VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

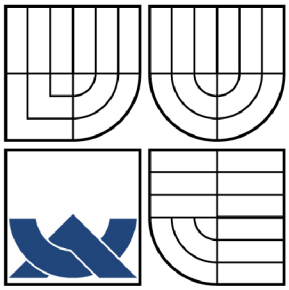
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

SIMULACE A ANALÝZA PROVOZU BLOKOVÉ ŠIFRY SE STATISTICKOU SAMOSYNCHRONIZACÍ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

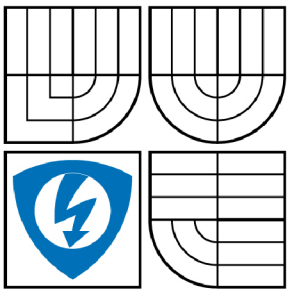
AUTOR PRÁCE Bc. MAREK KOPČAN
AUTHOR

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

SIMULACE A ANALÝZA PROVOZU BLOKOVÉ ŠIFRY SE STATISTICKOU SAMOSYNCHRONIZACÍ

SIMULATION AND ANALYSIS OF THE BLOCK CIPHER MODE WITH STATISTICAL SELF-SYNCHRONIZATION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. MAREK KOPČAN

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. KAREL BURDA, CSc.

SUPERVISOR

BRNO 2008

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ
Fakulta elektrotechniky
a komunikačních technologií
Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Kopčan Marek, Bc. ***ID:*** 88460
Ročník: 2 ***Akademický rok:*** 2007/08

NÁZEV TÉMATU:

Simulace a analýza provozu blokové šifry se statistickou samosynchronizací

POKYNY PRO VYPRACOVÁNÍ:

V rámci práce popište problematiku samosynchronizujících provozů blokových šifer a problematiku simulace náhodných dějů v přenosových kanálech. Na tomto základě navrhněte simulační model stanoveného samosynchronizujícího provozu blokové šifry. Model musí umožnit analýzu vlivu přenosových chyb na výstupní chybovost šifry a analýzu doby resynchronizace šifry na přenosový slip. Model prakticky ověřte pro vybraný typ blokové šifry. Získané výsledky prezentujte a interpretujte.

DOPORUČENÁ LITERATURA:

[1] STALLINGS, W.: Cryptography and Network Security. Prentice Hall, Upper Saddle River 2006.
[2] BURDA, K. Kryptologická a bezpečnostní analýza synchronizace blokových šifer pro MKS v reálných telekomunikačních systémech. Národní bezpečnostní úřad, Praha 2004.

Termín zadání: 11.2.2008 ***Termín odevzdání:*** 28.5.2008

Vedoucí projektu: doc. Ing. Karel Burda, CSc.

prof. Ing. Kamil Vrba, CSc.
předseda oborové rady

UPOZORNĚNÍ:

Prehlásenie

Prehlasujem, že svoju diplomovú prácu na tému "Simulácia a analýza prevádzky blokovej šifry so štatistickou samosynchronizáciou" som vypracoval samostatne pod vedením vedúceho diplomovej práce a s použitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, pričom som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a som si plne vedomý následkov porušenia ustanovení §11 a nasledujúcich autorského zákona č. 121/2000 Sb., vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovení §152 trestného zákona č. 140/1961 Sb.

V Brne dňa 23. 5. 2008

Poďakovanie

Touto cestou by som sa chcel poďakovať vedúcemu diplomovej práce, doc. Ing. Karlovi Burdovi, CSc. za pedagogické usmernenie, za podnetné rady a konzultácie, za trpezlivosť a podporu a v neposlednom rade za vytvorenie podmienok pracovného prostredia pre vznik mojej práce.

V Brne dňa 23. 5. 2008

Abstrakt

V súčasnej dobe narastá dôležitosť kryptografie exponenciálnym tempom. V dobe moderných technológii, kde sú informácie tou najcennejšou hodnotou, je žiadúce chrániť túto hodnotu. Informácie však potrebujeme aj vymieňať medzi sebou, ale dodržať pritom utajenie. Aby bolo možné preniesť informáciu v utajenom tvare, treba použiť špeciálny mód. Na prenosovej ceste sa však vyskytujú poruchy, proti ktorým nie je každý mód odolný. Preto sa vyvinuli módy, ktorých úlohou je čo najlepšie ochrániť šifrovanú informáciu pri prenose.

Táto práca sa zaoberá problematikou samosynchronizačných módov blokových šifier. Ide o ochranu prenášanej informácie proti rôznym druhom porúch počas prenosu informácie prenosovým kanálom. V práci budú vyšetrované dva druhy samosynchronizačných módov - $OCFB$ (Optimized Cipher FeedBack) a $SCFB$ (Statistical Cipher FeedBack). Oba tieto módy majú svoje výhody a nevýhody. Cieľom práce je analyzovať tieto módy a vytvoriť simulačný model, ktorý pomôže zlepšiť ich ďalší výskum.

Kľúčové slová

blokové šifry, Advanced Encryption System, samosynchronizačné módy, OCFB, Optimized Cipher FeedBack, SCFB, Statistical Cipher FeedBack, simulačný model, metóda Monte–Carlo

Abstract

There is a enormous rise in importance of cryptography. In age of hi-technologies, where information are the most valuable asset, is need to protect this value. But we need to transport information between us and keep information confidental. In this case we use special modes of block cipher because of defect in communication canal. Not all modes are able to deal with this problem. For this purpose, there are special modes.

This work deal with self-synchronization modes of block cipher. It is protection of tranfered information in communication canal against different types of defects. We will exam two self-synchronization modes - $OCFB$ (Optimized Cipher FeedBack) and $SCFB$ (Statistical Cipher FeedBack). Both have their advantages and disadvantages. The goal of this work is to provide analyse of both modes and to create simulation model. This model should help with further research of self-synchronization modes.

Keywords

block cipher, Advanced Encryption System, self-synchronization modes, OCFB, Optimized Cipher FeedBack, SCFB, Statistical Cipher FeedBack, simulation model, Monte–Carlo method

Obsah

Zoznam obrázkov

Úvod

Kryptografia je v dnešnej dobe veľmi dôležitou súčasťou každodenného života. Možno si to väčšina z nás ani neuvedomuje, ale prichádza s ňou do styku takmer neustále. Napríklad výber z bankomatu alebo platba platobnou kartou. Tieto jednoduché operácie sprevádzajú náš každodenný život a využívajú práve kryptografiu. Kryptografia sa nepoužíva iba u finančných operácii, ale pre moderného človeka i u úplne bežných vecí, ako je prihlasovanie na počítač pod heslom (heslo je v počítači uložené v zašifrovanej podobe), posielaní emailu (email môže byť poslaný v zašifrovanej podobe) alebo používaní čipových kariet v knižniciach, školách, mestskej doprave (informácie uložené na karte sú často chránené proti zneužitiu práve kryptografiou).

Kryptoanalýza je opak kryptografie. Jej hlavným cieľom je analýza odolnosti kryptografických systémov a hľadanie metódy vedúcej k preniknutiu do týchto systémov. Hromadné použitie kryptografie prišlo s masovým rozšírením počítačov a následne sprístupnenia Internetu širokej verejnosti.

Odolnosť šifier proti útoku je najdôležitejšou vlastnosťou šifrovacieho systému. Ďalšou významnou vlastnosťou šifrovacieho systému je odolnosť voči chybám, pretože šifrový text je veľmi citlivý na akýkoľvek druh chýb. Tieto chyby spôsobujú vo výsledku nemožnosť správne dešifrovať blok šifrového textu v prípade blokových šifier, v závislosti na použitom móde to môže byť aj celý zvyšok správy.

Rozlišujeme rôzne typy chýb. Bitová chyba je chyba, ktorá nastane invertovaním bitu, to znamená zmenou hodnoty z 0 na 1 alebo naopak. Ďalším typom chýb sú chyby, keď sa počas prenosu prenosovým kanálom stratí niektorý bit, alebo naopak sa niektorý bit zduplikuje. Táto porucha sa anglicky nazýva slip a tento termín bude použitý aj ďalej. Slipy vznikajú z dôvodu rozdielnosti časových základní vysielača a prijímača, respektíve môže tento druh chyby vzniknúť pri prenose dlhej postupnosti bitov rovnakej hodnoty. Slip sa vyskytuje vždy v rozsahu základnej prenosovej jednotky (bit, byte). Automatickou elimináciou uvedeného javu umožňujú šifrové prevádzky so samosynchronizáciou. V súčasnej dobe sú publikované tri druhy prevádzky blokovej šifry tohto typu - prevádzka CFB, $OCFB$ a $SCFB$.

V rámci projektu bude popísaná problematika blokových šifier a ich prevádzkových režimov odolných voči slipom a bude popísaný súčasný stav problematiky. V ďalšej časti bude popísaná simulácia náhodných porúch a zvolené programové prostriedky pre simuláciu.

Kapitola 1

Blokové šifry

Blokové šifry sa radia medzi šifry symetrické, teda šifry, v ktorých sa používa rovnaký kľúč pre šifrovanie i dešifrovanie. Tieto šifry pracujú s pevne stanoveným počtom bitov, ktorý sa nazýva blok a s nemennou transformáciou. Pri šifrovaní sa vezme napríklad ako vstupný súbor 128 bitov otvoreného textu a ako výstup je 128 bitov šifrovaného textu. Pre výstup je rozhodujúci druhý vstup - tajný kľúč. Dešifrovanie je podobné, dešifrovací algoritmus vezme na vstupe 128 bitový blok šifrového textu a pomocou tajného kľúča získa pôvodný otvorený text. Pre šifrovanie správ, ktoré sú dlhšie ako je dĺžka bloku, sa musí správa algoritmicky upraviť.

1.1 Vlastnosti blokových šifier

Blokové šifry majú charakteristické vlastnosti, ktoré umožňujú ich použitie v mnohých aplikáciách. Medzi vlastnosti blokových šifier patria:

Veľkosť kľúča - efektívna bitová dĺžka kľúča definuje hornú hranicu bezpečnosti danej šifry (na tom závisí rýchlosť útoku hrubou silou). Dlhšia dĺžka kľúča typicky znamená nejakú nadbytočnú cenu, ako je zložitejšie generovanie, prenos a uloženie.

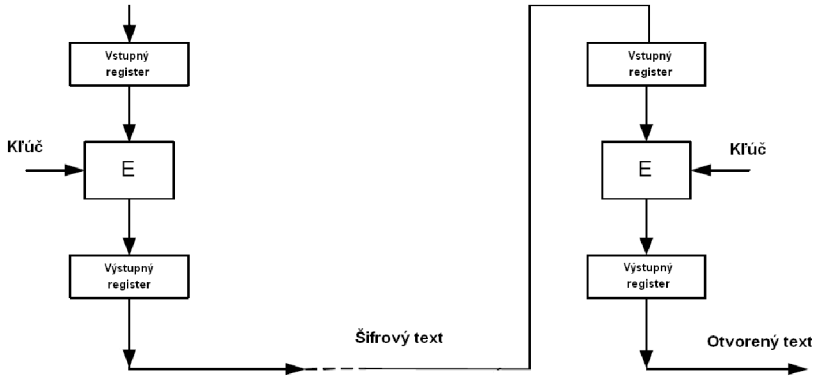
Rýchlosť šifrovania - rýchlosť šifrovania závisí na zložitosti šifrovacieho algoritmu, zvoleného druhu šifry a tiež na móde blokovej šifry.

Veľkosť bloku - veľkosť bloku ovplyvňuje ako bezpečnosť (väčší je lepší), tak zložitosť riešenia (väčší blok je náročnejší na implementáciu).

Zložitosť kryptografického zobrazenia - zložitosť algoritmu ovplyvňuje cenu implementácie ako z hľadiska vývoja a zdrojov, tak i z hľadiska rýchlosti a výkonnosti samotného šifrovania v reálnom čase.

Expanzia dát - všeobecne je vhodné, a v niektorých prípadoch dokonca nutné, aby sa pri šifrovaní nezväčšovalo množstvo dát (aby neprebiehala expanzia dát). Napriek tomu niektoré techniky majú za výsledok expanziu dát.

Propagácia chyby - dešifrovanie šifrového textu obsahujúceho chybu v jedinom bite môže vyústiť v rozdielny efekt na výsledný text, podľa toho, aký algoritmus bol použitý. Propagáciu chyby ovplyvňuje napríklad dĺžka bloku alebo použitý mód.



Obrázok 1.1: Základný mód blokových šifier – *ECB*

Blokové šifry patria medzi symetrické šifry, čo znamená že používajú jeden a ten istý kľúč pre šifrovanie a dešifrovanie, z čoho vyplývajú ako isté výhody, tak aj niektoré obmedzenia.

Výhody symetrickej kryptografie:

1. Symetrické šifry môžu byť navrhnuté pre rýchlejšie šifrovanie dát, u niektorých hardwarových riešení ide o stovky megabitov za sekundu
2. Kľúče pre symetrické šifry sú relatívne krátke pri zachovaní dostatočnej bezpečnosti
3. Symetrické šifry môžu byť použité ako základ pre ďalšie rôzne kryptografické mechanizmy, ako sú napríklad generátory pseudonáhodných čísel, hashovacie funkcie a podobne
4. Symetrické šifry môžu byť kombinované a skladané do seba, čím sa zvyšuje zabezpečenie šifrového textu

Nevýhody symetrickej kryptografie:

1. Pri komunikácii dvoch entít musí zostať kľúč tajomstvom na oboch stranách
2. Vo veľkej sieti je treba mať uložené veľké množstvo párov kľúčov
3. Pri komunikácii medzi dvomi entitami je konvenciou meniť často kľúče, optimálne je mať nový kľúč pre každú komunikačnú reláciu, čo môže byť komplikované, pretože prenos kľúča vyžaduje zabezpečený kanál

1.2 IV - Inicializačný vektor

IV je blok bitov, ktorý je požadovaný pri prúdových alebo blokových šifrách na vykonanie niektorých módov. Je nutný pri módoch ako *CBC* alebo *CFB*, kde sa používa ako "nultý" blok šifrovaného textu.

Dĺžka *IV* závisí na šifrovacom algoritme a veľkosti bloku tohto algoritmu. Hodnotu *IV* musí

poznať aj príjemca šifrovanej správy, aby mohol správu dešifrovať (je možné sa naň pozerať ako na ďalší kľúč). Je množstvo spôsobov ustanovenia IV, napríklad výpočtom určitej hodnoty (zvyčajne inkrementáciou), alebo meraním niektorých parametrov, ako je napríklad aktuálny čas, ID odosielateľa a/alebo príjemcu. Môže byť kombinovaných viacero spôsobov a/alebo sa môže hashovať, záleží na algoritme. Ak je IV ustanovený náhodne, šifrátor musí brať do úvahy aj možne kolízie.

V operačných módoch, kde sa inicializačný vektor používa, je tiež možné využívať metódu solenia IV. Spočíva v tom, že komunikujúcej strane sa síce predáva hodnota IV, ale k šifrovaniu sa použije iná hodnota IV' (tzv. "osolený IV"), ktorá sa na oboch stranách vypočíta z IV a kľúča K nejakým definovaným spôsobom. Napríklad to môže byť hashovacia hodnota, vypočítaná zo zreťazenia hodnôt. Bezpečnostnou výhodou je, že skutočná použitá hodnota IV' sa nikde neobjavuje na komunikačnom kanáli. Metódou solenia sa zaoberá norma PKCS#5.

Kapitola 2

Módy blokových šifier

Pomocou rôznych módov blokových šifier môžeme dosiahnuť niektoré zaujímavé vlastnosti, ako je napríklad lepšia odolnosť voči chybám, efektívnejšia ochrana proti slipom a zlepšenie ďalších vlastností. Základné módy sú:

ECB *(Electronic code book)* - v tomto móde sú rovnaké bloky otvoreného textu transformované do rovnakých blokov šifrového textu. Tento mód je pre blokové šifry natívny, je najjednoduchší a aj najmenej bezpečný. Výhodou je, že vzniknutá chyba spôsobí, že blok v ktorom sa chyba vyskytla, nie je možné dešifrovať. Toto sa však týka iba tohto jediného bloku.

CBC *(Cipher block chaining)* - na základe použitia reťazenia blokov, kde každý blok závisí aj na predchádzajúcom bloku šifry. Dosahuje sa to aplikovaním operácie XOR na predchádzajúci a aktuálny blok a následným šifrovaním. Tým dosiahneme zlepšenie vlastností, nakoľko rovnaké bloky otvoreného textu budú transformované do rôznych blokov šifrového textu. Nevýhodou je propagácia chyby, ktorá na rozdiel od módu *ECB*, znemožňuje dešifrovať kvôli reťazeniu celý zvyšok správy.

CTR *(Counter mode)* - využíva čítač, ktorý sa aktualizuje pri každom šifrovaní bloku, najčastejšie pričítaním jedničky. Cieľom je zaručiť maximálnu periodicitu hesla. Oproti módu *CBC* tu nie je žiadna spätná väzba, teda nie je nutné poznať predchádzajúci blok.

CFB *(Cipher Feedback)* a OFB *(Output Feedback)* - tieto módy budú podrobne popísané ďalej.

Použitie niektorého zo základných módov *ECB*, *CBC*, CFB alebo OFB je vhodné takmer pre každú aplikáciu, naproti tomu väčšina rôznych zložitých módov môže priniesť minimálne zvýšenie bezpečnosti oproti zvýšeniu komplexnosti alebo výpočtovej náročnosti základného algoritmu. Existuje i celá rada ďalších módov, ktoré sú viac či menej obdobou doteraz uvedených módov alebo ich kombináciou.

Módy *ECB*, *CBC* a CTR sú základné módy všeobecne dobre známe a nebudú popisované. Z pohľadu tejto práce sú dôležité módy CFB, OFB, $OCFB$ a $SCFB$, pretože sú to módy ktoré umožňujú blokovej šifre získať vlastnosť samosynchronizácie, teda vlastnosť ktorá je charakteristická pre prúdové šifry. Preto sa rôznymi úpravami základných módov dospelo k vyvinutiu niektorých módov, ktoré samosynchronizáciu umožňujú.

Vlastnosti samosynchrónnych šifier:

1. Samočinná synchronizácia - samočinná synchronizácia je možná, i keď sú niektoré časti šifrového textu zmazané alebo nejaké nové vložené, pretože dešifrovanie závisí na fixnom počte predchádzajúcich znakov šifrového textu. Takže šifra je schopná znovu zaviesť správne dešifrovanie automaticky po strate synchronizácie a iba s fixným počtom nedešifrovaných znakov správy.

2. Obmedzená propagácia chyby - vieme že stav samosynchrónnej šifry závisí na n predchádzajúcich znakoch šifrového textu. Ak je jeden znak šifrového textu zmenený (zmazaný alebo vložený nový) počas prenosu, potom dešifrovanie až n ďalších znakov môže byť nesprávne, ale potom už nasleduje správne dešifrovanie správy.

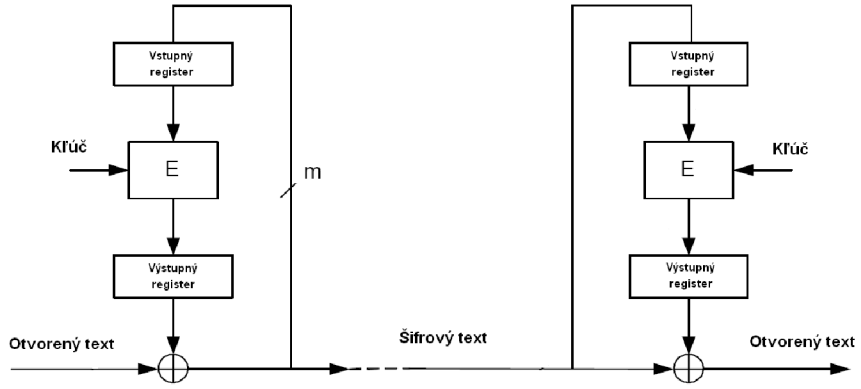
3. Difúzia štatistík vstupného textu - nakoľko každý znak šifrového textu ovplyvní celý nasledujúci šifrový text (každý šifrový znak je použitý pre generovanie nasledujúceho bloku hesla), tak štatistické vlastnosti vstupného textu sú rozprestreté do celého šifrového textu. Takže samosynchronizujúce šifry sú odolnejšie proti útokom založeným na redundancii vstupného textu než synchrónne šifry.

Aj keď sa zdá, že operačných módov je dosť, nie je tomu tak. Nové potreby vyžadujú nové módy, príkladom nech je nutnosť šifrovať dáta a súčasne počítať ich zabezpečovací kód *MAC* (súčasné použitie módu *CBC* a *MAC*), čo môže byť na niektorých výpočtovo obmedzených zariadeniach pomalé alebo príliš náročné na systémové prostriedky, hlavne na pamäť. Každý mód má však svoje špecifiká a odstraňuje špecifické nedostatky, preto National Institute of Standards and Technology (NIST) vyhlásil iniciatívu na tvorbu módov [1].

Všetky uvedené operačné módy definujú normy FIPS PUB 81, ANSI X3.106, ISO 8732 a ISO/IEC 10116. V ďalšej časti budú popísané iba módy, ktoré majú schopnosť obnoviť stratu synchronizácie, teda sú odolné voči slipu. Sú to módy CFB, OFB, $OCFB$ a $SCFB$.

2.1 CFB (Cipher FeedBack mode)

V tomto móde sa bloková šifra chová ako prúdová šifra s vlastnou synchronizáciou. Ako už bolo zmienené, niektoré aplikácie (hlavne sieťové), aby mohla byť daná šifra použitá na bloky menšej veľkosti než je veľkosť používaná v algoritme šifrovania. Ako príklad si môžeme vziať napríklad terminál - ten by mal byť schopný predať serveru každý vložený znak (avšak pri dĺžke bloku šifry 64 bitov by bolo treba najskôr zadať celkom 8 znakov, než by mohli byť tieto zašifrované a odoslané serveru). V niektorých prípadoch dokonca komunikačná cesta nemusí dosahovať šírku, po ktorej sa majú zašifrované dáta prenášať, veľkosti bloku šifry. Taký spôsob šifrovania nesplňuje ani základný *ECB* mód šifry, ani mód *CBC*8.



Obrázok 2.1: Šifrovanie v móde CFB

Práve tento problém rieši CFB. Princíp šifrovania a dešifrovania 8-bitového CFB módu pre 64-bitovú blokovú šifru je ukázaný na obr. 2.1. Na začiatku musí byť posuvný register naplnený IV - veľkosť registru odpovedá veľkosti bloku šifry. Pomocou štandardného šifrovacieho mechanizmu (pre danú šifru) zašifrujeme obsah posuvného registra a z výsledku si vezmeme najľavejších tj. najviac významných 8 bitov (vo všeobecnom n-bitovom CFB by sme vzali n-bitov). Na prvých 8 bitov otvoreného textu a túto hodnotu potom aplikujeme operáciu XOR, čím získame prvých 8 bitov šifrového textu. Obsah posuvného registru posunieme o 8 bit (vo všeobecnom prípade o n-bitov) doľava. Príslušných 8 bitov šifrovaného textu potom vložíme na miesto najnižšieho bytu v posuvnom registri – tj. vlastne ako keby nasunieme tieto bity sprava do posuvného registru (týmto spôsobom je zaistená spätná väzba). Obsah posuvného registru opäť zašifrujeme a vyberieme najľavejší byte, vykonáme XOR s ďalšími 8 bitmi otvoreného textu atď.

Dešifrovanie je obdobné. Rovnakým spôsobom ako pri šifrovaní získame z IV prvý podkľúč k (8 najľavejších bytov zašifrovaného obsahu posuvného registru). Ten potom použijeme pre XOR s prvými 8 bitmi šifrového textu, čím získame pôvodný byte otvoreného textu. Pritom každú jednotku (tu 8 bitov) prijatého šifrového textu nasúvame do posuvného registru rovnakým spôsobom ako pri šifrovaní. Týmto spôsobom pokračujeme v dešifrovaní prichádzajúcej postupnosť jednotiek šifrového textu.

Všeobecné n-bitové CFB je tiež možné graficky vyjadri ako na obr. 2.1. Tomu odpovedá matematický zápis:

$ST_0 = IV$,
$ST_i = OT_i \oplus E_K(ST_i - 1), i = 1,2,...,n.$
Predpis pre odšifrovanie v móde CFB:
$ST_0 = IV$,
$OT_i = ST_i \oplus E_K(ST_i - 1), i = 1,2,...,n.$

kde OT_i je $i-$tý blok otvoreného textu a ST_i je $i-$tý blok šifrového textu.

Dôležitým požiadavkom je, aby obidve strany (odosielateľ i príjemca) mali správne inicializovaný posuvný register (jeden spôsob môže byť taký, že ako prvých N bitov kde N je dĺžka bloku šifry, sa vyšle IV, a potom až nasleduje samotný obsah správy. IV tiež môže byť predaný cez inú komunikačnú cestu). Rovnako ako u CBC nemusí byť IV tajný (pretože bez znalosti kľúča nie je samotné IV príliš platné). Napriek tomu je z hľadiska bezpečnosti vhodné, aby každá správa používajúca rovnaký kľúč bola zašifrovaná použitím iného IV. Tu by som tiež rád upozornil že neznalosť IV pri znalosti kľúča spôsobí nesprávne dešifrovanie iba prvého bloku šifrového textu, ostatné bloky už budú dešifrované správne. Týmto sa poukazuje na to, že v tomto prípade nie je možné sa na IV pozerať ako na "ďalší" kľúč. U CFB závisí výsledok šifrovania danej jednotky otvoreného textu na predchádzajúcich častiach otvoreného textu.

Prevádzka CFB sa podľa veľkosti použitých bytov označuje ako $h-CFB$ prevádzka, pričom v praxi sa najčastejšie používa varianta $h = 1,8$ alebo n. Prevádzka CFB je odolná voči slipom, ktorých dĺžka je celistvým násobkom bytu h. K obnoveniu synchronizácie dochádza po N bytoch, tj. po bloku n bitov. V prípade slipu s inou dĺžkou je potrebné synchronizáciu obnoviť externe alebo použiť prevádzku $1-CFB$. Nevýhodou prevádzky CFB je neefektívne využitie šifrátoru, pretože z N možných bytov hesla sa využíva iba jediný.

Vlastnosti módu CFB:

1. Mód CFB vďaka reťazeniu sa rovnaké bloky otvoreného textu zašifrujú do rôznych blokov šifrovej správy, pričom takisto ako u OFB nie je potrebné tajiť inicializačnú hodnotu IV.

2. Reťaziaci mechanizmus zaisťuje, že blok šifrového textu závisí ako na bloku na príslušnom bloku otvoreného textu, takisto aj na všetkých predchádzajúcich blokoch.

3. Propagácia chyby - jeden alebo viac bitová chyba v jednom bloku šifrového textu ovplyvní dešifrovanie tohto bloku i ďalších N/n blokov šifrového textu, kde N je dĺžka bloku a n počet znakov pre šifrovanie (typicky 1 alebo 8).

4. Oprava chýb - CFB mód je samosynchrónny mód na obnovenie synchronizácie potrebuje N/n blokov.

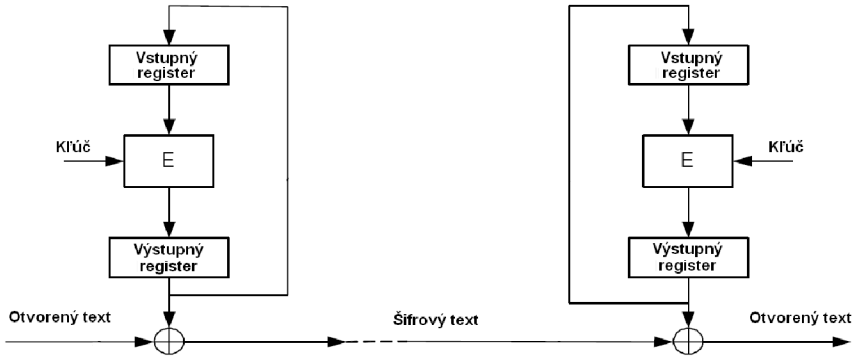
5. Výkon - pre bloky $n < N$ výkon klesá, pretože každým zavolaním šifrovacej funkcie je šifrovaný alebo dešifrovaný iba blok dĺžky n.

2.2 OFB (Output FeedBack mode)

Tento operačný mód prevádza blokovú šifru na prúdovú. Sám o sebe nedokáže ochrániť pred slipom (nemá schopnosť samosynchronizácie), ale je využitý v móde $SCFB$, ktorý bude popísaný ďalej. Používa náhodnú inicializačnú hodnotu IV k nastaveniu odpovedajúceho konečného automatu do

náhodnej polohy. Automat potom produkuje postupnosť hesla tak, že je privedené na vstup blokovej šifry a zašifrovaním je produkovaný ďalší blok hesla. Toto heslo aplikujeme pomocou operácie XOR na otvorený text. Prvý blok hesla sa získa zašifrovaním IV. Tento spôsob netvorí "skutočnú" spätnú väzbu medzi šifrovým a otvoreným textom tak, ako tomu je u iných módov (CBC, CFB), tomu to sa niekedy hovorí vnútorná spätná väzba (internal feedback).

OFB má vlastnosť čistej (synchrónnej) prúdovej šifry, pretože heslo je generované úplne autonómne bez vplyvu otvoreného a šifrového textu.



Obrázok 2.2: Šifrovanie v móde OFB

Predpis pre zašifrovanie v móde OFB:

$H_0 = IV = ST_0$,
pre $i = 1,2,...,n : \{H = E_K(H), ST_i = OT_i \oplus H_i\}$
Predpis pre odšifrovanie v móde OFB:
$H_0 = IV = ST_0$,
pre $i = 1,2,...,n : \{H = E_K(H), OT_i = ST_i \oplus H_i\}$

Pomerne zaujímavou vlastnosťou OFB je to, že aktuálne heslo na vstupe môžeme pri znalosti IV a kľúča K počítať takpovediac offline, tj. bez toho aby sme poznali otvorený (pri šifrovaní) či šifrový (pri dešifrovaní) text. To samozrejme môže výrazne urýchliť operácie šifrovania i dešifrovania. Ešte ako poznámku uvedieme, že pre OFB nám postačí znalosť šifrovacieho algoritmu blokovej šifry (dešifrovací algoritmus sa nikde nevyužíva).

Pre IV platia rovnaké pravidlá ako u CFB. Avšak, ako už bolo zmienené u CFB, neznalosť IV spôsobí chybné dešifrovanie všetkých blokov šifrového textu.

Modus OFB poskytuje synchrónnu prúdovú šifru. Heslo generuje konečným automatom, ktorý má maximálne $2N$ vnútorných stavov. Po tomto počte krokov sa produkcia hesla musí nutne opakovať. Dĺžka periódy hesla je preto maximálne $2N$, jej konkrétna dĺžka je určená hodnotu IV a môže sa pohybovať náhodne v rozmedzí od jednej do $2N$. Štruktúra hesla je značne závislá na tom, či spätná väzba je plná alebo nie. OFB/CFB je vhodné pre šifrovanie postupnosti znakov, obyčajne 8-bitových, kedy je vhodné, aby každý znak bol spracovávaný nezávisle.

Vlastnosti módu OFB:

1. Dva rovnaké bloky otvoreného textu nie sú šifrované do dvoch rovnakých blokov šifrového textu, pokiaľ nie sú použité rovnaké hodnoty IV.

2. Heslo je úplne nezávislé na vstupnom texte.

3. Propagácia chyby - jedno alebo viac bitová chyba v ktoromkoľvek bloku šifrového textu ovplyvní iba tento blok. Naviac sa táto chyba objaví iba na odpovedajúcich miestach vo výslednom texte, takže je možné tieto chyby spätne opraviť.

4. Oprava chýb - OFB mód umožňuje obnoviť bitové chyby v šifrovom texte, ale nemôže sa sám resynchronizovať po strate synchronizácie. V tomto prípade je nutná explicitná obnova synchronizácie - tento mód nie je samosynchronizačný.

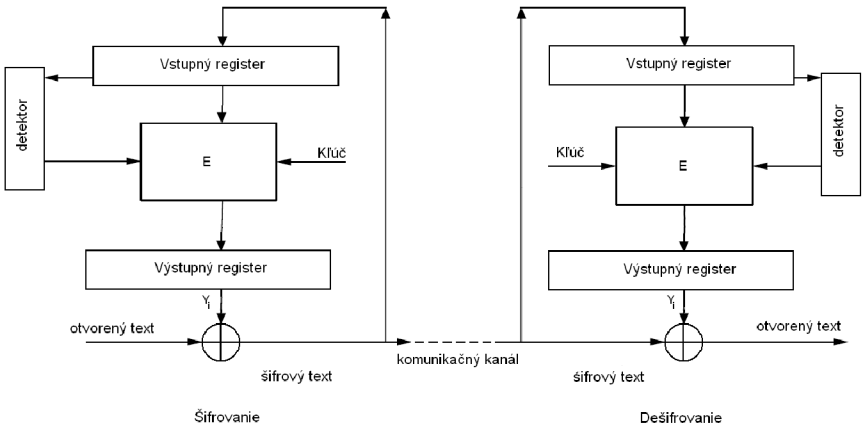
5. Výkon - výkon tohto módu klesá, ale vďaka tomu, že kľúč nezávisí ani na otvorenom texte ani na šifrovom texte, tak môže byť prepočítaný dopredu, čo umožní určité zrýchlenie.

2.3 OCFB (Optimized Cipher Feedback Mode)

Z jednotlivých módov blokových šifier je vhodný na vyriešenie problému slipu iba mód CFB. A tento mód je jediný, ktorý umožňuje šifrovanie blokov, ktoré sú kratšie ako pevne stanovená dĺžka bloku, čo znamená že sa nemusí prenášať celý blok šifry.

Takisto aj slip môže vzniknúť iba ako základná jednotka prenosu, to znamená že v sieťach ISDN, v ktorých sú prenášané byty, môže vzniknúť slip iba v dĺžke 8 bitov.

Bloková šifra použitá v prevádzkach CFB, $OCFB$ a $SCFB$ pracuje s blokmi o dĺžke n bitov, pri čom platí že dĺžka bloku šifry je celistvým násobkom N dĺžky bytov, tj platí $n = N.h$. Jednotlivé prevádzky sa líšia spôsobom odvodenia bytov hesla Y_i.

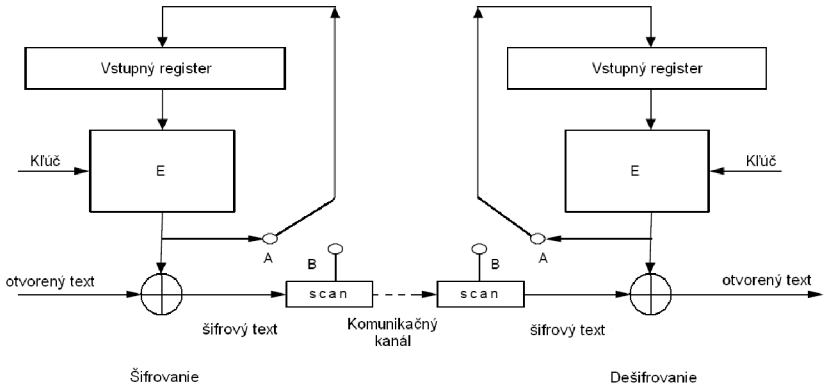


Obrázok 2.3: Šifra v móde $OCFB$, ktorý umožňuje opraviť dešifrovanie aj po výskyte slipu

Modifikáciou CFB je prevádzka označovaná $OCFB$ (”Optimized CFB” - Optimalizovaný CFB mód). Princíp prevádzky $h-OCFB$ je ilustrovaný na obr 2.3. Na začiatku je vstupný register blokového šifrátora naplnený inicializačným vektorom IV. Tento vektor je zašifrovaný a vo výstupnom registri sa objavia byty X_1 až X_N, ktoré sú postupne použité ako N bytov hesla Y_i. Byty kryptogramu C_i sa vo vysielači i prijímači priebežne ukladajú do vstupného posuvného registru. Potom čo je zašifrovaný N-tý byte, je vo vstupnom registri nová n-tica bitov, ktorá je zašifrovaná. Získa sa tak ďalších N bytov hesla. Uvedená činnosť sa periodicky opakuje s výnimkou, kedy sa v ľavej časti vstupného registra objaví dopredu stanovená kombinácia s bitov. Táto kombinácia sa nazýva synchronizačná sekvencia. Pokiaľ je takáto sekvencia príslušným detektorom zistená, potom sa vykoná nové šifrovanie i cez to, že doteraz nebolo využitých všetkých N bytov hesla. Výhodou popísanej prevádzky je vyššia efektívnosť pri generovaní hesla. S každým spustením blokového šifrátora sa získa až N bytov hesla oproti jedinému bytu v prevádzke CFB. Z hľadiska slipov je $h-OCFB$ odolná voči slipom s dĺžkou celistvého násobku h bitov. K obnoveniu synchronizácie dôjde až keď sa v kryptogramu vyskytne synchronizačná sekvencia. Túto sekvenciu rozozná šifrátor i dešifrátor. Pokiaľ nedošlo k žiadnym prenosovým chybám alebo k výskytu ďalšieho slipu tak sa vo vstupných registroch vysielača i prijímača nachádzajú totožné bitové bloky. Vysielač i prijímač tak odvodia rovnaký blok hesla, ktorý použijú k dešifrovaniu rovnakého bytu kryptogramu. Tak sa obidve strany dostanú opäť do synchronizácie. Stredná doba obnovy synchronizácie v tomto prípade závisí na pravdepodobnosti výskytu synchronizačnej sekvencie.

2.4 SCFB (Statistical Cipher FeedBack mode)

Ďalším samosynchronizujúcou prevádzkou blokovej šifry je prevádzka *SCFB* (”Statistical *CFB*” - Štatistická *CFB*). Principiálne je prevádzka *SCFB* hybrid medzi prevádzkou *OFB* a *OCFB*. Za normálnych podmienok sa využíva *OFB* prevádzka a priebežne sa sleduje postupnosť bitov kryptogramu. V prípade výskytu synchronizačnej sekvencie v kryptograme sa vykoná resynchronizácia. Princíp *SCFB* je ilustrovaný na obr. 2.4. Na začiatku sa do vstupného registru vloží inicializačný vektor IV, ktorý sa zašifruje a výsledok šifrovania sa uloží do výstupného registru. Byty X_1 až X_N z tohto registru sú postupne použité ako N bytov hesla Y_i. Byty kryptogramu C_i sú vo vysielači i prijímači ukladané do posuvného registra. Za normálnej situácie sa po využití všetkých bytov hesla obsah výstupného registru skopíruje do vstupného registru, nový obsah tohto registru sa znova zašifruje a popísaná činnosť sa periodicky opakuje (prakticky prevádzka *OFB*). Výnimkou je situácia, keď sa v kryptograme vyskytne synchronizačná sekvencia. Keď je táto sekvencia detekovaná na konci registra, tak je obsah registru zašifrovaný a výsledok šifrovania je použitý ako heslo (prakticky prevádzka *OFB*). Systém sa potom vráti k normálnemu spôsobu prevádzky až do výskytu novej synchronizačnej sekvencie. Z hľadiska slipov je $h - SCFB$ odolná voči slipom s dĺžkou rovnou celistvému násobku h bitov. K obnoveniu synchrónnosti dôjde, až sa v kryptograme vyskytne synchronizačná sekvencia. Stredná doba obnovy synchronizácie v tomto prípade závisí na pravdepodobnosti výskytu synchronizačnej sekvencie.



Obrázok 2.4: Šifra v móde *SCFB*, ktorý umožňuje opraviť dešifrovanie aj po výskyte slipu

Kapitola 3

Navrhnuté riešenie

3.1 Simulácie porúch

Táto práca sa zaoberá poruchou slip, teda vložením alebo stratou základnej prenosovej jednotky na prenosovej ceste, ktorá predstavuje metalické, optické alebo bezdrôtové vedenie. Najčastejšie je základnou jednotkou prenosu jeden bit. Je stanovená rozhodovacia hladina, ktorá závisí od typu prenosu, táto hladina je napríklad stanovená v jednotkách napätia. Keďže prenosový kanál predstavuje fyzické vedenie, simulácia vzniku slipu by bola veľmi problematická. Preto bolo rozhodnuté, že simulácia bude prebiehať softwarovo.

Ako bolo spomenuté v úvode, slip vzniká z dôvodu rozdielnych časových základní vysielača a prijímača, kedy časová základňa prijímača je o určitú odchýlku "rýchlejšia", respektíve "pomalšia" oproti časovej základni vysielača, čo sa vo výsledku prejavuje vznikom slipu. Ďalšou možnosťou vzniku slipu je prenos dlhých postupností bitov rovnakej hodnoty, kedy v tomto prípade odchýlka v časovej základní spôsobí nesprávnu identifikáciu počtu týchto bitov. Pre zabezpečenie použiteľnosti simulačného softwarového systému je treba presne definovať niektoré akcie, ktoré sa nevyskytujú pri simulácii bez porúch. Ďalej uvádzame popis základných činností a ich realizáciou v práci:

- *špecifikácia porúch*
  Spočíva v určení, ktoré poruchy a akých typov sa budú modelovať v priebehu simulácie. Spravidla sa určujú všetky poruchy, ktoré sme schopní v danom obvode modelovať. Táto množina sa potom na základe určitých techník redukuje.
  V našom prípade ide o poruchy invertovania bitu a tzv. slip. Táto práca sa zaoberá iba slipmi, preto sa budú simulovať výhradne tieto poruchy.

- *vkladanie (injekcia) porúch*
  Musí byť možnosť špecifikovať typ poruchy a miesto v obvode, kde sa daná porucha vyskytuje. Vložením poruchy dochádza k transformácii modelu bezporuchového obvodu na model obvodu s poruchu.
  V simulovanom kryptosystéme uvažujeme iba jeden typ poruchy a to je slip. Miesto vkladania porúch bolo zvolené na výstup šifrátora. Toto odpovedá modelu, kedy systém okamžite prenáša zašifrované dáta cez prenosový kanál, teda porucha bude vkladaná ihneď po zašifrovaní jedného bloku.

- *šírenie porúch*
  Simulačný systém musí byť schopný zabezpečiť šírenie vplyvu poruchy systémom. Konkrétny spôsob záleží na použitej technike simulácie porúch.
  Vloženie poruchy spôsobí stratu synchronizácie, tým simulujeme nedokonalosť vysielača a

prijímača a ich rozdielnych časových základní. Vloženie slipu spôsobí, že text od tohto miesta nebude možné naďalej správne šifrovať.

- *detekcia porúch*
  Systém musí byť schopný určiť, kedy je porucha pozorovateľná v niektorom mieste kryptosystému alebo na jeho výstupe.
  Toto miesto bude identifikované na základe neschopnosti systému správne dešifrovať šifrový text od miesta vloženia slipu.

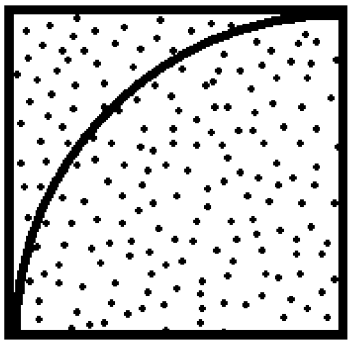
3.2 Metóda Monte-Carlo

Prístup k riešeniu úloh, ktorý na získanie výsledkov využíva vhodný pravdepodobnostný model, sa vo všeobecnosti nazýva metóda Monte-Carlo. Použitím tejto metódy možno riešiť veľmi rôznorodé úlohy, pričom môže ísť o určenie tak hodnôt deterministických veličín, ako aj charakteristík náhodných veličín. Pri samotnej realizácii pravdepodobnostného modelu vychádzame práve z možnosti vytvárať náhodné veličiny s daným rozdelením pravdepodobnosti, pričom model zobrazuje ich vzťah k skúmaným veličinám.

V praktických úlohách býva vzťah medzi náhodnými veličinami a hľadaným výsledkom veľmi zložitý, pričom záleží aj na rozdelení pravdepodobnosti náhodných veličín. Analytické riešenie úlohy je často zložité alebo ho dokonca nepoznáme. Vtedy môžeme výsledok určiť použitím metódy Monte-Carlo.

Ak poznáme rozdelenie pravdepodobnosti pre jednotlivé elementárne procesy, z ktorých sa skúmaný jav skladá, môžeme modelovať rozdelenie pravdepodobnosti určitej konfigurácie systému.

Pre názornosť môžeme uviesť príklad:



Obrázok 3.1: Štvrťkružnica vpísaná do štvorca

Máme určiť hodnotu čísla π (podľa obrázku 3.1). Je jasné, že ak vpíšeme do štvorca štvrťkružnicu, bude pre plochu štvorca a štvrťkružnice platiť:

$$S_{st} = r^2 \tag{3.1}$$

$$S_{kr} = \frac{r^2}{4} \tag{3.2}$$

Ak by sme poznali obe plochy, tak môžeme číslo π vyjadriť ako

$$\pi = 4\frac{S_{kr}}{S_{st}} \tag{3.3}$$

Predpokladajme, že budeme na štvorec hádzať šípky. Predpokladajme tiež, že budú na štvorec dopadať úplne náhodne a rovnomerne (oboje je veľmi dôležité). Potom môžeme prehlásiť, že počet zásahov je priamo úmerný ploche. Môžeme teda vzorec upraviť na tvar:

$$\pi = 4\frac{n_{kr}}{n_{st}} \tag{3.4}$$

kde n_{kr} je počet zásahov do štvrťkružnice a n_{st} je počet všetkých zásahov do štvorca a zároveň aj do štvrťkružnice.
Dôležitý je odhad chyby, ktorej sa touto metódou dopustíme. Matematický popis presnosti metódy Monte-Carlo je nasledujúci: Uvažujeme o nejakej udalosti A, pričom pravdepodobnosť jej výskytu je p. Označíme V_i náhodnú veličinu ktorá nadobúda hodnotu 1 ak udalosť v i-tom pokuse nastala a hodnotu 0 ak udalosť A nenastala. Počet výskytov m udalosti A v n vykonaných pokusoch je hodnotou náhodnej veličiny M, pričom:

$$M = \sum_{i=1}^{n} V_i \tag{3.5}$$

Predpokladáme, že pokusy sú nezávislé. Pretože pravdepodobnosť p odhadujeme pomocou hodnoty m/n, určíme pre náhodnú veličinu M/n jej strednú hodnotu:

$$E\left(\frac{M}{n}\right) = \frac{1}{n}E(M) = \frac{1}{n}\sum_{i=1}^{n} E(V_i) = E(V_i) = 0.(1-p) + 1.p = p \tag{3.6}$$

a rozptyl

$$D\left(\frac{M}{n}\right) = \frac{1}{n^2}D(M) = \frac{1}{n^2}\sum_{i=1}^{n} D(V_i) = D(V_i) = \frac{1}{n}\Big(\big(0-p\big)^2\big(1-p\big) + \big(1-p\big)^2 p\Big) = \frac{p(1-p)}{n} \tag{3.7}$$

Našou úlohou je určiť vzťah, $\varepsilon > 0$ také, aby vzťah

$$\left|\frac{M}{n}\right| - p \geq \varepsilon \tag{3.8}$$

platil s pravdepodobnosťou rovnajúcou sa nanajvýš nejakej malej hodnote $\delta > 0$, teda aby bola splnená nerovnosť

$$P\Big(\left|\frac{M}{n} - p\right| \tag{3.9}$$

Inými slovami povedané to znamená, že vzťah

$$\left|\frac{M}{n} - p\right| < \varepsilon \tag{3.10}$$

platí s pravdepodobnosťou väčšou ako $1-\delta$.
Z Čebyševovej nerovnosti dostávame, že pre každú náhodnú veličinu X s konečným rozptylom $D(X) = \sigma^2$ a strednou hodnotou $E(X)$ pre ľubovolné $\lambda > 0$ platí

$$P\Big(\big|X-E(X)\big| \geq \lambda\sigma\Big) \leq \frac{1}{\lambda^2}\frac{M}{n} - p\Big| < \varepsilon \tag{3.11}$$

Ak dosadíme do tohto vzťahu za X náhodnú veličinu M/n, ako aj jej strednú hodnotu a rozptyl, dostaneme

$$P\left(\left|\frac{M}{n} - p\right| > \lambda\sqrt{\frac{p(1-p)}{n}}\right) \leq \frac{1}{\lambda^2} \tag{3.12}$$

Porovnaním vidíme, že s pravdepodobnosťou väčšou ako $1-\delta$ platí

$$\left|\frac{M}{n} - p\right| > \lambda\sqrt{\frac{p(1-p)}{n\delta}} \tag{3.13}$$

a pre dané δ a ε vzťah platí pre

$$n > \frac{p(1-p}{\varepsilon^2\delta} \tag{3.14}$$

Napríklad pre $p =$½vzťah

$$P\left(\left|\frac{M}{n} - p\right| > 0.02\right) < 0.05 \tag{3.15}$$

platí pre $n > 12500$ pokusov.

Odhad chyby, ktorej sa dopustíme môžeme zlepšiť, ak pravdepodobnosť nejakej udalosti nahradíme jej relatívnou početnosťou. Dôležité je že táto chyba má rád $1/\sqrt{n}$. Z toho vidieť, že na to, aby sme zvýšili presnosť výsledku získaného metódou Monte-Carlo o jeden rád, treba zvýšiť počet vykonaných pokusov stokrát. V praktických aplikáciách dostávame výsledky, ktorých relatívna chyba je 0,01 až 0,001. Je vhodné si uvedomiť, či si charakter úlohy vyžaduje vysokú presnosť výsledku alebo nie.

Metóda Monte-Carlo je závislá na kvalitnom generátore náhodných čísel, pretože vychádza z pravdepodobnostného modelu. Generátor náhodných čísel nie je vo väčšine prípadov počítačový program. Tieto sú označované ako generátory pseudonáhodných čísel (ide napríklad o funkciu random() v jazyku Pascal a podobne). Tieto generátory totiž čísla generujú napríklad odvodením zo systémového času. Preto skôr či neskôr v tomto pseudonáhodnom signáli nájdeme určitú periódu a rozloženie týchto čísel nezodpovedá náhodnému (Gaussovskému, normálovému) rozloženiu.

3.3 Návrh riešenia

Simulačný systém bude využívať metódu Monte-Carlo pre výpočet pozície, na ktorú bude slip umiestnený. Táto práca sa zaoberá účinkami slipu na módy blokových šifier a nie slipom samotným. Preto bolo zvolené, že slip bude simulovaný tak, že po výpočte pozície slipu sa bit z tejto pozície odoberie (to znamená že nastane prípad, keď sa bit stratí). Metóda Monte-Carlo je charakteristická daným vstupným rozdelením pravdepodobnosti, ktoré bolo zvolené pre tento prípad ako exponenciálne. Použitá bude metóda inverznej funkcie. Pre naše potreby bude vytvorená náhodná veličina s definovaným rozdelením (exponenciálne rozdelenie) distribučnú funkciu $F(x)$.

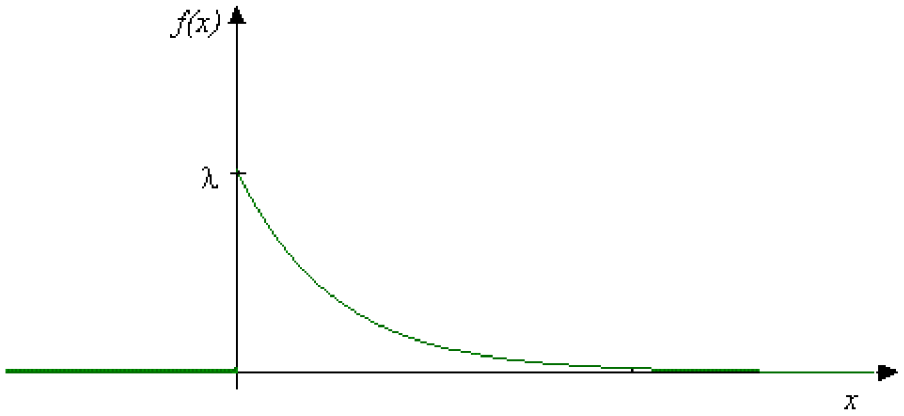
Pre distribučnú funkciu platí, že $F(x) < 0,1 >$, taktiež definujeme náhodnú veličinu $RND < 0,1)$.

Je dané:

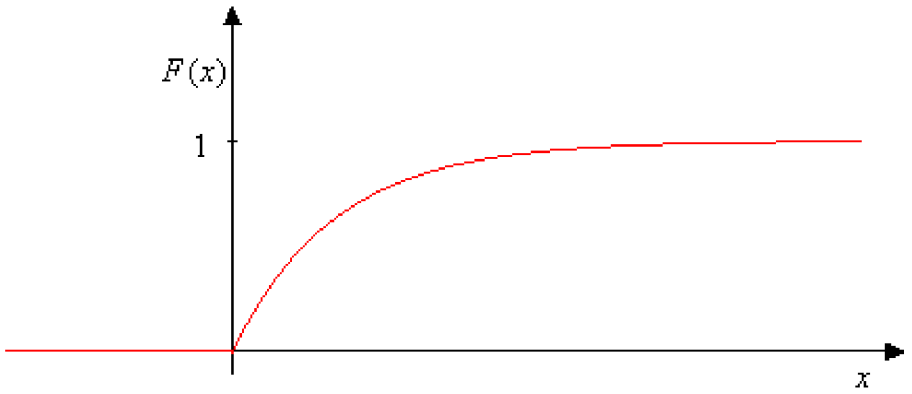
$F(x) = RND$
$X = F - 1(RND)$ - náhodná veličina s rozdelením podľa $F(x)$

Bude použitý nasledujúci výpočet pre pozíciu slipu:
$F(x) = 1 - e^{-\lambda x}$ - o vstupné rozdelenie bolo zvolené exponenciálne
$RND = 1 - e^{-\lambda x}$
$e^{-\lambda x} = 1 - RND$
$-\lambda x = ln(1 - RND)$

$x = \frac{1}{\lambda} \ln(1 - RND)$ - pozícia, na ktorej sa bude slip nachádzať



Obrázok 3.2: Funkcia hustoty exponenciálneho rozdelenia.



Obrázok 3.3: Distribučná funkcia exponenciálneho rozdelenia.

Pre potreby projektu bude použitá funkcia *random*() v jazyku Pascal na vygenerovanie náhodnej poruchy. Tu je vzorový kód ktorý bude použitý pre generovanie:

```
function GetPosition(lambda : integer) : integer;
var
  RND : double;
begin
  RND := Random;
  result := round((1/lambda)*ln(1-RND));
end;
```

Táto funkcia vráti pozíciu v bloku, na ktorý bude vložený slip. Ďalej definujeme veličiny:

- výberový priemer $\overline{X}$

$$P(x) = \frac{\sum_{i=1}^{N} x_i}{N} \tag{3.16}$$

- výberový rozptyl

$$R(x) = \frac{\sum_{i=1}^{N} x_i^2 - \frac{(\sum_{i=1}^{N} x_i)}{N}}{N-1} \tag{3.17}$$

- konfidenčný interval

$$P(x) = \sqrt{\frac{R(x)}{N(1-\delta)}} \tag{3.18}$$

kde δ je je konfidenčný koeficient, zvyčajne je δ v intervale $< 0,95;1)$.

Výberový priemer určí rozsah konfidenčného intervalu, teda presnosť metódy.
Platí:

$$E(\overline{X}) = E(\frac{\sum_1 x_i}{N}) = \frac{1}{N}\sum_{i=1}^{N} E(x_i) = \frac{N.E(X)}{N} = E(X) \tag{3.19}$$

$$D(\overline{X}) = D(\frac{\sum x_i}{N}) = \frac{1}{N^2}\sum D(x_i) = \frac{N.D(X)}{N^2} = \frac{D(X)}{N} \tag{3.20}$$

Platí Čebyševova nerovnosť

$$P\{\left|X - E(X)\right| < \varepsilon\} \geq 1 - \frac{D(X)}{\varepsilon^2} \tag{3.21}$$

$$P\{\left|\overline{X} - E(\overline{X})\right| < \varepsilon\} \geq \delta \in < 0,95;1) \tag{3.22}$$

Určíme δ ako

$$\delta = 1 - \frac{D(\overline{X})}{\varepsilon^2} = 1 - \frac{D(X)}{N\varepsilon^2} \Rightarrow \varepsilon\sqrt{\frac{D(X)}{N(1-\delta)}} \tag{3.23}$$

a pre ε platí

$$\varepsilon = f(x,\delta)\frac{1}{\sqrt{N}} \tag{3.24}$$

$$D(X) \approx R(X) \tag{3.25}$$

kde ε je maximálna chyba ktorej sa môžeme dopustíme.

3.4 Zvolené programové vybavenie

Bloková šifra bude realizovaná v módoch *CFB*, *OCFB* a *SCFB*, základný šifrovací algoritmus bol zvolený *AES* (Advanced Encryption Standard) s veľkosťou bloku 128 bitov a s veľkosťou kľúča 128 bitov, nakoľko je veľmi dobre známy a je v súčasnosti asi najrozšírenejší šifrovací algoritmu v symetrickej kryptografii. Takisto aj hardwarová alebo softwarová implementácia algoritmu je jedna z menej náročných. Celý kryptosystém sa skladá z troch častí, a to z vysielača, prenosovej cesty a prijímača.

Simulovať sa bude reálny kryptosystém, ktorý bude dáta šifrovať a okamžite odosielať do prijímača k dešifrovaniu. Slipy budú simulované pri odosielaní, to znamená na výstupe šifrátora.

Kryptosystém bude implementovaný softwarovo a pre túto implementáciu bol zvolený programovací jazyk Delphi. Samotná realizácia bude prebiehať vo vývojovom prostredí Borland Developer Studio 2006 pre Windows.

Pomocou zvoleného programového vybavenia budeme simulovať náhodné deje v prenosovom kanáli, ktoré spôsobujú tzv. slipy a budú analyzované popísané módy blokovej šifry.

V reálnom svete predstavuje prenosový kanál najčastejšie metalické vedenie, po ktorom sa informácie prenášajú vo forme núl a jedničiek. Toto je ovplyvňované vonkajšími nežiadúcimi vplyvmi, ktoré sú známe ako šum a sú príčinou týchto porúch. Tieto poruchy sa interferenčne pripoja k signálu a nie je možné ich oddeliť zo signálu. Toto je veľký problém pre prenos šifrových dát, kde takáto porucha môže spôsobiť nemožnosť správne dešifrovať správu. Slip je porucha, ktorú spôsobujú rozdielne časové základne prijímača a vysielača. Táto chyba takisto znemožňuje správne dešifrovať zvyšok správy. Popis vkladania a generovania slipu bol popísaný vyššie. Po vložení slipu bude šifrový text dešifrovaný použitím niektorého z módov, ktoré majú schopnosť samosynchronizácie. Súčasne budú analyzované a skúmané jednotlivé charakteristiky daného módu. Všetky ďalšie výsledky budú uvedené v diplomovej práci vrátane celého softwarového riešenia.

Kapitola 4

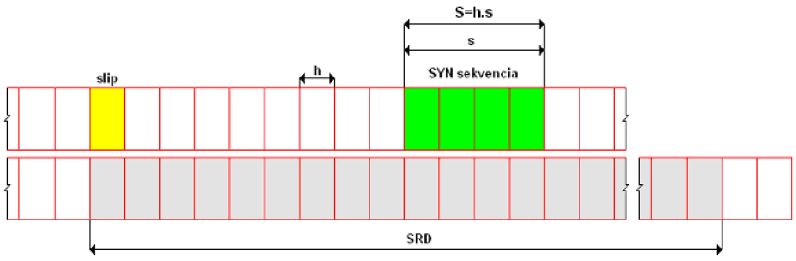
Grafické rozhranie aplikácie

V nasledujúcej časti bude popísaná štruktúra programu, jednotlivé funkcie a zdrojové súbory. Tiež budú uvedené príklady komentovaných zdrojových kódov a popis funkcií.

Pre návrh grafického rozhrania bol použitý základný súbor komponent, ktoré ponúka vývojové štúdio Borland Developer Studio (BDS). Rozhranie bolo navrhované s ohľadom na jednoduchosť a efektivitu použitia a čo najlepšie možnosti využitia analýzy.

4.1 Sledované parametre - SRD a EPF

Pri samotnej analýze prevádzky blokovej šifry so samosynchronizačným módom je hlavným sledovaným parametrom Synchronization Recovery Delay (*SRD*) - resynchronizačná vzialenosť. Je to vzdialenosť v bitoch od miesta posledného výskytu slipu po prvý bit správne dešifrovaného textu.

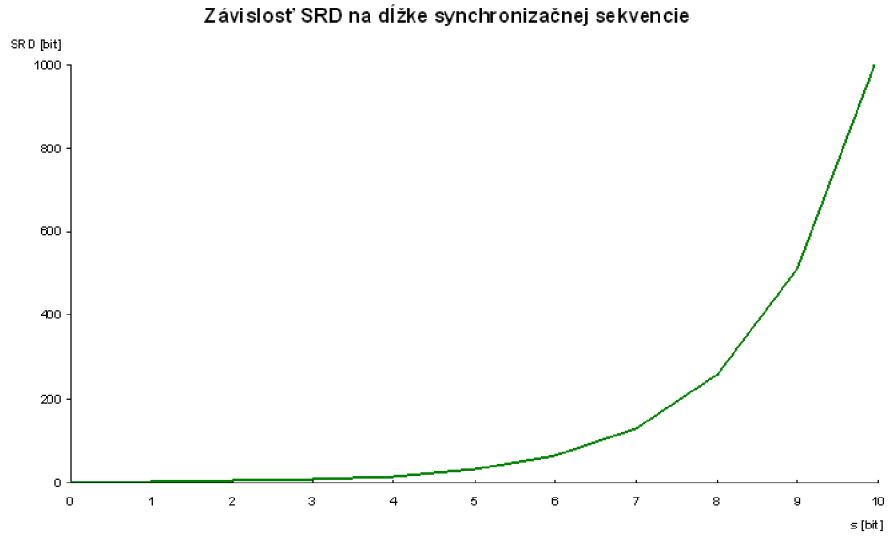


Obrázok 4.1: SRD - Synchronization Recovery Delay

Táto vzdialenosť závisí na pravdepodobnosti výskytu synchronizačnej sekvencie, a pravdepodobnosť výskytu synchronizačnej sekvencie zasa závisí na dĺžke synchronizačnej sekvencie. Každý mód má inú resynchronizačnú vzdialenosť, vzhľadom na svoj návrh. Napríklad v prípade *CFB* módu je resynchronizačná vzdialenosť rovná dĺžke bloku šifry. Je to najmenšia resynchronizačná vzdialenosť, ktorú je možné dosiahnuť. Naproti tomu, tento mód je veľmi neefektívny, pretože z N možných bitov sa využíva iba h bitov vzniknutého hesla. Tento mód je však odolný iba slipom s celistvým násobkom h, ak vznikne slip inej dĺžky je potrebné obnoviť synchronizáciu iným spôsobom alebo použiť $h = 1$, ktorý sa zosynchronizuje pri akomkoľvek slipe. *SRD* tiež závisí na zvolenej dĺžke bloku.

Samosynchronizačné módy *SCFB* a *OCFB* majú *SRD* vždy horšie. Z toho vyplýva, že používanie

týchto módov je vhodné iba vtedy, ak je dôležitým požiadavkom efektívne využitie blokového šifrátora. Z uvedeného grafu vyplýva, že *SRD* významne závisí na dĺžke synchronizačnej sekvnecie, preto by sme sa mali snažiť o čo najmenšie hodnoty najmä s (dĺžky synchronizačnej sekvencie), ale aj N (dlzky bloku) a h (dĺžky bytu).



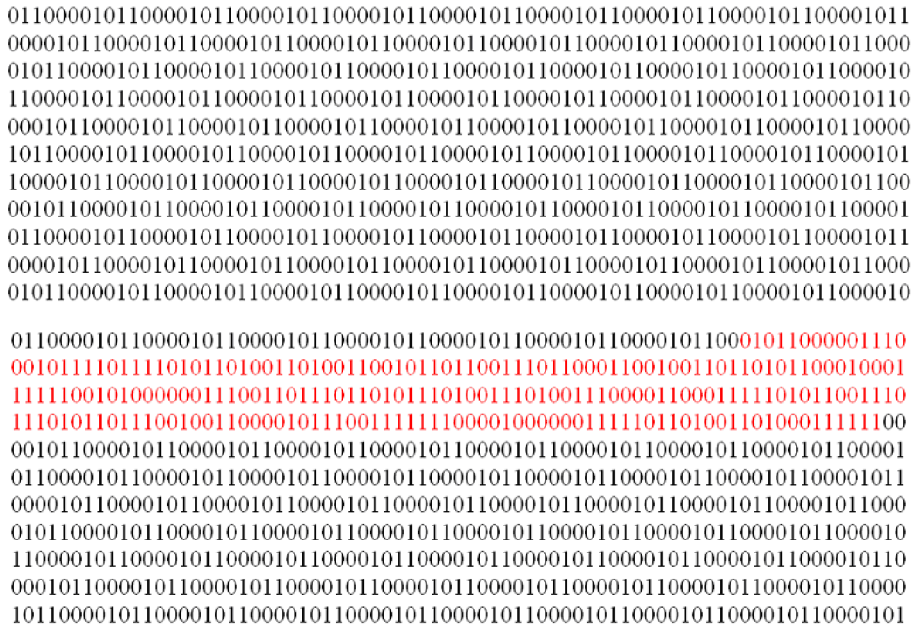
Obrázok 4.2: Nárast SRD so zväčšovaním dĺžky synchronizačnej sekvencie pri móde *OCFB*

Na tomto mieste by som chcel upozorniť, že synchronizácia a prenos môžu byť bitové alebo bytové. Pre samosynchronizačný operačný mód *OCFB* bolo *SRD* odvodené v [1], kde je výsledok je *SRD* definované ako $h.[2S + (N - S)]$, kde h je počet bitov bytu, S je dĺžka synchronizačnej sekvencie v bytoch a N je dĺžka bloku v byte. Takto uvedený vzťah udáva strednú hodnotu *SRD* v bitoch. Mód *SCFB* ma *SRD* horšie, pretože po nájdení synchronizačnej sekvencie sa na dĺžku získavania nového inicializačného vektoru synchronizácia zablokuje, preto má menej príležitostí sa synchronizovať ako mód *OCFB*.

Vo výstupe z programu bola vyznačená sekcia, kde nastal slip a takisto sekcia, kde nastala opätovná resynchronizácia.

Začiatok farebne odlíšenej sekvencie je miesto výskytu slipu. Za bežných okolností by nebolo možné text dešifrovať správne, ale vďaka použitiu samosynchronizačného módu je možné správne dešifrovanie. Červenou farbou je vyznačená sekvencia od miesta výskytu slipu až po opätovné získanie synchronizácie. Na obrázku hore môžeme vidieť postupnosť bitov pred zašifrovaním, na obrázku dole je dešifrovaný text.

Ďalším parametrom, ktorý budeme sledovať, je parameter *EPF* (Error Propagation Factor). Ide o údaj, ktorý určuje, ako dlho trvá obnovenie správneho šifrovania po chybe. Táto práca je primárne zameraná na analýzu samosynchronizačných módov, ktoré poskytujú ochranu pred slipom. Vlastnosti týchto módov tiež môžu ochrániť pred chybou slip, preto bude tento parameter skúmaný ďalej.



Obrázok 4.3: Ukážka vplyvu slipu s vyznačením poškodeného dešifrovaného textu

4.2 Softwarová štruktúra programu

Jadrom celého šifrovania je zdrojový súbor AES.PAS, v ktorej sa nachádza samotné jadro šifrovacieho algoritmu AES a taktiež aj jednotlivé módy tohto algoritmu. Celý algoritmus je zapuzdrený v objekt TAES, čo je trieda ktorá obsahuje všetky potrebné funkcie pre inicializáciu šifry a tiež jej správnu prevádzku. Taktiež sú v tomto zdrojovom súbore nadefinované koštanty a dátové typy použité pri šifrovaní.

Funkcia GetSet a procedúra PutSet sa používajú výlučne pre interné nastavenie blokovej šifry. Procedúra KeySetupEnc vykoná expanziu kľúča CipherKey pred šifrovaním. Keďže šifra AES využíva rundy (rundou nazývame jeden krok v internom algoritme), expanzia kľúča rozloží kľúč na viacero podkľúčov a tie sú potom použité pre jednotlivé rundy. Počet rund závisí na dĺžke kľúča.

Procedúra KeySetupDec vykoná expanziu kľúča CipherKey pred dešifrovaním. Vykoná sa expanzia ale podkľúče sú v tomto prípade určené pre dešifrovanie. Procedúry Encrypt a Decrypt - sú to vnútorné funkcie šifry, samotné jadro algoritmu *AES*.

Zdrojový súbor Routines.pas obsahuje rôzne pomocné funkcie a procedúry, ide najmä o konverzné rutiny a taktiež funkciu určenú pre simuláciu poruchy slip.

```
type
  TRijndael = class
  private
    function KeySetupEnc(var rk: ExpandedKey; const cipherKey: array of byte Block;
                         keyBits: longint): longint;
    function KeySetupDec(var rk: ExpandedKey; const cipherKey: array of byte Block;
                         keyBits: longint): longint;
    function GetSet(const pt: array of BYTE Block; pos: LONGINT): SET32;
    procedure PutSet(var ct: array of BYTE Block; pos: LONGINT; st: SET32);
    procedure Encrypt(const rk: ExpandedKey; Nr: longint; const pt: array of byte Block;
                      p0: longint; var ct: array of byte Block; c0: longint);
    procedure Decrypt(const rk: ExpandedKey; Nr: longint; const ct: array of byte Block;
                      c0: longint; var pt: array of byte Block; p0: longint);
  public
    // Electronic CodeBook Mode
    function EncryptECB(const Key: array of byte; PlainText: ansistring): ansistring;
    function DecryptECB(const Key: array of byte; CipherText: ansistring): ansistring;
    // Cipher FeedBack Mode
    function EncryptCFB(const IV: array of byte; const Key: array of byte;
                        PlainText: ansistring): ansistring;
    function DecryptCFB(const IV: array of byte; const Key: array of byte;
                        CipherText: ansistring): ansistring;
    // Output FeedBack Mode
    function EncryptOFB(const IV: array of byte; const Key: array of byte;
                        PlainText: ansistring): ansistring;
    function DecryptOFB(const IV: array of byte; const Key: array of byte;
                        CipherText: ansistring): ansistring;
    // Optimized Cipher FeedBack Mode
    function EncryptOCFB(const IV: array of byte; const Key: array of byte;
                         PlainText: ansistring): ansistring;
    function DecryptOCFB(const IV: array of byte; const Key: array of byte;
                         CipherText: ansistring): ansistring;
    // Statistical Cipher FeedBack Mode
    function EncryptSCFB(const IV: array of byte; const Key: array of byte;
                         PlainText: ansistring): ansistring;
    function DecryptSCFB(const IV: array of byte; const Key: array of byte;
                         CipherText: ansistring): ansistring;
end;
```

4.3 Realizácia samosynchronizačných módov

V programe, ktorý je súčasťou tejto práce, je možné použiť tri druhy samosynchronizačných módov - mód *CFB* (Cipher Feedback), *OCFB* (Optimized Cipher Feedback mode) a mód *SCFB* (Statistical Cipher Feedback mode). V programe sú realizované všetky tri módy, ale pre naše potreby budeme využívať iba módy *OCFB* a *SCFB*. Implementované boli aj ďalšie základné módy, ale tieto používať nebudeme. Mód *CFB* je pri použití pre účely samosynchronizácie veľmi neefektívny.

Mód *SCFB* je kombinovaný mód *OFB* a *CFB*. V základnej polohe funguje tento mód ako *OFB*. Ak je však nájdená synchronizačná sekvencia, ďalších N bitov je získaných z výstupu šifrátora čo je mód *CFB*. Po N bitoch sa získané bity použijú ako nový inicializačný vektor. Týmto spôsobom sa dosahuje možnosť samosynchronizácie. Počas získavania N bitov je hľadanie synchronizačnej sekvencie vypnuté.

```
function TRijndael.EncryptSCFB(const IV, Key: array of byte; PlainText:
                               ansistring; SyncPattern : string): ansistring;
var
  Nr, i, j: LONGINT;
  ek: ExpandedKey;
  pt, ct, vt, st : Block;
  ptB, ctB, vtB, stB : BitBlock;
  keybits, syncc, cnt: integer;
  resultB, PlainTextB : array of byte;
  sync: boolean;
begin
  if Length(PlainText) = 0 then exit;
  keyBits := KeySize * 8;
  Nr := KeySetupEnc(ek, Key, KeyBits);
  SetLength(PlainTextB,(Length(PlainText)*8));
  SetLength(resultB,(Length(PlainText)*8));
  for I := 0 to BlockSize - 1 do
    pt[i] := IV[i];
  ByteToBinArray(PlainText,PlainTextB);
  Encrypt(ek, Nr, pt, 0, ct, 0);
  for I := 0 to BlockSize - 1 do // for next loop
    pt[i] := ct[i];
  ByteToBinArray(ct,ctB);
  ByteToBinArray(pt,ptB);
  cnt := 0;
  sync := false;
  syncc := 0;
  for I := 0 to (Length(PlainText)*8) - 1 do begin
    ctB[cnt] := ctB[cnt] xor PlainTextB[i];
    resultB[i] := ctB[cnt];
    cnt := cnt + 1;
    if sync then begin
      stB[syncc] := resultB[i];
      inc(syncc);
      if syncc = (BlockSize*8) then begin
        BinArrayToByte(stB,st);
        Encrypt(ek, Nr, st, 0, ct, 0);
        for j := 0 to BlockSize - 1 do // for next loop
          pt[j] := ct[j];
        ByteToBinArray(pt,ptB);
        ByteToBinArray(ct,ctB);
        sync := false;
        syncc := 0;
        cnt := 0;
        if CompareSync(SyncPattern,resultB,i) then begin
          sync := true;
        end;
      end;
    end else begin
      if i > 2 then begin
        if CompareSync(SyncPattern,resultB,i) then begin
         sync := true;
        end;
      end;
    end;
     if cnt = BitBlockSize then begin
      BinArrayToByte(ptB,pt);
      Encrypt(ek, Nr, pt, 0, ct, 0);
      for j := 0 to BlockSize - 1 do // for next loop
        pt[j] := ct[j];
      ByteToBinArray(ct,ctB);
      ByteToBinArray(pt,ptB);
      cnt := 0;
    end;
  end;
  BinArrayToByte(resultB,result);
  result := B64Encode(result);
end;
```

Pri *OCFB* sa používa posuvný register, kam sa ukladajú bity z výstupu šifrátora. Ak je detekovaná synchronizačná sekvencia, je vykonané nové šifrovanie aj napriek tomu, že nebolo využitých všetkých N bitov hesla. Oproti módu *SCFB* je hľadanie synchronizačnej sekvencie stále činné, čo umožňuje módu *OCFB* viac príležitostí pre synchronizáciu. Naproti tomu je však menej efektívny, pretože sa vykoná nové šifrovanie aj keď nebolo využitých všetkých N bitov hesla.

```
function TRijndael.EncryptOCFB(const IV, Key: array of byte; PlainText: ansistring): ansistring;
var
  Nr, i, j: LONGINT;
  ek: ExpandedKey;
  pt, ct: Block;
  ptB,ctB : BitBlock;
  PlainTextB : array of byte;
  resultB : array of byte;
  keybits: integer;
  cnt: int64;
begin
  keyBits := KeySize * 8;
  Nr := KeySetupEnc(ek, Key, KeyBits);
  result := '';
  cnt := 0;
  SetLength(PlainTextB,(Length(PlainText)*8));
  SetLength(resultB,(Length(PlainText)*8));
  ByteToBinArray(PlainText,PlainTextB);
  for I := 0 to BlockSize - 1 do
    pt[i] := IV[i];
  Encrypt(ek, Nr, pt, 0, ct, 0);
  ByteToBinArray(ct,ctB);
  ByteToBinArray(pt,ptB);
  for I := 0 to (Length(PlainText)*8) - 1 do begin // XOR statement
    ctB[cnt] := ctB[cnt] xor PlainTextB[i];
    resultB[i] := ctB[cnt];
    ShiftRegisterAdd(ctB[cnt], ptB);
    cnt := cnt + 1;
    if (ptB[0] = 0) and (ptB[1] = 1) and (ptB[2] = 0) then begin  //010
      BinArrayToByte(ptB,pt);
      Encrypt(ek, Nr, pt, 0, ct, 0);
      ByteToBinArray(ct,ctB);
      cnt := 0;
    end else begin
      if cnt = (BlockSize*8) then begin
        BinArrayToByte(ptB,pt);
        Encrypt(ek, Nr, pt, 0, ct, 0);
        ByteToBinArray(ct,ctB);
        cnt := 0;
      end;
    end;
  end;
  BinArrayToByte(resultB,result);
  result := B64Encode(result);
end;
```

U týchto módov sa využíva šifrovacia funkcia takisto pre dešifrovanie. Tento kód je prispôsobený pre $h = 1$, to znamená že používa bit ako základnú prenosovú a šifrovaciu jednotku. Keďže módy sú implementované softwarovo v prostredí operačného systému Windows, najmenšou jednotkou ktorú môžeme použiť je jeden byte, ktorý sa skladá z ôsmich bitov. Preto je tu nutnosť prevádzať jednotlivé znaky na bity a tie ďalej spracovávať. Táto implementácia je z hľadiska softwarového využitia pod operačným systémom veľmi neefektívna.

Takisto samosynchronizačné módy nie sú vhodné pre implementáciu v prostredí operačných systémov, pretože účelom týchto módov je ochrániť prenášaný signál pred slipom. Na toto ale používajú operačné systémy napr. protokol TCP/IP, ktorý využíva vlastné ochranné mechanizmy

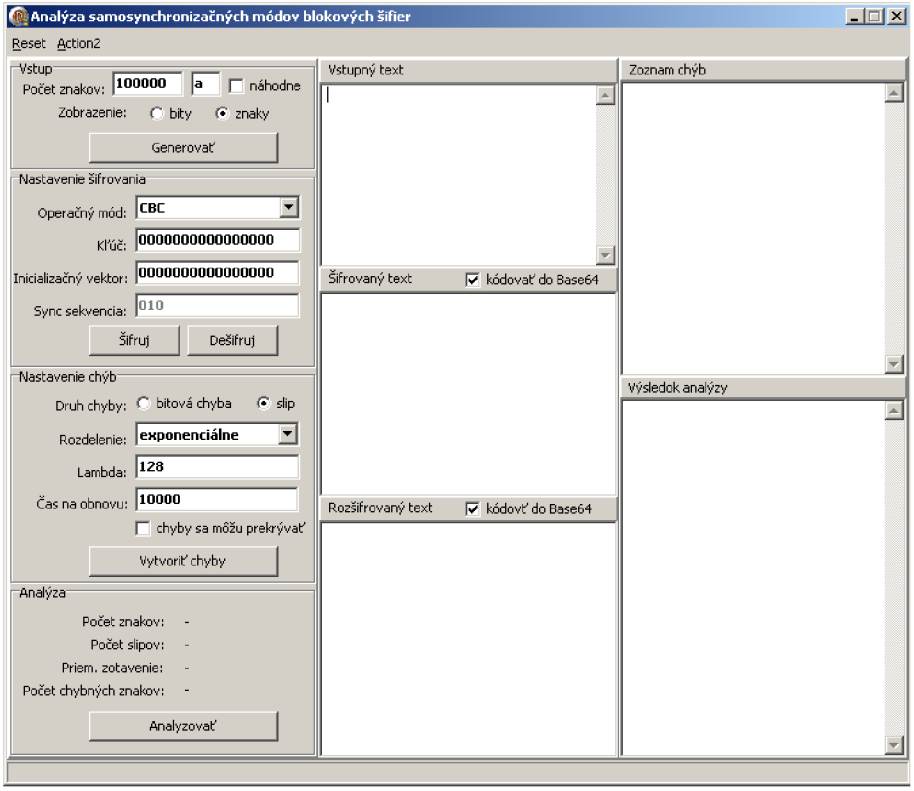
nielen pred chybami typu slip, ale aj pred chybami. Implementácia je preto vhodná pre hardwarové šifrovacie systémy, resp. pre systémy ktoré používajú vlastný prenosový mechanizmus bez využitia protokolov alebo iných ochranných možností. Zariadeniami, ktoré môžu využívajú samosynchronizačné módy sú napríklad telefóny alebo faxy.

4.4 Grafické rozhranie

Súčasťou tejto práce je software, v ktorom boli implementované popisované metódy. Pri porovnávaní jednotlivých metód bol kladený najväčší dôraz na výsledky metód *SCFB* a *OCFB*, ktoré umožňujú samosynchronizáciu. Výstupom z programu analýza doby resynchronizácie a šírenie chyby. Tieto výsledky budú využiteľné hlavne pre daľší vývoj a zdokonaľovanie módov so samosynchronizáciou.

Navrh simulačného modelu bol popísaný skôr a jeho implementácia prebehla podľa tohto návrhu. Model má analyzovať vplyv prenosových chýb na výstupnú chybovosť. Toto je dosiahnuté grafickým rozlíšením chybného dešifrovania spôsobeného prenosovou chybou. V modeli sa chybne dešifrovaná sekvencia prejaví červeným sfarbením danej časti.

Ďalším požiadavkom bola analýza doby resynchronizácie. Toto je taktiež splnené, model zobrazuje *SRD* vypočítané podľa zvolených parametrov.



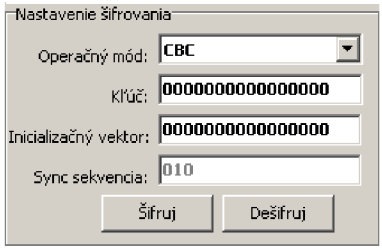
Obrázok 4.4: Grafické rozhranie programu

Rozhranie pozostáva z niekoľkých logicky rozdelených častí. Prvou časťou je nastavenie vstupu. V prípade že užívateľ chce vygenerovať vstupný text, použije sa prvá časť.

Obrázok 4.5: Grafické rozhranie - nastavenie vstupu

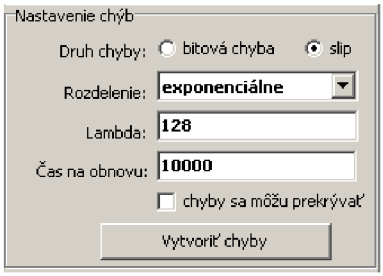
Užívateľ zvolí počet znakov, ktorý má byť vygenerovaný. Podobne môže zvoliť znak, ktorý sa má vygenerovať. Pri použití voľby 'náhodne' sa vygenerujú náhodné znaky zo sady *Base*64. Nastavenie zobrazenia určí, či sa celý proces bude zobrazovať ako znaky (zvolená voľba "znaky") alebo ako postupnosť bitov (volba "bity"). Po stlačení tlačidla "Generovať"' sa v poli "Vstupný text" objaví vstupná postupnosť.

Časť 'Nastavenie šifrovania' slúži pre nastavenie šifrovacieho algoritmu *AES*.



Obrázok 4.6: Grafické rozhranie - nastavenie šifrovania

Základom je zvolenie jedného z implementovaných módov, ktoré sa volia v roletovacom menu. Ďalším vstupom je kľúč šifrovania a inicializačného vektoru, ktoré sú nutné pre šifrovanie. V prípade samosynchronizačných módov SCFB a OCFB je nutné zadať aj synchronizačnú sekvenciu v podobe postupnosti bitov. Použitím tlačidiel "Šifrovať"' alebo "Dešifrovať" sa vstupný text zašifruje, resp. dešifruje. Význam synchronizačnej sekvencie bol popísaný v predchádzajúcich kapitolách.

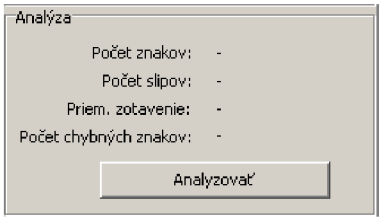


Obrázok 4.7: Grafické rozhranie - nastavenie chýb

V časti ”Nastavenie chýb” sa zvolí druh chyby. Volí sa medzi bitovou chybou (to znamená chybou vzniknutou invertovaním bitu pri prenose) alebo chybou slip.

Táto časť slúži pre nastavenie simulácie prenosového kanálu, v ktorom dochádza k chybám. Ako bolo popísané v predchádzajúcich kapitolách, pre simuláciu je použitá metóda Monte-Carlo s exponenciálnym vstupným rozdelením. Parameter λ určuje strednú vzdialenosť medzi dvoma chybami. Pomocou tohto parametru sa vypočíta pozícia, na ktorú je následne chyba vložená. Nepoužitie voľby ’chyby sa môžu prekrývať’ znamená, že šifre poskytneme pri dešifrovaní dostatok času na to, aby sa obnovilo správne šifrovanie textu (pokiaľ to mód umožňuje). Je to z dôvodu, že náhodné generovanie chýb môže vytvoriť niekoľko chýb v krátkom slede za sebou, čo znehodnotí analýzu. V reálnej prevádzke však môže nastať aj takáto situácia, preto túto simuláciu možno uskutočniť s použitím voľby ”chyby sa môžu prekrývať”. Tlačidlom ”Vytvoriť chyby” sa do šifrovaného textu zanesú chyby.

Dôležitou časťou je analýza, kde sa zobrazujú výsledky jednotlivých simulácii.



Obrázok 4.8: Grafické rozhranie - analýza

V tejto časti nie sú nutné žiadne nastavenia, použije sa tlačidlo ”Analyzovať” a zobrazia sa výsledky analýzy. Pri analýze sa porovnáva vstupný text a výstupný text po bitoch a zisťujú sa rozdiely, ktoré sú potom vyhodnotené. Súčasťou grafického rozhrania sú jednotlivé polia, v ktorých sa zobrazuje spracovávaný text. V poli ”Vstupný text” sa zobrazuje text, s ktorým budeme pracovať, teda vstup pre blokovú šifru. V poli šifrovaný text je výstup z blokovej šifry, teda po šifrovaní zvoleným módom. Toto je text, na ktorom sa simuluje prenosový kanál a do neho sú zanesené chyby. V poli ”Rozšifrovaný text” sa zobrazuje text po dešifrovaní a v tomto texte je skúmaný vplyv chýb počas prenosu.

Ďalšie dve polia slúžia pre analýzu chybovosti - v poli ”Zoznam chýb” sa zobrazujú pozície jednotlivých chýb v šifrovom texte. V poli ’Výsledok analýzy’ sú potom zobrazené resynchronizačné vzdialenosti, respektíve dĺžka šírenie chyby (závisí na zvolenom type chyby).

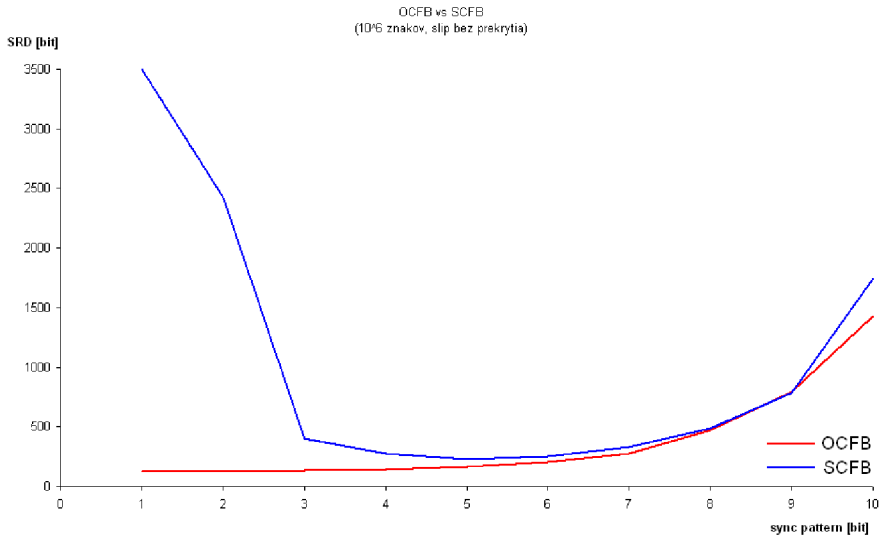
Kapitola 5

Výsledky

V programe sú použité módy pre ktoré platí $h = 1$, pretože táto hodnota je použitá ako východzia pre všetky ďalšie h. Slipy boli generované náhodne použitím softwarového generátora s exponenciálnym rozdelením. Vo výsledku budeme skúmať dobu resynchronizácie v dvoch prípadoch. Prvý je ten, že ďalší slip je vygenerovaný až potom, čo sa systém obnoví synchronizáciu po vzniku predchádzajúceho slipu. Toto je zabezpečené tým, že ďalší slip je vygenerovaný približne po 10^5 bitoch. Tým dosiahneme že systém má po každom slipe dostatok času na synchronizáciu. Toto však nie je reálne chovanie systému, pretože v reálnej prevádzke sa môže napr. vyskytovať niekoľko slipov hneď za sebou. V takomto prípade však vznikne situácia, že prvý slip nemá dostatok času na synchronizáciu a je prekrytý ďalším slipom. Vo výsledkoch budeme uvádzať oba tieto prípady.

Podobne budeme skúmať šírenie chyby v dešifrovanom texte. Budú opäť uvedené dva prípady – s prekrytím chýb a bez prekrytia chyb. Pre tieto prípady platia rovnaké podmienky ako v prípade slipov bez prekrytia a s prekrytím.

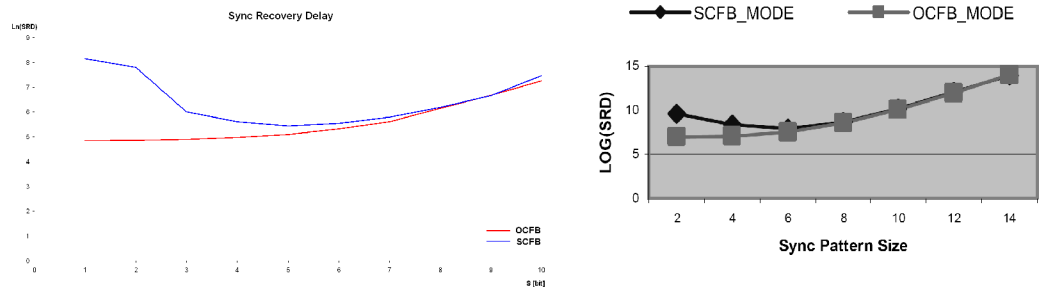
5.1 Výsledky SRD bez prekrytia slipov



Obrázok 5.1: Porovnanie SRD medzi módmi *OCFB* a *SCFB*

Zvolené vstupné rozdelenie bolo exponenciálne. Parameter λ v prípade exponenciálneho rozdelenia určuje strednú vzdialenosť medzi dvoma slipmi. Na základe tohto rozdelenia budú generované slipy a λ pritom určuje strednú vzdialenosť jednotlivých slipov. Podrobnejšou analýzou samosynchronizačných módov sa zaoberá H. Heys vo svojich prácach [6], [7] a [8]. Tieto práce je možné použiť k verifikácii nášho simulačného modelu.

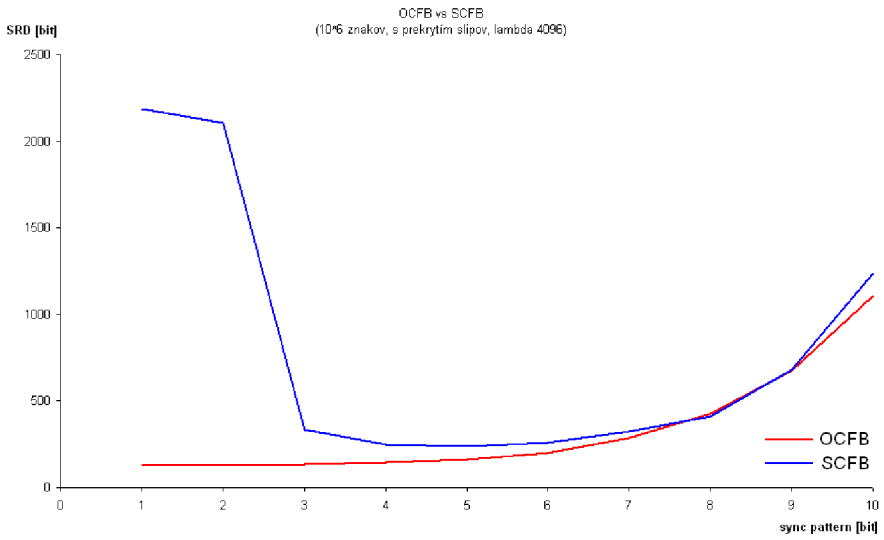
Z predchádzajúceho grafu vidíme, že pri móde *OCFB* narastá *SRD* približne exponenciálne s narastajúcou dĺžkou synchronizačnej sekvencie. Pri malých dĺžkach synchronizačnej sekvencie S $(S < 4)$ má tento mód veľa príležitostí na synchronizáciu. S ďalším narastaním S synchronizačná vzdialenosť narastá približne exponenciálne. Mód *SCFB* má pre malé dĺžky S veľké hodnoty. Je to z dôvodu konštrukcie tohto módu. Pri malých hodnotách S mód zablokuje synchronizáciu veľmi často. V tomto prípade je kritické miesto výskytu slipu. V prípade že zasiahne synchronizačnú sekvenciu, potrebuje mód dlhú dobu na obnovu. To sa prejaví na celkovom *SRD*. S rastúcou dĺžkou S hodnota *SRD* klesá a približne od dĺžky $S = 6$ má rovnaký priebeh ako pri móde *OCFB*. Namerané výsledky sme overili porovnaním s výsledkami zverejnenými v [8]. Z obrázku č. 5.2 je vidieť, že sme dospeli k zhodným výsledkom ako H. Heys.



Obrázok 5.2: Porovnanie získaných výsledkov SRD a výsledkov uvedených v [8]

5.2 Výsledky SRD s prekrytím slipov

V tomto prípade môže dôjsť k tomu, že slipy budú vygenerované príliš blízko seba. Tým nebude mať systém dostatok času na to, aby sa po prvom slipe zosynchronizoval.



Obrázok 5.3: Porovnanie SRD pre módy *OCFB* a *SCFB* so strednou vzdialenosťou slipov $\lambda = 4096$

Tento simulačný model je simuláciou reálnej prevádzky, pretože slipy sa vyskytujú úplne náhodne na základe exponenciálneho rozdelenia. Parameter rozdelenia λ v tomto prípade určuje strednú vzdialenosť jednotlivých slipov. Výsledky sú porovnateľné s prvou simuláciou bez prekrytia slipov, oba módy mali dostatok času na svoju synchronizáciu.

Mód potrebuje po slipe určitú dobu (*SRD*) na obnovenie synchronizácie. Keďže slipy sú generované úplne náhodne, môže nastať prípad, keď budú dva slipy vygenerované príliš blízko seba. To znamená, že prvý slip nebude mať dostatok času aby sa zosynchronizoval. Vo výsledku sa to prejaví ako dlhšia sekvencia poškodeného textu. Preto uvádzam percentuálne porovnanie chybných znakov v otvorenom texte v porovnaní s prípadom, kedy sa slipy neprekrývali.

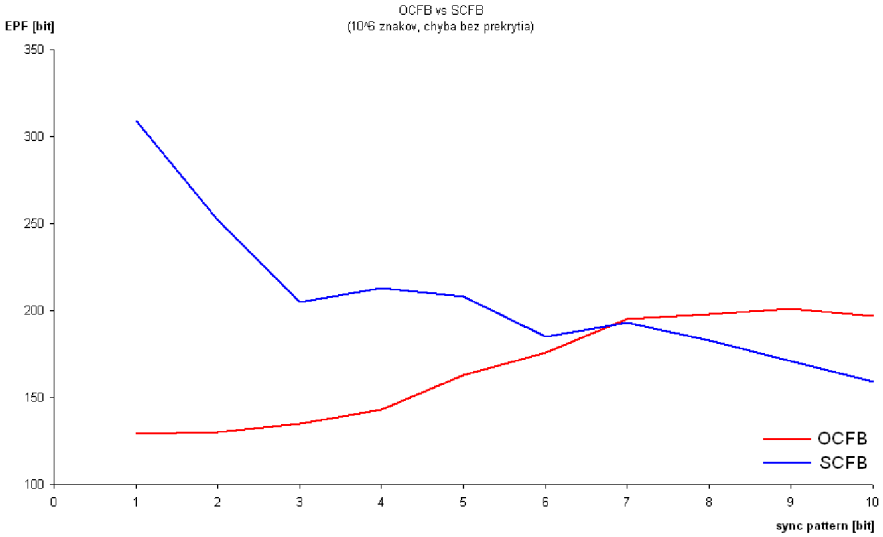
Mód	bez prekrytia	s prekrytím
OCFB	4,860	4,164
SCFB	13,167	9,762

Obrázok 5.4: Porovnanie chybných znakov na výstupe (v %)

Uvedené hodnoty platia pre dĺžky synchronizačných sekvencii od 1 do 10. Menšie hodnoty pre slipy s prekrytím sú spôsobené spomínaným vznikom dvoch slipov príliš blízko seba. V takom prípade nemá systém dostatok času na synchronizáciu prvého slipu, na synchronizáciu ďalšieho slipu však už môže mať dostatok času. To spôsobuje výsledné menšie poškodenie dešifrovaného textu. Podobne môžeme pozorovať, že mód *SCFB* ma výrazne väčšie poškodenie textu z dôvodu, že pri malých hodnotách S dosahuje vysoké hodnoty *SRD* a to je dané návrhom tohto módu.

5.3 Výsledky EPF bez prekrytia chýb

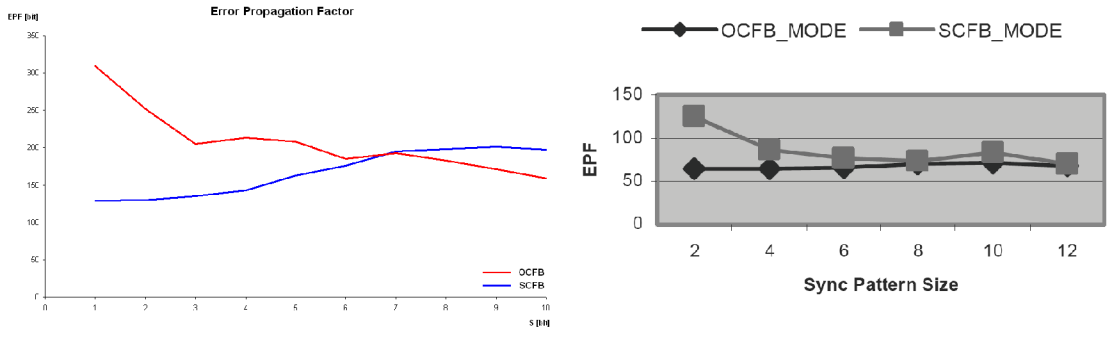
V tomto prípade porovnávame spomínane módy pri šírení chyby (EPF - Error Propagation Factor).



Obrázok 5.5: Porovnanie EPF medzi módmi *OCFB* a *SCFB*

Z grafu môžeme pozorovať, že pre mód *OCFB* má najnižšie hodnoty *EPF* pri malých hodnotách S. So vzrastajúcimi hodnotami S narastá hodnota *EPF* približne lineárne. Je to spôsobené tým, že pri väčších hodnotách S nedochádza k novému šifrovaniu tak často ako pri menších hodnotách. Tým sa zväčšuje aj vplyv chyby v otvorenom texte. Ďalej môžeme pozorovať, že mód *SCFB* má pri malých hodnotách S pomerne veľké hodnoty *EPF*. S ďalším nárastom S šírenie chyby klesá a približne od vzdialenosti $S = 6$ má približne lineárny priebeh. Toto je opäť dané konštrukciou módu *SCFB*. Kritické je taktiež miesto výskytu chyby. Podobne ako pri slipe, aj v prípade chyby, ak sa vyskytne v synchronizačnej sekvencii, znamená to veľmi dlhú obnovu. Dôvodom je zablokovanie synchronizácie počas získavania nového inicializačného vektoru.

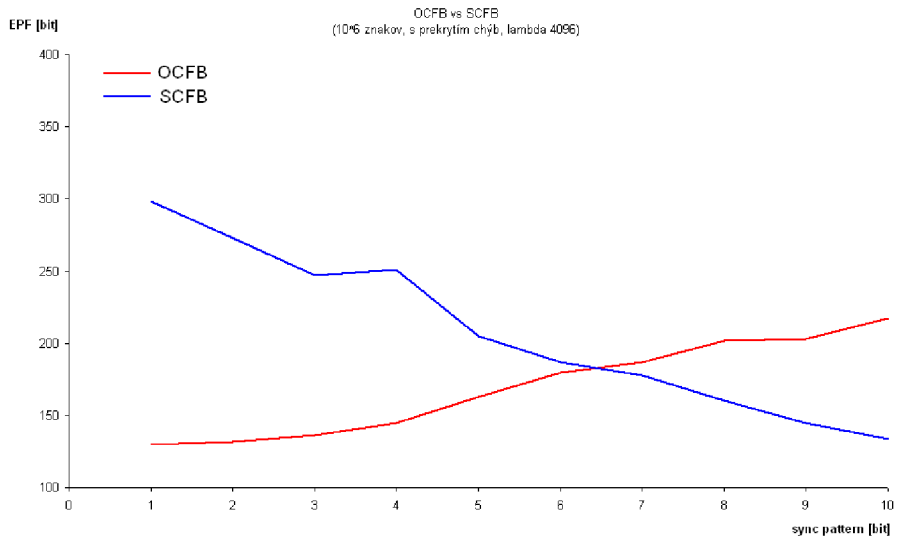
Správnosť našich záverov sme overili porovnaním s grafmi a hodnotami z [8].



Obrázok 5.6: Porovnanie získaných výsledkov EPF a výsledkov podľa [8]

5.4 Výsledky EPF s prekrytím chýb

Pre oba módy boli vykonaná rovnaká simulácia s exponenciálnym rozdelením medzi chybami. Stredná vzdialenosť medzi chybami je $\lambda = 4096$.



Obrázok 5.7: Porovnanie EPF pre módy *OCFB* a *SCFB* so strednou vzdialenosťou chýb $\lambda = 4096$

Namerané hodnoty obnovy po chybe sú podľa predpokladov. Zhodujú sa taktiež s prípadom analýzy chyby bez prekrytia, pretože oba módy majú dostatok času na zotavenie. Pre úplnosť uvádzam tiež percentuálne hodnoty poškodenia otvoreného textu.

Mód	bez prekrytia	s prekrytím
OCFB	2,098	2,063
SCFB	2,610	2,511

Obrázok 5.8: Porovnanie % chybných znakov na výstupe

Podobne ako v v časti 5.3, aj v prípade chýb s prekrytím je výsledné poškodenie dešifrovaného textu mierne nižšie oproti voľbe bez prekrytia. Ako sme spomenuli, príčinou je vznik slipov vo vzdialenosti kratšej ako je *SRD*. Hodnoty pre oba módy v tomto prípade vykazujú minimálne odchýlky.

Kapitola 6

Záver

V práci sme analyzovali samosynchronizačné módy *OCFB* a *SCFB*. Tieto módy boli simulované softwarovo vo vývojovom prostredí Borland. V súčasnosti sú popísané tri samosynchronizačné módy - *CFB*, *OCFB* a *SCFB*. Použitie každého z týchto módov nesie svoje výhody a nevýhody. Najdôležitejším parametrom v prípade samosynchronizačných módov je *SRD* (Synchronization Recovery Delay) - stredná doba resynchronizácie, ktorá určuje za ako dlho po výskyte slipu sa obnoví správne dešifrovanie. Z tohto hľadiska je najvhodnejší mód *CFB*, ktorý má *SRD* rovné dĺžke bloku, jeho hlavnou nevýhodou je však neefektívne využívanie šifry. Tento mód totiž z n možných bitov použije práve jeden. Toto je nežiadúce vo vysokorýchlostných systémoch, kde je potrebné šifrovať veľké množstvá dát a tento mód je nevhodný. Módy *SCFB* a *OCFB* naproti tomu využívajú bity hesla vzniknutého šifrovaním efektívnejšie, problémom je ale že vzrastá hodnota *SRD*. Nárast hodnoty *SRD* spôsobí však vačšiu stratu prenášanej informácie. Na tomto mieste si treba uvedomiť požiadavky na systém. Účelom výskumu je poskytnúť mód, ktorý by poskytol minimálne *SRD* pri efektívnom využití šifrovacieho algoritmu.

Bol vytvorený simulačný model, ktorý spĺňa požiadavky zadania. Simuláciou bolo zistené, že mód *OCFB* má najmenšie *SRD* pri najkratšej synchronizačnej sekvencii. Toto je však za cenu menej efektívneho využitia šifrátora. So stúpajúcou dĺžkou synchronizačnej sekvencie je sa využitie zlepšuje. So stúpajúcou dĺžkou synchronizačnej sekvencie narastá *SRD* približne exponenciálne, preto je na mieste voliť dĺžku synchronizačnej sekvencie podľa požiadaviek na systém. Pokiaľ ide o analýzu šírenia chyby, priebeh je približne konštantný. Mód dosahuje dobre výsledky, pretože obmedzuje šírenie chyby.

Mód *SCFB* naproti tomu má najefektívnejšiu dĺžku synchronizačnej sekvencie približne 4 - 5 bitov pri dĺžke bloku 128 bitov. Od tejto dĺžky sa priebeh podobá priebehu módu *OCFB*. Efektivita využitia šifrátora je väčšia, pretože využíva všetky bity hesla v prípade nájdenia synchronizačnej sekvencie. Aj v tomto prípade je voľba tohto módu oprávnená ak požadujeme vyššiu efektivitu využitia šifrátora, čo sa odvíja od požiadaviek na systém. V prípade analýzy šírenie chyby bolo zistené, že s rastúcou dĺžkou synchronizačnej sekvencie klesá doba na zotavenie z chyby.

Z výsledkov je zrejmé, že či už *SRD* alebo *EPF* závisia na mieste výskytu chyby alebo slipu v šifre. To znamená či zasiahne synchronizačnú sekvenciu alebo blok. Najviac sa táto vlastnosť prejavuje pri móde *SCFB*, čo však vyplýva z jeho návrhu.

Zadanie práce bolo splnené, bol navrhnutý a implementovaný simulačný model pre samosynchronizačné módy. Podobne aj získané výsledky potvrdili teoretické predpoklady. Je možný aj ďalší rozvoj projektu, napríklad rozšírenie parametrov alebo módov ktoré je možné použiť. Takisto v prípade analytického nástroja je možné zlepšenie a zefektívnenie jeho činnosti. Získané výsledky budú použité v ďalšom výskume a zdokonaľovaní samosynchronizačných módov.

Použité výrazy

algoritmus - postup (najčastejšie matematický), ktorý umožňuje na základe mnohých transformácii premenu otvoreného textu na text šifrový a naopak (v prípade symetrických algoritmov)

útok - týmto termínom označujeme v kryptológii pokus a/alebo postup k prelomeniu šifrovacieho algoritmu, pričom vychádza z niečoho známeho (šifrový text, časť kľúča a pod.). Útok by mal byť prakticky vyskúšaný, ale existuje veľa teoretických útokov

bit - základná jednotka informácie, nadobúda hodnôt 0 alebo 1

byte - skupina 8 bitov, ktoré tvoria jeden znak. To znamená, že maximálne môžeme týmto spôsobom zakódovať 256 znakov. Je to základná veľkostná jednotka informácií v počítači

otvorený text - nezašifrovaný text

šifrový text - otvorený text, na ktorý bol aplikovaný niektorý zo šifrovacích algoritmov

šifrátor - osoba alebo zariadenie alebo počítačový program, ktorý šifruje

šifrovanie, dešifrovanie - v symetrickej kryptografii vzájomne opačné postupy - algoritmy, ktoré za pomoci kľúča transformujú otvorený text do formy nečitateľnej pre človeka alebo počítač

Literatúra

[1] `http://csrc.nist.gov/groups/ST/toolkit/BCM/`
National Institute of Standards and Technology
Ústav pre normy a techniku v USA

[2] *STALLINGS, W.: Cryptography and Network Security*
Prentice Hall, Upper Saddle River 2006

[3] *BURDA, K.: Kryptologická a bezpečnostní analýza synchronizace blokových šifer pro MKS v reálných telekomunikačních systémech*
Národní bezpečnostní úřad, Praha 2004

[4] *NEUSCHL, Š., BLATNÝ J., ŠAFAŘÍK J., ZENDULKA J.: Modelovanie a simulácia*
ALFA, Bratislava 1988

[5] *ALKASSAR, A., GERALDY, A., PFITZMANN, B., SAGEDHI, A.: Mode of Operation*
8th International Workshop on Fast Software Encryption, Yokohama, April 2001, Proceedings in LNCS 2355, Springer-Verlag

[6] *HEYS, M.: Analysis of the Statistical Cipher Feedback Mode of Block Ciphers*
IEEE Transactions on Computers, vol. 52, no. 1, Jan. 2003, pp. 77-92

[7] *HEYS, M.: An Analysis of the Statistical Self-Synchronization of Stream Ciphers*
Proceedings IEEE INFOCOM 2001, 22-26 April 2001, Anchorage, Alaska, USA, Volume 2, pp. 897-904

[8] *YANG, F., HEYS, H.: Comparison of Two Self-Synchronizing Cipher Modes*
Queen's 22nd Biennial Symposium on Communications, Kingston, Ontario, Jun. 2004

[9] *BURDA, K.: Resynchronization interval of Self-synchronizing Modes of Block Cipher*
IJCSNS International Journal of Computer Science and Network Security, VOL.7 No.10, October 2007

Príloha A

Advanced Encryption Standard

Algoritmus Rijndael bol jedným z kandidátov na Advanced Encryption Standard (AES). Podľa návrhu by mala táto šifra odolávať prelomeniu niekoľko desaťročí. Úlohou tohto štandardu bolo nahradiť algoritmus DES, ktorý prestal vyhovovať požiadavkám na bezpečné šifrovanie.

Blokovú šifru Rijndael prihlásili do súťaže známi kryptológovia Joan Daemen a Vincent Rijmen. Aj keď tento algoritmus podporuje rôzne dĺžky blokov, pre štandard AES bola stanovená dĺžka 128 bitov. Dĺžka kľúča je voliteľná – 128, 192, 256 bitov, čo je N_k =(4, 6 alebo 8) 32 bitových slov. Rijndael je veľmi flexibilný. Aj keď jeho popis uvedieme v bytoch, je možné efektívne ho zapísať v 32 bitových slovách. Návrh je priamočiarý a za základ sú použité operácie v rôznych algebraických štruktúrach. Pracuje sa s prvkami Galoisovho telesa $GF(2^8)$ a s polynómami, ktorých koeficienty sú prvky z $GF(2^8)$. Príslušné operácie s nimi je možné vykonávať buď tabuľkovo, alebo výpočtom priamo, čo je v prvom prípade výhodné pre implementáciu softwarovú a v druhom prípade hardwarovú. Bytovo orientovaný návrh tiež umožňuje optimalizovať programový kód pre rôzne mikroprocesory. Pre operácie šifrovania a dešifrovania síce nie je možné využiť úplne totožný hradware, značnú časť jeho prvkov však použiť možné je.

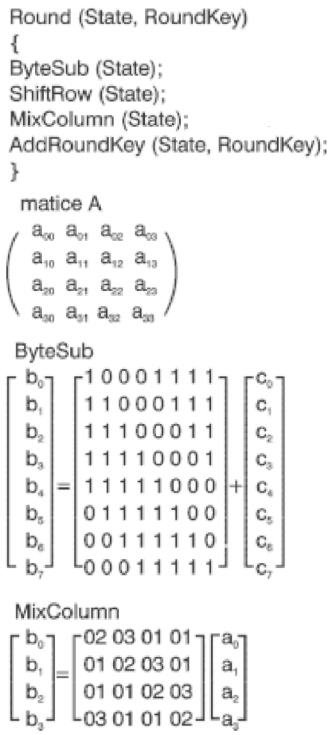
Prvky v Galoisovom teleso $GF(2^8)$ majú osem bitov (b_7, ..., b_0), nereprezentujú však byty, ale polynómy (b_7x^7, ..., b_1x^1,b_0x^0). Násobenie týchto prvkov je preto zavedené nie ako násobenie bytov, ale ako násobenie im odpovedajúcich polynómov, a to modulo m(x) = $x_8 + x_4 + x_3 + x_1 + 1$.

Takže napríklad ”57”(v apostrofoch píšeme bežne vyjadrenie bitov (b_7,...., b_0) krát ”83”je rovné ”C1”, alebo $(x_6 + x_4 + x_2 + x_1 + 1)$ $(x_7 + x_1 + 1)$ = $(x_7 + x_6 + 1)$ mod $m(x)$.

Rijndael pracuje v rundách. Ich počet $N_r = 10, 12, 14$ je určený podľa toho, aký dlhý je šifrovací kľúč. Pre dlhšie kľúče sa teda použije viacej rund. Pred operáciou zašifrovania (alebo v jej priebehu, tzv. *on-the-fly*) sa vypočíta $4 + N_r4$ rundových kľúčov (32bitových slov). Prvé štyri sa ”naxorujú”na otvorený text (tzv. ”whitening”). Potom prebehne N_r rund a v každej z nich sa použijú 4 rundové kľúče. Na začiatku sa 16 bytov otvoreného textu naplní postupne po stĺpcoch (tj. zhora dole a zľava doprava) do matice bytov A = $(a_{ij})_{i=0..3,j=0..3}$ a na ňu sa v rovnakom poradí postupne ”naxoruje”16 bytov tvoriacich prvé štyri rundové kľúče.

Potom prebehne N_R rund, podľa pseudokódu na obrázku 6.1, kde ”State”znamená stav matice A. Pripomeňme, že prvky matice A sú síce byty, ale pri násobení sú chápané ako prvky GF(2^8). Sčítanie týchto prvkov (pri operácii MixColumn) je bežná operácia XOR. Výsledný šifrový text sa opäť vyberá po stĺpcoch z matice A.

Všetky rundy sú rovnaké, až na poslednú, kde je malá zmena – nevykonáva sa operácia MixColumn. Teraz k jednotlivým operáciám z obrázku 6.1:



Obrázok 6.1: Jedna runda v algoritme Rijndael

ByteSub - bytová substitúcia ($a \rightarrow b$), ktorú aplikujeme na každý byte $a_{i,j}$ matice A. Najskôr vypočítame multiplikatívnou inverziou prvu a, tj. $c = a^{-1} \bmod m(x)$, a potom byte c transformujeme na b substitúciou S podľa obrázku 6.1. Substitúciu nemusíme počítať podľa vzorca, ale môžeme si ju uložiť ako pevnú tabuľku.

ShiftRow - vykoná v matici A cyklickú rotáciu jej prvkov jednotlivých riadkoch doľava, a to tak, že prvý nechá bez zmeny, druhý rotuje o jednu pozíciu, tretí o dve a štvrtý o tri.

MixColumns - zvýši zložitosť prvkov v rámci každého stĺpca matice A. Vstupom tejto transformácie sú všetky prvky daného stĺpca (na obrázku je označený a), a výstupom ich novej hodnoty (b). Tak bude napríklad b_0 =”02” $a_0 \oplus$ ”03” $a_1 \oplus$ ”01” $a_2 \oplus$ ”01” a_3.

AddRoundKey - touto operáciou sa nakoniec, opäť po stĺpcoch, ”naxorujú”po riadkoch jednotlivé byty rundových kľúčov, ktoré sú na rade.

Odšifrovanie prebieha trochu inak ako šifrovanie, ale využíva jeho stavebné prvky. Zostáva popísať výpočet rundových kľúčov zo šifrovacieho kľúča.

Šifrovací kľúč *key* (obrázok 6.2) o N_k 32 bitových slovách (4, 6 alebo 8) sa naplní počiatok pomocného poľa 32 bitových slov $W[0...N_k - 1]$. Toto pole sa potom expanduje tak, že každé nové W je vypočítané ako $W[i] = W[i - N_k] \oplus$ temp, kde temp je $W[i-1]$ alebo jeho modifikácia - obrázok Y. Pri modifikácii sa využíva operácie cyklického posuvu bytov slova temp o jeden doprava (RotByte), ďalej známe substitúcie bajtov SubByte, a to aplikované na každý byte premennej temp, a pole konštánt Const[].

Obaja autori dokazujú skvelé vlastnosti stavebných blokov schémy i odolnosť voči lineárnej a diferenciálnej kryptoanalýze. Pretože schéma pre zašifrovanie i odšifrovanie (v hardware) sa líši,

```
for i = Nk to 4*Nr + 3 do
{
  temp = W[i - 1];
  if (i mod Nk = 0)
    temp = SubByte(RotByte(temp)) ⊕ Const[i / Nk];
  if ((i mod Nk = 4) AND (Nk = 8))
    temp = SubByte(temp);
  W[i] = W[i - Nk] ⊕ temp;
}
```

Obrázok 6.2: Expanzia kľúča

nie je tu riziko slabých kľúčov. Ekvivatelncii kľúčov (čo je prípad, kedy rôzne šifrovacie kľúče dávajú i rovnaké sady rundových kľúčov) bráni podľa autorov nelineárnej expanzii.

Pri tejto šifre je cenený hlavne jej priehľadný návrh, založený na rôznych algebraických operáciách. Šifra je flexibilná pri realizácii na rôznych typoch procesorov s veľmi malými nárokmi i veľkosťou kódu. Pri tom vykazuje ešte dostatočnú rýchlosť. Je vhodná i pre paralelné spracovanie a je odolná voči fyzickým typom útokov.

LICENČNÍ SMLOUVA

POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Bc. Marek Kopčan
Bytem: Dolná Poruba 363, 91444, Dolná Poruba
Narozen/a (datum a místo): 18.6.1984, Trenčín

(dále jen "autor")

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií
se sídlem Údolní 244/53, 60200 Brno 2
jejímž jménem jedná na základě písemného pověření děkanem fakulty:
prof. Ing. Kamil Vrba, CSc.

(dále jen "nabyvatel")

Článek 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - ☐ disertační práce
   - ☑ diplomová práce
   - ☐ bakalářská práce

   jiná práce, jejíž druh je specifikován jako .....

(dále jen VŠKP nebo dílo)

Název VŠKP: Simulace a analýza provozu blokové šifry se statistickou samosynchronizací
Vedoucí/školitel VŠKP: doc. Ing. Karel Burda, CSc.
Ústav: Ústav telekomunikací
Datum obhajoby VŠKP: .....

VŠKP odevzdal autor nabyvateli v:

- ☑ tištěné formě - počet exemplářů 1
- ☑ elektronické formě - počet exemplářů 1

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.

4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - ☑ ihned po uzavření této smlouvy
   - ☐ 1 rok po uzavření této smlouvy
   - ☐ 3 roky po uzavření této smlouvy
   - ☐ 5 let po uzavření této smlouvy
   - ☐ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: .....

....

Nabyvatel Autor