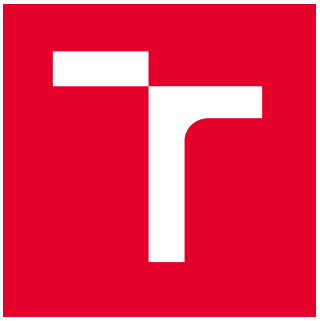


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

OPERAČNÍ MÓDY SYMETRICKÝCH ŠIFER

BLOCK CIPHER MODE OF OPERATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Alois Kunert**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Ing. Václav Zeman, Ph.D.**
SUPERVISOR

BRNO 2022

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Alois Kunert ***ID:*** 221555

Ročník: 3 ***Akademický rok:*** 2021/22

NÁZEV TÉMATU:

Operační módy symetrických šifer

POKYNY PRO VYPRACOVÁNÍ:

Práce je zaměřena na analýzu moderních symetrických šifrovacích algoritmů. V teoretické části práce bude provedeno srovnání vlastností standardizovaných i nově používaných symetrických šifrovacích algoritmů včetně rozboru operačních módů a popisu možností implementace. Na základě uvedeného rozboru bude navržena a realizována webová aplikace pro šifrování souborů.

DOPORUČENÁ LITERATURA:

[1] Ferguson, Niels, Schneier, Bruce. Practical Cryptography. Wiley, 2003.

[2] Menezes, Alfred J., Oorschot, Paul C. van, Vanstone, Scott A.. Handbook of Applied Cryptography. 2001.

Termín zadání: 7.2.2022 ***Termín odevzdání:*** 31.5.2022

Vedoucí práce: doc. Ing. Václav Zeman, Ph.D.

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Tato bakalářská práce se zabývá Operačními módy symetrických šifer. Tato práce se zaměřuje jak na základní operační módy, tak i na novější operační módy, které poskytují i dodatečné funkce.
V semestrální práci byl realizován koncepční návrh webové aplikace, který byl v této práci realizován.
Byla provedena analýza základních operačních módů implementovaných ve webové aplikaci. Pro porovnání jejich rychlostí oproti základním módům realizovaným v semestrální práci.

KLÍČOVÁ SLOVA

Operační mód, Balení klíče, šifrování zprávy s možností vytvoření MAC, základní operační módy, GCM, CCM, CMAC, KW a KWP.

ABSTRACT

This Bachelor thesis deals with Operation modes of symmetric ciphers. This thesis deals with both basic operation modes and with newer operation modes that offer additional functions.
A conceptual design was made in semestral thesis, which was realised in this thesis.
Finally an analysis of the basic operation modes, implemented in the web application ,was made to make a comparison between the measurments taken in the semestral thesis.

KEYWORDS

Operation modes, Key Wrapping, Encryption of a message with an option to create a MAC, Basic operation modes, GCM, CCM, CMAC, KW and KWP.

KUNERT, Alois. *Operační módy symetrických šifer*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2022, 81 s. Bakalářská práce. Vedoucí práce: doc. Ing. Václav Zeman, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Alois Kunert
VUT ID autora:	221555
Typ práce:	Bakalářská práce
Akademický rok:	2021/22
Téma závěrečné práce:	Operační módy symetrických šifer

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

Úvod do bakalářské práce,

Tato práce se věnuje oblasti Operačních módů symetrických šifer. Zejména se věnuje rozboru jednotlivých operačních módů a jejich porovnání podle určitých parametrů a jejich realizaci ve webové aplikaci pro výukové účely.

Hlavním cílem bylo provést rozbor jednotlivých operačních módů, porovnat je, vytvořit koncepční návrh webové aplikace a realizovat jej a vytvořit webovou aplikaci pro výukové účely.

Tato webová aplikace je zaměřena na realizaci operačních módů symetrických šifer. Uživatel bude schopen používá všechny operační módy uvedené v bakalářské práci s pomocí dvou symetrických šifer.

Cíle práce

Práce je zaměřena na analýzu moderních symetrických šifrovacích algoritmů. V teoretické části práce bude provedeno srovnání vlastností standardizovaných i nově používaných symetrických šifrovacích algoritmů včetně rozboru operačních módů a popisu možností implementace. Na základě uvedeného rozboru bude navržena a realizována webová aplikace pro šifrování souborů.

1 Teoretický podklad práce

1.1 Symetrické šifry

Symetrické šifry jsou šifry, které používají k šifrování a dešifrování jeden klíč na-rozdíl od asymetrických šifer kde se používá veřejný a soukromý klíč. Na-rozdíl od asymetrických šifer jsou symetrické šifry mnohonásobně rychlejší.

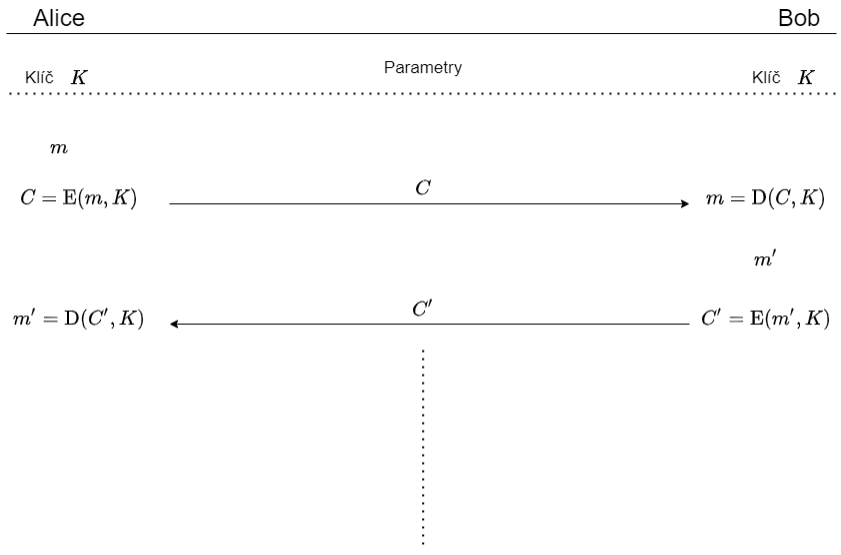
Symetrické šifry se dělí na proudové a blokové. Proudové šifry šifrují bit po bitu, zatímco blokové šifrují jednotlivé bloky bitů.

Proudové šifry jsou šifry v nichž je zpráva kombinována s klíčovým proudem, který je vytvořen z klíče. Proudové šifry jsou rychlejší než blokové a jsou i méně náročné na hardwarové požadavky.

Blokové šifry jsou šifry, ve kterých se zpráva rozdělí na jednotlivé bloky o fixní velikosti a následně jsou tyto bloky postupně šifrovány.

Základní schéma šifrované komunikace

Mějme dvě osoby Alici a Boba, které si chtějí posílat zprávy po síti a chtějí spolu komunikovat tak aby jejich komunikace byla utajená, tj. data která si posílají by měli být tajná. Pro vedení takovéto komunikace je potřeba nejprve aby měla Alice i Bob k dispozici funkci, která umí dešifrovat D a funkci, která umí šifrovat P a klíč K, který bude použit jak při šifrování, tak i dešifrování. Šifrovaná komunikace mezi Alicí a Bobem je znázorněna v obrázku 1.1.



Obr. 1.1: Šifrovaná komunikace mezi Alicí a Bobem

Tato komunikace probíhá tak, že Alice nejprve vytvoří zprávu m, kterou chce Bobovi poslat. Proto ji zašifruje do kryptogramu $C = \mathrm{E}(m, K)$, a ten přes síť pošle Bobovi. Ten dešifruje získaný kryptogram pro získání původní zprávy $m = \mathrm{D}(C, K)$. Poté chce poslat odpověď m' Alici tak ji zašifruje $C' = \mathrm{E}(m', K)$ a pošle po síti nový kryptogram C'. Alice si jej dešifruje a získá Bobovu odpověď $m' = \mathrm{D}(C', K)$. Další komunikace probíhá stejným způsobem.

1.1.1 Základní terminologie

Pro účely tohoto textu je zde používána jednotná terminologie. Tato terminologie je následující:

- K - šifrovací klíč
- IV - inicializační vektor
- P - otevřený text (plaintext)
- p_i - i-tý vstupní blok
- p_i^*- konečný vstupní blok
- o_i - i-tý mezivýpočtový blok
- C - šifrový text/ kryptogram (ciphertext)
- c_i - i-tý šifrový blok
- c_i^*- konečný šifrový blok
- $\mathrm{E}(P, K)$ - šifrování textu P klíčem K
- $\mathrm{D}(C, K)$ - dešifrování textu C klíčem K
- $A \oplus B$ - XOR A a B, neboli exkluzivní součet A a B
- LSB_s - s nejméně důležitých bitů
- MSB_s - m nejméně důležitých bitů
- $A||B$ - zřetězení řetězců A a B
- $\mathrm{len}(P)$ - délka P v bitech
- ICB - počáteční counter blok
- ctr_i - i-tý counter blok

1.1.2 DES

DES, neboli Data Encryption Standard, je blokový šifrovací algoritmus ze 70. let 20 století. Tato šifra byla v roce 1977 zvolena Národním Institutem standardů a technologii (USA) za šifrovací standard FIPS 46. Původně byla užívána státními institucemi, poté se rozšířila i do soukromého sektoru. V dnešní době se DES považuje za nebezpečnou šifru kvůli malé velikosti klíče.

1.1.3 3DES

Triple DES, neboli Triple Data Encryption Algorithm vznikl v roce 1995 jako odpověď na již zastaralou šifru DES, která se svým 56-bitovým klíčem již nebyla považována za bezpečnou.

Samotný 3DEA je odvozenec DESu. Samotný rozdíl je takový, že 64-bitové bloky jsou šifrovány za pomoci 64-bitového klíče namísto 56 bitového jako tomu bylo u DESu, ale k šifrování se užívá pouze 56 bitů jako tomu bylo u DESu, zbývajících 8 bitů jsou určeny ke kontrole chyb. Těchto 8 bitů slouží k tomu, aby z každého osmice bitů udělali liché číslo.

1.1.4 Twofish

Twofish byl jedním z kandidátů na AES (Advanced Encryption Standard).Twofish je symetrická bloková šifra s bloky o velikosti 128 bitů. Twofish používá S-boxy, MDS(maximum distance separable) matice, Pseudo-Hadamardovu transformací a bělení (Whitenning).

1.1.5 AES

AES nebo-li Advanced Encryption Standard, je šifrovací standard, který je nástupcem DES. AES je odvozeninou blokové šifry Rijndael, což byla jedna z šifer, co se účastnila výběrového řízení pro nový šifrovací standard. AES byl zveřejněn v roce 2001 americkým Národním institutem standardů a technologií (NIST).

AES je blokovou šifrou s bloky o velikosti 128 bitů a klíčem o velikosti 128, 192 a 256 bitů. Počet rund r závisí na velikosti klíče, tj.: pro AES-128 je $r = 10$, AES-192 je $r = 12$ a AES-256 je $r = 14$.

1.2 Operační módy symetrických šifer

Operační módy jsou módy, kterou mohou zvýšit míru bezpečnosti daného kryptosystému. Operační módy se používají u blokových šifer tak, že určují jakým způsobem má být daný operační mód použit a jaké dodatečné operace se mají ještě provádět.

1.2.1 Electronic Codebook Mode

Electronic Codebook Mode, neboli ECB je jeden z nejjednodušších operačních módů. Tento mód specifikuje, že šifra musí být použita na každý blok nezávisle na sobě.

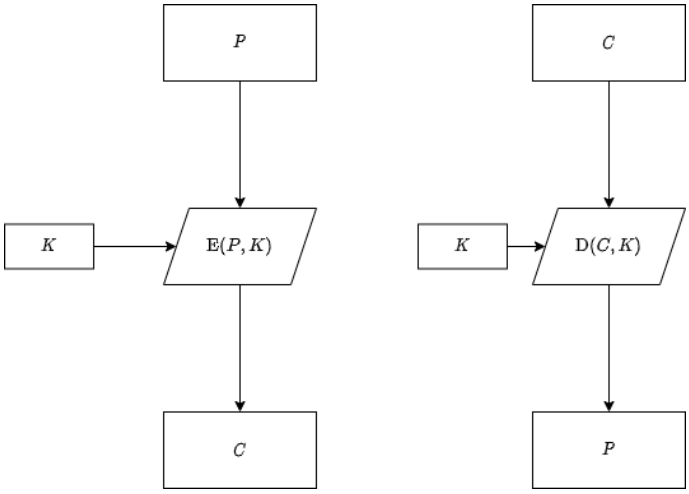
Toto přináší výhodu, že se může šifrovat více zpráv nezávisle na sobě, čímž dosahujeme vyšší rychlosti. Při dešifrování platí stejná pravidla jako u šifrování a to sice každý blok musí být dešifrován nezávisle na ostatních.[1]

ECB nijak nezvedá míru zabezpečení. Schéma ECB je zobrazeno na obrázku 1.2. Šifrování v ECB:

$$c_i = \mathrm{E}(p_i, K), \text{ kde } i = 1, \ldots, n. \tag{1.1}$$

Dešifrování v ECB:

$$p_i = \mathrm{D}(c_i, K), \text{ kde } i = 1, \ldots, n. \tag{1.2}$$



Obr. 1.2: Operační mód ECB, šifrování vlevo, dešifrování vpravo

1.2.2 Cipher Block Chaining Mode

Cipher Block Chaining Mode, neboli CBC je operační mód u kterého dochází k řetězení(chaining) bloků. CBC potřebuje pro svoji funkci Inicializační vektor IV, pro kombinaci s prvním blokem, tento inicializační vektor nemusí být tajný, ale musí být nepředvidatelný, tj. nelze předpovídat tvar dalších inicializačních vektorů.[1]

CBC funguje na principu řetězení, a to sice tak, že na první řetězec se ještě před jakýmkoliv šifrováním naváže inicializační vektor, toto vázaní probíhá jako exkluzivní součet, následně proběhne samotné šifrování a výstup šifry je následně použit jako inicializační vektor pro druhý řetězec, atd..[1]

Dešifrování probíhá tak, že je nutné začít u prvního článku. První článek se dešifruje tak, že se první šifrovaný text dešifruje a poté je exkluzivně sečten s inicializačním vektorem. Pro získání dalších článků se postupuje tak, že se nejprve dešifruje příslušný článek a poté se exkluzivně sečte s předchozím článkem, který je v podobě šifrovaného textu.[1]

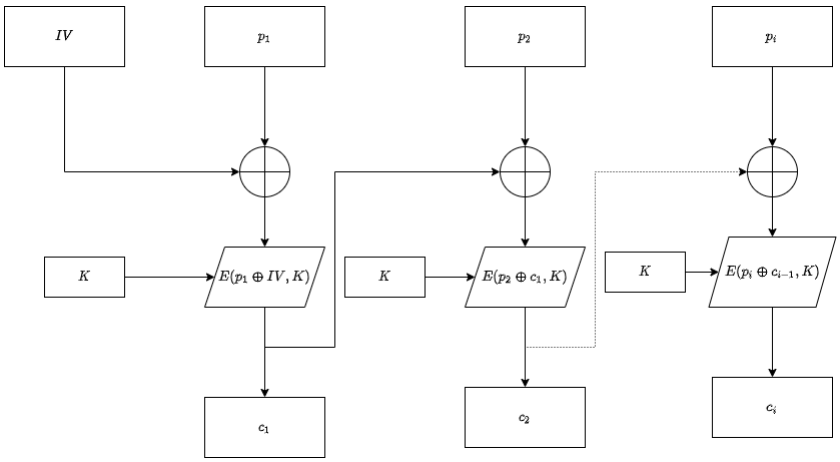
CBC zvedá míru zabezpečení tak, že v ideálním případě je potřeba mít k dispozici celý šifrovaný řetězec pro úspěšné dešifrování celé zprávy. U CBC v průběhu šifrování nelze mít spuštěno více šifrovacích vláken najednou, jelikož jeden blok závisí na druhém, ale při dešifrování jsou nám dostupné všechny bloky, tudíž lze dešifrovat paralelně na více blocích najedou.[1]
Schéma šifrování v CBC je vyobrazeno na obrázku 1.3 a schéma dešifrování je vyobrazeno na obrázku 1.4.

Šifrování v CBC:

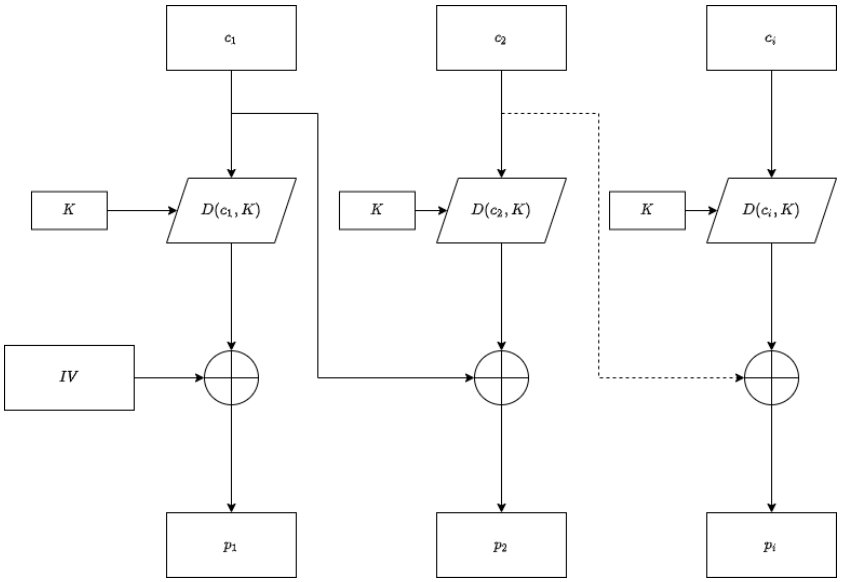
$$\begin{aligned} c_1 &= \mathrm{E}(p_1 \oplus IV, K); \\ c_i &= \mathrm{E}(p_i \oplus c_{i-1}, K), \text{ pro i} = 2 \ \ldots n. \end{aligned} \tag{1.3}$$

Dešifrování v CBC:

$$\begin{aligned} p_1 &= \mathrm{D}(c_1, K) \oplus IV; \\ p_i &= \mathrm{D}(c_i, K) \oplus c_{i-1}, \text{ pro i} = 1 \ \ldots n. \end{aligned} \tag{1.4}$$



Obr. 1.3: Šifrování u módu CBC



Obr. 1.4: Dešifrování u módu CBC

1.2.3 Cipher Feedback Mode

Cipher Feedback Mode, neboli CFB je operační mód, který pracuje na podobném principu jako CBC, ale na rozdíl od CBC se na vstupu používá inicializační vektor.

Tento mód mimo inicializační vektor IV používá parametr s takový, že $1 \leq s \leq b$, kde b je velikost bloku používané šifry v bitech.[1]

CBC funguje tak, že se do vstupu šifry nejprve vloží IV následně se zanechá právě s bitů a zbytek je zahozen, těchto s bitů je následně užito tak že jsou exkluzivně sečteny s otevřeným textem a tím nám vzniká šifrovaný text. Ten se následně používá namísto inicializačního vektoru pro další blok a také se posílá dále.[1]

Při dešifrování je opět nutné znát inicializační vektor, který je opět použit jako vstup do šifry z níž se na výstupu ponechá jenom s bitů a tento výstup je exkluzivně sečten s prvním šifrovaným textem a tím získáme první část zprávy. Pro získání dalších bloků se již místo IV používá šifrovaný text a to sice tak že pro získání i-té části zprávy se používá $(i-1)$-tý šifrovaný text namísto IV a i-tý šifrovaný text pro exkluzivní součet.[1]

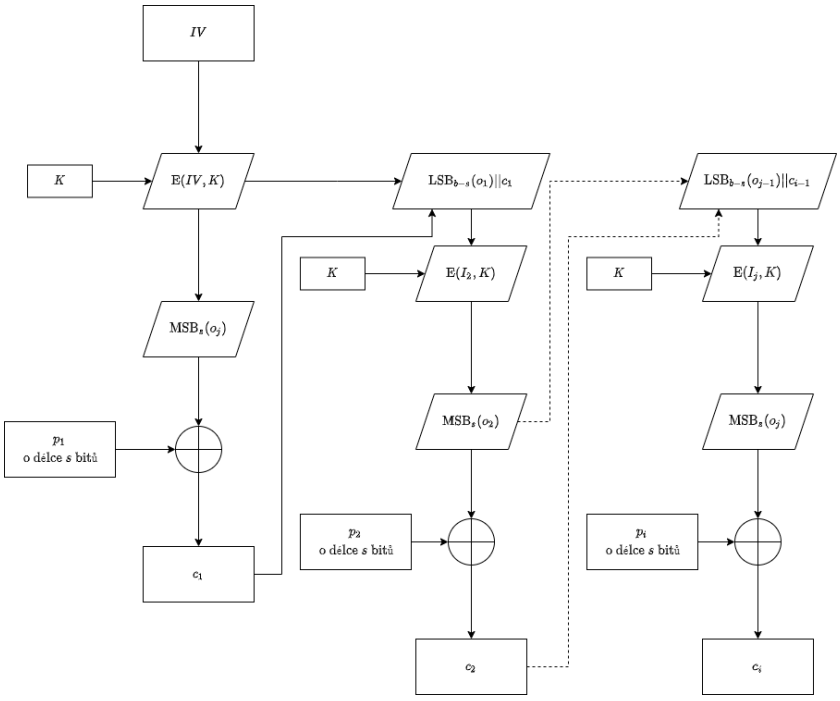
Schéma pro šifrování v CFB je vyobrazeno v obrázku 1.5 a schéma dešifrování je vyobrazeno v obrázku 1.6.

Šifrování v CFB:

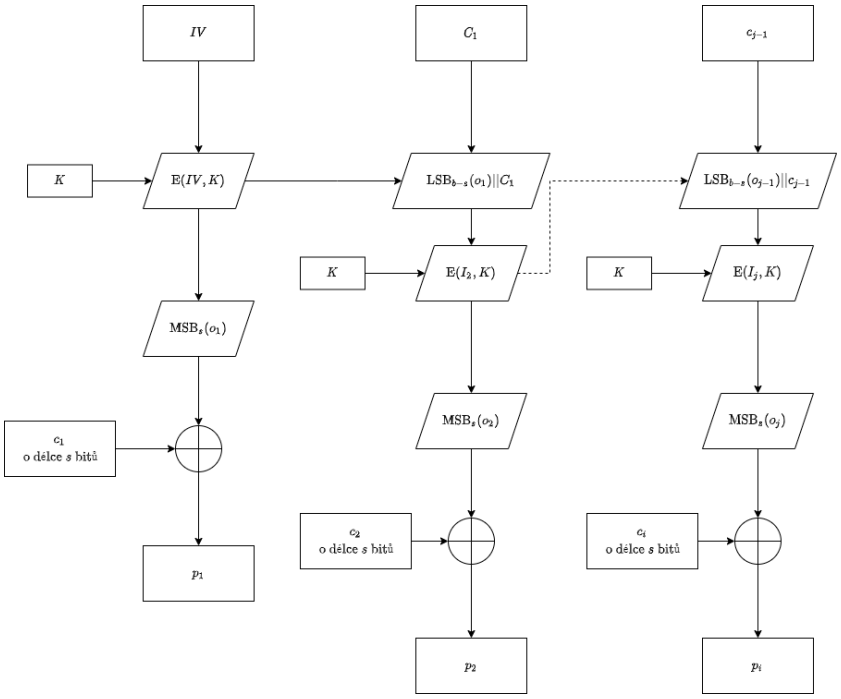
$$
\begin{aligned}
&o_1 = \mathrm{E}(IV, K);\\
&I_j = \mathrm{LSB}_{b-s}(o_{j-1}||c_{j-1}) \text{ ,kde j} = 2,\ \ldots n;\\
&o_j = \mathrm{E}(I_j, K) \text{ ,kde j} = 2,\ \ldots n;\\
&c_j = p_j \oplus \mathrm{MSB}(o_j) \text{ ,kde j} = 1,\ 2,\ \ldots n;\\
&c_j^* = p_j^* \oplus \mathrm{MSB}_s(o_j).
\end{aligned}
\tag{1.5}
$$

Dešifrování v CFB:

$$
\begin{aligned}
&o_1 = \mathrm{E}(IV, K);\\
&I_j = \mathrm{LSB}_{b-s}(o_{j-1}||c_{j-1}) \text{ ,kde j} = 2,\ \ldots n;\\
&o_j = \mathrm{E}(I_j, K) \text{ ,kde j} = 2,\ \ldots n;\\
&p_j = c_j \oplus \mathrm{MSB}(o_j) \text{ ,kde j} = 1,\ 2,\ \ldots n;\\
&p_j^* = c_j^* \oplus \mathrm{MSB}_s(o_j).
\end{aligned}
\tag{1.6}
$$



Obr. 1.5: Šifrování u módu CFB



Obr. 1.6: Dešifrování u módu CFB

1.2.4 Output FeedBack Mode

Output FeedBack Mode, neboli OFB je operační mód, který používá princip řetězení.

OFB funguje tak, že inicializační vektor IV je zašifrován a výsledný šifrovaný text je použit jako IV pro další blok dat a také pro exkluzivní součet se zprávou a tímto exkluzivním součtem se získává výsledný šifrovaný text.[1]

Při dešifrování je potřeba znát IV ten se nejprve zašifruje a poté je exkluzivně sečten s prvním šifrovaným textem čímž se získá první část zprávy. Dále je opět výstup po šifrování IV zašifrován a je opět použit k exkluzivnímu součtu s druhým šifrovaným textem a tak se pokračuje dokud není zpracována celá zpráva.[1]

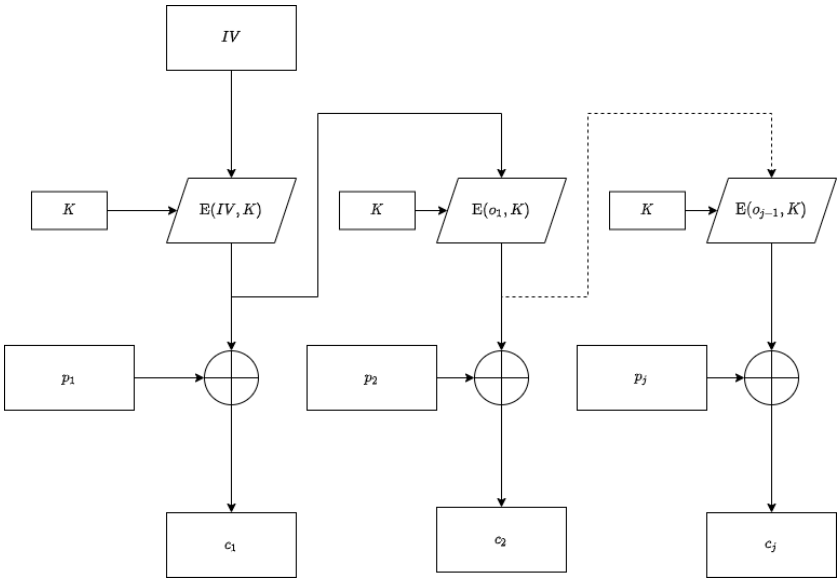
Schéma pro šifrování v OFB je vyobrazeno v obrázku 1.7 a schéma dešifrování je vyobrazeno v obrázku 1.8.

Šifrování v OFB:

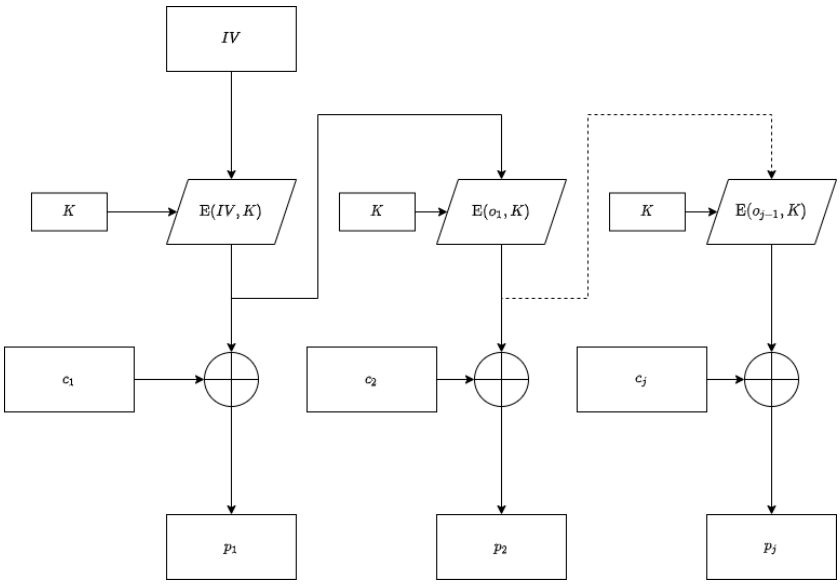
$$\begin{aligned} o_1 &= E(IV, K); \\ o_j &= \mathrm{E}(o_{j-1}, K), \text{ kde } j = 2, \ldots, n; \\ c_j &= p_j \oplus o_j, \text{ kde } j = 1, 2, \ldots, n. \end{aligned} \tag{1.7}$$

Dešifrování v OFB:

$$\begin{aligned} o_1 &= E(IV, K); \\ o_j &= \mathrm{E}(o_{j-1}, K), \text{ kde } j = 2, \ldots, n; \\ p_j &= c_j \oplus o_j, \text{ kde } j = 1, 2, \ldots, n. \end{aligned} \tag{1.8}$$



Obr. 1.7: Šifrování v módu OFB



Obr. 1.8: Dešifrování v módu OFB

1.2.5 Counter Mode

Counter Mode, neboli CTR mode je operační mód, který používá pro zvýšení zabezpečení tzv. counter, neboli počítadlo, to má za účel zajistit aby se každý blok dat lišil od následujícího.[1]

Při šifrování je šifrován counter, který je pak exkluzivně sečten s prostým textem. a tím je získán šifrový text. Pro poslední blok může dojít k situaci, kde je prostý text o menší délce u než je je velikost bloku, v tomto případě se exkluzivně sečte u nejdůležitějších bitů s prostým textem a zbytek bitů se zahodí.[1]

Při dešifrování je šifrován counter, který je pak exkluzivně sečten s šifrovým textem a tím je získán prostý text. Pro poslední blok může dojít k situaci, kde je prostý text o menší délce u než je je velikost bloku, v tomto případě se exkluzivně sečte u nejdůležitějších bitů s prostým textem a zbytek bitů se zahodí.[1]

Schéma pro šifrování v Counter módu je vyobrazeno v obrázku 1.9 a schéma dešifrování je vyobrazeno v obrázku 1.10.

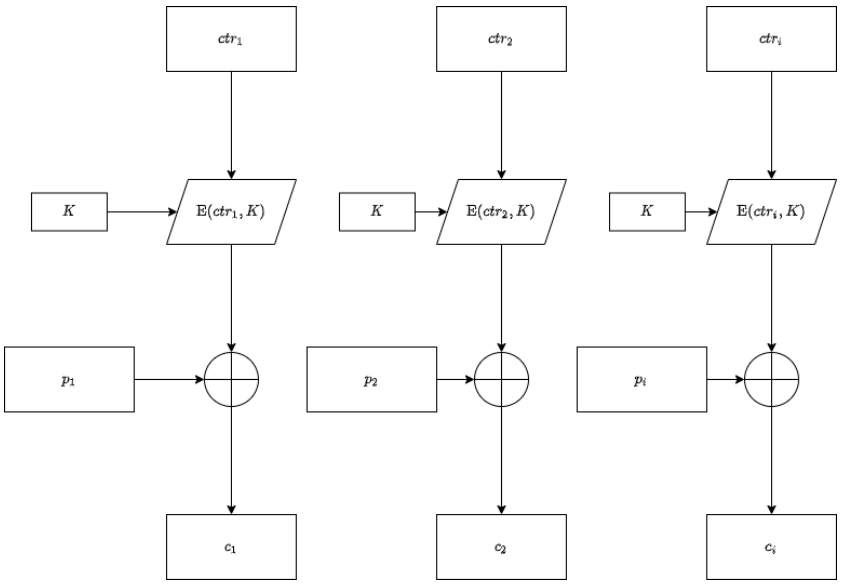
V rovnicích jak pro šifrování v CTR ale i pro dešifrování je o_i mezivýpočtová proměnná.

CTR šifrování:

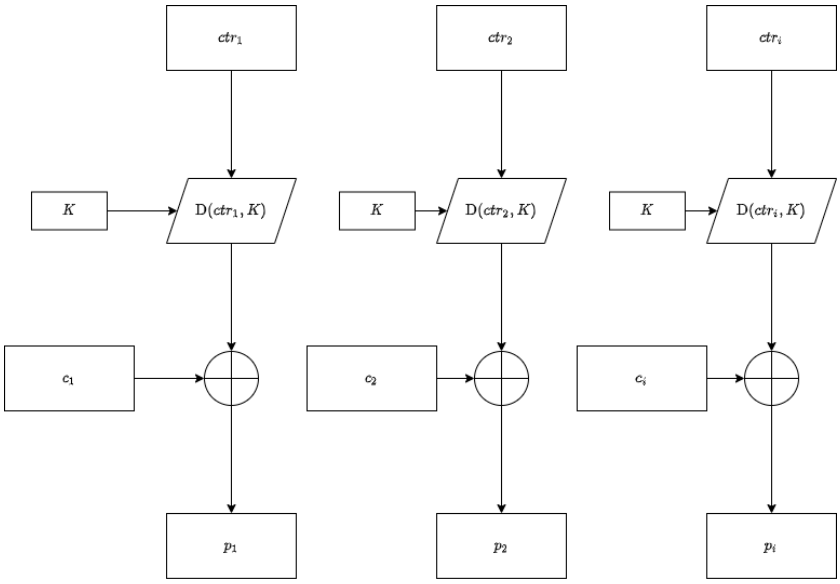
$$\begin{aligned} o_i &= \mathrm{E}(ctr_i, K), \text{ kde } i = 1, 2 \ldots n; \\ c_i &= p_i \oplus o_j, \text{ kde } i = 1, 2 \ldots n. \end{aligned} \tag{1.9}$$

CTR dešifrování:

$$\begin{aligned} o_i &= \mathrm{E}(ctr_i, K), \text{ kde } i = 1, 2 \ldots n; \\ p_i &= c_i \oplus o_j, \text{ kde } i = 1, 2 \ldots n. \end{aligned} \tag{1.10}$$



Obr. 1.9: Šifrování v módu Counter



Obr. 1.10: Dešifrování v módu Counter

1.3 Message Authentication Code

Message Authentication Code, neboli MAC je kód sloužící k zajištění integrity a autenticity dat. MAC neslouží jako obvyklá podepisovací funkce, a to sice z toho důvodu, že k podpisu se mimo samotné zprávy používá i šifrovací klíč.[2]

MAC se může kombinovat s operačním módem CBC na tzv. CBC-MAC, kde se jednotlivé podepsané bloky řetězí do jednoho řetězu.

1.3.1 CMAC

CMAC, neboli Cipher-Based message authentication mode je operační mód sloužící k použití u blokových šifer.

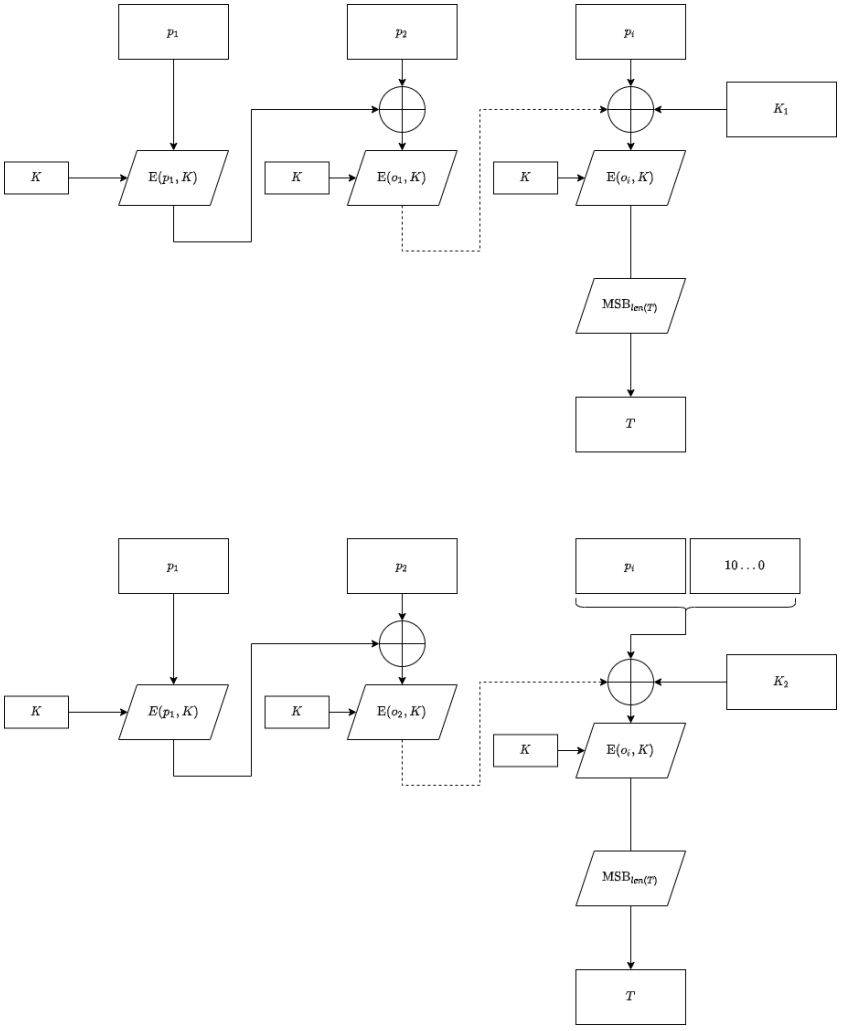
CMAC využívá operačního módu CBC pro zkomplikování možností útoků na výsledný šifrový text. samotný algoritmus postupuje tak, že se nejprve vygenerují dva podklíče K_1 a K_2 z původního klíče K, následně se zpráva naformátuje příslušný počet bloků, načež mohou nastat dvě varianty:

1. Délka zprávy je celočíselným násobkem velikosti bloku. V tomto případě je zpráva rozdělena na příslušný počet bloků a poslední blok je exkluzivně sečten s K_1 a výsledné pole bloků je výstupní zpráva.[2]
2. Délka zprávy není celočíselným násobkem velikosti bloku. V tomto případě je zpráva rozdělena na co nejvíce kompletních bloků. A poslední řada bitů, která nebyla dostatečně veliká na vytvoření posledního bloku, je je doplněna o jednu '1' a o tolik '0' dokud řada bitů nedosáhne velikosti bloku. A poslední blok je exkluzivně sečten s K_2.[2]

Tvorba klíčů K_1 a K_2 se provádí následujícím způsobem:

$$
\begin{aligned}
&L = \mathrm{E}(0^{128}, K);\\
&K_1 = \begin{cases} L << 1 & \text{pokud } \mathrm{MSB}_1(L) = 0;\\ (L << 1) \oplus R_{128} & \text{pokud } \mathrm{MSB}_1(L) \neq 0; \end{cases}\\
&K_2 = \begin{cases} K_1 << 1 & \text{pokud } \mathrm{MSB}_1(K_1) = 0;\\ (K_1 << 1) \oplus R_{128} & \text{pokud } \mathrm{MSB}_1(K_1) \neq 0. \end{cases}
\end{aligned}
\tag{1.11}
$$

,kde $R_{128} = 0^{120}10000111$.[2]
Poté následuje CBC řetězení s $IV = 0$. Načež proběhne na konci exkluzivní součet posledního bloku buď s K_1 nebo K_2. A na konci je výsledek zkrácen podle délky MAC, která závisí na klíči a následný výstup je MAC. Schéma tvorby štítku je vyobrazeno na obrázku1.11. [2]



Obr. 1.11: CMAC, 1. varianta nahoře a 2. dole

Ověření MAC probíhá tak, že si příjemce vytvoří z přijaté zprávy vlastní MAC a pokud se přijatý a vytvořený MAC rovnají tak nedošlo k manipulaci se zprávou.

1.3.2 CCM

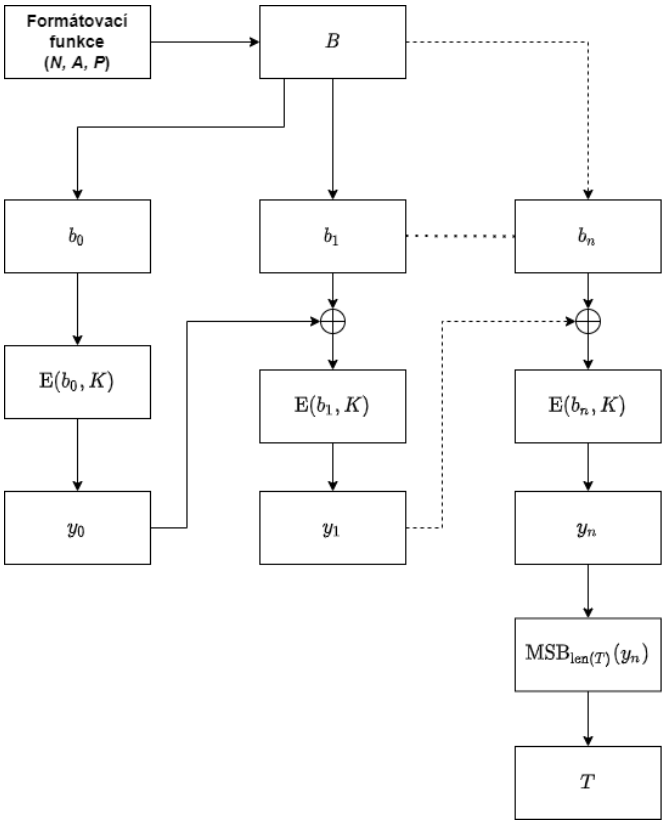
CCM, neboli Counter with Cipher block chaining Message authentication code je operační mód sloužící k autentizaci, který kombinuje mód CBC-MAC a counter mód. CCM je mód uzpůsobený pro šifry používající bloky o velikosti 128 bitů.[3]

CCM potřebuje na svém vstupu nonce N, vstupní data P a spojené data A. Ty jsou následně formátovány pomocí formátovací funkce, která z vytvoří řadu bloků $b_0, b_1, \ldots, b_n$. Tato formátovací funkce má vlastnost, kde pokud je použit stejný nonce ale jiné P a A tak by nemělo dojít ke kolizi, kde $b_i = b'_i$. Dále blok b_0 unikátně určuje nonce a také je rozdílný od dalších counter bloků, které se v CCM použijí. Schéma procesu získávání štítku je vyobrazeno v obrázku1.12 načež schéma při němž je získán samotný kryptogram C je vyobrazen na obrázku 1.13.
Šifrování v CCM:

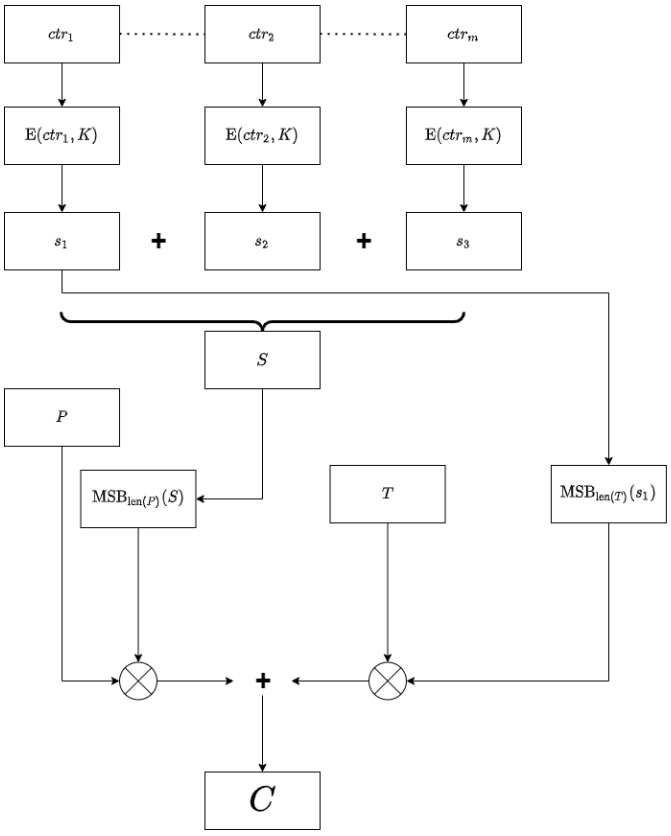
$$
\begin{aligned}
&\text{Použije se formátovací funkce na } (N, A, P) \text{ pro vytvoření bloků } B;\\
&y_0 = \mathrm{E}(b_0, K);\\
&y_i = \mathrm{E}(b_i \oplus y_{i-1}, K) \text{ ,kde } i = 1, 2, \ldots, n;\\
&T = \mathrm{MSB}_{\mathrm{len}(T)}(Y_n);\\
&\text{Vytvoří se } m \text{ counter bloků } ctr \text{ , kde } m = \lceil \mathrm{len}(P)/128 \rceil;\\
&s_i = \mathrm{E}(Ctr_i, K), \text{ kde } i = 0, 1, \ldots, m;\\
&S = s_1 || s_2 || \ldots || s_m;\\
&c = (P \oplus \mathrm{MSB}_{\mathrm{len}(P)}(S)) || (T \oplus \mathrm{MSB}_{\mathrm{len}(T)}(s_0)).
\end{aligned}
\tag{1.12}
$$

Ověřovaní v CCM:

$$
\begin{aligned}
&\text{Pokud je délka } C \leq \mathrm{len}(T), \text{ tak je ověření automaticky NEPLATNÉ;}\\
&\text{Vytvoří se } m \text{ counter bloků } Ctr \text{ , kde } m = \lceil \mathrm{len}(P)/128 \rceil;\\
&s_i = \mathrm{E}(ctr_i, K), \text{ kde } i = 0, 1, \ldots, m;\\
&P = \mathrm{MSB}_{\mathrm{len}(C)-\mathrm{len}(T)}(C) \oplus \mathrm{MSB}_{\mathrm{len}(C)-\mathrm{len}(T)}(S);\\
&T = \mathrm{LSB}_{\mathrm{len}(T)}(C) \oplus \mathrm{MSB}_{\mathrm{len}(T)}(s_0);\\
&\text{Nesplňuje-li } (N, A, P), \text{ výše zmíněné vlastnosti pro vytvoření bloků}\\
&b_0, b_1, \ldots, b_n \text{ tak je ověření NEPLATNÉ;}\\
&y_0 = \mathrm{E}(b_0, K);\\
&y_i = \mathrm{E}(b_i \oplus y_{i-1}, K) \text{ ,kde } i = 1, 2, \ldots, n;\\
&\text{Pokud se } T \neq \mathrm{MSB}_{\mathrm{len}(T)}(y_n) \text{ tak je ověření NEPLATNÉ, jinak se vrací } P.
\end{aligned}
\tag{1.13}
$$



Obr. 1.12: Šifrování v CCM po získání T



Obr. 1.13: Šifrování v CCM, získání C

Formátovací funkce

Pro potřeby přehlednosti se délka v bitech len(Q) bude zapisovat jako q. Toto platí analogicky pro další proměnné tedy len(T) = t, atd.
Formátovací funkce (N, A, P), kde N je nonce A jsou autorizovaná data a P je otevřený text, má následující podmínky:

$$\begin{aligned} &t \in \{4, 6, 8, 10, 12, 14, 16\}; \\ &q \in \{2, 3, 4, 5, 6, 7, 8\}; \\ &n \in \{7, 8, 9, 10, 11, 12, 13\}; \\ &n + q = 15; \\ &a < 2^{64}; \\ &p < 2^{8q}. \end{aligned} \tag{1.14}$$

Délky uvedené v podmínkách jsou uvedeny v násobcích osmi, tedy pokud je $q = 3$, tak Q je o délce 24 bitů.Q je řetězec bitů o délce $q \cdot 8$ reprezentující p.[3]

Formátování kontrolní informace a nonce

Vedoucí oktet v prvním bloku b_0, obsahuje čtyři vlajky pro kontrolu informací: dva jednotné bity,Rezervovaný a Adata('0' pokud $a = 0$, '1' pokud $a > 0$), a dvě trojice pro zakódování t a q. Rezervovaný bit je rezervován pro případné budoucí rozšíření formátovaní. Parametr t je kódován jako $\frac{t-2}{2}$, a parametr q je kódován jako $q - 3$.
Zbývajících 15 oktetů prvního bloku jsou určeny pro nonce a binární reprezentaci

Číslo bitu	7	6	5	4	3	2	1	0
Obsah	Rezervováno	Adata	$[(t-2)/2]$			$[q-1]$		

délky zprávy v q oktetech.

Číslo oktetu	0	$1 \dots 15 - q$	$16 - q \dots 15$
Obsah	Vlajky	N	Q

Formátování autorizovaných dat A

Data A jsou kódována následujícím způsobem:

- Pokud $0 < a < 2^{16} - 2^8$, tak se a kóduje jako dva oktety.
- Pokud $2^{16} - 2^8 \leq a < 2^{32}$, tak se a kóduje jako $0\text{xff}\|0\text{xfe}\|[a]_{32}$, kde $[a]_{32}$ je binární reprezentace a o délce 32 bitů.
- Pokud $2^{32} \leq a < 2^{64}$, tak se a kóduje jako $0\text{xff}\|0\text{xff}\|[a]_{64}$, kde $[a]_{64}$ je binární reprezentace a o délce 64 bitů.

Formátování Counter bloků

Counter bloky Ctr_i se formátují podle následující tabulky:

Číslo oktetu	0	$1 \dots 15 - q$	$16 - q \dots 15$
Obsah	Vlajky	N	$[i]_{8q}$

Formátování vlajkového bloku vypadá následovně:

Číslo oktetu	7	6	5	4	3	2	1	0
Obsah	Rezervovaný	rezervovaný	0	0	0	$[q-1]$		

1.3.3 GCM

GCM, neboli Galois/Counter mode je operační mód, který zajišťuje autentizaci tak i zvedá míru zabezpečení. GCM je mód uzpůsobený pro šifry s blokem o velikosti 128 bitů.[4]

GCM má dvě funkcionality. První je, že zašifruje tajná data a vytvoří autentizační štítek jak k tajným datům tak i k neutajovaným datům, které jsou součástí zprávy. Druhá je, že pouze zašifruje tajná data a vytvoří k ním MAC štítek. Druhá verze se nazývá GMAC.[4]

GCM má na vstupu vstupní text I, dodatečná autentizovaná data A a inicializační vektor IV. GCM se v tomto případě snaží zachovat utajení I a vytvořit k němu MAC štítek a vytvořit pouze MAC štítek pro data A.[4]

Vstupní trojice I, IV a A musí splňovat tyto podmínky:

$$\begin{aligned} &\text{len}(P) \leq 2^{39} - 256; \\ &\text{len}(A) \leq 2^{64} - 1; \\ &1 \leq \text{len}(IV) \leq 2^{64-1}. \end{aligned} \tag{1.15}$$

, a zároveň I, A a IV musí být o délce, která je násobkem osmi.

Na výstupu je šifrový text C, který má stejnou délku jako P a štítek T. Délka štítku t v bitech může mít velikost: 128, 120 112, 104 nebo 96 bitů, v některých případech může mít velikost 64 nebo 32 bitů. GCM používá:

Inkrementovací funkci

Mějme $s \in \mathbf{N}$ a bitový proud X takový, že $\text{len}(X) \geq s$. Pak[4]:

$$\text{inc}_s(X) = \text{MSB}_{\text{len}(X)-s}(X) \mid\mid [\text{int}(\text{LSB}_s(X)) + 1 \mod 2^s]_s. \tag{1.16}$$

,kde $\text{int}(\text{LSB}_s(X))$ reprezentuje celočíselnou hodnotu $\text{LSB}_s(X)$. Tedy inkrementovací funkce inkrementuje s pravých bitů modulo 2^s ostatní bity se nemění.

Násobení bloků

Mějme dva vstupní bloky X, Y pak[4]:

$$\begin{aligned} &x_0, x_1 \ldots x_{127} \text{ reprezentuje řadu bitů v } X; \\ &Z_0 = 0^{128} \text{ a } V_0 = Y; \\ &Z_{i+1} = \begin{cases} Z_i & \text{pokud } x_i = 0; \\ Z_i = Z_i \oplus V_i & \text{pokud } x_i = 1; \end{cases} \\ &V_{i+1} = \begin{cases} V_i >> 1 & \text{pokud } \text{LSB}_1(V_i) = 0; \\ Z_i = (V_i >> 1) \oplus R & \text{pokud } \text{LSB}_1(V_i) = 1; \end{cases} \\ &\text{Výstupem násobení je } Z_{128} \text{ ,neboli } X \cdot Y. \end{aligned} \tag{1.17}$$

GHASH funkce

GHASH je hašovací funkce používající násobení bloků fixním parametrem. GHASH má jako prerekvizitu blok H neboli hašovací podklíč. H je generován jako šifrový výstup nulového bloku.[4]

Mějme na vstupu bitový proud X, takový, že $\text{len}(X) = 128m$, kde $m \in \mathbf{N}$, tak:

$$\begin{aligned}
&x_1, x_2, \ldots x_m \text{ jsou jednotlivé bloky, které dohromady tvoří } X;\\
&y_0 = 0^{128};\\
&y_i = (y_{i-1} \oplus x_i) \cdot H, \text{ kde } i = 1, \ldots, m;\\
&\text{Výstupem je } y_m.
\end{aligned} \tag{1.18}$$

GCTR funkce

GCTR funkce je counter funkce, která určuje hodnotu počítadla neboli counteru.[4]

Mějme na vstupu počáteční counterový blok ICB a bitový proud X o libovolné délce, pak:

$$\begin{aligned}
&\text{Je-li } X \text{ prázdný tak je výstupem prázdný bitový proud } Y;\\
&n = \lceil \text{len}(X)/128 \rceil;\\
&x_1, x_2, \ldots x_n \text{ jsou jednotlivé bloky, které dohromady tvoří } X;\\
&ctr_1 = ICB;\\
&ctr_i = \text{inc}_{32}(ctr_{i-1}) \text{ , kde } i = 2, \ldots, n;\\
&y_i = x_i \oplus \text{E}(ctr_i, K) \text{ , kde } i = 1, \ldots, n-1;\\
&y_n^* = x_n^* \oplus \text{MSB}_{\text{len}(x_n^*)}(\text{E}(ctr_n, K));\\
&\text{Výstupem je } Y = Y_1 || Y_2 || \ldots || Y_n^*.
\end{aligned} \tag{1.19}$$

Algoritmus pro šifrování v GCM

Mějme na vstupu inicializační vektor IV, prostý text P a autentizovaná data A. Pak šifrování probíhá[4]:

$$
\begin{aligned}
&H = \mathrm{E}(0^{128}, K);\\
&j_0 = \begin{cases} IV||0^{31}||1 & \text{, pokud } \mathrm{len}(IV) = 96;\\ j_0 = \mathrm{GHASH}_H(IV||0^{s+64}||[\mathrm{len}(IV)]_{64}) & \text{, pokud } \mathrm{len}(IV) \neq 96; \end{cases}\\
&\text{, kde } s = 128\lceil \mathrm{len}(IV)/128\rceil - \mathrm{len}(IV);\\
&C = \mathrm{GCTR}_K(\mathrm{inc}_{32}(j_0), P);\\
&u = 128 \cdot \lceil \mathrm{len}(C)/128\rceil - \mathrm{len}(C);\\
&v = 128 \cdot \lceil \mathrm{len}(A)/128\rceil - \mathrm{len}(A);\\
&S = \mathrm{GHASH}_H(A||0^v||C||0^u[\mathrm{len}(A)]_{64}||[\mathrm{len}(C)]_{64});\\
&T = \mathrm{MSB}_i(\mathrm{GCTR}_K(j_0, S));\\
&\text{Výstupem je dvojice šifrového textu a štítku}(C, T).
\end{aligned}
\tag{1.20}
$$

Nejprve se vytvoří hašovací podklíč H jako výstup šifry, do které se vloží nulový blok o velikosti 128-bitů. Poté se vytvoří prvotní counter blok J_0. Poté je vytvořen šifrový text C. Následně jsou bloky A a C rozšířeny o co nejméně nulových bitů tak, aby výsledná velikost byla násobkem velikosti bloku, zřetězení těchto bloků je následně doplněno o 64-bitovou reprezentaci velikosti A a C na to je následně použit GHASH čímž je získán blok S. A nakonec je blok S zašifrován s prvotním counterovým blokem J_0 výstup je poté zkrácen na požadovanou velikost autentizačního štítku.[4]

Algoritmus pro dešifrování a ověřovaní v GCM

Mějme na vstupu inicializační vektor, šifrový text, autentizovaná data a MAC štítek. Pak dešifrování a ověřování probíhá následovně[4]:

$$
\begin{aligned}
&\text{Pokud } \mathrm{len}(IV), \mathrm{len}(A) \text{ nebo } \mathrm{len}(C) \text{nejsou podporovány, nebo } \mathrm{len}(T) \neq t \\
&\text{, potom je automaticky ověření NEPLATNÉ;} \\
&H = E(0^{128}, K); \\
&j_0 = \begin{cases} IV||0^{31}||1 & \text{, pokud } \mathrm{len}(IV) = 96; \\ \mathrm{GHASH}_H(IV||0^{s+64}||[\mathrm{len}(IV)]_{64}) & \text{, pokud } \mathrm{len}(IV) \neq 96; \end{cases} \\
&P = \mathrm{GCTR}_K(\mathrm{inc}(j_0), C); \\
&u = 128 \cdot \lceil \mathrm{len}(C)/128 \rceil - \mathrm{len}(C); \\
&v = 128 \cdot \lceil \mathrm{len}(A)/128 \rceil - \mathrm{len}(A); \\
&S = \mathrm{GHASH}_H(A||0^v||C||0^u[\mathrm{len}(A)]_{64}||[\mathrm{len}(C)]_{64}); \\
&T' = \mathrm{MSB}_t(GCTR_K(j_0, S)); \\
&\text{Pokud } T = T', \text{ tak je kontrola platná a je bezpečná dál pracovat s } P \\
&\text{, v opačném případě je kontrola NEPLATNÁ.}
\end{aligned}
\tag{1.21}
$$

1.3.4 Zabalování klíče

Balení klíče neboli Key Wrapping je proces, ve kterém je klíč zapouzdřen do obalu za účelem jeho zabezpečení buď při přenosu na síti nebo pro zabezpečení na nedůvěryhodném úložišti.

KW a KWP

KW, neboli AES Key wrap a KWP, neboli AES Key wrap with padding jsou operační módy určené pro Key Wrapping. Oba tyto módy jsou určeny pro AES, tedy velikost bloku je omezena na 128-bitů, ale je možné tento mód použít i za pomoci jiné 128 bitové šifry.[5]

Existují i varianty pro 3DES, které se jmenují TW a TKW, viz[5].

Funkce pro balení W

Mějme balící funkci W, která má jako svoji prerekvizitu klíč K pro 128-bitovou šifru a na svém vstupu má proud S, který se skládá z n polo-bloků, kde $n \geq 3$, a na svém výstupu má šifrový text C Pak balení W(S) probíhá následovně[5]:

Nejprve se inicializují proměnné:

$$\begin{aligned} &s = 6(n-1); \\ &\text{Mějme polo-bloky } s_1, s_2, \ldots, s_n \text{ takové, že } S = s_1||s_2||\ldots||s_n; \\ &A^0 = s_1; \\ &R_i^0 = s_i, \text{ kde } i = 2, \ldots, n. \end{aligned} \tag{1.22}$$

Poté pro $t = 1, \ldots, s$, následují mezi-výpočty:

$$\begin{aligned} A^t &= \mathrm{MSB}_{64}(E(A^{t-1}||R_2^{t-1}, K)) \oplus [t]_{64}; \\ R_i^t &= R_{i+1}^{t-1}, \text{ kde } i = 2, \ldots, n-1; \\ R_n^t &= \mathrm{LSB}_{64}(E(A^{t-1}||R_2^{t-1}, K)). \end{aligned} \tag{1.23}$$

Výstup je následující:

$$\begin{aligned} c_1 &= A^s; \\ c_i &= R_i^s, , \text{ kde } i = 2, \ldots, n; \\ C &= c_1||c_2||c_3||\ldots||c_n. \end{aligned} \tag{1.24}$$

Funkce pro rozbalování W^{-1}

Funkce pro rozbalování má stejné prerekvizity jako funkce pro balení, a na vstupu je C, skládající se z n polo-bloků, kde $n \geq 3$, a na svém výstupu má S. Pak rozbalování $\mathrm{W}^{-1}(C)$ vypadá následovně:
Nejprve se inicializují proměnné[5]:

$$\begin{aligned} &s = 6(n-1); \\ &\text{Mějme polo-bloky } c_1, c_2, \ldots, c_n \text{ takové, že } C = c_1||c_2||\ldots||c_n; \\ &A^s = c_1; \\ &R_i^s = c_i, \text{ kde } i = 2, \ldots, n. \end{aligned} \tag{1.25}$$

Poté pro $t = s, s-1, \ldots, 1$, následují mezi výpočty[5]:

$$\begin{aligned} A^{t-1} &= \mathrm{MSB}_{64}(D((A^t \oplus [t]_{64})||R_n^t, K)); \\ R_2^{t-1} &= \mathrm{LSB}_{64}(D((A^t \oplus [t]_{64})||R_n^t, K)); \\ R_{i+1}^{t-1} &= R_i^t, \text{ kde } i = 2, \ldots, n-1. \end{aligned} \tag{1.26}$$

Výstup je následující:

$$\begin{aligned} s_1 &= A^0; \\ s_i &= R_i^0, , \text{ kde } i = 2, \ldots, n; \\ S &= s_1||s_2||s_3||\ldots||s_n. \end{aligned} \tag{1.27}$$

KW

Mějme 64-bitový základ hodnoty pro kontrolu integrity $ICV1 = 0\text{xA6A6A6A6A6A6A6A6}$. A mějme na vstupu P o podporované délce, pak balení klíče probíhá následovně[5]:

$$\begin{aligned} &S = ICV1||P; \\ &C = W(S). \end{aligned} \tag{1.28}$$

Pro rozbalování klíče potřebujeme$ICV1$ a na vstupu potřebujeme C. Pak rozbalování klíče probíhá následovně[5]:

$S = W^{-1}(C)$;
Pokud $\text{MSB}_{64}(S) \neq ICV1$, pak došlo k narušení integrity dat a tedy C je NEPLATNÉ;
$P = \text{LSB}_{64(n-1)}(S)$.

(1.29)

KWP

KWP se od KW liší tím, že se ještě využívá padding, což je proces, ve kterém se přidávají nuly na konec textu jako vycpávka.[5]
Mějme 32-bitový základ hodnoty pro kontrolu integrity $ICV2 = 0\text{xA65959A6}$. A mějme na vstupu P o podporované délce, pak balení klíče probíhá následovně[5]:

$$\begin{aligned} &\text{len}(PAD) = 8 \cdot \lceil \text{len}(P)/64 \rceil - \text{len}(P)/8; \\ &PAD = 0^{8 \cdot len(PAD)}; \\ &S = ICV2||[\text{len}(P)/8]_{32}||P||PAD; \\ &C = \begin{cases} E(S,K) & \text{, pokud len}(P) \leq 64; \\ W(S) & \text{,pokud len}(P) > 64. \end{cases} \end{aligned} \tag{1.30}$$

Pro rozbalování klíče potřebujeme$ICV2$ a na vstupu potřebujeme C. Pak rozbalování klíče probíhá následovně[5]:

Nechť n je počet polo-bloků v C

$$S = \begin{cases} D(C,K) & \text{, pokud } n = 2; \\ W^{-1}(C) & \text{, pokud } n > 2; \end{cases}$$

Pokud $\mathrm{MSB}_{32}(S) \neq ICV2$, tak došlo k narušení integrity a

rozbalování končí SELHÁNÍM;

$\mathrm{len}(P) = \mathrm{int}(\mathrm{LSB}_{32}(\mathrm{MSB}_{64}(S)))$;

$\mathrm{len}(PAD) = 8(n-1)\mathrm{len}(P)$;

Pokud $len(PAD) < 0$, nebo $\mathrm{len}(PAD) > 7$, tak došlo k narušení integrity a

rozbalování končí SELHÁNÍM;

Pokud $\mathrm{LSB}_{8len(PAD)}(S) \neq 0^{8len(PAD)}$, tak došlo k narušení integrity a

rozbalování končí SELHÁNÍM;

$P = \mathrm{MSB}_{8len(P)}(\mathrm{LSB}_{64(n-1)(S)})$.

(1.31)

1.4 Porovnání jednotlivých operačních módů a jejich implementace

1.4.1 Základní operační módy

Mějme skupinu základních operačních módů: ECB, CBC, CFB, OFB a CTR. Tyto módy lze považovat za základní, z toho důvodu, že se pouze snaží zvednou míru zabezpečení dané šifry a jejich použití nám neposkytuje jakékoliv jiné benefity. Z tohoto důvodu budou porovnávány ve své vlastní skupině, jelikož poskytují pouze jednu funkci.

1.4.2 Možnosti implementace základních operačních módů

ECB

Operační mód ECB, jako jeden z nejjednodušších a lze implementovat například těmito dvěma způsoby, a to sice:

1. Šifrování a dešifrování probíhá v přímém pořadí, tj. šifrují/dešifrují se jeden blok za druhým.
2. Šifrování a dešifrovaní probíhá na jednotlivých vláknech/procesech tak, aby se mohlo šifrovat/dešifrovat více bloků najednou.

Tedy lze mít jednoduchou implementaci, která bude postupně šifrovat blok po bloku, tím lze mít implementaci, která bude mít nízkou spotřebu paměti, avšak nebude příliš rychlá. Za to druhá možnost implementace bude více náročná na paměť i na výpočetní možnosti zařízení.

CBC

Šifrování u CBC lze implementovat pouze jedním způsobem, kvůli tomu, že je pro získání C_i nutné znát C_{i-1} mimo C_1, které se získává za pomoci IV, to vynucuje šifrovat postupně a to sice tak, že se bude stejně jako možnosti 1 u ECB šifrovat postupně po jednotlivých blocích a nelze je vykonávat paralelně.

Při dešifrování avšak existuje více možností implementace, a to sice tak že se buď bude dešifrovat stejně jako u ECB postupně, nebo lze rozdělit dešifrování jednotlivých bloků na více vláken/procesů, jelikož všechny potřebné informace pro dešifrování jsou již k dispozici a není nutné vyčkávat na mezivýsledky.

CFB

Šifrování u CBC lze implementovat pouze jedním způsobem a to sice z toho důvodu, že stejně jako tomu je u CBC je nutné pro získání C_i nutné znát C_{i-1}, mimo C_1. To tedy umožňuje šifrovat pouze tak, že se šifruje po jednotlivých blocích

Při dešifrování je avšak možné dešifrovat více způsoby prvním je jako u ECB, postupné dešifrování blok po bloku a druhým je dešifrovat tak, že se získá O_{j-1}, a tím se umožní dešifrovat P_j, kde $j = 2, \ldots, n$. Takže je možné takto dešifrovat po dvojicích. Avšak existuje i třetí možnost, kde se nejprve vypočítají všechny O_i, kde $i = 1, \ldots, n$ a následně lze dešifrovat všechny bloky najednou, nebo po libovolně velkých sekcích.

OFB

Šifrování u OFB lze provádět více způsoby, tedy postupně blok po bloku, nebo se vypočítají všechny mezihodnoty O a následně se šifruje po sekcích.

Při dešifrování lze opět dešifrovat po blocích , nebo se opět vypočítají všechny mezihodnoty O_i a dešifruje se po blocích.

Counter

Šifrování i dešifrování u Couter módu lze provádět postupně, nebo po více blocích najednou, avšak je nutné vzít v potaz změnu counteru Ctr.

1.4.3 Implementace benchmarkingu pro porovnání základních operačních módů

Jako vlastnost, která se bude porovnávat se zvolila časová náročnost t. Jako testovací data se pro každé měření vygenerovalo 200 souborů, ve formátu *.bin*, které byly o velikostech:

- 1 MB
- 10 MB
- 100 MB

Pro porovnání operačních módů bylo zvoleno programovací prostředí .NET 6.0 a programovací jazyk C#. Pro účely využití kryptografických funkcí byla zvolena knihovna *System.Security.Cryptography*, která v sobě má zabudované základní kryptografické funkce. Dále byla zvolena šifra AES-128. Klíč a inicializační vektor se vygenerovali jednou pro právě jednu velikost testovacích dat.

Bylo provedeno testování s využitím naprosto základní formy implementace jednotlivých módů, tj. bloky se šifrovaly/dešifrovaly ve frontě za sebou.

Statistika byla provedena tak, že pro každou velikost vstupních souborů bylo provedeno šifrování i dešifrování šestkrát, tedy bylo pro každou velikost bylo u každého operačního módu naměřeno právě 1200 hodnot. Tyto hodno se ukládaly do pro každou sadu o 200 souborech do vlastního *.txt* souboru. Formát ukládání a jednotlivé hodnoty měření lze nalézt v příloze A.

Pro naměřené hodnoty lze předpokládat, že operační mód ECB by měl být nejrychlejší, jelikož jako jediný mód z měřené skupiny nepřidává jakékoliv dodatečné operace.

.

1.4.4 Naměřená data

U měřených dat byl zjišťován modus, medián, variační rozpětí a aritmetický průměr. Ty byly vypočítávány ze všech hodnot, tj ze všech 1200 naměřených časů.

Měření probíhalo na zařízení s operačním systémem Windows 10, na zařízení nebyly spuštěny jakékoliv jiné aplikace a zařízení bylo připojeno k síti.

Data jsou vyobrazena v tabulce, v níž první sloupec reprezentuje velikost vstupních souborů, druhý reprezentuje modus měřených hodnot v $[ms]$, třetí je medián v $[ms]$, čtvrté je variační rozpětí v $[ms]$ a poslední je průměr.

Například mějme tabulku 1.2. Tato tabulka ve svém druhém řádku říká, že pro 1200 naměřených časů při šifrování souborů o velikosti 1 MB byl zjištěn modus 4 ms, medián 6 ms, variační rozpětí 583 ms a průměr 9,4475.

ECB

Po měření byly vypočítány hledané hodnoty, ty jsou znázorněné v tabulce1.1 a v tabulce 1.2.

Vstup [MB]	Modus [ms]	Medián [ms]	Variační Rozpětí [ms]	Průměr [ms]
1	4	6	583	9,4475
10	162	202	1494	249,045
100	1881	1949	3300	1970,6

Tab. 1.1: Získané časové statistiky z šifrování ECB

Vstup [MB]	Modus [ms]	Medián [ms]	Variační Rozpětí [ms]	Průměr [ms]
1	2	4	488	9,09289
10	33	75	544	92,78
100	1640	1674	2965	1703,132

Tab. 1.2: Získané časové statistiky z dešifrování ECB

CBC

Po měření byly vypočítány hledané hodnoty, ty jsou znázorněné v tabulce1.3 a 1.4

Vstup [MB]	Modus [ms]	Medián [ms]	Variační Rozpětí [ms]	Průměr [ms]
1	4	6	556	19,505
10	67	178	1090	216,838
100	1837	1904	9777	1979,75

Tab. 1.3: Získané časové statistiky z šifrování CBC

Vstup [MB]	Modus [ms]	Medián [ms]	Variační Rozpětí [ms]	Průměr [ms]
1	2	4	809	11,47
10	32	80	725	102,587
100	1517	1646,5	3016	1682,3

Tab. 1.4: Získané časové statistiky z dešifrování CBC

CFB

Po měření byly vypočítány hledané hodnoty, ty jsou znázorněné v tabulce1.6 a 1.5

Vstup [MB]	Modus [ms]	Medián [ms]	Variační Rozpětí [ms]	Průměr [ms]
1	6	5	397	9,703
10	55	67	485	88,28
100	1678	1757	2972	1803

Tab. 1.5: Získané časové statistiky z dešifrování CFB

Vstup [MB]	Modus [ms]	Medián [ms]	Variační Rozpětí [ms]	Průměr [ms]
1	69	79	812	94,1475
10	136	204	938	248,145
100	1951	1933	4586	2000,89

Tab. 1.6: Získané časové statistiky z šifrování CFB

OFB

Po měření byly vypočítány hledané hodnoty, ty jsou znázorněné v tabulce1.7 a 1.8

Vstup [MB]	Modus [ms]	Medián [ms]	Variační Rozpětí [ms]	Průměr [ms]
1	13	12	421	24,0283
10	233	248,5	306,893	306,38933
100	2500	2601,5	2063	2634,243

Tab. 1.7: Získané časové statistiky z šifrování OFB

Vstup [MB]	Modus [ms]	Medián [ms]	Variační Rozpětí [ms]	Průměr [ms]
1	13	18	518	22,313
10	129	131	265	129,7269
100	2700	2667	2661	2690,314

Tab. 1.8: Získané časové statistiky z dešifrování OFB

Counter

Po měření byly vypočítány hledané hodnoty, ty jsou znázorněné v tabulce1.9 a 1.10

Vstup [MB]	Modus [ms]	Medián [ms]	Variační Rozpětí [ms]	Průměr [ms]
1	12	86	1112	106,7092
10	203	247	1157	309,7625
100	2836	2636	1847	2677,66

Tab. 1.9: Získané časové statistiky z šifrování CTR

Vstup [MB]	Modus [ms]	Medián [ms]	Variační Rozpětí [ms]	Průměr [ms]
1	16	17	293	18,72
10	125	129	217	133,245
100	2273	2388	2461	2432

Tab. 1.10: Získané časové statistiky z dešifrování CTR

1.4.5 Porovnání jednotlivých módu na základě naměřených hodnot

1 MB vstup

Pro vstupní soubory o velikost 1 MB při šifrování a dešifrování se ukázal operační mód ECB jako nejrychlejší, což souhlasí s předpokladem v kapitole 1.4.3.

10 MB vstup

Pro vstupní soubory o velikosti 10 MB se při šifrování ukázal operační mód CFB jako nejrychlejší, toto zjištěni bylo proti původnímu předpokladu, že operační mód ECB by měl být nejrychlejší, viz 1.4.3.

Při dešifrování se ukázal operační mód ECB jako nejrychlejší, což souhlasí s původním předpokladem v kapitole 1.4.3.

100 MB vstup

Pro vstupní soubory o velikosti 100 MB se při šifrování ukázal operační mód CFB jako nejrychlejší, toto zjištěni bylo proti původnímu předpokladu, že operační mód ECB by měl být nejrychlejší, viz předpoklad v kapitole 1.4.3.

Při dešifrování se ukázal operační mód CBC jako nejrychlejší, což je v rozporu s původním předpokladem v kapitole 1.4.3.

Nejstabilnější operační mód

Vzhledem k tomu, že byly naměřeny velké variační rozpětí, tak se ukázalo jako velmi nutné porovnat jednotlivé operační módy i z pohledu stability, tzn. jaký operační

mód měl nejmenší variační rozpětí.

Při šifrování se jako obecně nejstabilnější operační mód ukázal mód OFB. Při dešifrování se jako nejstabilnější operační mód ukázal mód CTR.

Možné důvody rozdílných výsledků mezi předpokládaným výsledkem a naměřenými výsledky

Pro 1 MB vstup jako jediný výsledek shodoval s předpokladem v kapitole 1.4.3. Mezi potenciální důvody, proč došlo k rozdílným výsledkům patří:

- Dočasné omezení výpočetních prostředků zařízení na němž probíhalo měření
- Vyšší využití disku, na němž probíhalo měření

2 Koncepční návrh Webové aplikace

2.1 Vývojové prostředí .NET

Pro návrh webové aplikace, která se bude realizovat v bakalářské práci bylo zvoleno vývojové prostředí .NET a programovací jazyk C#.

2.1.1 C#

C# je objektově orientovaný programovací jazyk pocházející z rodiny C. Tento programovací jazyk je založen na jazyku C a jazyku Java.

2.2 Funkce aplikace

2.2.1 Implementace jednotlivých operačních módů

Aplikace bude obsahovat všechny operační módy uvedené v této práci, tj. ECB, CBC, CFB, OFB, CTR, CCM, GCM, KW a KWP. Tyto módu budou dostupné jen u těch šifer, u kterých jsou splněny základní podmínky.

2.2.2 Šifrování a dešifrování souborů/textu

Aplikace by měla být schopna šifrovat a dešifrovat jak soubory, tak i prostý text s uživatelskými vstupy klíče/inicializačního vektoru, nebo za pomoci vlastních vygenerovaných parametrů.
Pro možnosti uživatele bude aplikace obsahovat výběr mezi více typy šifer, mezi nimiž si uživatel bude moci vybírat. Mezi tyto šifry bude patřit AES s všemi podporovanými velikostmi klíče, Twofish také se všemi podporovanými velikostmi klíče.
Aplikace bude realizována tak, aby byla schopná šifrovat libovolně formátovaný soubor.

2.2.3 Grafická ukázka jednotlivých operačních módů

Aplikace by měla být schopna ukázat uživateli graficky za pomoci animovaného schématu, jakým způsobem fungují jednotlivé operační módy.
Tato funkce bude realizovaná jak při vykonávání šifrování/dešifrování tak i u definic jednotlivých operačních módů.

2.2.4 Definice a vysvětlivky k jednotlivým operačním módům

Aplikace by měla mít u každého operačního módu jeho zběžnou definici s vysvětlivkami. Dále bude existovat záložka definice, kde bude podrobně vysvětlen každý operační mód.

3 Realizace Webové aplikace

Realizovaná aplikace je přiložena v příloze B.
Stránka se nachází po spuštění na https://localhost:7012.

3.1 Segmentace na oddíly

3.1.1 Domovská stránka

Na domovské stránce lze nalézt základní informace o tématu bakalářské práce a základní informace o problematice operačních módů symetrických šifer.

3.1.2 AES

Na záložce AES lze nalézt základní informace o šifře a následné odkazy k jednotlivým operačním módům.

3.1.3 Twofish

Na záložce Twofish lze nalézt základní informace o šifře a následné odkazy k jednotlivým operačním módům.

3.1.4 Technická dokumentace

V technické dokumentaci si lze rozkliknout jednotlivé operační módy, implementované ve webové aplikaci, pro získání dodatečných informací o nich.

3.2 Manipulace se vstupy

Vstupní soubory a texty jsou přijímány webovou aplikací na principu REST Controlleru.

3.2.1 Vstupní a výstupní soubory

Aplikace je schopna mít na vstupu jakýkoliv formát souboru o velikosti 1 B až přibližně 40 MB.
Vstupní soubory jsou přijímány a ukládány do složky **ToEncrypt** pro šifrování a do **ToDecrypt** v případě dešifrování.
Vstupní soubor je po uložení a přečtení smazán. A následný výstup šifry, tedy

šifrovaný/dešifrovaný soubor, je následně uložen do složky **Encrypted** nebo do **Decrypted** přečten smazán a je jako výstup vrácen uživateli.

3.2.2 Padding souborů

Vzhledem k tomu, že ne každý vstupní soubor bude právě o rozměrech $x \cdot 128$ bitů, kde $x \in \mathrm{N}$. Byla vytvořena vlastní realizace paddingu souborů tak, aby výsledné rozšířený soubor byl právě o velikosti $x \cdot 128$ a bylo možné je zašifrovat.
Pro tuto formu je určena funkce **FileReadAllBytesWithPadding** realizovaná ve třídě **Base_Functions** viz kapitola 3.3.
Padding je použit pouze u módů, které nejsou uzpůsobeny vstupním souborům o neideálních velikostech.

3.3 Základní funkce

Základní funkcí se pro potřeby tohoto textu bude rozumět funkce realizována ve třídě **Base_Functions**. Tyto funkce jsou odděleny z toho důvodu, že funkcionalita, kterou realizují se užívá v téměř všech operačních módech, nebo jsou pro lepší přehlednost delegovány do jiné třídy.
Mezi základní funkce patří:

1. byte[] FileReadAllBytesWithPadding(string path, string location, string fileName),
2. List<byte[]> FileRemovePadding(List<byte[]> input),
3. bool FileSizeCheck(IFormFile input),
4. bool KeySizeCheck(byte[] key),
5. bool IVCheck(byte[] IV),
6. byte[] byteArrayXor128(byte[] first, byte[] second),
7. byte[] CounterBlockCreator(),
8. byte[] CounterIncrement(byte[] counter),
9. byte[] GetLSBFrom128ByteArray(byte[] input,UInt16 range),
10. byte[] GetMSBFrom128ByteArray(byte[] input,UInt16 range),
11. byte[] byteArrayXorVariableSize(byte[] first, byte[] second, UInt16 size),
12. byte[] concatVariableSize(byte[] first, byte[] second),
13. byte[] FileOpenPaddedAndDelete(string location, string subSection, string filename),
14. byte[] FileOpenAndDelete(string location, string subSection, string filename).

FileReadAllBytesWithPadding(string path, string location, string fileName)

Tato funkce přečte soubor a pokud soubor nesplňuje vstupní rozměry tedy není o velikosti $x \cdot 16$ bytů tak je nejprve přidělen na konec znak 'f' tolikrát dokud není splněna podmínka pro rozměry souboru a následně je přidělen jeden nový kontrolní blok, který je ve formátu $\{\mathbf{fffffffffffff}xy\}$, kde x a y reprezentují kolik bylo přiděleno 'f' na konec souboru mimo kontrolní blok. Číslo x může být 0 nebo 1 a číslo y může být $0 - 9$.
Mezi vstupní parametry této funkce patří **string path, location, fileName**. Path reprezentuje cestu v uložišti k složce kde se nachází aplikace. location upřesňuje podsložku, kde se soubor nachází, typicky **ToEncrypt, ToDecrypt**. fileName reprezentuje jméno souboru.

FileRemovePadding(List<byte[]> input)

Tato funkce na již dešifrovaném souboru odstraní padding. Vstupní parametr List<byte[]> input reprezentuje celý dešifrovaný soubor.
Tato funkce se podívá na poslední blok a pokud je rozpoznán kontrolní blok viz kapitola 3.3, pokud je rozpoznán tak jej odstraní a odstraní i příslušný počet znaku 'f'.

FileSizeCheck(IFormFile input)

Tato funkce je volána v kontrolérech, aby bylo ověřeno, že vstupní soubor splňuje maximální velikost 40MB. Vrací TRUE, když je soubor o vhodných rozměrech, v opačném případě vrací FALSE.

KeySizeCheck(byte[] key)

Obdobně jako FileSizeCheck kontroluje rozměry šifrovacího klíče. Vrací TRUE při splnění podmínek, v opačném případě vrací FALSE.

IVCheck(byte[] IV)

Obdobně jako FileSizeCheck kontroluje rozměry inicializačního vektoru. Vrací TRUE při splnění podmínek, v opačném případě vrací FALSE.

byteArrayXor128(byte[] first, byte[] second)

Tato funkce provádí exkluzivní součet dvou polí byte[16].

CounterBlockCreator()

Tato funkce vytvoří startovní CounterBloky.

CounterIncrement(byte[] counter)

Přičítá jedničku ke CounterBloku.

GetLSBFrom128ByteArray(byte[] input,UInt16 range)

Tato funkce vrací **range** LSB bitů ze vstupního pole byte[16], tedy z **input**.

GetMSBFrom128ByteArray(byte[] input,UInt16 range)

Tato funkce vrací **range** MSB bitů ze vstupního pole byte[16], tedy z **input**.

byteArrayXorVariableSize(byte[] first, byte[] second, UInt16 size)

Tato funkce provede exkluzivní součet dvou byte[] o velikosti **size** a vrátí jejich výsledek.

concatVariableSize(byte[] first, byte[] second)

Tato funkce spojí dva byte[] a výsledný byte[] vrací.

FileOpenPaddedAndDelete(string location, string subSection, string filename)

Přečte soubor s polstrováním, smaže jej a vrátí data souboru v podobě byte[].

FileOpenAndDelete(string location, string subSection, string filename)

Přečte soubor bez polstrovaní, smaže jej a vrátí data souboru v podobě byte[].

3.4 Implementace šifer

3.4.1 AES

Pro šifru AES obdobně jako pro Twofish byla použita knihovna BouncyCastle dostupná z[6].
Tato třída obsahuje metodu:

1. Advanced_Encryption_Standard(byte[] key),
2. byte[] Encrypt(byte[] input),
3. byte[] Decrypt(byte[] input).

Advanced_Encryption_Standard(byte[] key)

Tato metoda je konstruktorem celé třídy. Tento konstruktor inicializuje šifrovací engine a uloží klíč.

Encrypt(byte[] input)

Tato funkce šifruje vstup a vrací zašifrovaný blok. Vstupní pole **input** reprezentuje jeden blok, tudíž musí být o velikosti 16, neboť 16 bytů je 128 bitů.

Decrypt(byte[] input)

Tato funkce dešifruje vstup a vrací dešifrovaný text. Stejně jako tomu je ve funkci Encrypt musí být input o délce 16, neboť input reprezentuje jeden blok.

3.4.2 Twofish

Pro šifru Twofish byla použita knihovna BouncyCastle dostupná z [6].
Twofish obsahuje 2 funkce, a to sice:

1. byte[] Encrypt(byte[] input, byte[] key),
2. byte[] Decrypt(byte[] input, byte[] key).

Encrypt(byte[] input, byte[] key)

Tato funkce šifruje vstup a vrací zašifrovaný blok.

Decrypt(byte[] input, byte[] key)

Tato funkce dešifruje vstup a vrací dešifrovaný blok.

3.5 Implementace operačních módů

Všechny operační módy jsou implementování ve třídě "Cipher_manipulator", tato třída obsahuje dodatečné funkce, které jsou používány během šifrování/-dešifrování. Jedná se o funkce:

1. byte[] combineSplitArray(List<byte[]> input),
2. byte[] combineSplitArrayVariableLength(List<byte[]> input, UInt16 length),
3. List<byte[]> splitByteArray(byte[] source),
4. List<byte[]> splitByteArrayVariableLength(byte[] source, UInt16 length),
5. byte[] resizeFromEnd128(byte[] input, UInt16 length),
6. byte[] resizeFromEndAny(byte[] input, UInt64 length),

7. List<byte[]> combineSplitArrayInto128(List<byte[]> input, UInt16 length),
8. byte[] ConvertToByteArrayFromBitArray(BitArray bitArray).

combineSplitArray(List<byte[]> input)

Tato funkce spojí vstupní list bloků do jednoho pole byte[].

combineSplitArrayVariableLength(List<byte[]> input, UInt16 length)

Tato funkce je určena pro operační mód CFB, KW a KWP. Funkce spojí rozdělený list bloků, které jsou o délce length $*$ 8 bitů.

splitByteArray(byte[] source)

Tato funkce je určena pro rozdělení vstupního pole na 128 bitové bloky.

splitByteArrayVariableLength(byte[] source, UInt16 length)

Tato funkce je určena pro operační mód CFB, KW a KWP. Funkce rozdělí vstupní pole na bloky o délce length $*$ 8 bitů.

resizeFromEnd128(byte[] input, UInt16 length)

Tato funkce ořízne blok o délce byte[16] na byte[16 − length] tak, že odebere 16 − length prvních členů pole.

resizeFromEndAny(byte[] input, UInt16 length)

Tato funkce ořízne pole o libovolné o délce byte[$input.length$] na byte[$input.length$− length] tak, že odebere $input.length$ − length od začátku pole.

combineSplitArrayInto128(List<byte[]> input, UInt16 length)

Tato funkce je určena pro operační mód CFB. Funkce spojí podbloky užívané módem CFB na standardní bloky o 128 bitech.

ConvertToByteArrayFromBitArray(BitArray bitArray)

Tato funkce konvertuje pole bitů zpět na pole bytů.

3.5.1 Základní operační módy

Všechny metody realizující základní operační módy splňuji standardní formátování a to sice ve tvaru "Encrypt_*MÓD*_*Šifra*(...)"pro šifrování a "Decrypt_*MÓD*_*Šifra*(...)"pro dešifrování.
Tedy například Encrypt_ECB_Twofish(...) je metoda realizující šifrování pomocí šifry Twofish za použití operačního módů ECB.
Funkce každého operačního módu má na svém vstupu cestu k souboru, klíč, délku klíče a jméno souboru. Další vstupní parametry se liší v závislosti na operačním módu.

3.5.2 Módy s MAC

Módy s MAC mimo CMAC, KW a KWP splňují standardní formátování jako tomu je u základních operačních módů.
Operační mód CMAC má následující formátování "CMAC_*Šifra*"a pro kontrolu MAC "CMAC_CHECK_*Šifra*".
Key wrapping má následující formátování "*Šifra*_Wrapping", "*Šifra*_Unwrapping"a KWP má následující formátování "*Šifra*_WrappingWithPadding"a "*Šifra*_UnWrappingWithPadding".

3.5.3 Omezení a stanovení velikostí vstupních parametrů

Pro potřeby implementace byly omezeny vstupní parametry, a to sice:

- Velikost vstupního souboru,
- Parametr s u operačního módu CFB.

Vstupní soubor musí být menší nebo roven velikosti 40MB, tato podmínka platí i pro asociovaná data viz například mód GCM.
U operačního módu CFB byl omezen parametr s na velikost: $32, 64, 128$ v aplikaci na $s = 4, 8, 12$. Více informací o parametru s se nachází v kapitole 1.2.3.
Velikost MAC byla omezena dle doporučení uvedených v příslušných dokumentacích. Povolené délky MAC jsou:

1. CMAC: libovolný nenulový násobek 8 bitů menší než 128. [2]
2. GCM: 96, 104, 112, 120, 128 bitů.[4]
3. CCM: 32, 48, 64, 80, 96, 112, 128 bitů.[3]

Délky klíčů odpovídají povoleným délkám klíčů pro jednotlivé šifry, tedy 128, 192, 256 bitů jak pro AES tak pro Twofish. Klíče se zadávají v aplikaci v ASCII symbolech.

Délky inicializačních vektorů jsou omezeny pro všechny operační módy. U všech operačních módů užívající inicializační vektor je mimo GCM je jich délka omezena na 16 ASCII symbolů (128 bitů). U operačního módů GCM je délka omezena vrcholem 512 bitů tedy 64 ASCII symbolů a minimum je více než 8 bytů, tedy platná délka je 2 - 64 ASCII symbolů.
Parametry pro operační mód CCM nejsou nijak omezeny mimo již stanovená omezení a omezení uvedená v kapitole 1.3.2.

3.6 Tvorba statistiky

Pro porovnání vlastní implementace operačních módů s implementací v systémové knihovně jazyku C# a pro zjištěni rychlosti jednotlivých módů byla vytvořena vytvořena třída **StatMaker** a pomocná třída **StopwatchForStats**. Třída **StopwatchForStats** pouze realizuje stopky pro měření.

3.6.1 StatMaker

V StatMakeru se nachází pouze jedna metoda, a to sice MakeStats().
Tato funkce, obdobně jako tomu bylo pro měření v semestrální práci, vygeneruje 200 souborů o velikostech 1 MB, 10 MB a 100 MB. Pro tyto velikosti byl měřen čas šifrování a dešifrování. V těchto časech byla také zohledněna doba přečtení souboru jeho šifrování/dešifrování a následně uložení jeho výstupu.
Pro měření byl stanoven jeden pevně daný klíč a pevně daný inicializační vektor. Klíč byl ve tvaru "bQeThVmYq3t6w9z$"a inicializační vektor byl ve tvaru "bQeThVmYq3t6w9za".
U operačního módu CFB byly naměřeny všechny možné varianty dle povolených parametrů v aplikaci, viz kapitola 3.5.3, to sice pro $s = 32, 64$ a 128. Více informací o parametru s v kapitole1.2.3.

3.6.2 Naměřené hodnoty

Každý ze základních módů má naměřeno 200 hodnot pro šifrování a dešifrování pro každou velikost vstupních souborů. Z těchto hodnot byl vypočítán průměr, modus a směrodatná odchylka. A byl realizován jednoduchý histogram rozložení naměřených hodnot.
Všechny měřené hodnoty byly naměřeny pouze pro šifru AES.
Měření probíhalo na zařízení s operačním systémem Windows 10, na zařízení nebyl spuštěn jakýkoliv jiný program mimo webovou aplikaci, a prohlížeč v

němž bylo pouze spuštěno měření. Zařízení bylo během měření připojeno k síti.
Všechny naměřené časy lze nalézt v příloze C.

ECB

Po měření byly vypočítány hledané hodnoty, ty jsou znázorněné v tabulce 3.1 a v tabulce 3.2.

Vstup [MB]	Průměr [ms]	Modus [ms]	Směrodatná odchylka [ms]
1	61,95	57	6,201
10	701,36	724	51,872
100	6802,06	6623	327,680

Tab. 3.1: Získané časové statistiky z šifrování ECB.

Vstup [MB]	Průměr [ms]	Modus [ms]	Směrodatná odchylka [ms]
1	92,11	73	27,043
10	776,77	735	69,914
100	7790,9	7799	291,743

Tab. 3.2: Získané časové statistiky z dešifrování ECB.

CBC

Po měření byly vypočítány hledané hodnoty, ty jsou znázorněné v tabulce 3.3 a v tabulce 3.4.

Vstup [MB]	Průměr [ms]	Modus [ms]	Směrodatná odchylka [ms]
1	75,95	72	5,539
10	920,14	965	93,514
100	9288,83	8862	507,450

Tab. 3.3: Získané časové statistiky z šifrování CBC.

CFB

Po měření byly vypočítány hledané hodnoty, ty jsou znázorněny v příslušných tabulkách pro jednotlivé velikosti parametru $s = 32, 64, 128$.

Vstup [MB]	Průměr [ms]	Modus [ms]	Směrodatná odchylka [ms]
1	97,14	83	24,077
10	870,84	762	126,941
100	9480,88	8980	575,438

Tab. 3.4: Získané časové statistiky z dešifrování CBC.

Pro $s = 32$ jsou hodnoty znázorněné v tabulce 3.5 a 3.6. Pro $s = 64$ jsou hodnoty znázorněné v tabulce 3.7 a 3.8. Pro $s = 128$ jsou hodnoty znázorněné v tabulce 3.9 a 3.10.

Vstup [MB]	Průměr [ms]	Modus [ms]	Směrodatná odchylka [ms]
1	156,5	153	7,721
10	1700,19	1697	47,8
100	16784,74	16842	237,25

Tab. 3.5: Získané časové statistiky z šifrování CFB, pro $s = 32$.

Vstup [MB]	Průměr [ms]	Modus [ms]	Směrodatná odchylka [ms]
1	203,01	198	28,945
10	1883,77	1856	99,114
100	17403,93	17379	227,645

Tab. 3.6: Získané časové statistiky z dešifrování CFB, pro $s = 32$.

Vstup [MB]	Průměr [ms]	Modus [ms]	Směrodatná odchylka [ms]
1	301,28	228	12,722
10	3339,84	3311	78,258
100	34797,48	34512	838,735

Tab. 3.7: Získané časové statistiky z šifrování CFB, pro $s = 64$.

Vstup [MB]	Průměr [ms]	Modus [ms]	Směrodatná odchylka [ms]
1	432,06	419	39,973
10	3515,82	3554	82,574
100	36218,63	35469	817,085

Tab. 3.8: Získané časové statistiky z dešifrování CFB, pro $s = 64$.

Vstup [MB]	Průměr [ms]	Modus [ms]	Směrodatná odchylka [ms]
1	83,68	79	5,930
10	907,7	866	51,566
100	8801,18	8848	108,568

Tab. 3.9: Získané časové statistiky z šifrování CFB, pro $s = 128$.

Vstup [MB]	Průměr [ms]	Modus [ms]	Směrodatná odchylka [ms]
1	100,75	90	18,545
10	918,86	948	58,772
100	8677,51	8734	130,086

Tab. 3.10: Získané časové statistiky z dešifrování CFB, pro $s = 128$.

OFB

Po měření byly vypočítány hledané hodnoty, ty jsou znázorněné v tabulce 3.11 a v tabulce 3.12.

Vstup [MB]	Průměr [ms]	Modus [ms]	Směrodatná odchylka [ms]
1	66,1	61	6,231
10	749,48	786	53,911
100	7187,98	7084	140,591

Tab. 3.11: Získané časové statistiky z šifrování OFB.

Vstup [MB]	Průměr [ms]	Modus [ms]	Směrodatná odchylka [ms]
1	83,07	66	22,935
10	710,81	640	77,639
100	7304,7	7235	160,127

Tab. 3.12: Získané časové statistiky z dešifrování OFB.

CTR

Po měření byly vypočítány hledané hodnoty, ty jsou znázorněné v tabulce 3.13 a v tabulce 3.14.

Vstup [MB]	Průměr [ms]	Modus [ms]	Směrodatná odchylka [ms]
1	75,95	72	5,539
10	920,14	965	93,514
100	9288,83	8862	507,450

Tab. 3.13: Získané časové statistiky z šifrování CTR.

Vstup [MB]	Průměr [ms]	Modus [ms]	Směrodatná odchylka [ms]
1	97,14	83	24,077
10	870,835	762	126,941
100	9480,88	8980	575,437

Tab. 3.14: Získané časové statistiky z dešifrování CTR.

3.7 Závěr ze statistik

Jako nejrychlejší operační mód se pro velikost vstupního souboru 1MB ukázal mód ECB, pro 10MB s ukázal jako nejrychlejší ECB, pro 100MB byl nejrychlejší také ECB.
Ukázalo se, že implementace těchto operačních módů v systémové knihovně jazyku C# je značně rychlejší, než vlastní implementace použita ve webové aplikaci.

3.7.1 Zjištění u operačního módu CFB

U operačního módu CFB došlo ke zjištění že časy naměřené pro $s = 32$ byly značně rychlejší než $s = 64$. Tyto rozdíly naznačují tabulky 3.15, 3.16, 3.17, 3.18 a 3.19.

Vstup [mb]	Průměr pro $s = 32$ [ms]	Průměr pro $s = 64$ [ms]	Průměr pro $s = 128$ [ms]
1	156,5	301,28	83,68
10	1700,19	3339,84	907,7
100	16784,74	34797,48	8801,18

Tab. 3.15: Porovnání naměřených časů při šifrování u módu CFB.

Vstup [mb]	Průměr pro $s = 32$ [ms]	Průměr pro $s = 64$ [ms]	Průměr pro $s = 128$ [ms]
1	203,01	432,06	100,75
10	1883,77	3515,82	918,86
100	17403,93	36218,63	8677,51

Tab. 3.16: Porovnání naměřených časů při dešifrování u módu CFB.

Vstup [mb]	Poměr vůči $s = 128$ pro $s = 32$	Poměr vůči $s = 128$ pro $s = 64$
1	1,870	3,600
10	1,873	3,679
100	1,907	3,953

Tab. 3.17: Porovnání naměřených časů v poměrech, při šifrování u módu CFB.

Vstup [mb]	Poměr vůči $s = 128$ pro $s = 32$	Poměr vůči $s = 128$ pro $s = 64$
1	2,014	4,288
10	2,050	3,826
100	2,005	4,174

Tab. 3.18: Porovnání naměřených časů v poměrech, při dešifrování u módu CFB.

	Průměrný poměr vůči $s = 128$ pro $s = 32$	Průměrný poměr vůči $s = 128$ pro $s = 64$
Šifrování	2,014	3,745
Dšifrování	2,023	4,096

Tab. 3.19: Celkový průměrný poměr rychlostí vůči $s = 128$.

Tyto hodnoty ukazují, že pro $s = 32$, kde se používá pouze čtvrtina bloku pro získání šifrovaného/dešifrovaného bloku je rychlost v průměru dvakrát

pomalejší než když je použit celý blok jako tomu je u $s = 128$. Naproti tomu při $s = 64$, kdy je použita polovina bloku, měřené hodnoty ukázaly, že je průměrně 3,92 krát pomalejší.

Tento rozdíl je pravděpodobně způsoben tím, že náročnost početních operací do určité míry zvýhodňuje menší parametr s. Avšak zároveň menší velikost parametru s také způsobuje, že se úměrně zvyšuje počet nutně proveditelných operací. To naznačuje, že pro $s = 32$ je změna rychlosti těchto operací natolik zásadní, až je navýšení počtu operací méně zásadním faktorem.

To tedy znamená, že při použiti $s = 64$ je zrychlení operací nedostatečně zásadním faktorem a množství nutných operací (dvojnásobné) je dominantním faktorem.

Závěr

V bakalářské práci byla realizována webová aplikace, jejíž koncepční návrh byl realizován v semestrální práci.

Tato aplikace realizuje 10 operačních módů, jmenovitě ECB, CBC, CFB, OFB, CTR, CMAC, CCM, GCM, KW a KWP. Každý z těchto módů je realizován jak pro šifru AES tak i pro Twofish. Webová aplikace také obsahuje technickou dokumentaci, která vysvětluje jak každý operační mód funguje.

Bylo provedeno měření rychlosti základních operačních módů implementovaných ve webové aplikaci, tedy ECB, CBC, CFB, OFB a CTR. Toto měření bylo provedeno pro zjištění rychlosti těchto módů a pro porovnání vlastní implementace těchto módů vůči implementaci v systémové knihově programovacího jazyku C#.

Literatura

[1] National Insitute of Standards and Technology: *Recommendation for Block Cipher Modes of Operation*[online].NIST Special Publication 800-38A. NIST: 12/2001 [cit. 18. 11. 2021]. Dostupné z URL: <https://csrc.nist.gov/publications/detail/sp/800-38a/final>

[2] National Insitute of Standards and Technology: *Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication*[online].NIST Special Publication 800-38B. NIST: 5/2005. Poslední aktualizace 6. 10. 2016 [cit. 18. 11. 2021]. Dostupné z URL: <https://csrc.nist.gov/publications/detail/sp/800-38b/final>

[3] National Insitute of Standards and Technology: *Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality*[online].NIST Special Publication 800-38C. NIST: 5/2004. Poslední aktualizace 20. 7. 2007 [cit. 18. 11. 2021]. Dostupné z URL: <https://csrc.nist.gov/publications/detail/sp/800-38c/final>

[4] National Insitute of Standards and Technology: *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*[online].NIST Special Publication 800-38D. NIST: 11/2007.[cit. 28. 11. 2021]. Dostupné z URL: <https://csrc.nist.gov/publications/detail/sp/800-38d/final>

[5] National Insitute of Standards and Technology: *Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping*[online].NIST Special Publication 800-38F. NIST: 12/2012.[cit. 30. 11. 2021]. Dostupné z URL: <https://csrc.nist.gov/publications/detail/sp/800-38f/final>

[6] The Legion of the Bouncy Castle: *The Bouncy Castle Crypto Package For C Sharp*[online]Poslední aktualizace 6. 4. 2022 [cit. 18. 4. 2022]. Dostupné z URL: <https://www.bouncycastle.org/csharp/index.html>

Seznam symbolů a zkratek

P	otevřený text
C	šifrovaný text
K	šifrovací klíč K
$\mathbf{E}(p_i, K)$	šifrování textu p_i klíčem K
$\mathbf{D}(c_i, K)$	dešifrování textu c_i klíčem K
I	mezi výpočtový vstupní text
$\mathbf{MSB}_s$	s nejdůležitějších bitů
$\mathbf{LSB}_s$	s nejméně důležitých bitů
IV	inicializační vektor
o_i	i-tý mezivýpočtový blok
T	vnitřní proměnná CCM, reprezentující MAC
$A\|\|B$	zřetězení A a B, neboli spojení dvou bitových proudů A a B
$\lceil A/B \rceil$	horní celá část podílu čísel A a B
$\mathbf{len}(A)$	délka A v bitech
Ctr_i	i-tý counter blok
ICB	počáteční counterový blok
$A \oplus B$	XOR A a B, neboli exkluzivní součet A a B

Seznam příloh

A Statistiky měření

Statistika se skládá ze souboru *Statistika*, v tomto souboru se nachází pět podsložek *CBC*, *CFB*, *CTR*, *ECB* a *OFB*.
Každá z těchto podložek obsahuje statistiku ve formě souboru ve formátu *.xlsx* a podsložky *Šifrování* a *Dešifrování*, v každé z těchto složek je možné nalézt jednotlivé textové dokumenty, které jsou výstupem jednotlivé sady. Soubory jsou pojmenované takovýmto způsobem *Statistika_enc/dec_číslo sady_velikost vstupních dat_operační mód.txt*, *enc* je zkratka pro šifrování, *dec* je zkratka pro dešifrování. Příklad názvu souboru: *Statistika_dec_4_100_ECB.txt*, tedy jedná se o statistiku dešifrování čtvrté sady se vstupem o velikosti 100 MB v operačním módu ECB. Statistiky ve formátu *.xlsx* obsahují šest listů, každý list je pojmenovaný ve formátu *Statistika_enc/dec_velikost vstupních dat_operační mód*. Každý list obsahuje data relevantní pro daný list, tj. data všech sad pro danou velikost vstupu, a 7 grafů, z nichž 6 tvoří grafy každé sady a sedmý je graf průměrných hodnot.

B Webová aplikace

Webová aplikace je odevzdána ve standardním Visual Studiu formáu. Tedy primární složka pro otevření kódu samotné aplikace je soubor "Bakalář.sln", který je solution celé webové aplikace.

C Statistiky měření u webové aplikace

Statistika měření se skládá z 5 podsložek, každá je pro vlastní operační mód. Soubory v těchto složkách jsou formátovány následujícím způsobem: "Recorded_*velikost v MB*_šifrování / dešifrování.txt", šifrováni se značí ENC a dešifrování DEC. Dále se tam nachází příslušný .xlxs soubor kde jsou zpracovaná data s grafy.