolíme-li dvě dostatečně velká prvočísla p a q, můžeme s pomocí počítače získat výsledek jejich vzájemného vynásobení prakticky okamžitě. Takto získaný výsledek N je ovšem téměř nemožné rozložit v rozumném čase zpět na dvě prvočísla [1].

2.3.1.1 Postup algoritmu RSA

1.) V prvním kroku se zvolí dvě prvočísla p a q, která mají podobnou velikost, a spočítá se jejich součin $n = pq$. Dále je vypočtena Eulerova funkce

$$\varphi(n) = (p-1) \times (q-1)$$

V dalším kroku je zvoleno celé číslo e tak, aby platilo $1 < e < \varphi(n)$ a zároveň, že největší společný dělitel čísel e a $\varphi(n) = 1$. Nakonec je nalezeno číslo d, pro které platí, že $(de) = 1\ (mod\ \varphi(n))$. Po tomto kroku již čísla p a q nepotřebujeme. Veřejný klíč tvoří dvojice (e,n), soukromý klíč dvojice (d,n).

Šifrování je pak dáno vztahem

$$C = M^e\ (mod\ n)$$

a dešifrování zprávy probíhá podle vztahu

$$M = C^d\ (mod\ n).$$

2.3.1.2 Bezpečnost algoritmu RSA

Z důvodu technické náročnosti na provedení faktorizace, je algoritmus RSA považován za bezpečný, pokud při generování klíče použijeme dostatečně veliká prvočísla. Podle doporučení NIST tuto podmínku splňují klíče o délce alespoň 2048 bitů [5].

Dále je nutné zajistit zachování tajemství. V tomto případě tedy nesmíme prozradit prvočísla p a q. Pokud by kdokoliv přišel na jednoho z těchto činitelů, může snadno dopočítat soukromý klíč nutný pro dešifrování zprávy.

Největší hrozbou pro algoritmus RSA je případný objev rychlého způsobu faktorizace [2]. Jako možný příklad by mohl posloužit Shorův faktorizační algoritmus, který ovšem vyžaduje kvantový počítač. Pokud by se nám takovýto algoritmus podařilo spustit na opravdovém kvantovém počítači, pak by RSA šifra pravděpodobně přestala být bezpečnou.

3 Využití kryptografie

Jak již bylo zmíněno v úvodu, kryptografie je základem informační bezpečnosti. Existuje množství protokolů a systémů, které zajišťují bezpečný přenos dat právě pomocí kryptografie. Jako příklad můžeme uvést TLS - kryptografický protokol zabezpečující bezpečný přenos dat a šifrování pro protokoly aplikační vrstvy jako jsou HTTP, FTP a SMTP. Obdobným protokolem je SSH, který zajišťuje bezpečnou komunikaci například při vzdáleném přístupu mezi dvěma počítači.

V praxi se často používají hybridní systémy - například oba zmíněné protokoly. Základní ideou je, že se strany dohodnou na používaných algoritmech. Následně jedna strana vygeneruje privátní a veřejný klíč. Veřejný klíč poté pošle druhé straně. Ta zašifruje klíč k symetrickému algoritmu pomocí obdrženého veřejného klíče
a odešle ho zpět první straně, která ho může dešifrovat pomocí svého privátního klíče. Veškerá následná komunikace probíhá pomocí symetrického šifrování s vyměněným klíčem. Mezi dnes nejpoužívanější algoritmy symetrické kryptografie patří šifry popsané v předchozí kapitole, jmenovitě pak především AES, Blowfish/Twofish, 3DES a ChaCha20. K bezpečnému předání klíče je pak zpravidla využíváno asymetrické šifry RSA.

V moderním světě mají kryptografické algoritmy i jiné využití nežli pouhé utajení obsahu zprávy. Jako příklad může posloužit zajištění integrity dat, nebo autentizace. V případě počítačové bezpečnosti má termín autentizace dva možné významy. Prvním z nich je ověření původu dat, druhým pak ověření identity entit [1]. Mluvíme tedy o autentizaci jednostranné, nebo párové.

3.1 Zajištění integrity dat

V rámci bezpečnosti dat je nutné nejen utajit jejich obsah, ale také zajistit jejich integritu. Cílem je tedy zajistit, že přenášená data nebudou modifikována či jinak poškozena další neoprávněnou osobou [11]. Existuje mnoho možností jak tohoto cíle dosáhnout, jedním z nich je využití symetrické kryptografie. Při odesílání je

aplikována hashovací funkce, která přijímá jako vstup zprávu a tajný klíč [1]. Tato funkce převede zprávu na krátký bitový řetězec, který je poté přiložen k zašifrované zprávě. Příjemce pak aplikuje stejnou funkci se stejným klíčem a porovná výsledek s přiloženým řetězcem. Pokud se obě hodnoty navzájem neliší, příjemce má jistotu, že s daty nikdo nemanipuloval [1].

3.2 Autentizace

Autentizace, neboli ověření je nutnou součástí bezpečnosti. Existuje mnoho technik jak uživatelé mohou ověřit svou bezpečnost. Jako příklad může posloužit nějaké tajemství, například heslo nebo PIN u platební karty. Další možností je ověření pomocí něčeho vlastněného, například kartou nebo jiným autentizátorem [1]. V některých případech se také používá něco charakteristického pro uživatele jako jsou biometrické údaje [1]. Tyto možnosti lze samozřejmě různě kombinovat. K autentizaci uživatele lze využít kryptografie podobným způsobem jako při zajišťování integrity dat.

V případě jednostranné autentizace mluvíme například o přihlášení uživatele k počítači nebo vzdálenému serveru. Pokud musíme ukládat uživatelské údaje do databáze, není vhodné, aby se v databázi nacházeli v podobě plaintextu. V takovém případě je zvykem ukládat hesla v hashované podobě. Pokud se uživatel chce přihlásit, je aplikována hashovací funkce na zadané heslo a výsledek porovnán s údajem v databázi. Jelikož jsou hashovací funkce jednostranné, je zajištěna bezpečnost uživatelského hesla i v případě, že dojde k úniku dat.

Někdy však potřebujeme dokázat, že data přišla v nezměněné podobě od specifické osoby. V takovém případě je symetrická kryptografie nedostačující a je nutné použít digitální podpis [1]. Digitální podpis využívá především asymetrické kryptografie. Jelikož asymetrické algoritmy bývají výpočetně náročné, je na zprávu zpravidla použita hashovací funkce, jejíž produkt má pevnou délku. Tento hash je poté šifrován pomocí soukromého klíče, který jednoznačně identifikuje uživatele. Příjemce dešifruje tuto hodnotu pomocí dodaného veřejného klíče a dopočítá dle domluveného postupu novou hash hodnotu zprávy [12]. Pokud se obě hodnoty

shodují, byla ověřena nejen integrita dat, ale také odesílatel zprávy. Využití digitálních podpisů je různé. Můžeme například chtít ověřit, že smlouva pochází od našeho zaměstnavatele a nebyla nikým pozměněna. Vývojáři software mohou dodávat digitální podpis pro své aplikace, aby zaručili bezpečný původ spustitelných souborů [12].

4 Analýza vybraných algoritmů

Při analýze kryptografických algoritmů musíme uvážit kritéria, podle kterých budeme algoritmus zkoumat. Nejčastěji uvažujeme takové vlastnosti, které ovlivňují bezpečnost nebo výpočetní náročnost šifry.

Při analýze bezpečnosti šifrovacích algoritmů nás zajímá nejkratší známa doba, za kterou lze šifrování prolomit. V první řadě by matematický popis algoritmu neměl obsahovat žádné zjevné nedostatky. Analýza takového rozsahu je velmi náročná, a proto bývají jednotlivé algoritmy veřejně publikovány, aby bylo možné případné nedostatky najít a opravit. Dalším důležitým aspektem je délka klíče a v případě blokových šifer také délka bloku.

Krátký klíč činí šifru zranitelnou vůči útokům hrubou silou. Z matematického hlediska můžeme vyjádřit počet klíčů jako 2^n, kde n představuje délku klíče. Dále platí, že s každým přidaným bitem zdvojnásobujeme počet možných klíčů. Uvážíme-li v rámci ilustrace problému nejrychlejší superpočítač světa Summit, který dosáhl maximálního výkonu 200 petaflops [13], a zvolíme si, že k vyzkoušení jedné kombinace je potřeba výkonu 1000 flops, pak dostáváme hodnotu $\frac{200\times10^{15}}{1000} = 2 \times 10^{14}$. Tato hodnota nám dává množství kombinací, které je superpočítač schopný vyzkoušet za sekundu. V zájmu jednoduchosti budeme dále uvažovat, že všechny klíče lze vyzkoušet ve stejném čase. Pomocí těchto hodnot můžeme pak orientačně spočítat jak dlouho by trvalo prolomení šifry. V případě DES $\frac{2^{56}}{2\times10^{14}} = 360$. Superpočítač by byl tedy schopný prolomit šifru DES s klíčem o délce 56 bitů nejdéle za 6 minut. Uvážíme-li, že v mnoha případech nám stačí otestovat prvních 50% klíčů, dostaneme hodnotu 3 minuty. Pro 64 bitový klíč nám poté vychází nejdelší doba nutná k prolomení jako 25 hodin. AES s klíčem o délce 128 bitů by bylo prolomeno za $5{,}3 \times 10^{16}$ let. Pro porovnání nám může posloužit stáří vesmíru, které se odhaduje na přibližně 13,8 bilionu let, tedy $13{,}8 \times 10^9$. Z těchto údajů lze usoudit, že pro dnešní účely je 128 bitový klíč dostatečný.

U blokových šifer dále platí, že malá velikost bloku činí šifru zranitelnou vůči narozeninovým útokům [14]. Tento typ útoku vychází z teorie pravděpodobnosti,

konkrétně pak narozeninového problému. Narozeninový problém nám říká, že ve skupině 23 lidí je 50% pravděpodobnost, že dva lidé budou narozeni ve stejný den [3]. Ve skupině 50 lidí je pak šance již 97%. Tato skutečnost je dána tím, že nehledáme jeden konkrétní údaj - v tomto případě den, ale postačí nám kterýkoliv den v roce. Tuto skutečnost můžeme využít v kryptografii v rámci kolizního útoku. Budeme-li uvažovat blokovou šifru, která pracuje v režimu CBC, pak platí, že zpráva otevřeného textu M je rozdělena na bloky m_i. Šifrování pak můžeme zapsat jako $c_i = E(m_i \oplus c_{i-1})$. Hledáme takové dvě zprávy m_i a m_j, které nám vyprodukují stejný ciphertext, tedy $c_i = c_j$. Za předpokladu, že oba bloky byly zašifrovány pomocí stejného klíče, pak můžeme napsat, že platí:

$$m_i \oplus c_{i-1} = m_j \oplus c_{j-1}$$

a po přepsání

$$m_i \oplus m_j = c_{i-1} \oplus c_{j-1}.$$

Z tohoto vztahu můžeme odvodit XOR dvou plaintextů [14]. Situace je ještě horší, pokud známe podobu jednoho z plaintextů, například bloku m_j. V takovém případě můžeme také jednoduše dopočítat blok m_i ze vztahu:

$$m_i = c_{i-1} \oplus c_{j-1} \oplus m_j$$

Ačkoliv se může zdát, že podobná kolize je nepravděpodobná, pro šifry o délce bloku n platí, že většina módů šifer přestává být bezpečná při šifrování více jak $2^{n/2}$ bloků za použití stejného klíče [14]. Tato skutečnost se týká především šifer s bloky o délce 64bitů, pro které se slabina začne objevovat při zašifrování více jak 32GB stejným klíčem. Jedná se tedy například o algoritmy DES, 3DES a Blowfish.

Z hlediska výpočetní složitosti jsou nejčastěji uváděny dvě různá kritéria. Jedná se o složitost časovou a prostorovou [3]. Časová složitost nám udává rychlost algoritmu, prostorová složitost pak paměťové nároky. Z matematického hlediska se nejčastěji využívá "velké O notace". Pomocí této notace můžeme vyjádřit, jak roste náročnost algoritmu v závislosti na množství vstupních dat. Výhodou této

notace je, že není systémově závislá [3]. Jelikož symetrické šifry pracují na blocích pevné délky, jejich časová i prostorová náročnost je rovna O(1) pro jeden blok zprávy.

Dalším způsobem, jak znázornit výkonnost šifrovacího algoritmu, je pomocí praktického měření propustnosti. V takovém případě nejčastěji měříme rychlost šifrování a dešifrování dat o určité velikosti a výsledky uvádíme například v MB/sec, nebo v čase nutném na dokončení dané operace. Výsledky se také občas uvádějí v cyklech/byte. V takovém případě nám výsledek říká, kolik hodinových signálů CPU je potřeba vykonat na zpracování jednoho bytu dat. Tuto hodnotu lze vypočítat pomocí následující rovnice [15] $\frac{Množství\ CPU \times Využití\ CPU \times CPU\ frekvence}{Propustnost\ v\ byte/sec}$. Při takto měřených údajích je nutno pamatovat na to, že jsou závislá na použitém systému a vstupních datech.

Při výběru nového šifrovacího standardu AES byly algoritmy analyzovány a hodnoceny na základě následujících kritérií [16]: Bezpečnost algoritmu, výpočetní náročnost, paměťová náročnost, vhodnost pro softwarovou a hardwarovou implementaci, jednoduchost, flexibilita a licenční podmínky.

V rámci další podkapitoly byla provedena analýza symetrických algoritmů popsaných v kapitole 2.2, s cílem znázornit rozdíly ve výpočetní náročnosti jednotlivých šifer v závislosti na použitých parametrech.

4.1 Způsob testování

Všechny algoritmy byly otestovány v jazyce Python 3.7.3 na počítačové sestavě s procesorem Intel Core i5-4570 s frekvencí 3.20 GHz a operačním systémem Windows 10. Jelikož Python poskytuje mnoho nástrojů pro účely kryptografie a v komunitě neexistuje jednoznačný názor, kterou knihovnu používat, byly vybrány a porovnány dvě nejpoužívanější knihovny vhodné pro účely testování. První vybraná knihovna je *PyCryptoDome,* nástupce dříve nejpoužívanější kryptografické knihovny *PyCrypto*. Narozdíl od ostatních nástrojů spravovaných PyCA (Python Cryptographic Authority) tato *PyCryptoDome* využívá vlastní

C knihovnu. Druhou vybranou knihovnou je *Cryptography*, jenž využívá C knihovnu OpenSSL.

Všechna měření byla zpravidla provedena třikrát a ze získaných údajů byl vypočítán aritmetický průměr. Naměřené údaje jsou uváděny v MB/s. Hodnota koeficient odpovídá $\frac{rychlost\ Cryptography}{rychlost\ PyCryptoDome}$ a platí tedy, že knihovna *Cryptography* je rychlejší pro hodnoty větší než 1 a pomalejší pro hodnoty menší než 1. V průběhu měření byly vypnuty všechny nesystémové aplikace. Testovací scénáře byly vytvořeny tak, aby odpovídaly různým situacím při skutečném použití. Jedná se o následující:

1. Závislost rychlosti šifrování blokových šifer na použitém módu. Testovány byly módy ECB, CBC a CFB na zprávě o velikosti 30MB. Takto velká zpráva byla vybrána proto, aby byl zajištěn již stabilizovaný bit rate.

2. Rychlost šifrování a dešifrování algoritmu AES v závislosti na velikosti klíče. Test byl proveden v módu CBC na zprávách různé délky. Cílem tohoto testu je ověřit vliv počtu šifrovacích rund při různých velikostech klíče.

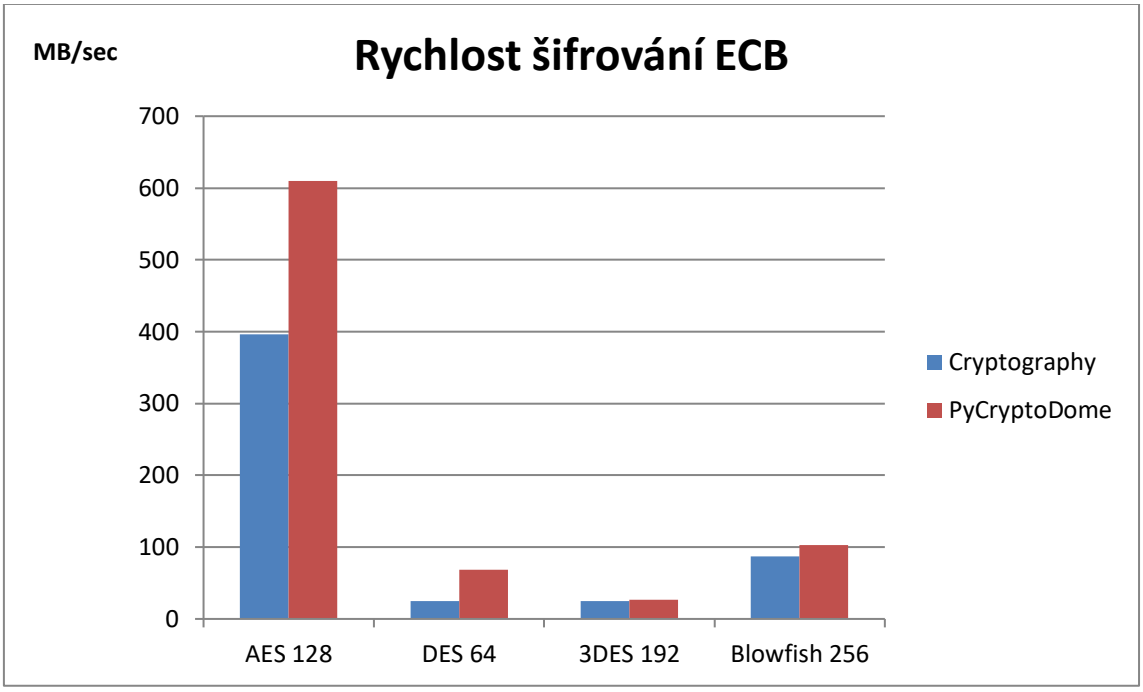
3. Rychlost šifrování a dešifrování vybraných symetrických algoritmů v závislosti na velikosti zprávy. Jelikož nelze režim ECB doporučit pro běžné používání, probíhalo měření v režimu CBC pro blokové šifry.

4. Test rozptýlení informací u blokových šifer. V první části tohoto testu jsou vygenerovány dvě zprávy, které se navzájem liší o jeden bit. Obě tyto zprávy jsou následně zašifrovány pomocí stejného klíče a inicializačního vektoru. Výsledný zašifrovaný zprávy by se měly navzájem lišit alespoň v 50% bitů. Test je také proveden v režimech ECB a CBC při změně jednoho bitu klíče a stejné zprávě.

4.2 Výsledky testování rychlosti šifer při různých režimech

Tabulka 4.1 Rychlost šifrování blokových šifer v módu ECB [vlastní zpracování]

Název algoritmu	Cryptography (MB/sec)	PyCryptoDome (MB/sec)	Koeficient
AES 128	395.9	609.5	0.65
DES 64	25.1	68.1	0.37
3DES 192	24.8	26.5	0.94
Blowfish 256	87.1	102.4	0,85



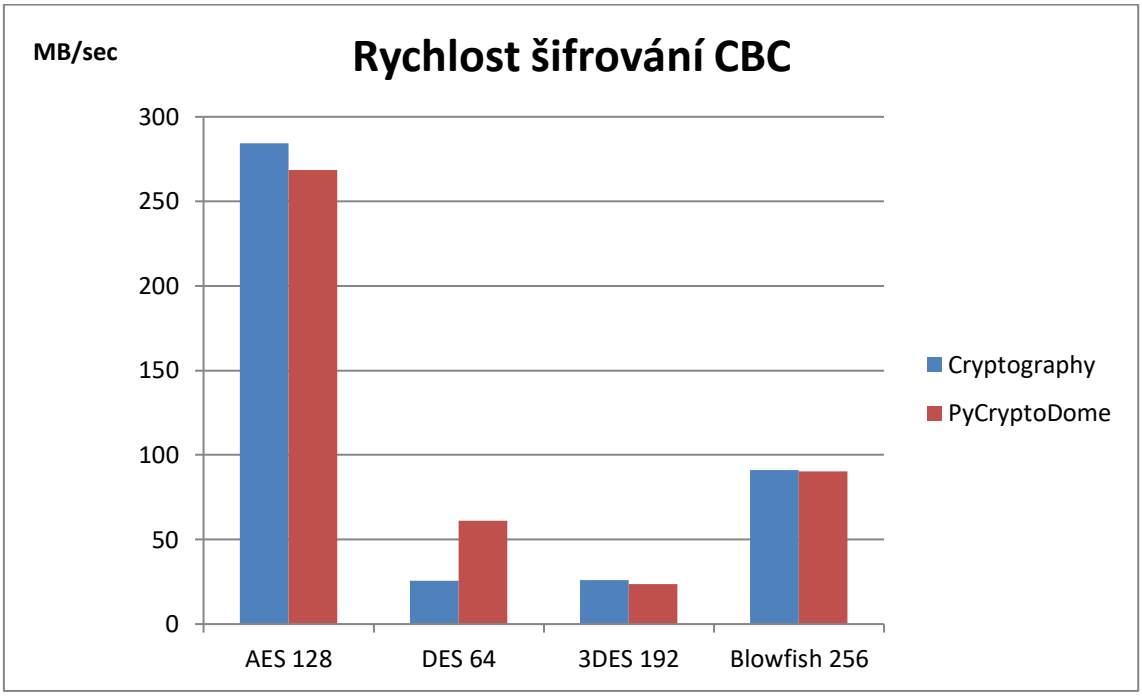
Obrázek 4.1 Graf rychlosti šifrování zprávy velikosti 30MB v módu ECB [vlastní zpracování]

Z naměřených výsledků je vidět znatelný rozdíl ve výkonu blokových šifer obou knihoven. Pro všechny testované algoritmy platí, že PyCryptoDome nabízí rychlejší implementaci. Největší rozdíl je vidět u algoritmu DES s velikostí klíče 64, kdy je knihovna PyCryptoDome téměř 3x rychlejší nežli Cryptography. Takto znatelný rozdíl je dán tím, že knihovna Cryptography implementuje všechny varianty DES jako 3DES, v tomto případě tedy DES se třemi stejnými klíči. Celkově se jako

nejrychlejší algoritmus projevil AES, který je 4.5x rychlejší než Blowfish v případě Cryptography a 6x rychlejší v knihovně PyCryptoDome. Naopak nejpomalejší jsou varianty 3DES, které jsou oproti AES až 20x pomalejší.

Tabulka 4.2 Rychlost šifrování blokových šifer v módu CBC [vlastní zpracování]

Název algoritmu	Cryptography (MB/sec)	PyCryptoDome (MB/sec)	Koeficient
AES 128	284.4	268.5	1.06
DES 64	25.8	61.2	0.42
3DES 192	25.9	23.8	1.08
Blowfish 256	91.2	90.4	1.01

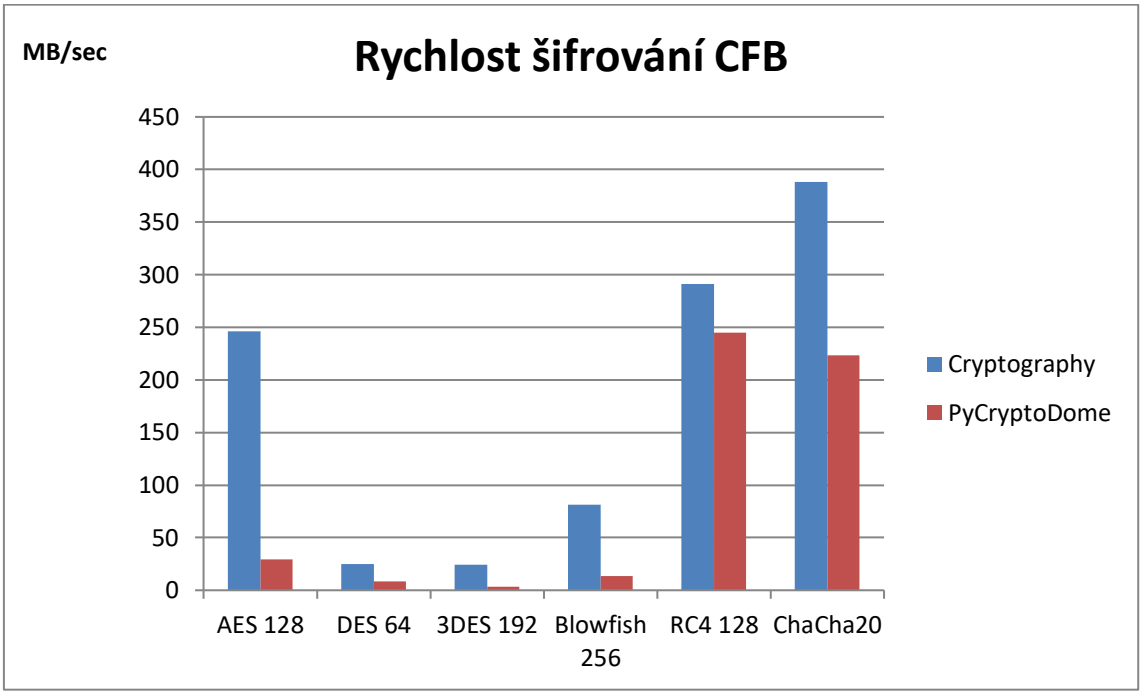


Obrázek 4.2 Graf rychlosti šifrování zprávy velikosti 30MB v módu CBC [vlastní zpracování]

Při použití módu CBC jsou výsledky s výjimkou DES mnohem vyrovnanější s nepatrnou výhodou pro knihovnu Cryptography. Oproti režimu ECB lze vidět znatelný pokles v rychlosti pro algoritmus AES, který je pomalejší přibližně o 28% v případě Cryptography a o 56% v knihovně PyCryptoDome. U ostatních algoritmů je rozdíl maximálně 10%.

Tabulka 4.3 Rychlost šifrování proudových šifer a blokových šifer v módu CFB [vlastní zpracování]

Název algoritmu	Cryptography (MB/sec)	PyCryptoDome (MB/sec)	Koeficient
AES 128	246.2	29.3	8.4
DES 64	25.2	8.7	2.9
3DES 192	24.3	3.1	7.84
Blowfish 256	81.4	13.6	5.99
RC4 128	290.9	244.6	1.19
ChaCha20	387.9	223.3	1.74



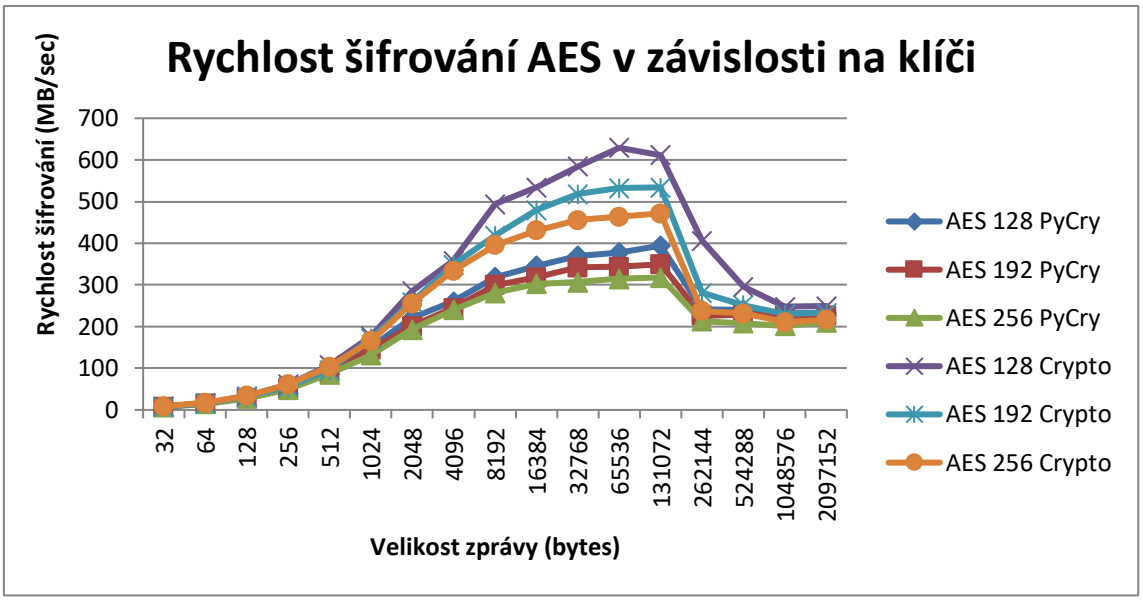
Obrázek 4.3 Graf rychlosti šifrování zprávy velikosti 30MB pro proudové šifry a blokové šifry v módu CFB [vlastní zpracování]

Jelikož se blokové šifry v módu CFB chovají jako proudové šifry, byly zahrnuty pro tento test i zástupci proudových šifer RC4 a ChaCha20.

Již na první pohled lze z grafu vyčíst obrovský rozdíl mezi oběma knihovnami v případě blokových šifer. Algoritmy Cryptography jsou jednoznačně rychlejší a například algoritmus AES se rychlostně blíží proudové šifře RC4.

Dle očekávání se zástupci proudových šifer ukázaly jako rychlejší pro šifrování proudu dat. Jako nejrychlejší algoritmus pro knihovnu Cryptography se ukázal ChaCha20, který byl také celkově nejrychlejším. Pro PyCryptoDome pak zvítězila šifra RC4, která se ukázala jako nepatrně rychlejší než vlastní implementace ChaCha20. Z blokových šifer je opět nejrychlejší algoritmus AES následovaný šifrou Blowfish pro obě testované knihovny.

4.3 Výsledky testování AES v závislosti na délce klíče

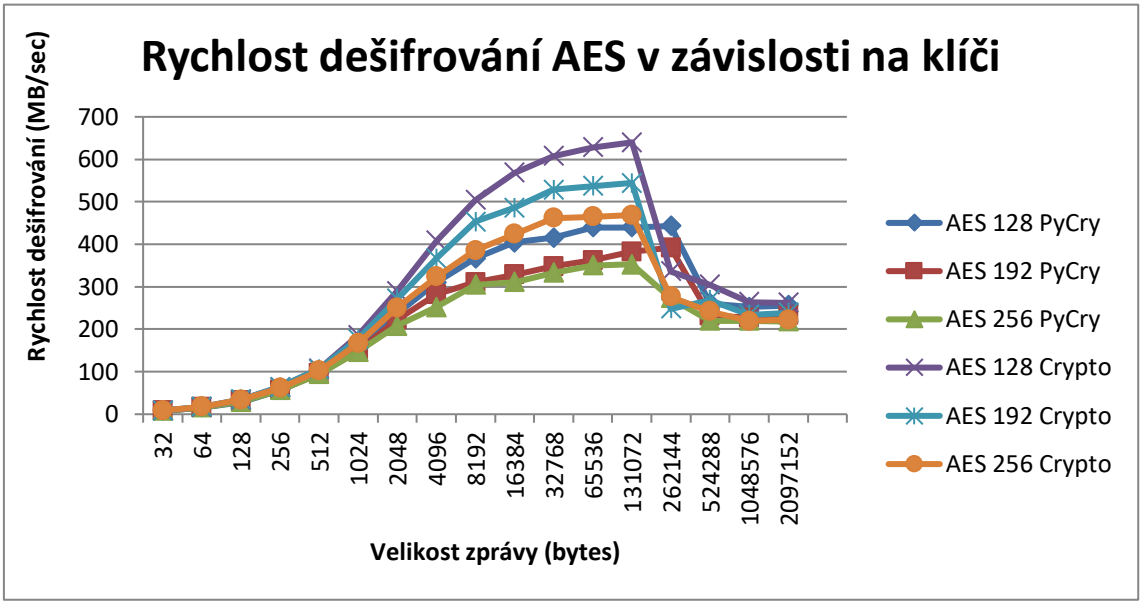


Obrázek 4.4 Graf rychlosti šifrování algoritmu AES v závislosti na velikosti klíče [vlastní zpracování]

Z naměřených hodnot lze vidět, že všechny varianty následují stejný trend. Pro zprávy nad 256 bytů se začínají objevovat rozdíly v závislosti na délce klíče. Největší rozdíly se objevují pro zprávy kolem velikosti 131072 bytů pro knihovnu PyCryptoDome, kdy je 128 bitová varianta o 11% rychlejší nežli 192 bitová a o 19% rychlejší nežli 256 bitová. U knihovny Cryptography jsou největší rozdíly ve zprávách o velikosti 262144 bytů. Naměřené údaje ukazují, že 128 bitový klíč je o 31% rychlejší oproti 192 bitovému a o 42% rychlejší oproti 256 bitovému. V těchto hodnotách také algoritmy dosahují naměřeného rychlostního maxima, po kterém následuje rychlý propad a stabilizace bit rate. Podstatně menší, avšak obdobný trend lze sledovat také v jiných pracích, například v měření provedeném

serverem panthema.net[6] u knihovny OpenSSL. Po stabilizaci je 128 bitová varianta rychlejší přibližně o 2% a 10% vůči větším klíčům v případě PyCryptoDome a o 7% a 15% pro Cryptography.

Z grafu lze také vidět znatelný rozdíl v rychlosti obou knihoven v intervalu 512- 1048576 bytů. Nejvyšší naměřena hodnota pro Cryptography je 629 MB/sec, zatímco pro PyCryptoDome je to 394 MB/sec.



Obrázek 4.5 Graf rychlosti dešifrování algoritmu AES v závislosti na velikosti klíče [vlastní zpracování]

Dešifrování následuje stejný trend jako šifrování. Knihovna Cryptography je rychlejší s výjimkou zpráv kolem velikosti 262144 bytů, kdy 128 a 192 bitovým variantám PyCryptoDome trvá o něco delší dobu, než stabilizují svoji rychlost. Po stabilizaci je AES128 v obou případech rychlejší přibližně o 10% vůči AES192 a o 15% vůči AES256.

4.4 Výsledky testování symetrických algoritmů v závislosti na velikosti zprávy

Tabulka 4.4 Rychlost šifrování knihovny Cryptography v závislosti na velikosti zprávy [vlastní zpracování]

AES128 (MB/sec)	DES64 (MB/sec)	3DES192 (MB/sec)	Blowfish256 (MB/sec)	RC4-128 (MB/sec)	ChaCha20 (MB/sec)	**Velikost zprávy (bytes)**
8.82	6.55	6.45	7.86	9	8.61	**32**
15.85	10.03	10.11	14.44	17.4	17.46	**64**
33.12	14.49	14.09	25.84	34.15	34.13	**128**
60.78	18.09	17.57	39.64	65.21	66.29	**256**
107.35	20.62	21.16	55.21	108.3	118.92	**512**
175.93	22.56	22.6	72.81	188.12	222.23	**1024**
285.19	24	24.01	81.38	314.26	396.36	**2048**
358.14	25.08	24.63	94.25	440.85	657.99	**4096**
493.69	24.67	24.75	97.06	549.17	1006.22	**8192**
533.82	25	24.6	100.93	615.15	1239.78	**16384**
584.45	25.24	24.45	103.39	667.31	1472.27	**32768**
629.16	25.27	25.08	100.58	695.22	1596.22	**65536**
611.62	24.94	25	100.3	714.84	1728.13	**131072**
406.35	23.42	23.42	83.29	300	1112.71	**262144**
295.39	23.42	23.42	81.02	312	501.25	**524288**
247.75	23.49	24.16	82.06	278.98	358.78	**1048576**
248.89	23.28	23.71	81.71	273.07	354.44	**2097152**

Tabulka 4.5 Rychlost dešifrování knihovny Cryptography v závislosti na velikosti zprávy [vlastní zpracování]

AES128 (MB/sec)	DES64 (MB/sec)	3DES192 (MB/sec)	Blowfish256 (MB/sec)	RC4-128 (MB/sec)	ChaCha20 (MB/sec)	**Velikost zprávy (bytes)**
8.72	6.47	6.22	7.88	8.95	8.69	**32**
17.02	10.14	9.64	14.52	17.49	17.87	**64**
33.86	14.57	14.23	25.36	34.43	35.07	**128**
63.56	17.45	17.93	38.35	65.25	68.08	**256**
106.66	20.31	21.4	56.09	108.31	120.78	**512**
185.28	22.3	21.64	72.4	194.92	231.37	**1024**
290.21	24.24	23.96	84.59	303.31	412.74	**2048**
408.21	24.37	23.95	90.63	428.14	686.48	**4096**
504.32	24.86	24.85	97.24	541.45	1017.2	**8192**
568.79	25.07	24.94	96.4	614.69	1255.19	**16384**
608	24.74	25.6	101.15	655.53	1532.42	**32768**
628.39	25.46	25.08	99.49	687.37	1653.77	**65536**
640	23.71	25.15	99.16	681.74	1727.49	**131072**
336.28	23.89	22.86	83.7	320	1108.77	**262144**
304.77	23.42	23.71	83.12	295.39	476.34	**524288**
264.13	23.93	24.16	83.48	264.13	365.43	**1048576**
262.57	24.16	24.16	81.71	262.57	357.45	**2097152**

Mezi šifrováním a dešifrováním nebyly naměřeny žádné znatelné rozdíly v rychlosti. Jednotlivé algoritmy lze rozdělit na dvě skupiny. Blokové šifry DES a Blowfish mají relativně stabilní bit rate pro zprávy nad 2048 bytů. Proudové šifry a AES následují trend známý z minulého testu. Jejich rychlost rapidně roste až do velikosti zprávy 131072 bytů s následujícím rychlým poklesem. Bit rate této skupiny je stabilní pro zprávy o velikosti nad 1MB.

Jako nejrychlejší se ukázal algoritmus ChaCha20, který je rychlostně srovnatelný s algoritmy AES a RC4 pro menší zprávy, ale s rostoucí velikostí šifrovaných dat se zvyšuje i rozdíl v rychlosti. Naopak nejpomalejší je algoritmus DES, který po stabilizaci bit rate dosahuje více jak 10x menší rychlost oproti AES a proudovým šifrám.

Tabulka 4.6 Rychlost šifrování knihovny PyCryptoDome v závislosti na velikosti zprávy [vlastní zpracování]

AES128 (MB/sec)	DES64 (MB/sec)	3DES192 (MB/sec)	Blowfish256 (MB/sec)	RC4-128 (MB/sec)	ChaCha20 (MB/sec)	**Velikost zprávy (bytes)**
8.2	7.02	5.99	7.35	13.72	9.49	**32**
15.52	12.28	9.22	13.88	24.89	16.51	**64**
29.78	20.66	13.07	23.22	42.02	34.34	**128**
56.62	30.23	16.28	37.46	72.51	53.7	**256**
95.83	40.26	18.85	55	118.44	98.61	**512**
150.75	49.12	21.55	67.02	196.8	201.97	**1024**
221.31	54.93	21.76	79.54	254.35	217	**2048**
260.23	58.19	21.75	84.92	277.34	238	**4096**
318.85	60.17	22.11	90.41	353.34	282	**8192**
345.95	60	22.2	90.25	335	327	**16384**
370.31	60	22.69	95.35	325.15	334.29	**32768**
377.78	60.85	22.72	93.34	368	347.2	**65536**
394.37	62.75	21.82	96.97	373.34	306.67	**131072**
241.32	60.55	22.86	91.43	336	219.43	**262144**
240	57.03	21.58	82.29	254.17	224	**524288**
231.33	55.66	21.89	83.12	219.43	234.67	**1048576**
232.73	56.27	21.7	82.59	256	219.43	**2097152**

Tabulka 4.7 Rychlost dešifrování knihovny PyCryptoDome v závislosti na velikosti zprávy [vlastní zpracování]

AES128 (MB/sec)	DES64 (MB/sec)	3DES192 (MB/sec)	Blowfish256 (MB/sec)	RC4-128 (MB/sec)	ChaCha20 (MB/sec)	**Velikost zprávy (bytes)**
8.08	7.09	5.91	7.34	10.33	7.61	**32**
16.06	12.5	9	13.73	19.73	15.52	**64**
28.61	20.56	13.24	21.3	37	27.65	**128**
56.41	29.27	16.12	39.14	66.08	50.27	**256**
96.61	40.49	18.29	54.57	106.31	84.7	**512**
160.51	48.9	20	69.82	163.37	138.56	**1024**
236.7	56.05	21.86	81.92	222.58	181.47	**2048**
310	59.58	21.67	90.88	270.55	222.85	**4096**
366.61	61.15	22.37	93.64	299.5	247.62	**8192**
404.18	62.63	22.82	95.32	320.89	275.95	**16384**
415.63	62.95	22.11	97.05	332.73	285.95	**32768**
440	61.64	22.36	97	337.05	291.9	**65536**
439.44	61.54	22.86	98.47	341.18	297.94	**131072**
443.08	56.99	21.58	85.64	336.42	219.71	**262144**
260.75	57.44	22.59	83.81	230	207.06	**524288**
251.81	56.89	21.7	84.22	227.1	207.57	**1048576**
256	58.19	22.27	85.34	227.56	207.57	**2097152**

Algoritmy knihovny PyCryptoDome lze opět rozdělit do dvou skupin. Zatímco u blokových šifer DES a Blowfish je průběh stabilní a neobjevuje se znatelný rozdíl mezi rychlostí šifrování a dešifrování, situace se trochu změnila v případě druhé skupiny. Ačkoliv algoritmy AES, RC4 a ChaCha20 následují stejný trend rapidního poklesu rychlosti po dosažení maxima, následovaný stabilizací bit rate, nejedná se již o tak veliký rozdíl jako v případě knihovny Cryptography. Dále lze u těchto algoritmů pozorovat větší rozdíly v rychlosti šifrování a dešifrování. Algoritmus AES dosahuje maximální rychlosti dešifrování 443 MB/sec, zatímco maximální

naměřená rychlost šifrování je pouze 394 MB/sec. Po stabilizaci bit rate je průměrná rychlost dešifrování přibližně o 10% vyšší. Naopak u proudových šifer je situace opačná, kdy šifrování větších dat probíhá o 10-15% rychleji nežli dešifrování

Tabulka 4.8 Koeficient rozdílu rychlosti šifrování použitých knihoven [vlastní zpracování]

AES128	DES64	3DES192	Blowfish256	RC4-128	ChaCha20	**Velikost zprávy**
1.08	0.94	1.08	1.07	0.66	0.91	**32**
1.03	0.82	1.1	1.05	0.7	1.06	**64**
1.12	0.71	1.08	1.12	0.82	1	**128**
1.08	0.6	1.08	1.06	0.9	1.24	**256**
1.13	0.52	1.13	1.01	0.92	1.21	**512**
1.17	0.46	1.05	1.09	0.96	1.11	**1024**
1.29	0.44	1.11	1.03	1.24	1.83	**2048**
1.38	0.44	1.14	1.11	1.59	2.77	**4096**
1.55	0.42	1.12	1.08	1.56	3.57	**8192**
1.55	0.42	1.11	1.12	1.84	3.8	**16384**
1.58	0.43	1.08	1.09	2.06	4.41	**32768**
1.67	0.42	1.11	1.08	1.89	4.6	**65536**
1.56	0.4	1.15	1.04	1.92	5.64	**131072**
1.69	0.39	1.03	0.92	0.9	5.08	**262144**
1.24	0.42	1.09	0.99	1.23	2.24	**524288**
1.08	0.43	1.11	0.99	1.28	1.53	**1048576**
1.07	0.42	1.1	0.99	1.07	1.62	**2097152**

Tabulka 4.9 Koeficient rozdílu rychlosti dešifrování použitých knihoven [vlastní zpracování]

AES128	DES64	3DES192	Blowfish256	RC4-128	ChaCha20	**Velikost zprávy**
1.08	0.92	1.06	1.08	0.87	1.15	**32**
1.06	0.82	1.08	1.06	0.89	1.16	**64**
1.19	0.71	1.08	1.2	0.94	1.27	**128**
1.13	0.6	1.12	0.98	0.99	1.36	**256**
1.11	0.51	1.18	1.03	1.02	1.43	**512**
1.16	0.46	1.09	1.04	1.2	1.67	**1024**
1.23	0.44	1.1	1.04	1.37	2.28	**2048**
1.32	0.41	1.11	1	1.59	3.09	**4096**
1.38	0.41	1.12	1.04	1.81	4.11	**8192**
1.41	0.41	1.1	1.02	1.92	4.55	**16384**
1.47	0.4	1.16	1.05	1.98	5.36	**32768**
1.43	0.42	1.13	1.03	2.04	5.67	**65536**
1.46	0.39	1.11	1.01	2	5.8	**131072**
0.76	0.42	1.06	0.98	0.96	5.05	**262144**
1.17	0.41	1.05	1	1.29	2.31	**524288**
1.05	0.43	1.12	1	1.17	1.77	**1048576**
1.03	0.42	1.09	0.96	1.16	1.73	**2097152**

S výjimkou 64 bitové varianty DES a RC4 pro zprávy menší nežli 512 bytů si téměř ve všech testovaných případech vede lépe knihovna Cryptography.

Největší rozdíly lze vidět v případě šifry ChaCha20, kdy implementace Cryptography dosahovala až 6x vyšších hodnot a i po stabilizaci bit rate zůstává přibližně o 50%-70% rychlejší. Znatelné rozdíly lze také pozorovat u šifer AES a RC4 pro zprávy o velikosti 2048-524288 bytů.

4.5 Výsledky testování rozptýlení informací

V první části tohoto testu byly vygenerovány dvě zprávy, které se navzájem liší o jediný bit. Obě tyto zprávy byly zašifrovány pomocí stejného klíče a inicializačního vektoru tak, aby výsledek nebyl ovlivněn žádnou jinou změnou. Měřena byla změna mezi zašifrovanými zprávami pomocí vzorce $\frac{Počet\ změněných\ bitů}{Celkový\ počet\ bitů}$. Aby byl test úspěšný, mělo by dojít ke změně alespoň poloviny bitů a změna by měla být distribuována skrz celou zprávu.

Jako zprávy byly zloveny následující:

Zpráva 1: abcdefghijklmnop

Zpráva 2: cbcdefghijklmnop

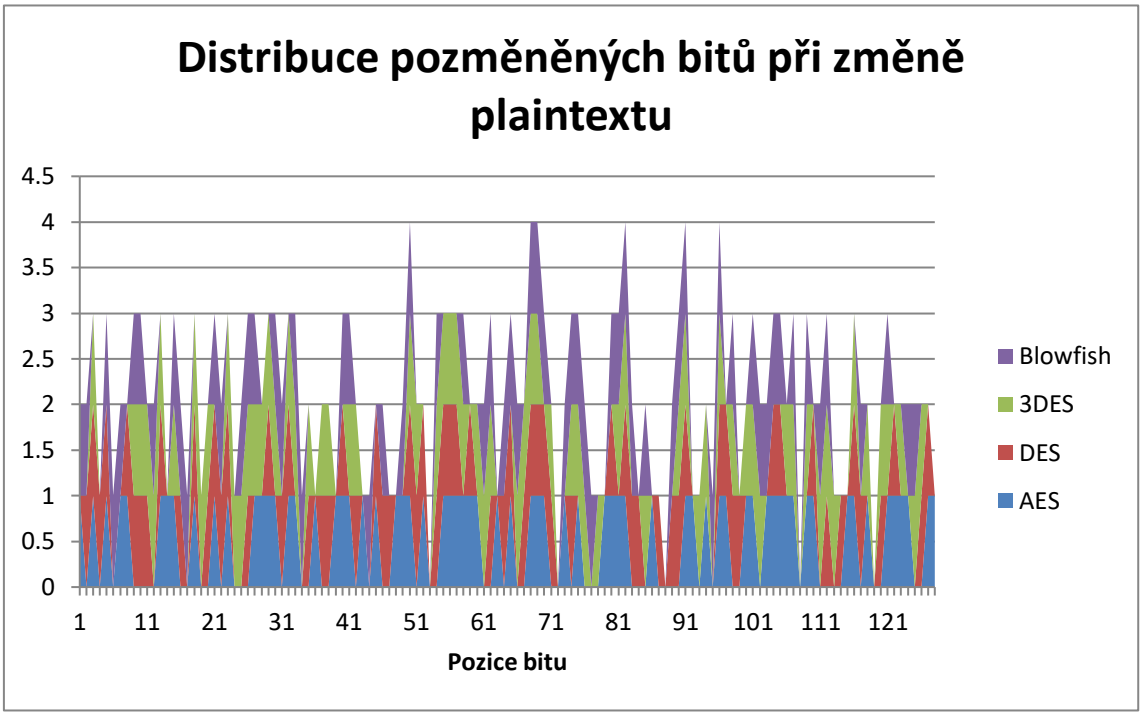
Všechna měření byla provedena pětkrát. Při těchto měřeních byl zaznamenán průměr spolu s nejnižší a nejvyšší hodnotou.

Tabulka 4.10 Vliv změny v plaintextu na ciphertext [vlastní zpracování]

Název algoritmu	Nejmenší změna (%)	Největší změna(%)	Průměr (%)
AES-128	51,6	54,7	52,8
DES-64	50,8	53,1	52,3
3DES-192	50	52,3	51,5
Blowfish	49,2	52,3	51,2

V průběhu všech měření došlo k jedinému případu u algoritmu Blowfish, kdy počet změněných bitů nedosáhl hranice 50%. Největší změny se projevovaly u algoritmu AES. Celkově z naměřených údajů lze říci, že všechny otestovaný algoritmy uspěly

v testu a minimální změna v původní zprávě má za následek velkou změnu v šifrovaném textu.



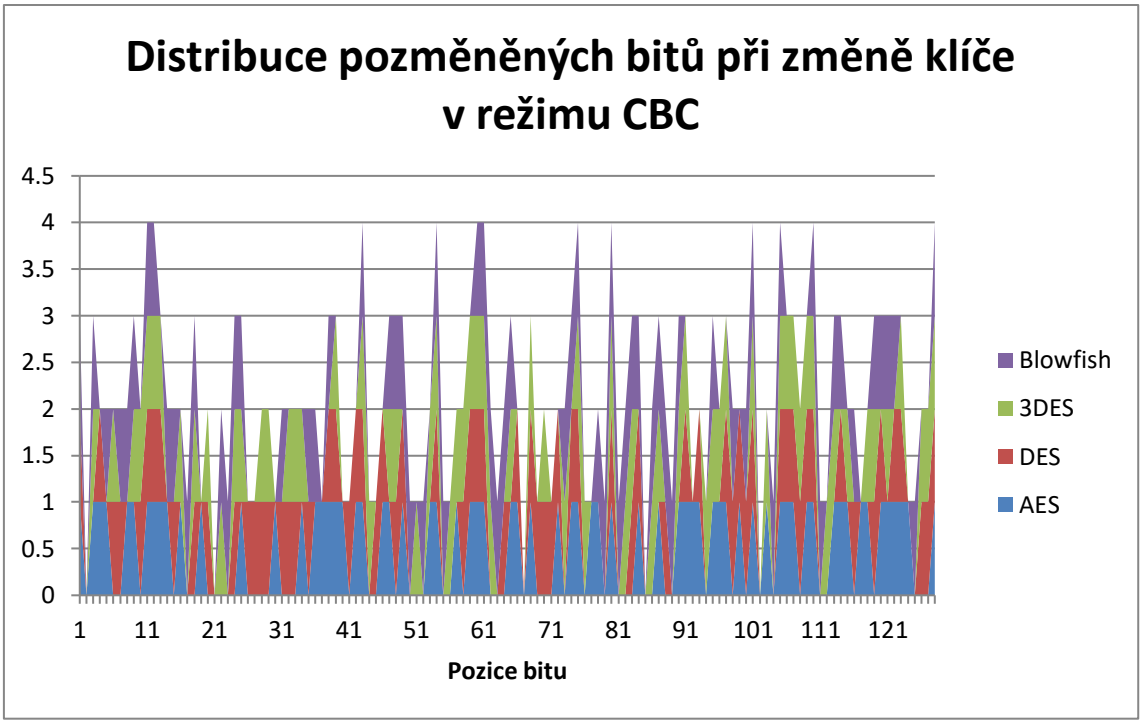
Obrázek 4.6 Ukázka distribuce změny bitů při změně v plaintextu [vlastní zpracování]

Z grafu je vidět, že všechny otestované algoritmy správně distribuovaly změnu v celé délce zašifrované zprávy. Ani jeden z algoritmů nevykazoval žádné známky anomálie, která by mohla ohrozit bezpečnost šifry.

V další části tohoto testu byla šifrována stejná zpráva za pomoci dvou různých klíčů. Otestovány byly režimy ECB a CBC. I v tomto testu by mělo dojít ke změně alespoň poloviny bitů rovnoměrně v celé délce zašifrovaného textu.

Tabulka 4.11 Vliv změny v klíči na ciphertext v módu CBC [vlastní zpracování]

Název algoritmu	Nejmenší změna (%)	Největší změna (%)	Průměr (%)
AES-128	50,8	52,3	51,6
DES-64	50,8	53,9	51,8
3DES-192	51,2	53,1	51,9
Blowfish	50,8	51,6	51,4

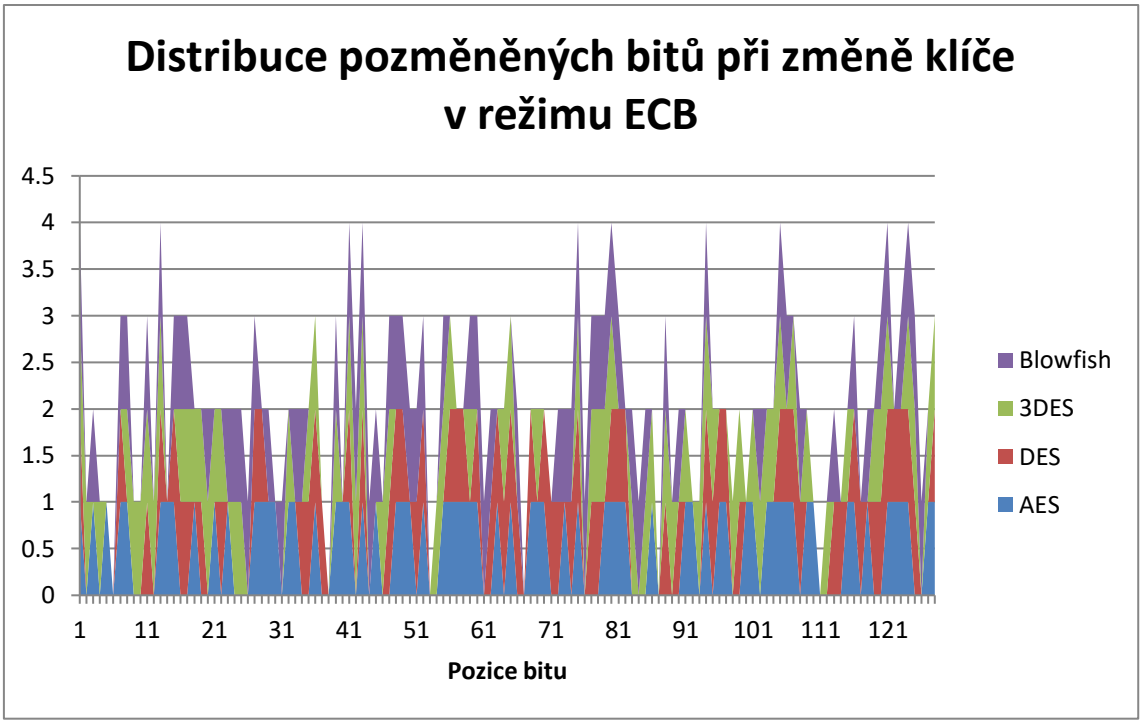


Obrázek 4.7 Ukázka distribuce změny bitů při změně klíče v režimu CBC [vlastní zpracování]

Všechny algoritmy obstály i v tomto testu s lehce vyrovnanějšími výsledky. Nejlépe se vedlo algoritmu 3DES, který dosáhl nejlepších výsledků jak v průměru, tak v nejnižší naměřené hodnotě. Žádný z algoritmů neměl nejnižší naměřenou hodnotu, která by nesplňovala kritérium 50%.

Tabulka 4.12 Vliv změny v klíči na ciphertext v režimu ECB [vlastní zpracování]

Název algoritmu	Nejmenší změna (%)	Největší změna (%)	Průměr (%)
AES-128	51,2	53,1	52,5
DES-64	50	53,9	51,4
3DES-192	51,6	53,1	52
Blowfish	50,8	52,3	51,5



Obrázek 4.8 Ukázka distribuce změny bitů při změně klíče v režimu ECB [vlastní zpracování]

Obdobně jako v předchozím testu nedošlo k výskytu žádného případu, kdy by algoritmus nesplnil požadované minimum. Nejvyšších hodnot v průměru dosahoval algoritmus AES, nejnižších pak opět algoritmus Blowfish. Celkově nejmenší i největší naměřená hodnota se objevila u algoritmu DES. U tohoto

algoritmu také došlo k neobvyklé anomálii v distribuci změn. V 80% testovaných případů bylo v prvních 10 bitech zprávy pozměněno méně než 3 bitů. Tento trend je také vidět v přiloženém grafu a stál by za další zkoumání na větším vzorku dat.

5 Shrnutí výsledků

V praktické části byly testovány čtyři blokové a dvě proudové šifry za pomoci dvou knihoven. Ačkoliv bylo před testy ověřeno, že obě knihovny produkují stejný výsledek při použití identických vstupů, vykazovaly nečekaně vysoké rozdíly ve všech testovacích scénářích. Z blokových šifer se ve všech výkonnostních testech nejlépe umístila šifra AES, která dosahovala mnohonásobně vyšší rychlosti zpracování dat nežli ostatní testované šifry. Důvodem takto znatelného rozdílu je pravděpodobně nejen optimalizace knihoven, ale také specializované instrukce v procesoru, které mají za účel zrychlit šifrování pomocí AES. V nejdůležitějším testu, tedy rychlost šifrování v závislosti na délce zprávy dosáhla stabilní rychlosti kolem 235 MB/sec. Maximální naměřená rychlost dosahovala 629 MB/sec pro knihovnu Cryptography a 394 MB/sec pro knihovnu PyCryptoDome. U této šifry byl také pozorován zajímavý trend, kdy rychlost zpracování dat před stabilizací rapidně klesla. Menší, ale podobný pokles rychlosti lze pozorovat i v jiných obdobných pracích, například u knihovny OpenSSL. Nejhůře si pak vedly algoritmy DES a 3DES. U algoritmu DES se objevil velmi vysoký rozdíl rychlosti v závislosti na použité knihovně, způsobený tím, že v případě knihovny Cryptography je algoritmus DES implementován jako 3DES se třemi stejnými klíči. Nejvyrovnanějších výsledků dosahoval algoritmus Blowfish, který ve všech případech s výjimkou módu CFB dosahoval průměrné rychlosti kolem 90 MB/sec. U blokových šifer lze pozorovat vysoké rozdíly rychlosti při různých módech v závislosti na použité knihovně. Zatímco v případě módu ECB jasně vedla knihovna PyCryptoDome, u které dosahovala rychlost šifry AES více jak dvojnásobku oproti CBC, ve všech ostatních případech měla navrch knihovna Cryptography.

U obou testovaných proudových šifer se objevil stejný pokles rychlosti jako v případě AES. Stejně tak lze pozorovat vysoké rozdíly v závislosti na použité knihovně. V případě PyCryptoDome jsou RC4 i ChaCha20 rychlostně srovnatelné s AES v CBC režimu. U knihovny Cryptography se šifra ChaCha20 ukázala jako nejrychlejší algoritmus v celém testu, dosahující několikanásobně vyšší maximální rychlosti než kterákoliv jiná šifra, včetně AES a RC4.

Všechny blokové šifry také obstály v testech rozptýlení informací, kdy byl měřen vliv minimální změny ve vstupu na výslednou podobu šifrované zprávy. Za zmínku pak stojí algoritmus DES v režimu ECB, kdy v 80% testovaných případů při změně klíče projevoval minimální změny v prvních 10 bitech zprávy.

6 Závěry a doporučení

Cílem této práce bylo poskytnout čtenáři úvod do oblasti kryptografie se zaměřením na symetrické šifry moderní kryptografie. V teoretické části byly vysvětleny základní pojmy, několik příkladů z historie a nejznámější moderní algoritmy. Jelikož se jedná o velmi rozsáhlou problematiku, nebylo možné zahrnout do práce všechny důležité pojmy. Pozornost nebyla věnována například hashovacím funkcím, detailnějšímu popisu možných útoků, či kvantové kryptografii.

V praktické části byl zkoumán vliv různých nastavení na výkon vybraných algoritmů. Dále byl testován rozdíl při použití dvou různých knihoven v jazyce Python. Dle očekávání se z blokových šifer nejlépe umístila šifra AES, která je v současné době uznávaná jako kryptografický standard. Na základě výsledků této práce by autor doporučil používat 128 bitovou variantu, která nabízí znatelný rozdíl v rychlosti oproti variantám s větším klíčem. Takto velký klíč je také považován za dostatečný a při správně implementaci by měla být šifra bezpečná. Ačkoliv režim ECB nabízí vyšší rychlost, nelze ho doporučit pro možná bezpečnostní rizika. Z tohoto důvodu byla většina testů provedena v režimu CBC. Z proudových šifer si nejlépe vedla šifra ChaCha20. Tuto šifru je vhodné používat v kombinaci s Poly1305, aby byla zajištěna integrita dat.
Ačkoliv obě testované knihovny dosahovaly velmi dobrých výsledků, knihovna Cryptography vedla téměř ve všech testech. Cryptography je také postavena na oficiální knihovně OpenSSL a kromě šifrovacích primitiv nabízí třídu Fernet. Jedná se o implemenetaci symetrické šifry AES v kombinaci s HMAC, která zajišťuje, že zpráva nemůže být přečtena/manipulována bez znalosti klíče.

Do budoucna by bylo vhodné rozšířit práci například o asymetrické algoritmy a hashovací funkce. Dále by bylo vhodné otestovat vliv kryptografie na výkon systému. Například by šlo porovnat negativní dopad na rychlost úložiště, či životnost baterie u mobilních zařízení.

7 Seznam použité literatury

[1] PIPER, F. C. a Sean MURPHY. Kryptografie. Praha: Dokořán, 2006. Průvodce pro každého. ISBN 80-7363-074-5.

[2] SINGH, Simon. Kniha kódů a šifer: tajná komunikace od starého Egypta po kvantovou kryptografii. 2. vyd. v českém jazyce. Přeložil Dita ECKHARDTOVÁ, přeložil Petr KOUBSKÝ. Praha: Dokořán, 2009. Aliter (Argo: Dokořán). ISBN 978-80-7363-268-7.

[3] SCHNEIER, Bruce. Applied cryptography: protocols, algorithms, and source code in C. 20th anniversary edition. Indianapolis, IN: Wiley, [2015]. ISBN 978-1-119-09672-6

[4] KOHNO, Tadayoshi, Niels FERGUSON a Bruce SCHNEIER. Cryptography engineering: design principles and practical applications. Indianapolis, IN: Wiley Pub., c2010. ISBN 978-0470474242.

[5] E. Barker and A. Roginsky. Transitioning the Use of Cryptographic Algorithms and Key Lengths. Draft NIST Special Publication 800-131A Revision 2. National Institute of Standards and Technology (NIST), July 2018.

[6] BINGMANN, Timo. Speedtest and Comparsion of Open-Source Cryptography Libraries and Compiler Flags. Panthema.net [online]. Karlsruhe: Timo Bingmann, 2008, 14. 7. 2008 [cit. 2019-08-07]. Dostupné z: https://panthema.net/2008/0714-cryptography-speedtest-comparison

[7] BERNSTEIN, Daniel. ChaCha, a variant of Salsa20. Cr.yp.to [online]. Lausanne: Workshop Record of SASC 2008, 2008, 20. 1. 2008 [cit. 2019-08-07]. Dostupné z: https://cr.yp.to/chacha/chacha-20080120.pdf

[8] ChaCha20. Libsodium [online]. 2019 [cit. 2019-08-01]. Dostupné z: https://libsodium.gitbook.io/doc/advanced/stream_ciphers/chacha20

[9] BURSZTEIN, Elie. Speeding up and strengthening HTTPS connections for Chrome on Android. Google Security Blog [online]. 2014 [cit. 2019-08-02]. Dostupné z: https://security.googleblog.com/2014/04/speeding-up-and-strengthening-https.html

[10] MILLER, Damien. ChaCha20 and Poly1305 in OpenSSH. Djm's personal weblog [online]. 2013, 24. 4. 2013 [cit. 2019-08-02]. Dostupné z: http://blog.djm.net.au/2013/11/chacha20-and-poly1305-in-openssh.html
[11] ROSE, Margaret. Data integrity. SearchDataCenter [online]. 2005 [cit. 2019-08-02]. Dostupné z: https://searchdatacenter.techtarget.com/definition/integrity
[12] What Is a Digital Signature?. Instantssl [online]. Instantssl [cit. 2019-08-02]. Dostupné z: https://www.instantssl.com/digital-signature
[13] JUNE 2019. Top 500 [online]. BERKELEY, Calif.; FRANKFURT, Germany; and KNOXVILLE, Tenn.: Top500, 2019 [cit. 2019-08-01]. Dostupné z: https://www.top500.org/lists/2019/06/
[14] BHARGAVAN, Karthikeyan Bhargavan a Gaëtan LEURENT. Sweet32: Birthday attacks on 64-bit block ciphers in TLS and OpenVPN. Sweet32 [online]. Sweet32, 2016 [cit. 2019-08-01]. Dostupné z: https://sweet32.info/
[15] Processor allocation. IBM Knowledge Center [online]. IBM, 2014, 12. 6. 2014 [cit. 2019-08-01]. Dostupné z: https://www.ibm.com/support/knowledgecenter/POWER6/iphb1/iphb1_vios_planning_sea_procs.htm
[16] ADVANCED ENCRYPTION STANDARD. National institute of standards and technology: COMPUTER SECURITY RESOURCE CENTER [online]. Gaithersburg, Maryland: NIST, 1997, 1997 [cit. 2019-08-04]. Dostupné z: https://csrc.nist.gov/csrc/media/publications/shared/documents/itl-bulletin/itlbul1997-02.txt
[17] PŘIBYL, Jiří a Jindřich KODL. Ochrana dat v informatice. Praha: České vysoké učení technické, 1996. ISBN 80-01-01664-1
[18] KOWALCZYK, Chris. Kerckhoffs's principle. Crypto-it [online]. 2013, 20. 7. 2013 [cit. 2019-08-03]. Dostupné z: http://www.crypto-it.net/eng/theory/kerckhoffs.html
[19] Announcing the ADVANCED ENCRYPTION STANDARD (AES). National institute of standards and technology [online]. Gaithersburg, Maryland: NIST, 2001, 26. 1. 2001 [cit. 2019-08-01]. Dostupné z: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf

[20] Wikimedia Commons contributors. Soubor:Vigenere-square.png [online]. Wikimedia Commons, the free media repository; 28. 8. 2017 [cit. 2019-07-10]. Dostupné z: https://commons.wikimedia.org/w/index.php?title=File:Vigenere-square.png&oldid=256609801.

8 Přílohy

Zdrojové kódy použité k testování šifer se nachází na přiloženém

UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Akademický rok: 2018/2019

Studijní program: Aplikovaná informatika
Forma studia: Prezenční
Obor/kombinace: Aplikovaná informatika (ai3-p)

Podklad pro zadání BAKALÁŘSKÉ práce studenta

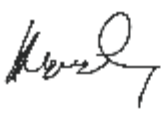
Jméno a příjmení:	**Daniel Havrda**
Osobní číslo:	**I1600533**
Adresa:	Řezníčkova 1076, Náchod, 54701 Náchod 1, Česká republika
Téma práce:	Analýza vybraných kryptografických algoritmů
Téma práce anglicky:	Analysis of selected cryptographic algorithms
Vedoucí práce:	doc. Ing. Vladimír Soběslav, Ph.D. Katedra informačních technologií

Zásady pro vypracování:

Bakalářská práce pojednává o problematice kryptografických algoritmů. Cílem práce je navrhnout a provést testy pro analýzu vybraných šifer s důrazem na výpočetní výkon a porovnat výsledky.

Seznam doporučené literatury:

[1] KARGER, M. WWW průvodce moderní kryptografií. Zlín: Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky, 2008. 64 s. Vedoucí bakalářské práce Ing. Karel Peršťka, Ph.D. [2] SINGH, S. Kniha kódů a šifer: tajná komunikace od starého Egypta po kvantovou kryptografii. Praha: Dokořán, 2003. 382 s. ISBN 80-865-6918-7. [3] STALLINGS, William. Cryptography and Network Security Principles and Practices. 4th edition. [s.l.] : Prentice Hall, 2005. 592 s. ISBN 0-13-1873164.

Podpis studenta:  Datum: 6.8.2019

Podpis vedoucího práce:  Datum: 6.8.2019