UNIVERZA V MARIBORU
FAKULTETA ZA ELEKTROTEHNIKO,
RAČUNALNIŠTVO IN INFORMATIKO

Tomi Jerenko

ANALIZA HITROSTI SIMETRIČNIH BLOČNIH ŠIFRIRNIH ALGORITMOV

Magistrsko delo

Maribor, december 2019

ANALIZA HITROSTI SIMETRIČNIH BLOČNIH ŠIFRIRNIH ALGORITMOV

Magistrsko delo

Študent:	Tomi Jerenko
Študijski program:	Študijski program 2. stopnje
	Informatika in tehnologije komuniciranja
Smer:	Varnost informacijskih sistemov in upravljanje z varnostjo
Mentor:	doc. dr. Marko Hölbl
Somentor:	Marko Kompara
Lektor:	Drago Meglič, prof.

Zahvala

Zahvaljujem se mentorju doc. dr. Marku Hölblu in somentorju asistentu Marku Kompari za pomoč pri izdelavi magistrskega dela.

Posebna zahvala velja staršem, ki so mi omogočili študij in na takšen ali drugačen način podpirali vse moje želje.

Zahvaljujem se tudi vsem prijateljem, ki so mi bili v oporo.

Analiza hitrosti simetričnih bločnih šifrirnih algoritmov

Ključne besede: simetrične šifre, bločne šifre, načini šifriranja, AES, hitrost

UDK: 004.6.056.55(043.2)

Povzetek

Šifriranje je najpogostejši način varovanja podatkov med prenosom preko javnih kanalov. Najpogosteje uporabljamo standard AES; ta spada med simetrične bločne šifre, ki za zagotavljanje zaupnosti, celovitosti in overjanja uporabljajo načine šifriranja. V magistrski nalogi smo s statističnimi metodami primerjali hitrosti različnih implementacij simetričnih bločnih šifer in načinov šifriranja. Ugotovili smo, da je pospešeno šifriranje oz. dešifriranje s šifro AES v povprečju 11,1-krat oz. 14,9-krat hitrejše kot nepospešeno. Nepospešeni AES je tudi hitrejši od ostalih primerljivih šifer. Najhitrejši AEAD-način šifriranja je OCB, med osnovnimi pa CTR pri šifriranju in CBC pri dešifriranju.

Performance Analysis of Symmetric Block Ciphers

Key words: symmetric ciphers, block ciphers, modes of operation, AES, performance

UDC: 004.6.056.55(043.2)

Abstract

Data encryption is the most common way to secure data during transmission over public channels. Mostly used is standard AES, which is a symmetric block cipher. To provide data confidentiality, integrity and authentication block ciphers use modes of operation. In the scope of the thesis statistical methods were used to compare the performance of different implementations of symmetric block ciphers and modes of operation. We have found hardware accelerated AES to be 11.1 and 14.9 times faster than non-accelerated for encryption and decryption respectfully. Non-accelerated AES was the fastest compared to other comparable ciphers. The fastest AEAD mode of operation is OCB, while CTR and CBC are the fastest common modes for encryption and decryption respectfully.

Kazalo

Kazalo slik

Kazalo tabel

Kazalo izsekov programske kode

Kazalo grafov

Seznam uporabljenih kratic

	Angleščina	Slovenščina
IT	Information Technology	informacijska tehnologija
AES	Advanced Encryption Standard	napredni šifrirni standard
CPU	Central Processing Unit	centralna procesna enota (CPE)
ECB	Electronic Codebook	elektronska knjiga kode
CBC	Cipher Block Chaining	veriženje blokov šifre
CTR	Counter	števec
GCM	Galois/Counter Mode	način Galois/števec
CCM	Counter With CBC-MAC	števec s CBC-MAC
OCB	Offset Codebook	knjiga kode z zamikom
XEX	XOR-Encrypt-XOR	XOR-šifriraj-XOR
CTS	Ciphertext Stealing	kraja čistopisa
XTS	XEX-Based Tweaked-Codebook Mode With CTS	XEX-temelječa prirejena elektronska knjiga s CTS
MAC	Message Authentication Code	koda za overjanje sporočila
DES	Data Encryption Standard	standard za šifriranje podatkov
RSA	Rivest-Shamir-Adleman	Rivest-Shamir-Adleman
ECC	Elliptic Curve Cryptography	kriptografija eliptičnih krivulj
DH	Diffie and Hellman	Diffie in Hellman
NIST	National Institute of Standards and Technology	Nacionalni inštitut standardov in tehnologij
XOR	Exclusive OR	izključni ALI
S-box	Substitution box	substitucijska škatla
FIPS	Federal Information Processing Standard	federalni standard za procesiranje informacij
EU	European Union	Evropska unija

	Angleščina	Slovenščina
IV	Initialization Vector	inicializacijski vektor
NI	New Instructions	novi ukazi
CpB	Cycles per Byte	cikli na bajt
IP	Internet Protocol	internetni protokol
AEAD	Authenticated Encryption with Associated Data	overjanje kriptograma s pripadajočimi podatki
DEA	Data Encryption Algorithm	algoritem za šifriranje podatkov
TDEA	Triple Data Encryption Algorithm	trojni algoritem za šifriranje podatkov
BIOS	Basic input/output system	osnovni vhodno-izhodni sistem
RAM	Random Access Memory	bralno-pisalni pomnilnik

1 Uvod

Število naprav, sodelujočih v internetnem omrežju, se v današnjem času hitro povečuje. Leta 2016 je število znašalo približno 16 milijard naprav: IoT-naprave (5,6 milijard), osebni računalniki, prenosniki in tablice (1,6 milijarde), mobilni telefoni (7,3 milijard) in stacionarni telefoni (1,4 milijarde). Zaradi novo nastajajočih aplikacij, poslovnih modelov, padanja cen naprav in povečanega fokusa na industrijski razvoj ter standardizacijo protokolov mobilne telefonije se v prihodnjih letih pričakuje porast naprav večine kategorij, predvsem IoT-naprav. Za leto 2022 je predvideno, da bo v internetnem omrežju sodelovalo 18,1 milijard IoT-naprav, 1,7 milijarde osebnih računalnikov, prenosnikov in tablic, 8,6 milijard mobilnih telefonov ter 1,3 milijarde stacionarnih telefonov, kar skupaj nanese 29 milijard naprav [1].

Ker po eni strani govorimo o porasti števila naprav, pa po drugi strani ne smemo zanemariti dejstva, da sočasno s tem raste tudi število takšnih in drugačnih internetnih zlorab. Skupina CyberEdge je v okviru poročila 'Cyberthreat Defense Report' opravila anketo, v kateri je sodelovalo 1200 kvalificiranih ljudi za IT-varnost iz organizacij z več kot 500 zaposlenimi iz 17 različnih držav in 19 različnih industrij. Ugotovili so, da je bilo omrežje organizacij ogroženo z uspešnim kibernetskim napadom med 78 in 80 odstotki (leta 2017, 2018 ter 2019). Anketirane organizacije so ocenile, da je najtežje braniti naprave, ki so neredno priključene na organizacijsko omrežje (pametni telefoni, tablice ter prenosniki) [10].

Eden izmed najpomembnejših načinov zaščite pred zlorabami je šifriranje podatkov. Glede na to, da je večina naprav, priključenih na internet, omejena z viri, kot sta zaloga energije in procesorska moč, je pomembno, da šifrirni algoritem ni samo varen, temveč tudi primeren iz različnih aspektov učinkovitosti – hitrost in poraba virov. Razvoja novega algoritma ali implementacije obstoječega se v večini primerov ne lotevamo sami

iz varnostnih, časovnih in stroškovnih razlogov. Najboljši šifrirni algoritmi so matematično dokazano varni in odporni na napad z grobo silo z zmogljivimi računalniki. Da se novi šifrirni algoritmi izkažejo za varne, morajo uspešno prestati množico že znanih napadov in tudi napade v prihodnosti. Zato je v večini primerov najprimerneje izbrati kriptografsko knjižnico, ki je testirana in uveljavljena ter hkrati ponuja vse funkcionalnosti, ki jih potrebujemo. Ameriški kriptografer Bruce Schneier pravi [12]: „Vsakdo, tako amater kot profesionalni kriptografer, lahko ustvari algoritem, ki ga sam ni sposoben razbiti. Težko pa je ustvariti takšen algoritem, ki ga nobeden drug ne more razbiti tudi po več letih analize.".

Eden izmed najpogostejših algoritmov za šifriranje podatkov je trenutno AES (Advanced Encryption Standard), saj je standardiziran, podprt v številnih knjižnicah in strojno implementiran na mnogih procesorjih. Iz vidika hitrosti se strojno pospešeno šifiranje uporablja, ker je hitrejše kot programsko, saj za specifičen algoritem uporablja optimiziran in posebej prilagojen set navodil ali pa celo preloži breme s procesorja na za to posebej prilagojen in temu namenjen čip [23].

Bločni šifrirni algoritem, kot je AES, se uporablja v kombinaciji z različnimi načini šifriranja (angl. Modes of Operation), saj šifra sama po sebi ne ponuja zadostne varnosti pri šifriranju podatkov, daljših od dolžine njenega bloka, in drugih lastnosti, ki jih želimo. Načini šifriranja dodajo šifri lastnosti, kot so: zaupnost (za večjo količino podatkov, kot je dolžina bloka), celovitost in sposobnost overjanja šifriranih podatkov, nekateri pa celo spremenijo bločno šifro v tokovno [11].

1.1 Identifikacija in opredelitev problema

Na spletu je moč najti različne kriptografske knjižnice. Nekaj od teh bomo uporabili za primerjavo hitrosti delovanja različnih šifrirnih algoritmov in načinov šifriranja. Glede na to, da je AES standardiziran in najbolj razširjen šifrirni algoritem, bomo proučili razliko hitrosti pri strojno pospešenem in nepospešenem izvajanju. Algoritem AES bomo primerjali tudi z nekaterimi drugimi primerljivimi šifrirnimi algoritmi pri nepo-

spešenem izvajanju. Ker bločne šifre uporabljamo v kombinaciji z različnimi načini šifriranja, bomo prav tako proučili njihovo hitrost v kombinaciji s strojno pospešenim izvajanjem šifre AES.

Cilj naloge je na zanesljiv in natančen način izmeriti hitrost šifriranja in dešifriranja podatkov z različnimi konfiguracijami in jih na strukturiran način med seboj primerjati. Merjenje časa izvajanja algoritma je neprimerno, ker je hitrost izvajanja procesorja odvisna od njegove frekvence delovanja (ta se lahko dinamično spreminja), in je zato primerneje meriti takte procesorske ure (angl. CPU clock cycles). Ker bodo testi izvajani na napravi z operacijskim sistemom, je število ciklov izvajanja opravil odvisno od časovnega razporejevalnika (angl. scheduler). Razporejevalnik izvaja 'pre-emption', kar pomeni, da je opravilo lahko kadar koli začasno prekinjeno, da se izvede neko drugo opravilo z višjo pomembnostjo (angl. priority) [13]. Te prekinitve so zelo kratke, vendar imajo pri merjenju hitrosti izvajanja hitrih operacij zelo velik vpliv. Svoje teste želimo torej oblikovati tako, da se čim bolj izognemo vplivu prekinitev na končne meritve. Navsezadnje ne smemo zanemariti niti izbire prevajalnika, saj ima ta prav tako vpliv na hitrost izvajanja algoritmov [14].

1.2 Namen

V magistrskem delu bomo predstavili šifre in njihov namen. Razložili bomo algoritem AES in njegovo strojno pospešeno delovanje, delovanje načinov šifriranja (ECB, CBC, CTR, GCM, CCM, OCB in XTS) in primerjali učinkovitost njihovih implementacij v kombinaciji s strojno pospešenim in nepospešenim izvajanjem algoritma AES128 med različnimi knjižnicami. Primerjali bomo tudi učinkovitost implementacije algoritma AES128 v načinu ECB z ostalimi simetričnimi bločnimi šiframi (TwoFish, Serpent, Camellia, DES in 3DES) med različnimi knjižnicami. Eksperiment bomo zasnovali tako, da se bomo izognili vplivu časovnega razporejevalnika oz. vplivu drugih spremenljivk na hitrost izvajanja in merili število procesorskih taktov, potrebnih za šifriranje in dešifriranje različnih dolžin vhodnih podatkov. Predstavili bomo, kako na zanesljiv način izmerimo število procesorskih ciklov, potrebnih za izvajanje omenjenih algorit-

mov in načinov šifriranja. Dobljene meritve bomo s statističnimi testi primerjali in pokazali, katere implementacije šifer in načinov šifriranja so najhitrejše.

1.3 Raziskovalna vprašanja in hipoteze

Na podlagi identificiranega problema in pregleda literature smo oblikovali naslednja raziskovalna vprašanja:

- **RV$_1$**: Kakšna je razlika v hitrosti izvajanja med strojno pospešenim in nepospešenim izvajanjem algoritma AES?
- **RV$_2$**: Kakšna je razlika v hitrosti izvajanja med algoritmom AES in ostalimi izbranimi simetričnimi bločnimi šifrirnimi algoritmi?
- **RV$_3$**: V kolikšni meri izbrani načini šifriranja vplivajo na hitrost izvajanja algoritma AES?

Na osnovi raziskovalnih vprašanj smo oblikovali naslednje hipoteze:

- **H$_1$**: Strojno pospešeno izvajanje algoritma AES je hitrejše od nepospešenega.
- **H$_2$**: Nepospešeno izvajanje algoritma AES je v primerjavi s primerljivimi simetričnimi bločnimi šiframi najhitrejše.
- **H$_3$**: Izbira načina šifriranja vpliva na hitrost algoritma AES.

1.4 Predpostavke in omejitve

Da bi omejili obseg vseh možnih primerjav in istočasno omogočili nepristransko primerjavo, se bomo omejili na naslednje:

- pri izbiri algoritmov se bomo omejili na simetrične bločne šifre,

- zaradi velike izbire kriptografskih knjižnic se bomo omejili na 4 - glavni kriteriji pri izbiri so jezik implementacije (C ali C++) in šifrirni algoritmi ter načini šifriranja, ki jih knjižnice ponujajo,
- omejili se bomo na uporabo prevajalnika G++,
- primerjali bomo ECB-, CBC-, CTR-, CCM-, GCM-, OCB- in XTS-načine šifriranja,
- pri primerjavi strojno pospešenega in nepospešenega izvajanje algoritma AES se bomo omejili na AES128,
- algoritem AES128 bomo primerjali samo z algoritmi TwoFish, Serpent, Camellia, DES in 3DES v načinu ECB,
- AES128 bomo primerjali z ostalimi šiframi samo pri nepospešenem izvajanju,
- teste bomo izvajali na enem procesorskem jedru v eni niti.

1.5 Metode dela

Uporabili bomo naslednje znanstveno-raziskovalne metode:

- S pomočjo pregleda literature bomo proučili šifre in načine šifriranja. Nato bomo izbrali šifre, načine šifriranja ter knjižnice in zanje zasnovali okolje za izvedbo eksperimenta.
- S pregledom literature bomo raziskali, kateri dejavniki vplivajo na hitrost izvajanja šifer, in poskušali nepristransko zasnovati eksperiment.
- Z eksperimentom bomo pridobili meritve in poskusili odgovoriti na raziskovalna vprašanja RV1-RV3.
- S statistično analizo bomo potrdili oz. zavrgli hipoteze H1-H3.

1.6 Struktura

Najprej bomo v poglavju 2 predstavili vrste šifer, razložili njihovo delovanje, jih kategorizirali glede na uporabo in namen ter predstavili njihove prednosti in slabosti. Na kratko bomo pregledali zgodovino in predstavili povod za razvoj modernih šifer ter se nato poglobili v asimetrične in simetrične šifre. Nazadnje se bomo še dodatno poglobili v simetrične šifre, nato pa posamično spoznali bločne in tokovne šifre.

V poglavju 3 bomo predstavili šifre AES, DES, 3DES, Camellia, TwoFish in Serpent. Osredotočili se bomo na algoritem AES in ga spoznali podrobneje. Dodatno bomo razložili še AES-NI-set ukazov in njegovo delovanje.

V poglavju 4 bomo predstavili, kaj so načini šifriranja, kako delujejo, zakaj jih potrebujemo in kakšne lastnosti imajo. Podrobneje bomo obravnavali načine ECB, CBC, CTR, GCM, CCM, OCB in XTS.

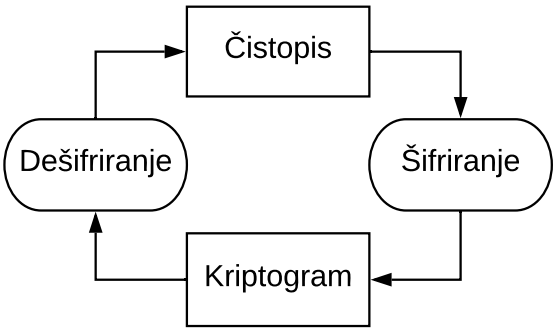
V poglavju 5 bomo predstavili napravo, na kateri bomo izvedli teste za preizkušanje zmogljivosti. Predstavili bomo tudi nastavitve za pripravo na testiranje, razložili bomo strukturo in delovanje testov, prav tako pa tudi način poganjanja testov.

V poglavju 6 bomo predstavili uporabljene statistične teste in predpripravo podatkov. Rezultate testov bomo na strukturiran način preučili in ugotovitve povzeli.

V zadnjem, 7. poglavju bomo na kratko povzeli namen magistrske naloge, rezultate, ugotovitve in probleme ter navedli možnosti za prihodnje raziskave.

2 Šifre

Ko govorimo o šifriranju sporočil, imamo v mislih model, prikazan na sliki 2.1. Šifre so sestavljene iz dveh primarnih procesov: iz procesa šifriranja, ki pretvori čistopis v kriptogram, in procesa dešifriranja, ki deluje obratno in pretvori kriptogram nazaj v čistopis. Glavni problem, ki ga šifriranje sporočil naslavlja, je zasebna izmenjava sporočila preko javnih kanalov. Preden sporočilo posredujemo prejemniku, ga želimo pretvoriti v takšno obliko, iz katere nihče drug, razen našega naslovnika, ne more izluščiti vsebine sporočila. Pretvorjeno sporočilo v neberljivi obliki pošljemo preko javnega (nezaupnega) kanala svojemu naslovniku, ta pa ga nato znova pretvori v prvotno obliko in pridobi vsebino posredovanega sporočila.



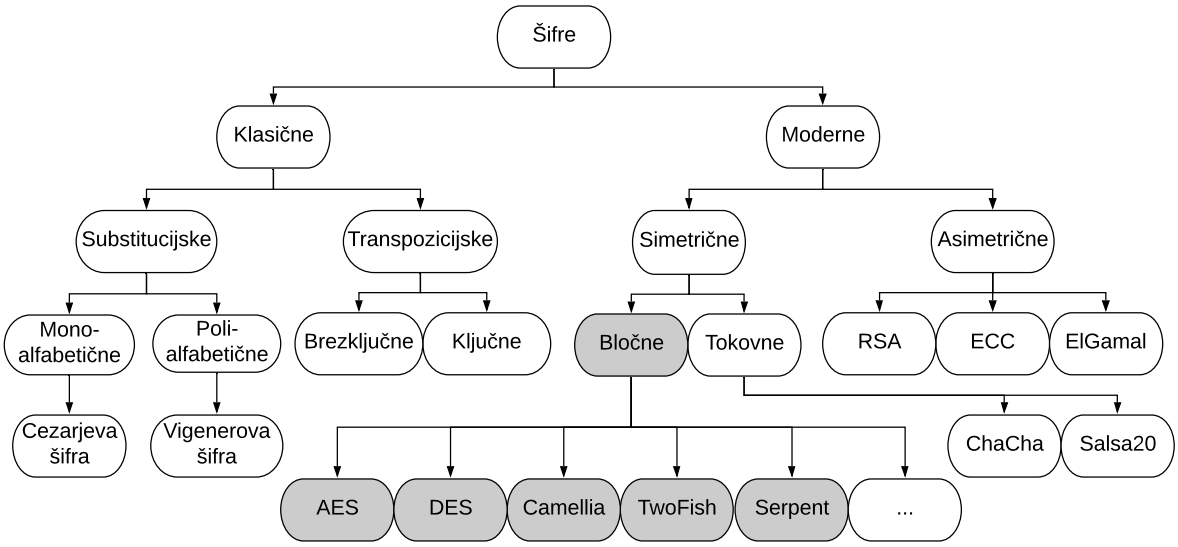
Slika 2.1: Proces delovanja šifre

Do nedavnega je bila osrednji del kriptografije samo zaupnost sporočil, kar pomeni pošiljanje sporočil v obliki, iz katere prisluškovalec ali prestreznik ne more pridobiti informacij, dandanes pa so se zaradi izmenjave digitalnih sporočil temu primerno uveljavile tehnike za zagotavljanje celovitosti (npr. zgoščevalne funkcije) in overjanja vsebine (npr. digitalni podpisi) [21]. Šifre delimo predvsem na 2 glavni kategoriji, prikazani sta na sliki 2.2: klasične in moderne [30].

Klasične šifre so šifre iz preteklosti in danes veljajo za neprimerne, saj kriptogram večine šifer razkriva statistične informacije čistopisa [21]. Delijo se na substitucijske,

kar pomeni menjavo znakov z drugimi znaki (menjava črk z drugimi črkami), in transpozicijske, kar pomeni menjavo vrstnega reda znakov znotraj sporočila. Substitucijske se naprej razdelijo še na monoalfabetične, ki uporabljajo samo eno vrsto substitucije za celotno sporočilo (primer: Cezarjeva šifra), in monoalfabetične, ki uporabljajo več različnih substitucij za eno sporočilo (primer: Vigenerova šifra)[30].

Glavna prednost modernih šifer pred klasičnimi je, da je njihova osnovna enota šifriranja bajt ali bit. Posledično so primernejše za šifriranje informacij v digitalni obliki. Delijo se na simetrične (podpoglavje 2.3) in asimetrične (podpoglavje 2.2). V eksperimentalnem delu se bomo osredotočili samo na simetrične bločne šifre, ki so na sliki 2.2 zasenčene.



Slika 2.2: Klasifikacija šifer [30]

Simetrične šifre veljajo za hitrejše, saj uporabljajo preprostejše operacije kot asimterične. Poleg kompleksnejših matematičnih operacij, asimetrične šifre tudi računajo z mnogo večjimi števili. Simetrične šifre potrebujejo krajše ključe za doseganje enake stopnje varnosti. Algoritmi zagotavljajo različne stopnje varnosti, te pa so odvisne od algoritma samega ter dolžine uporabljenega ključa. Varnost definiramo na naslednji način: dva algoritma sta primerljive stopnje varnosti za ključa A in B, če je za razkritje skrivnih ključev A in B potrebno približno enako delo z rabo enake računalniške moči [20]. V tabeli 2.1 so podane bitne dolžine ključev s primerljivo stopnjo varnosti za algoritme AES (simetrični), RSA (asimetrični) in ECC (asimetrični)[28].

Tabela 2.1: Primerjava dolžin ključev pri enaki stopnji varnosti za šifro AES ter kriptosistema RSA in ECC [28]

AES (št. bitov)	RSA (št. bitov)	ECC (št. bitov)
128	3072	256
192	7680	384
256	15360	521

Simetrične in asimetrične šifre delujejo na različne načine, njihovo delovanje pa bomo podrobneje razložili v podpoglavjih 2.3 in 2.2. Prve so hitrejše in se zato uporabljajo za šifriranje sporočil, druge pa se zaradi posebnih lastnosti uporabljajo za izmenjavo ključev in digitalno podpisovanje. Zaradi različnih prednosti in slabosti jih zato kombiniramo. Eden izmed pogostih modelov rabe je: asimetrične šifre uporabimo za vzpostavitev simetričnega ključa med entitetami v komunikaciji, na podlagi katerega je s simetrično šifro varovana vsa nadaljnja komunikacija.

2.1 Zgodovina

Pojav šifer sega zelo daleč v zgodovino. Že v Svetem pismu je bila uporabljena šifra, imenovana Atabašev kod (angl. Atabash code), ki je delovala na preprostem principu menjave črk: menjava prve črke abecede z zadnjo, druge s predzadnjo itn. Čez nekaj časa je v svojih vojaških podvigih svojo šifro uporabljal tudi Julij Cezar, zato se ta šifra še danes imenuje Cezarjeva šifra. Delovala je na principu zamika črk za vrednost oz. ključ v desno (primer: če je ključ 4, A postane D, B postane E itn.)[21].

Uporaba šifer je bila zaradi političnih spletk v Evropi pogosta tudi v srednjem veku. Okrog leta 1470 je Leon Alberti izumil metodo, s katero je želel preprečiti uporabo frekvenčne analize (angl. frequency analysis) kriptograma za razkritje vsebine sporočila. Njegova ideja je bila uporabiti polialfabetično šifro, ki pri šifriranju v različnih delih uporablja različne abecede. Njegova metoda je bila napačno pripisana francoskemu kriptograferju Blaisu Vigeneru (org. Blaise de Vigenere), zato se ta šifra danes imenuje Vigenerova šifra [21].

Uporabljeno je bilo veliko fizičnih naprav, ki so bile namenjene šifriranju sporočil. Pred izumom programabilnega računalnika se je uporabljala slavna Enigma, in sicer jo je med drugo svetovno vojno uporabljala nemška vojska. To je bila rotorska naprava, ki je uporabljala polialfabetično šifro. Njeno prvotno različico so razbili poljski kriptoanalitiki, naprava pa je bila nato nadgrajena. Britanski kriptoanalitiki, ki jih je vodil Alan Turing, so razbili tudi nadgrajeno verzijo in tako izjemno vplivali na razplet druge svetovne vojne [21].

Razvoj digitalnih računalnikov je omogočil razvoj kompleksnejših kriptosistemov. V primerjavi s predhodnimi šiframi, pri katerih se je šifriralo samo tekstovna sporočila, je z modernimi šiframi mogoče šifriranje kakršnih koli vsebin, ki jih je mogoče pretvoriti v digitalno obliko, saj so šifre začele operirati na nivoju bitov. Ta razvoj je revolucionaliziral kriptografijo. Uporaba računalnikov je tako zahtevala razvoj kompleksnejših kriptosistemov, saj je rastoča računalniška moč pospešila njihovo razbijanje [21].

Leta 1976 sta Diffie in Hellman razvila prvi uporabni asimetrični protokol za izmenjavo ključev. Leto kasneje so Rivest, Adleman in Shamir razvili kriptosistem RSA, ki je bil drugi asimetrični algoritem. Ker oba algoritma uporabljata zelo velik ključ, se je razvila kriptografija eliptičnih krivulj (angl. ECC - Elliptic Curve Cryptography). Eliptične krivulje kot kriptografska platforma so bile prvič predlagane leta 1985, ko sta Neil Koblitz in Victor Miller neodvisno predlagala njihovo rabo nad končnimi polji (angl. Finite Fields) in ostalimi pripadajočimi množicami. Do sedaj so se izkazale za zelo uspešne, saj so hitrejše in potrebujejo manjšo dolžino ključev kot RSA-algoritem [21].

Zaradi pomembnosti rabe kriptografije je nastala potreba po standardizaciji. Tako so leta 1976 izdali standard za šifriranje podatkov (angl. Data Encryption Standard, DES), ki je bil posvojen kot standard simetričnega šifriranja. Originalna različica DES ne velja več za varno, saj je bilo nanjo izvedenih veliko uspešnih napadov, prvi pomembnejši že leta 1999. Temu je sledilo tekmovanje za izbiro varnega naslednika; zmagovalno rešitev sta razvila Vincent Rijmen in Joan Daemen, algoritem pa se je imenoval Rijndaelova šifra (angl. Rijndael cipher). Leta 2001 je nacionalni inštitut za standarde in tehnologijo v ZDA posvojil standardizacijo Rijndaelove šifre, ki se danes imenuje AES in je danes najpogosteje uporabljena simetrična bločna šifra [21].

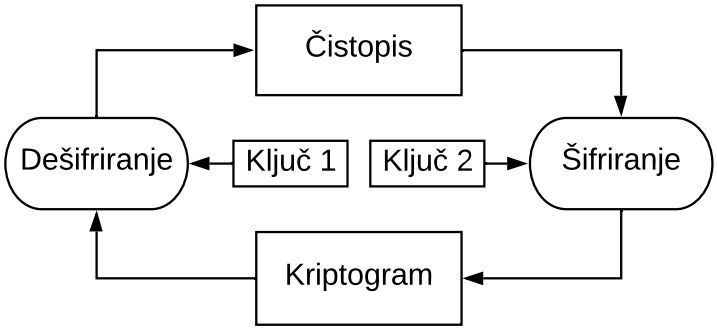
V zadnjih dvajsetih letih se je začela razvijati ideja kvantnih algoritmov. Možnost razvoja uporabnega kvantnega računalnika je tako pozvala raziskovalce k raziskovanju asimetričnih metod, ki bi bile varne pred kvantnimi računalniki (saj asimetrični algoritmi temeljijo na kompleksnih matematičnih problemih, ki bi jih lahko učinkovito rešili z uporabo kvantnih računalnikov). Tako je v zadnjih letih to področje postalo aktivna veja raziskav [21].

2.2 Asimetrične šifre

Glavna ideja asimetričih šifer oz. šifer z javnim ključem (angl. public key ciphers) je lastnost funkcije s stranskimi vrati (angl. trapdoor function). To je funkcija, s katero je računanje v eno smer preprosto, v drugo pa izjemno težko (časovno zahtevno), razen če poznamo stranska vrata. To pomeni, da asimetrična šifra uporablja 2 ključa: javni in zasebni. Javni ključ predstavlja vrednost, ki jo uporabnik deli z vsemi, zasebni ključ pa vrednost, ki jo uporabnik ohrani skrivno. Sam sistem, prikazan na sliki 2.3, deluje tako, da čistopis šifrira z enim, dešifrira v prvotno obliko pa z drugim ključem. Pri tem velja pravilo, da je vedno mogoče šifrirati samo z enim, dešifirati pa le z drugim ključem (šifrirati in nato dešifrirati samo z enim ključem ni mogoče). Vrstni red uporabe ključev je odvisen od primera uporabe: v primeru digitalnega podpisovanja šifriramo najprej z zasebnim, v primeru zagotovitve zaupnosti pa z javnim ključem.

Digitalno podpisovanje predstavlja tehniko, s katero želimo preveriti, ali je sporočilo res ustvarila določena entiteta. Digitalni podpis vsebine temelji na tajnosti zasebnega ključa. Ko je vsebina šifrirana z zasebnim ključem, lastnosti funkcije s stranskimi vrati zagotavlja, da jo je mogoče odšifrirati samo z javnim ključem. To pomeni, da vsak imetnik javnega ključa lahko preveri, ali je bila vsebina šifrirana s pripadajočim zasebnim ključem.

V primeru zagotavljanja zaupnosti se vsebina šifrira z javnim ključem. Zaradi lastnosti, ki jo omogoča funkcija s stranskimi vrati, je imetnik zasebnega ključa edini, ki je sposoben šifrirano vsebino dešifrirati.



Slika 2.3: Delovanje asimetrične šifre

Asimetrični sistemi so počasnejši od simetričnih šifer, saj temeljijo na računsko zahtevnih matematičnih problemih. Uporabljajo tudi večje dolžine ključev kot simetrični (tabela 2.1). Zato so večinoma uporabljeni za izmenjavo ključev simetričnih šifer in digitalno podpisovanje.

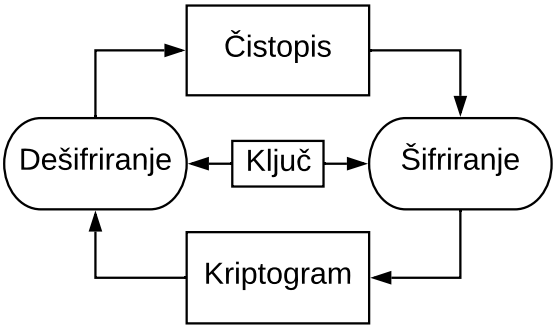
Sistemi, kot so RSA, ECC, Diffie-Hellman in ElGamal, temeljijo na principih računsko zahtevnih nerešenih matematičnih problemih, kar pomeni, da rešitev ne obstaja oz. da ne poznamo rešitve, ki bi ta problem rešila s kompleksnostjo polinomskega časa. Varnost teh algoritmov se zato meri v časovni kompleksnosti. Primera matematičnih problemov, na katerih temeljijo omenjeni sistemi, sta:

- Izračun diskretnega logaritma: če imamo množico G, generator množice g in element h iz množice G, želimo najti diskretni logaritem h osnove g v množici G. Drugače rečeno: želimo najti število x, da bo $x \coloneqq \log_g h \implies g^x = h$. Težavnost tega problema je odvisna od množice G [15]. Na tem problemu temeljijo ECC, ElGamal in Diffie-Hellman [21].

- Faktorizacija velikih števil v praštevila: za pozitivno celo število $n \geq 2$ želimo najti njegove faktorje $n \coloneqq p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_k^{\alpha_k}$, pri čemer p_i predstavlja praštevilo reda α_i. Faktorizacija trenutno velja za $P = NP$ problem, za katerega trenutno ni znano, kako zahteven je in ali obstaja preprosta rešitev, ki ga rešuje. Na njem temelji kriptosistem RSA [21].

2.3 Simetrične šifre

Simetrični sistemi v nasprotju z asimetričnimi uporabljajo samo en ključ. Kot je prikazano na sliki 2.4, simetrične šifre uporabljamo tako, da čistopis šifriramo in dešifriramo z enakim ključem. Ker so simetrične metode veliko učinkovitejše pri hitrosti in prostorski zahtevnosti kot asimetrične, se uporabljajo predvsem za šifriranje podatkov.

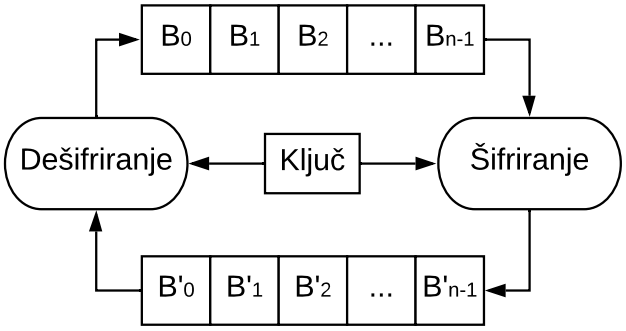


Slika 2.4: Delovanje simetrične šifre

2.3.1 Bločne šifre

Bločne šifre delujejo na principu iterativne uporabe preprostih operacij: permutacij in substitucij. Operacije izvajajo na blokih čistopisa dolžine n. Dolžina čistopisa, ki ga želimo šifrirati, mora biti enaka dolžini n ali njenemu večkratniku. Ker dolžine čistopisov v praksi teh pogojev ne zagotavljajo vedno, se uporabi tehnika (bitnega) zapolnjevanja (angl. padding). Zapolnjevanje čistopisa tukaj pomeni dodajanje bitov k čistopisu, da bo njegova dolžina enaka n oz. njen večkratnik [21]. Delovanje bločne šifre je prikazano na sliki 2.5, in sicer $B_0B_1 \ldots B_{n-1}$ predstavlja zaporedje bajtov bloka, (čistopis) dolžine n, $B'_0B'_1 \ldots B'_{n-1}$ pa njihovo preslikavo (kriptogram).

Uporaba preprostih operacij omogoča efektivno implementacijo šifre. Blok čistopisa je podvržen več iteracijam preslikav, zato te šifre imenujemo tudi iteracijske bločne šifre ali šifre produktov. Vsaka iteracija, ki lahko uporablja drugačen ključ (izpeljan iz glavnega ključa), se imenuje runda (angl. round). Za konstrukcijo iteracijskih šifer obstaja vrsta metod, med njimi bomo omenili substitucijsko-permutacijsko mrežo in Feisteljevo mrežo [21].



Slika 2.5: Delovanje bločne šifre

Substitucijsko-permutacijska mreža, ki jo uporablja šifra AES, temelji na iteracijah substitucij in permutacij. Za substitucije so uporabljene t. i. 'S-box' substitucijske škatle (angl. Substitution-box), ki so namenjene substitucijam majhnih blokov bitov z drugimi bloki bitov (1 bajt predstavlja blok 8 bitov). Permutacije so tiste, ki vplivajo na vse bite. Zaželeno je, da so izhodi permutacij čim bolj raznoliki, saj to povzroči raznolikost substitucij blokov bitov v naslednjih rundah. Ker se operacije izvajajo v rundah, je pomembno, da se jih izvede čim več, saj tako dosežemo, da kriptogram izgleda naključno in da izgubi morebitne vzorce, ki bi lahko razkrili kakšne informacije o čistopisu. Prav tako se blokom bitov v vsaki rundi dodajo bloki bitov ključa in tako šifrirano sporočilo naredi odvisno od njega [21].

Feisteljeva mreža (angl. Feistel network), na kateri temelji DES, predstavlja kriptografsko tehniko, ki se uporablja v konstrukcijah bločnih šifer. Deluje na način razdelitve bloka na dva enaka dela, pri čemer je na vsakem delu uporabljeno šifriranje v več iteracijah. Vsaka iteracija, njihovo število je odvisno od same šifrine implementacije, implementira permutacije in kombinacije, izpeljane iz ključa ali primarne funkcije [16]. Prednost Feisteljeve mreže je, da sta algoritma šifriranje in dešifriranja zelo podobna ali celo identična, saj tako zmanjšata dodatno delo pri implementaciji dešifriranja. Omogoča tudi dešifriranje vsebine z uporabo ključa v obratnem vrstnem redu [21].

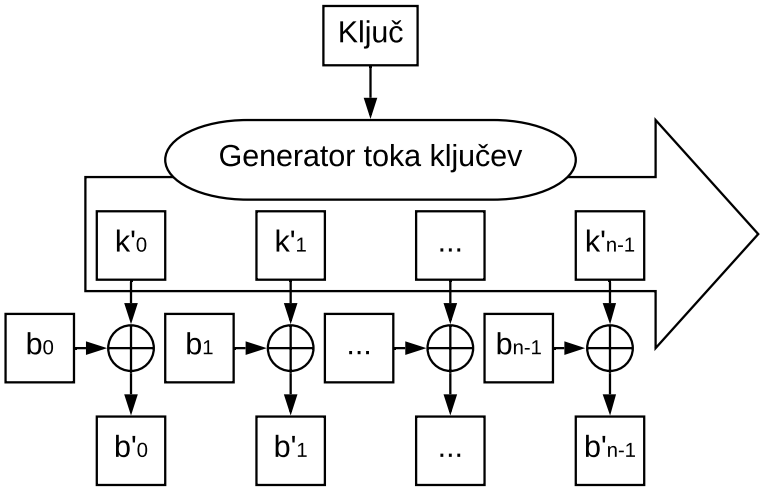
2.3.2 Tokovne šifre

Tokovne šifre v primerjavi z bločnimi ne potrebujejo prisotnosti celotnega bloka čistopisa pri šifriranju ali celotnega bloka kriptograma pri dešifriranju, temveč lahko to storijo

bit po bit. To lastnost dosežemo tako, da v procesu uporabimo generator psevdonaključnega ključa (angl. pseudorandom key generator). Takšne šifre šifrirajo bit po bit čistopisa in so tipično hitrejše od bločnih. Posledično je mogoče sinhrono šifriranje in dešifriranje, saj se v primerjavi z bločnimi šifrira ali dešifrira vsak bit posebej. Zato je lahko šifrirani bit takoj prenešen k prejemniku, ta pa ga takoj po prejetju dešifrira, saj mu ni treba čakati na prenos celotnega sporočila [21].

Generator psevdonaključnega ključa je psevdonaključna funkcija, ki iz podanega ključa generira tok ključa (angl. key stream). Ta tok izgleda kot naključno zaporedje bitov in je lahko poljubne dolžine, zato te šifre veljajo za poskus približka šifriranju z enkratno podlago (angl. one-time pad). Šifriranje bitov čistopisa in toka ključa izvedemo na način: $k'_0 \oplus b_0, k'_1 \oplus b_1, \ldots, k'_{n-1} \oplus b_{n-1} = b'_1 b'_2 \ldots b'_n$ [21].

Slika 2.6 predstavlja delovanje tokovne šifre, pri čemer število n predstavlja število bitov, ki jih želimo šifrirati, $b_0 b_1 \ldots b_{n-1}$ bite čistopisa, $k'_0 k'_1 \ldots k'_{n-1}$ psevdonaključne bite, izpeljane iz ključa s pomočjo generatorja, $b'_0 b'_1 \ldots b'_{n-1}$ šifrirane bite in simbol $\oplus$ XOR-operacijo.



Slika 2.6: Delovanje tokovne šifre

3 Bločne šifre

3.1 AES

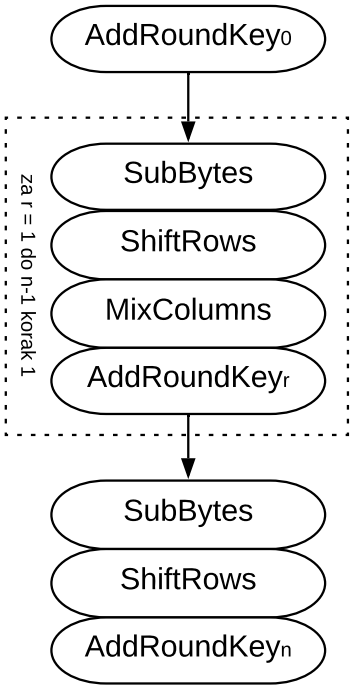
Nacionalni inštitut standardov in tehnologij (angl. NIST) je 26. novembra 2001 v publikaciji Federalni standardi za procesiranje informacij 197 (angl. FIPS 197) izdal napredni šifrirni standard AES (angl. Advanced Encryption Standard) [31]. Šifra se je prvotno imenovala Rijndael, njena avtorja, ki sta jo prvič izdala 11. junija 1998, pa sta Joan Daemen in Vincent Rijmen [24].

AES je bločna šifra, ki uporablja 128-, 192- ali 256-bitne ključe (na kratko AES128, AES192 in AES256) za šifriranje in dešifriranje 128-bitnih blokov v 10, 12 ali 14 rundah za omenjene dolžine ključev. Osnovna enota procesiranja je bajt, ki predstavlja zaporedje 8 bitov. Bajti so razdeljeni v matrike velikosti 4×4, za njihovo šifriranje in dešifriranje pa so uporabljene naslednje glavne funkcije [31]:

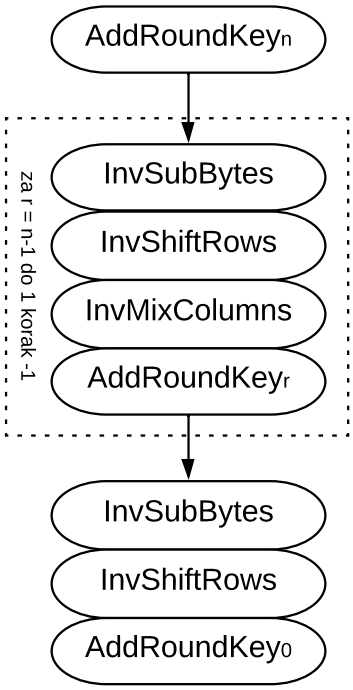
- $AddRoundKey()$: transformacija stanja, pri katerem je z operacijo $\oplus$ (XOR) dodan ključ runde (angl. round key). Dolžina ključa runde je vedno 128 bitov.
- $MixColumns()$ in $InvMixColumns()$: transformacija, ki v vseh stolpcih neodvisno premeša njihovo vsebino, da ustvari nove stolpce, in njen inverz.
- $ShiftRows()$ in $InvShiftRows()$: transformacija, ki ciklično zamakne vrednosti zadnjih treh vrstic stanja z različnimi dolžinami zamikov (angl. offset), in njen inverz.
- $SubBytes()$ in $InvSubBytes()$: transformacija, ki procesira stanje z uporabo konstantne nelinearne substitucijske tabele bajtov (S-box) in operira na vseh bajtih stanja neodvisno, in njen inverz.

Na sliki 3.1 je prikazan potek šifriranja. Algoritem razširi ključ K na $n+1$ ključev rund k, pri čemer n predstavlja število rund. Takoj na začetku s funkcijo $AddRoundKey()$ doda stanju prvi ključ k_0. Nato se pričnejo izvajati runde $r = 1, 2, \ldots, n-1$, v katerih se nad stanjem zaporedoma kličejo funkcije $SubBytes()$, $ShiftRows()$, $MixColumns()$ in $AddRoundKey()$. V funkciji $AddRoundKey()$ je v vsaki rundi dodan ključ runde k_r. V zadnji rundi, ki jo zaporedoma sestavljajo funkcije $SubBytes()$, $ShiftRows()$ in $AddRoundKey()$, je dodan še zadnji ključ runde k_n [31].

Pri dešifriranju, prikazanem na sliki 3.2, algoritem prične dodajati razširjeni ključ v obratnem vrstnem redu kot pri šifriranju. Prične z dodajanjem ključa k_n s funkcijo $AddRoundKey()$, v rundah $r = n-1, n-2, \ldots, 1$ zaporedoma kliče inverzne funkcije $InvSubBytes()$, $InvShiftRows()$, $InvMixColumns()$ in funkcijo $AddRoundKey()$. V funkciji $AddRoundKey()$ je v vsaki rundi dodan ključ runde k_r. V zadnji rundi kliče inverzne funkcije $InvSubBytes()$, $InvShiftRows()$ in funkcijo $AddRoundKey()$. V funkciji $AddRoundKey()$ je v zadnji rundi dodan ključ runde k_0 [31].



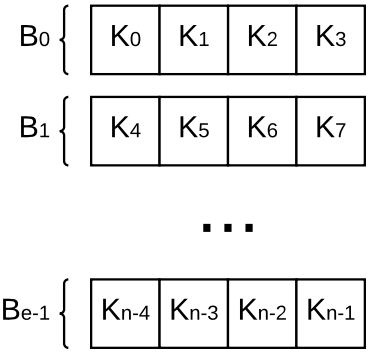
Slika 3.1: Delovanje AES-šifriranja [31]



Slika 3.2: Delovanje AES-dešifriranja [31]

3.1.1 Razširitev ključa

Algoritem v rutini razširitve ključa najprej iz ključa K dolžine n (16, 24 ali 32 bajtov) z zaporedjem bajtov $K = K_0 \dots K_{n-1}$ sestavi začetni set besed $B_0, \dots, B_{e-1}$, tako da je vsaka beseda sestavljena iz štirih bajtov ključa K, kot je prikazano na sliki 3.3. Iz prvih e besed (4, 6 ali 8) so kasneje generirane še ostale besede. Vsaka beseda je tako dolga 4 bajte, skupek 4 besed pa kasneje služi kot ključ runde v funkciji *AddRoundKey* (AES standardizirano operira z bloki bajtov velikosti 4×4)[31].



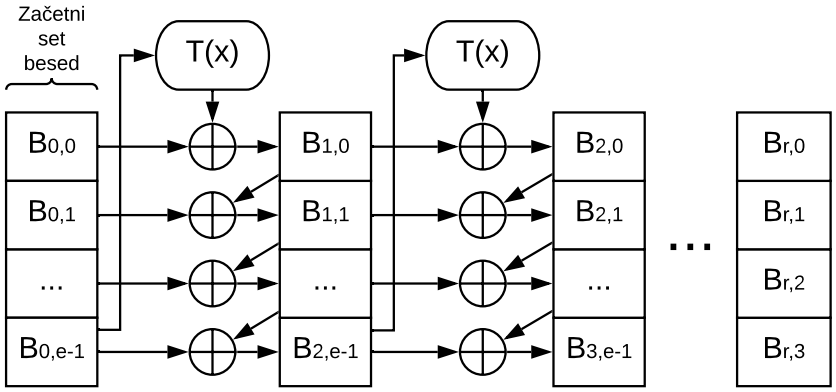
Slika 3.3: Začetni set besed pri razširjanju ključa AES [31]

Rutina še dodatno implementira funkciji, ki sta prikazani na sliki 3.5 in razloženi v nadaljevanju:

- $RotWord()$, ki zamakne bajte besede za eno pozicijo v levo,
- $SubWord()$, ki izvede S-box substitucijo nad besedo.

Kot je prikazano na sliki 3.4, so iz prvih e besed generirane še ostale (40, 46 ali 52 dodatnih besed za 128, 192 ali 256 bitni ključ): $B_{i,0} = T(B_{i-1,e-1}) \oplus B_{i-1,0}$ in $B_{i,j} = B_{i,j-1} \oplus B_{i-1,j}$, kjer je $1 \leq i \leq r$, $0 < j < e$, r število rund (10, 12 ali 14) in $T(x)$ transformacija besede. V zadnji iteraciji $i = r$ je $0 < j < 4$. V primeru rabe 256-bitnega ključa (14 rund) se vsako besedo $B_{i,4}$ še dodatno transformira s funkcijo $SubWord()$ [31].

Transformacija besede $T(x)$, ki smo jo omenili v prejšnjem odstavku, je prikazana na sliki 3.5. Bajti besed $B_{i,0}$ so zamaknjeni s funkcijo $RotWord()$, katere izhod je

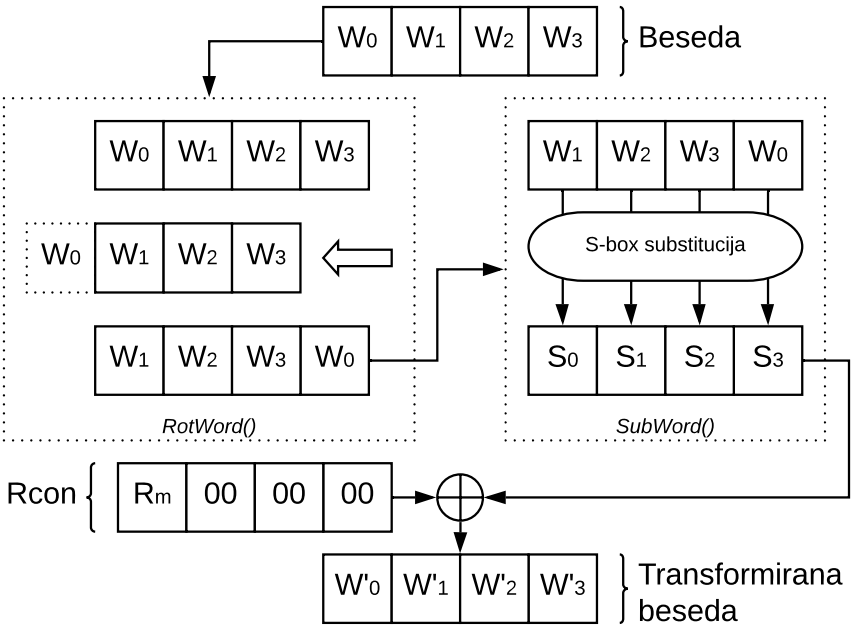


Slika 3.4: Generiranje nadaljnjih besed pri razširjanju ključa AES [31]

transformiran z S-box substitucijo v funkciji $SubWord()$. Na koncu je z operacijo $\oplus$ izhodu funkcije $SubWord()$ dodana konstanta $Rcon$ oblike $[R_m, 00, 00, 00]$, pri čemer je m: $0 < m < 10$ za AES128, $0 < m < 8$ za AES192 in $0 < m < 7$ za AES256. Bajt R_m konstante $Rcon$ zavzema naslednje vrednosti podane v tabeli 3.1 [31].

Tabela 3.1: Vrednosti prvega bajta R_m konstante $Rcon$ [31]

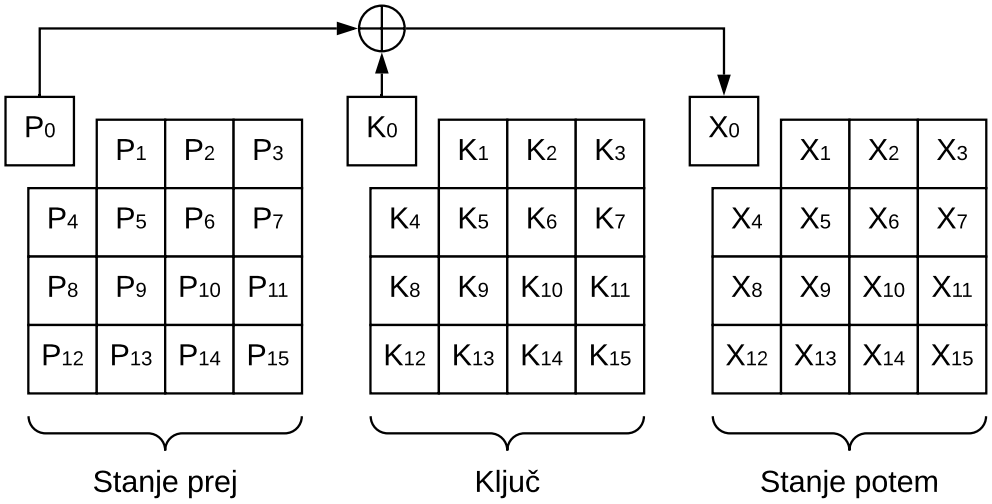
m	0	1	2	3	4	5	6	7	8	9
R_m(hex)	01	02	04	08	10	20	40	80	1b	36



Slika 3.5: Funkcija transformacije besede $T(x)$ pri razširjanju ključa AES [31]

3.1.2 AddRoundKey

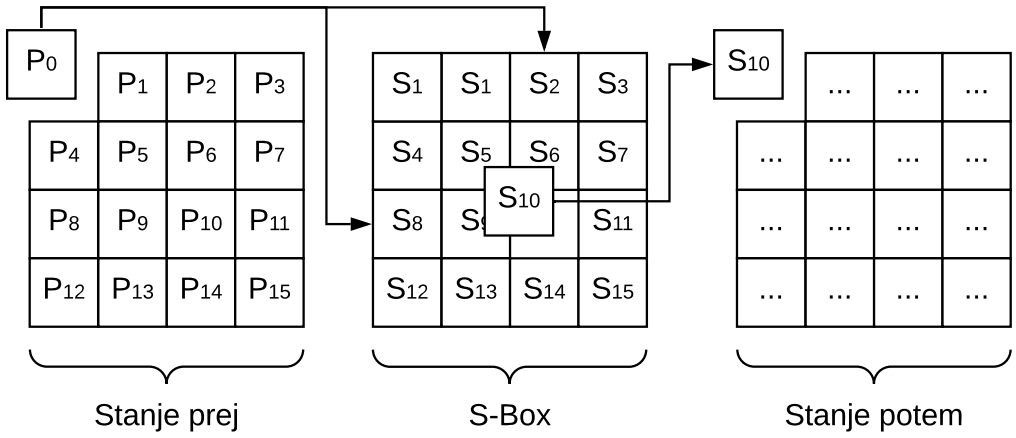
V funkciji $AddRoundKey()$ je stanju dodan ključ runde, kot je prikazano na sliki 3.6. Funkcija vsakič vzame 4 besede $B_0 \dots B_3$ (vsako besedo sestavljajo 4 bajti), generirane v fazi razširjanja ključa, po vrsti in uporabi vrednosti njihovih bajtov kot ključ runde: $B_0 = K_0K_1K_2K_3, \dots, B_3 = K_{12}K_{13}K_{14}K_{15}$. Bajti ključa runde $K_0 \dots K_{15}$ so združeni z bajti stanja $P_0 \dots P_{15}$ na način: $X_0 = P_0 \oplus K_0, \dots, X_{15} = P_{15} \oplus K_{15}$ [31].



Slika 3.6: Funkcija AddRoundKey šifre AES [31]

3.1.3 SubBytes in InvSubBytes

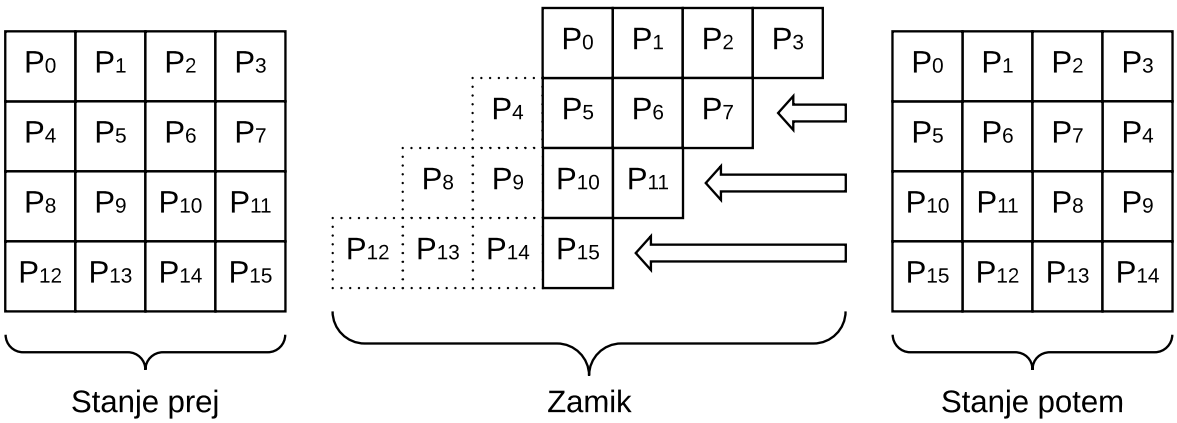
Funkcija $SubBytes()$ izvaja substitucijo stanja bajtov $P_0 \dots P_{15}$, tako da primerja njihovo vrednost v heksadecimalnem zapisu z vrednostmi stolpcev in vrstic v konstantni S-box tabeli. Stanje bajtov zamenja z vrednostmi, pri čemer se sekajo vrstice in stolpci. Prva heksadecimalna vrednost bajta predstavlja vrstico, druga pa stolpec substitucijske škatle. Na sliki 3.7 je prikazan primer substitucije bajtov. V primeru dešifriranja je uporabljena funkcija $InvSubBytes()$, ki deluje na enak način, le da uporablja inverz tabele S-box za substitucije vrednosti [31].



Slika 3.7: Funkcija SubBytes in tabela S-box šifre AES [31]

3.1.4 ShiftRows in InvShiftRows

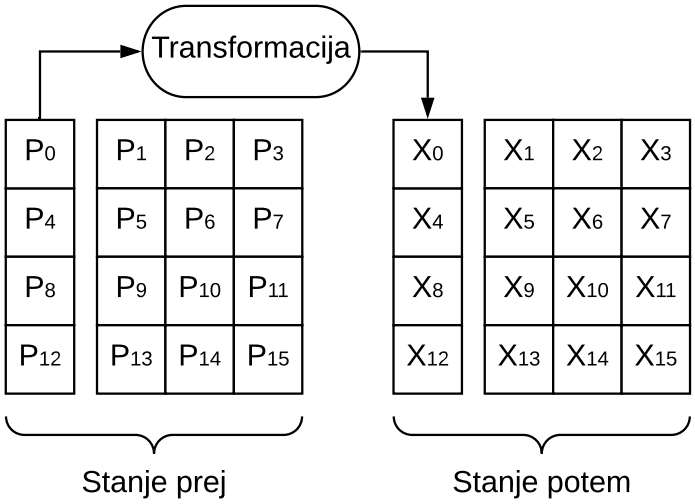
Funkcija $ShiftRows()$, prikazana na sliki 3.8, zamika vrstice bajtov stanja ciklično v levo. Prva vrstica ostane nespremenjena, druga je zamaknjena za eno pozicijo, tretja za 2 in četrta za 3. Pri dešifriranju je v inverzni funkciji $InvShiftRows()$ postopek enak, le da so vrstice zamaknjene v desno [31].



Slika 3.8: Funkcija ShiftRows šifre AES [31]

3.1.5 MixColumns in InvMixColumns

Funkciji $MixColumns()$ in $InvMixColumns()$ izvedeta transformacije nad stolpci matrike stanja, kot je prikazano na sliki 3.9.



Slika 3.9: Funkcija MixColumns šifre AES [31]

Izhodni bajti X_y so izračunani z množenjem definirane matrike s štirimi vhodnimi bajti P_y, pri tem y predstavlja stolpce matrike predhodnega stanja: $0 \leq y \leq 3$. Operacije so definirane v skupini $(\mathbf{GF}_{256}, +, \cdot)$, in sicer $\cdot$ predstavlja množenje in $+$ seštevanje v cikličnem polju $\mathbf{GF}_{256}$ [31]:

$$\begin{bmatrix} X_y \\ X_{y+4} \\ X_{y+8} \\ X_{y+12} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} P_y \\ P_{y+4} \\ P_{y+8} \\ P_{y+12} \end{bmatrix}$$

V funkciji $InvMixColumns()$ je uporabljen inverz definirane matrike [31]:

$$\begin{bmatrix} X_y \\ X_{y+4} \\ X_{y+8} \\ X_{y+12} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} P_y \\ P_{y+4} \\ P_{y+8} \\ P_{y+12} \end{bmatrix}$$

3.1.6 Novi ukazi AES

Novi ukazi (angl. New Instructions) algoritma AES (v nadaljevanju AES-NI) so bili zasnovani z namenom implementacije računsko zahtevnih in kompleksnih korakov algoritma na strojni opremi, da bi pospešili čas njegovega izvajanja. Izboljšava hitrosti je odvisna od procesorskega časa, ki ga aplikacija porabi za šifriranje in dešifriranje. Pri načinih šifriranja, pri katerih paralelizacija ni mogoča (npr. šifriranje s CBC), obravnavanih v poglavju 4, AES-NI doseže 2-3-kratno pospešitev, primerjaje z nepospešenim izvajanjem. Pri načinih šifriranja, kjer je paralelizacija mogoča (npr. CTR ali dešifriranje s CBC), je delovanje z AES-NI do 10-krat hitrejše [17].

Set ukazov je sestavljen iz šestih ukazov, ki za izvedbo porabijo veliko manj procesorskih ciklov kot navadna programska rešitev. Štirje so namenjeni šifriranju/dešifriranju in dva generiranju ključev rund [17]:

- $AESENC$: izvede posamezno rundo šifriranja,
- $AESENCLAST$: izvede zadnjo rundo šifriranja,
- $AESDEC$: izvede posamezno rundo dešifriranja,
- $AESDECLAST$: izvede zadnjo rundo dešifriranja,
- $AESKEYGENASSIST$: generira ključe rund, uporabljenih v šifriranju,
- $AESIMC$: pretvori ključe rund v obliko, primerno za dešifriranje.

3.2 DES, DEA in TDEA

Standard šifriranja podatkov (angl. Data Encryption Standard - DES), ki je bil opuščen 19. maja 2005 in se iz varnostnih razlogov ne uporablja več, specificira algoritem šifriranja podatkov (angl. Data Encryption Algorithm - DEA) in trojni algoritem šifriranja podatkov (angl. Triple Data Encryption Algorithm). DEA (v nadaljevanju DES, saj se algoritem tako naslavlja tudi v večini literature) je simetrična bločna šifra,

zasnovana na Feisteljevi mreži s 64-bitnim ključem. Izmed teh 64 bitov algoritem uporablja 56 naključno generiranih bitov za šifriranje, preostalih 8 pa za zaznavo napak. Algoritem se zaradi kratke dolžine ključev, kar dopušča napad z grobo silo, ne uporablja več [35].

Šifra DES, katere dolžina bloka čistopisa ali kriptograma je 64 bitov, šifrira blok čistopisa P tako, da najprej bite premeša v začetni permutacijski funkciji $IP(P)$, jih razdeli na dve 32-bitni polovici L_0 in R_0 in ju v 16 rundah še dodatno preračuna: $L_i = R_{i-1}$ in $R_i = L_{i-1} \oplus F(R_i, K_i)$ za $1 \leq i \leq 16$, pri čemer K_i predstavlja 46-bitni ključ runde, sestavljen iz izbranih bitov ključa K in $F(x, y)$ substitucijsko-permutacijsko funkcijo. Po končani zadnji rundi se bloka polovic L_{16} in R_{16} združita v obratnem vrstem redu, iz njiju pa se pridobi kriptogram: $C = IP^{-1}(R_{16}||L_{16})$, kjer $IP^{-1}(x)$ predstavlja inverz permutacijske funkcije $IP(x)$. Dešifriranje poteka na enak način kot šifriranje, le da namesto čistopisa P v algoritem podamo kriptogram C [35].

TDEA (v nadaljevanju 3DES) je simetrična bločna šifra, ki nad blokom čistopisa izvede šifro DES 3-krat. Šifrira na način $C = E_{K3}(D_{K2}(E_{K1}(P)))$ in dešifrira na način $C = P_{K1}(E_{K2}(D_{K3}(C)))$, pri čemer $K1, K2$ in $K3$ predstavljajo šifrirne ključe. Sam standard definira 3 različne izbire ključev: $K1 \neq K2 \neq K3$ kot prvo, $K1 = K3 \neq K2$ kot drugo in $K1 = K2 = K3$ kot tretjo [35]. Drugo in tretjo izbiro je NIST zaradi neprimernosti opustil [20].

3.3 Camellia

Camellia, simetrična bločna šifra, zasnovana na Feisteljevi mreži, operira z bloki dolžine 128 bitov in uporablja 128-, 192- ali 256-bitne ključe. Razvita na Japonskem (leta 2000) in izbrana kot priporočena v EU (leta 2003) je leta 2005 postala standardizirana šifra, ki je primerljive varnosti in hitrosti kot šifra AES [5]. V nadaljevanju bomo obravnavali samo šifriranje in dešifriranje s 128-bitnim ključem.

Algoritem šifrira blok čistopisa v 18 rundah, v 6. in 12. rundi pa še dodatno uporablja $FL(x, y)$ in $FL^{-1}(x, y)$ substitucijsko permutacijski funkciji. V primeru rabe

128-bitnega ključa se iz ključa K izračuna 64-bitne ključe rund kw_t, k_u in kl_v za $t, v = 1, ..., 4$ in $u = 1, ..., 18$. Najprej izračuna $L_0||R_0 = P \oplus (kw_1||kw_2)$, pri čemer sta L_0 in R_0 leva in desna polovica bitov bloka čistopisa. V rundah $r = 1, ..., 18$ se nato izvedejo naslednje operacije: $L_r = R_{i-1} \oplus F(R_i, K_i)$ in $R_r = L_{r-1}$, pri tem $F(x)$ predstavlja substitucijsko-permutacijsko funkcijo. V rundah $r = 6, 12$ se še naknadno izračuna $L_r = FL(L_r, kl_{2r/6-1})$ in $R_r = FL^{-1}(R_r, kl_{2r/6})$, pri čemer $FL(x)$ in $FL^{-1}(x)$ predstavljata substitucijsko-permutacijski funkciji. Po vseh 18 rundah se kriptogram pridobi na način: $C = (R_{18}||L_{18}) \oplus (kw_3||kw_4)$. Dešifriranje se izvede po enakem postopku kot šifriranje, le da namesto čistopisa kot vhodni blok podamo kriptogram, ključe rund pa dodajamo v obratnem vrstnem redu [19].

3.4 Serpent

Šifra Serpent, kandidatka na tekmovanju AES, je simetrična bločna šifra, ki operira s 128-bitnimi bloki in z variabilnimi dolžinami ključev. Prvotno je bila zasnovana na podlagi elementov šifre DES (uporabljene so bile iste S-box tabele), ki so jih v kasnejši verziji spremenili. V nadaljevanju bomo obravnavali samo šifriranje in dešifriranje s 128-bitnim ključem [22].

Algoritem šifrira blok čistopisa v 32 rundah $r = 0, ..., 31$, pri tem pa uporabi 33 128-bitnih ključev rund $K_0, ..., K_{32}$, generiranih iz ključa K. Blok čistopisa P postane prvi blok vmesnega stanja $B_0 = P$. V vsaki rundi je vmesnemu bloku stanja B_r najprej dodan ključ runde K_r. Nato se izvedejo substitucije vmesnega stanja z vrednostmi S-box tabel $S_r(x)$, takoj zatem pa se blok stanja razdeli na štiri 32-bitne besede: $X_0||X_1||X_2||X_3 = S_r(B_r \oplus K_r)$. Nad besedami $X_0, ..., X_3$ se izvede vrsta linearnih transformacij, kar po njihovi združitvi rezultira v naslednji blok runde B_{r+1}. Kriptogram C se pridobi v zadnji rundi, v kateri se namesto linearnih transformacij doda ključ K_{32} na način: $C = S_7(B_{31} \oplus K_{31}) \oplus K_{32}$ [22].

Postopek dešifriranja je podoben šifriranju. Za dešifriranje kriptograma se uporabijo inverzi linearnih transformacij, inverzi S-box tabel v obratnem vrstnem redu in ključi

rund prav tako v obratnem vrstnem redu [22].

3.5 TwoFish

Šifra TwoFish, kandidatka na tekmovanju AES, je simetrična bločna šifra, zasnovana na Feisteljevem omrežju, ki operira s 128-bitnimi bloki in z variabilnimi dolžinami ključev do 256 bitov. V nadaljevanju bomo obravnavali samo šifriranje in dešifriranje s 128-bitnim ključem [34].

Algoritem šifrira blok čistopisa P v šestnajstih rundah $r = 0, ..., 15$. Najprej ga razdeli na 4 32-bitne besede $X_{0,0}||X_{0,1}||X_{0,2}||X_{0,3} = P$ in generira 42 32-bitnih ključev rund $K_0, ..., K_{41}$ iz ključa K. Iz K še dodatno generirana 4 S-box tabele, ki so uporabljene v substitucijsko-permutacijski funkciji $F(x, y)$. Takoj na začetku so besedam dodani ključi rund na način: $X_{0,j} = X_{0,j} \oplus K_{0,j}$, $j = 0, ..., 3$. Vsaka runda izvede substitucije in permutacije z generiranimi S-box tabelami in doda ključe rund K_{2r+8} besedi $X_{r,0}$ in K_{2r+9} besedi $X_{r,1}$: $X_{r,0}, X_{r,1} = F(X_{r,0}, X_{r,1})$. Besede naslednjih rund so generirane na način: $X_{r+1,0} = (X_{r,0} \oplus X_{r,2}) >>> 1$, $X_{r+1,1} = X_{r,1} \oplus (X_{r,3} <<< 1)$, $X_{r+1,2} = X_{r,0}$ in $X_{r,1} = X_{r+1,3}$, pri tem $<<< 1$ predstavlja rotacijo elementa za 1 v levo in $>>> 1$ v desno. Po zadnji rundi se besede zamenjajo in vrnejo v prvotno stanje: $X_0 = X_{15,2}$ $X_1 = X_{15,3}$ $X_2 = X_{15,0}$ $X_3 = X_{15,1}$. Za pridobitev kriptograma se besedam dodajo še ključi rund: $C = (X_0 \oplus K_4)||(X_1 \oplus K_5)||(X_2 \oplus K_6)||(X_3 \oplus K_7)$ [34].

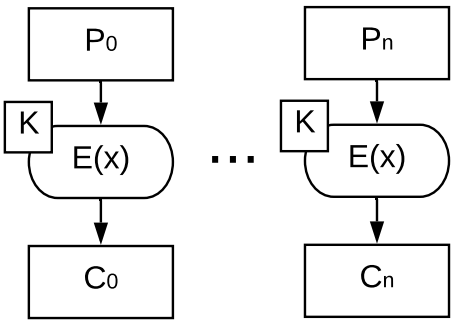
4 Načini bločnega šifriranja

Kot smo do sedaj že spoznali, simetrični bločni šifrirni algoritmi šifrirajo čistopis, razdeljen na dolžino njihovih blokov. Ta primitivni način se imenuje ECB-način šifriranja in potrebuje zapolnjevanje čistopisa ter ne zagotavlja popolne zaupnosti, zaščite celovitosti in overjanja kriptograma. Za zagotavljanje teh lastnosti se uporablja druge osnovne (CBC, CTR) ali napredne (GCM, CCM, OCB, XTS ...) načine šifriranja, ki te pomanjkljivosti odpravljajo (omenjene načine bomo spoznali v nadaljevanju).

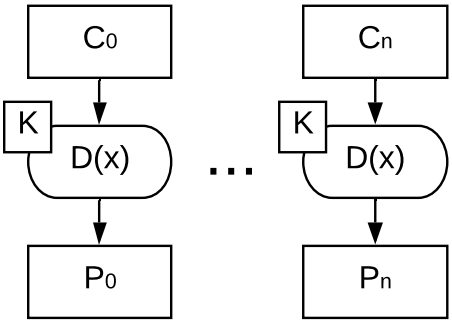
Nekateri načini spremenijo bločno šifro tudi v tokovno, kar pomeni, da nam čistopisa ni treba zapolnjevati (kriptogram je enake velikosti kot izvorno sporočilo, kar pomeni manj dodatnega dela pri prenosu ali potrebnega prostora pri shranjevanju kriptograma). Ko z načini šifriranja šifriramo ali dešifriramo, je pomembna tudi njihova sposobnost paralelizacije. To pomeni, da je bloke čistopisa ali kriptograma mogoče šifrirati ali dešifrirati vzporedno. Včasih želimo tudi, da so naši podatki overjeni. V ta namen se uporablja načine šifriranja, ki zagotavljajo overjeno šifriranje - AE (angl. authenticated encryption). V primerih, v katerih celotnega čistopisa ne smemo šifrirati, saj mora biti del podatkov prisoten v nešifrirani obliki (npr. glava v IP paketkih), uporabljamo načine šifriranja, ki zagotavljajo overjeno šifriranje s povezanimi podatki - AEAD (angl. authenticated encryption with associated data). Z njimi overimo šifrirane dele skupaj z nešifriranimi in jih tako povežemo v celoto.

4.1 ECB

Elektronska knjiga kode (angl. Electronic Codebook, v nadaljevanju ECB) je primitiven način šifriranja, ki predstavlja samo uporabo algoritma šifriranja ali dešifriranja bločne šifre. Šifriranje, prikazano na sliki 4.1, deluje tako, da se čistopis razdeli na manjše dele dolžine bloka šifre in se šifrira vsakega posebej. Naj bo d_B velikost bloka šifre in d_P dolžina bajtov čistopisa. Šifriramo na način: $C_i = E(B_i)$ za $n = \left\lceil \frac{d_B}{d_P} \right\rceil - 1$, pri čemer P_i predstavlja blok čistopisa, C_i blok šifriranega teksta, K ključ in $E(x)$ bločni algoritem šifriranja. Dešifriranje, prikazano na sliki 4.2, deluje na način: $P_i = D(C_i)$, pri tem je $D(x)$ algoritem dešifriranja [25].

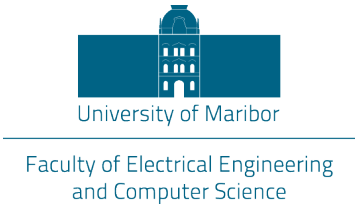


Slika 4.1: Delovanje šifriranja z načinom ECB [25]

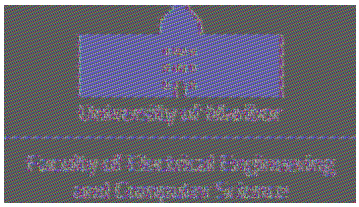


Slika 4.2: Delovanje dešifriranja z načinom ECB [25]

Način ECB se v praksi ne uporablja, saj ne ponuja zadostne zaupnosti, zaznave mešanja blokov ali overjanja. Na sliki 4.4 je prikazan logotip FERI, šifriran v načinu ECB (bloki so šifrirani neodvisno). V primerjavi z nešifriranim logotipom na sliki 4.3 so s šifrirane slike takoj razvidne podobnosti in informacije o izvornem sporočilu (FERI-logotipa). Na sliki 4.5 je prikazan kriptogram logotipa FERI, šifriran v načinu CBC, iz katerega ni razvidna nobena informacija.



Slika 4.3: Logotip FERI kot čistopis

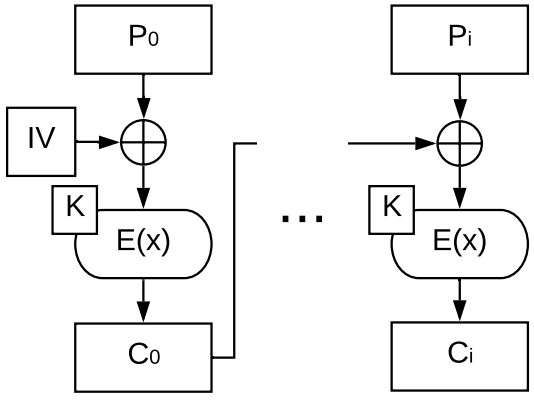


Slika 4.4: Logotip FERI, šifriran v načinu ECB

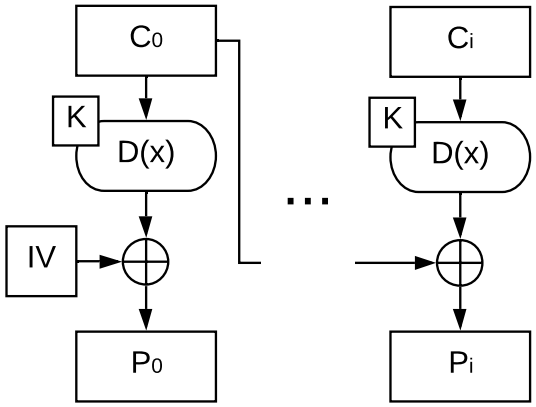
Slika 4.5: Logotip FERI, šifriran v načinu CBC

4.2 CBC

Veriženje blokov šifre (angl. Cipher Block Chaining, v nadaljevanju CBC) je način šifriranja, pri katerem so bloki čistopisa veriženi z operacijo $\oplus$ s prejšnjimi bloki kriptograma. V tem načinu se uporablja inicializacijski vektor IV, ki predstavlja javno, nepredvidljivo in enkratno vrednost, namenjeno maskiranju prvega šifriranega bloka in posledično vseh nadalnjih blokov (brez uporabe IV, bi različni čistopisi z enakimi začetki, šifrirani z enakim ključem, imeli enake začetke kriptogramov). Način CBC nam omogoči zaznavo spremembe vrstnega reda šifriranih blokov in zaupnost vsebine. Šifriranje, prikazano na sliki 4.6, šifrira prvi blok na način: $C_0 = E(P_0 \oplus IV)$. Nadaljnji bloki so šifrirani: $C_i = E(C_{i-1} \oplus P_i)$, za $0 < i < \left\lceil \frac{d_B}{d_P} \right\rceil$. Pri dešifriranju, prikazanem na sliki 4.7, se pridobi čistopis na način: $P_0 = D(C_0) \oplus IV$ in $P_i = D(C_i) \oplus C_{i-1}$ [25].



Slika 4.6: Delovanje šifriranja z načinom CBC [25]

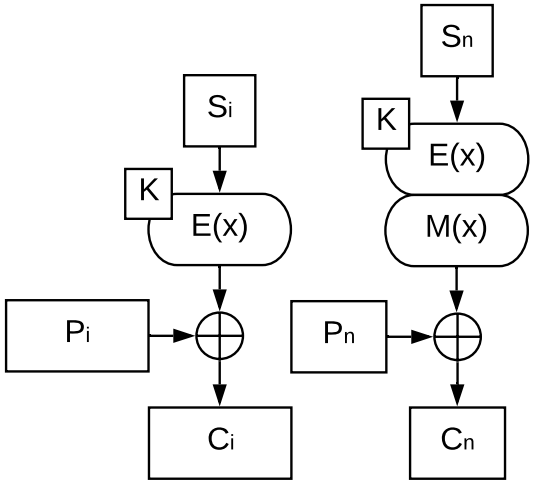


Slika 4.7: Delovanje dešifriranja z načinom CBC [25]

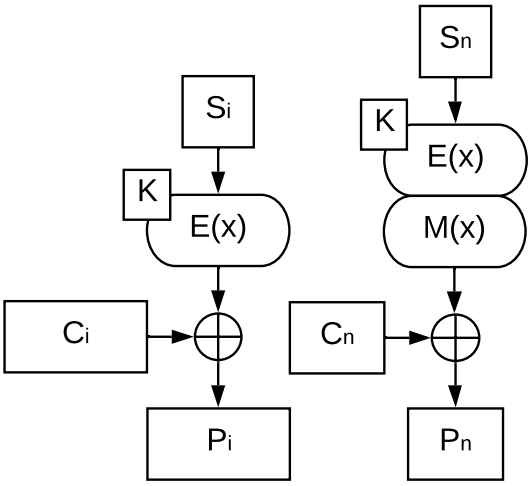
4.3 CTR

Števec (angl. Counter, v nadaljevanju CTR) je način šifriranja, pri katerem so bloki čistopisa združeni z operacijo $\oplus$ s šifriranimi številnimi vrednostmi. Zagotovljena mora biti lastnost, da so vse šifrirane vrednosti števcev med seboj različne. Šifriranje, prikazano na sliki 4.8, deluje na način: $C_i = E(S_i) \oplus P_i$, pri tem S_i predstavlja števec, $0 \leq i < n$ in $n = \left\lceil \frac{d_B}{d_P} \right\rceil - 1$. CTR je tokovni način šifriranja, kjer se zadnji blok šifrira

na način: $C_n = M(E(S_n)) \oplus P_n$. $M(x)$ predstavlja funkcijo, ki pridobi število najbolj uteženih bitov (dolžine bitov zadnjega bloka čistopisa). Dešifriranje, prikazano na sliki 4.9, deluje na podoben način kot šifriranje, le da so bloki kriptograma združeni s šifriranimi števci z operacijo XOR: $P_i = E(S_i) \oplus C_i$ in $P_n = M(E(S_n)) \oplus C_n$ [25].



Slika 4.8: Delovanje šifriranja z načinom CTR [25]



Slika 4.9: Delovanje dešifriranja z načinom CTR [25]

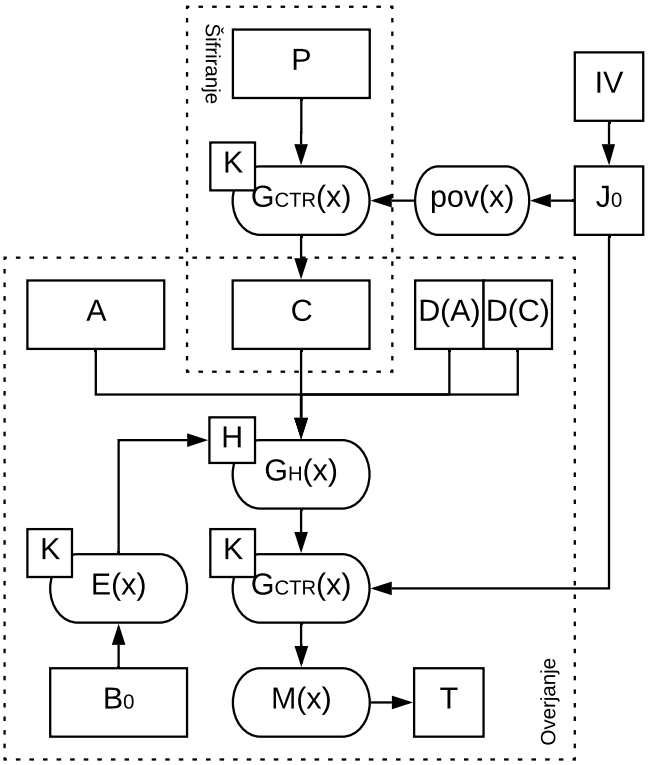
4.4 GCM

Galois/števec (angl. Galois/Counter, v nadaljevanju GCM) je AEAD-način šifriranja. Za šifriranje vsebine je uporabljena različica načina CTR, za overjanje pa zgoščevalna funkcija (angl. Hash function), definirana v Galoisovem polju. GCM vsebuje naslednje elemente [26]:

- H: ključ, ki se pridobi s šifriranjem ničlnega bloka B_0 (blok bitov '0'),
- G_H: zgoščevalna funkcija, ki uporablja ključ H,
- IV: inicializacijski vektor, ki je pretvorjen v začetni števec J_0,
- $G_{CTR}(x)$: bločna šifra $E(x)$ s CTR-načinom šifriranja in ključem K,
- $pov(x)$: funkcija, ki poveča vrednost števca,
- A: povezani podatki,

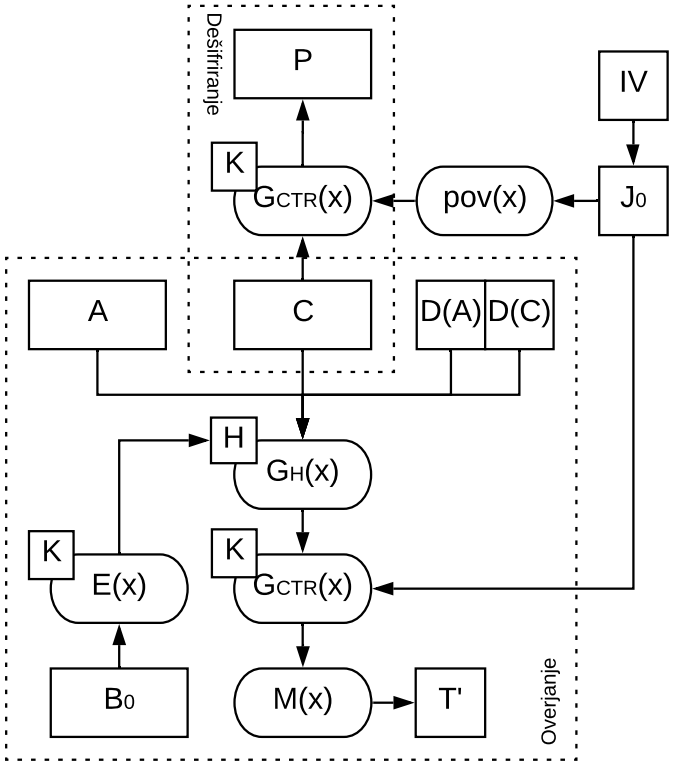
- $D(A)$: in $D(C)$: dolžini povezanih podatkov A in kriptograma C,
- $M(x)$: funkcija, ki vrne želeno število najbolj uteženih bitov,
- T: overitvena značka.

Pri šifriranju, prikazanem na sliki 4.10, se najprej pridobi ključ H s šifriranjem ničlnega bloka B_0 (blok zapolnjen z biti 0). Iz IV je generiran začetni števec J_0, ki je povečan in uporabljen kot prvi števec pri šifriranju čistopisa P v funkciji $G_{CTR}(x)$. Povezani podatki A in kriptogram C sta posamično zapolnjena na dolžino večkratnika dolžine bloka šifre $E(x)$ in sta nato združena z vrednostjo njunih dolžin $D(A)$ in $D(B)$ (vrednosti $D(A)$ in $D(B)$ sta dolžine 64 bitov) na način: $S = A||a||C||b||D(A)||D(C)$, pri čemer a in b predstavljata dolžini bitov, potrebni za zapolnitev A, C in $||$ pripenjanje elementov. S je podan v zgoščevalno funkcijo $G_H(x)$, ki uporablja ključ H. Zadnji blok zgoščene vrednosti je šifriran v funkciji $G_{CTR}(X)$, ki za števec uporabi J_0. Iz kriptograma zadnjega bloka zgoščene vrednosti je pridobljena značka T, tako da se mu odvzame želeno število najbolj uteženih bitov s funkcijo $M(x)$ [26].



Slika 4.10: Delovanje šifriranja z načinom GCM [26]

Dešifriranje, prikazano na sliki 4.11, deluje na enak način kot šifriranje, razlika je le v tem, da se namesto P dešifrira C. Po enakem postopku se pridobi tudi značka T', ki se nato primerja z značko T: če je $T \neq T'$, algoritem javi napako [26].



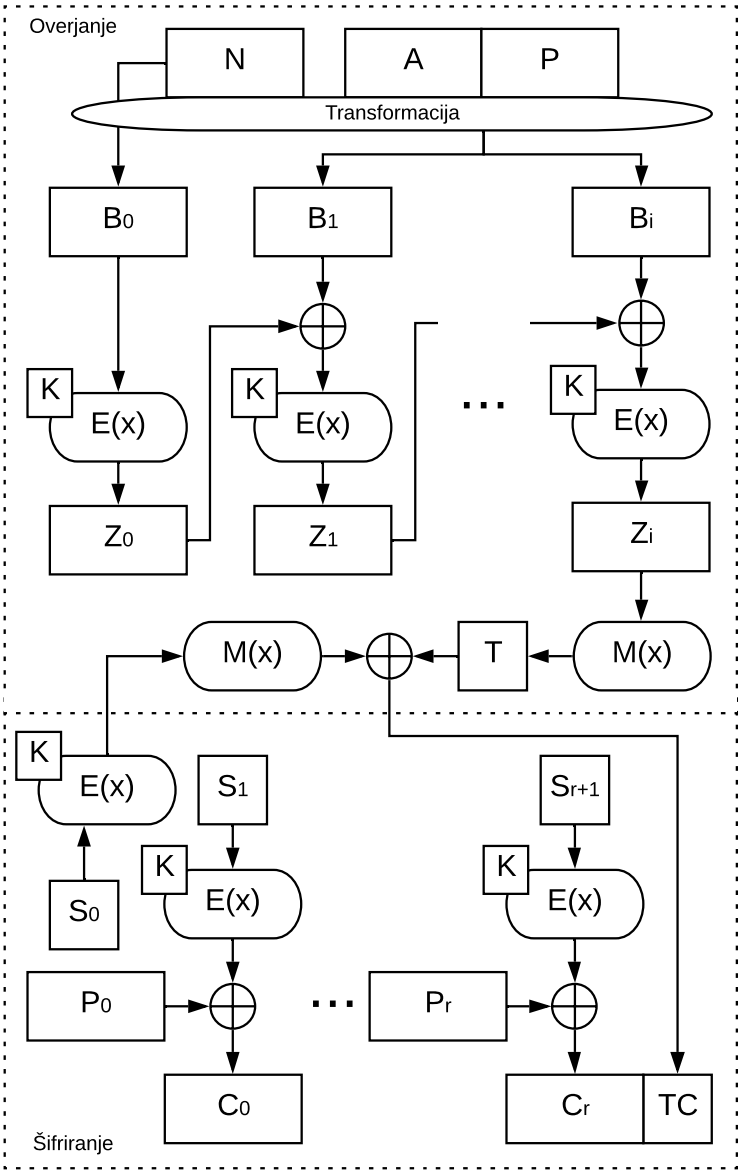
Slika 4.11: Delovanje dešifriranja z načinom GCM [26]

4.5 CCM

Števec z overjanjem veriženja blokov šifre (angl. Counter with CBC-MAC, v nadaljevanju CCM) je način AEAD-šifriranja, ki združuje osnovna načina CBC za overjanje in CTR za šifriranje [27].

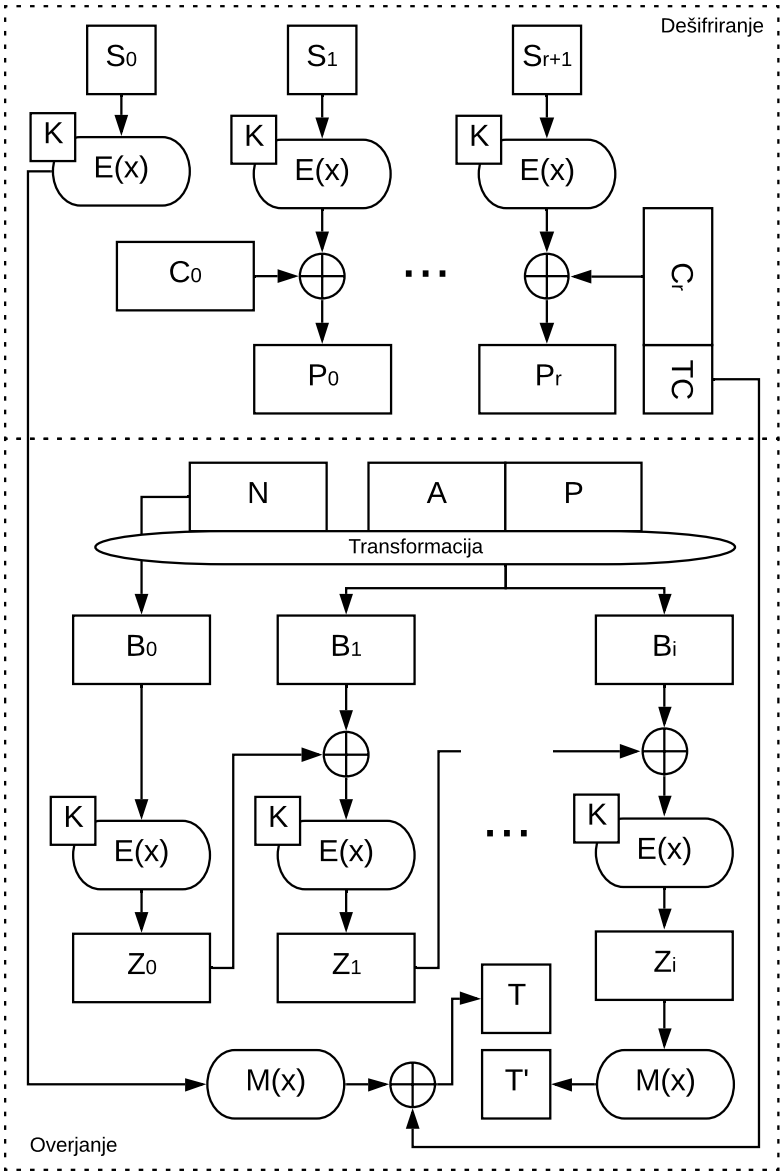
Pri šifriranju, prikazanem na sliki 4.12, se najprej izračuna značka T, katere vrednost predstavlja overjeno enkratno število N, čistopis P in povezane podatke A. Elementi (N, A, P) so formatirani v bajte $B_0 \dots B_i$, in sicer je N formatiran v B_0, (A, P) pa v $B_1 \dots B_i$. Bajti $B_0 \dots B_i$ so šifrirani z načinom CBC in izbrano bločno šifro v bajte $Z_0 \dots Z_i$. Zadnjemu bajtu Z_i je s funkcijo $M(x)$ odvzeto izbrano število najbolj uteženih bitov, kar nam poda overitveno značko T. V fazi šifriranja se naj-

prej generirajo števci $S_0, \ldots, S_{r+1}$. Z načinom CTR se šifrira čistopis $P = P_0 \ldots P_r$ na način $C_0 = E(S_1) \oplus P_0, C_1 = E(S_2) \oplus P_1, \ldots, C_r = E(S_{r+1}) \oplus P_r$. Na koncu se zadnjemu bloku C_r pripne šifrirano značko $TC = M(E(S_0)) \oplus T$ [27].



Slika 4.12: Delovanje šifriranja z načinom CCM [27]

Pri dešifriranju, prikazanem na sliki 4.13, se najprej generira števce $S_0, \ldots, S_{r+1}$ in nato s CTR-načinom dešifrira kriptogram $C = C_0 \ldots C_r$. Nato se iz šifrirane značke TC pridobi overitvena značka $T = M(E(S_0)) \oplus TC$. V fazi overjanja se na enak način kot pri šifriranju iz P (slika 4.12) pridobi overitvena značka $T' = M(Z_i)$. Znački T in T' se primerjata: če je $T = T'$, so elementi (N, A, P) uspešno overjeni, in če $T \neq T'$, algoritem javi napako [27].



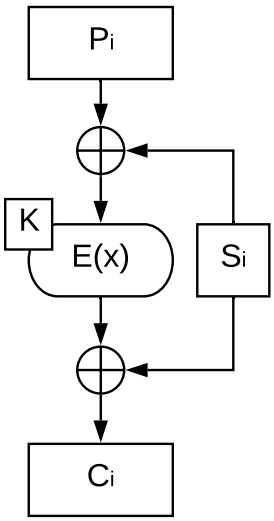
Slika 4.13: Delovanje dešifriranja z načinom CCM [27]

4.6 OCB

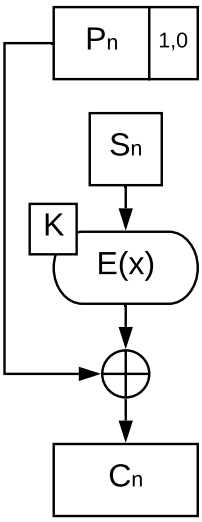
Knjiga kode z zamikom (angl. Offset Codebook, v nadaljevanju OCB) je način AEAD-šifriranja. Obstajajo 3 različice načina: OCB1 (2001), OCB2 (2004) in OCB3 (2011). V nadaljevanju bomo obravnavali najnovejšo različico in se nanjo sklicevali s kratico OCB [18].

Šifriranje, prikazano na sliki 4.14, je podobno kot način CTR. Vsak blok čistopisa P_i je šifriran v blok kriptograma C_i na način: najprej je enkratno število N pretvorjeno

v začetni števec S_0, nato pa izračunan kriptogram $C_i = E(P_i \oplus S_i) \oplus S_i$, pri čemer S_i predstavlja števec, povečan z vsako iteracijo za vrednost, izpeljano iz ključa. V primeru, da dolžina čistopisa ni večkratnik dolžine bloka šifre, se zadnji blok šifrira, kot je prikazano na sliki 4.15. Zadnji blok P_n je najprej zapolnjen z enim bitom vrednosti 1 in nato s toliko ničlnimi biti, da je velikost blok enaka velikosti bloka šifre. Zatem je poln blok združen z zadnjim šifriranim števcem S_n: $C_n = E(S_n) \oplus P_n$ [33].



Slika 4.14: Delovanje šifriranja z načinom OCB [33]



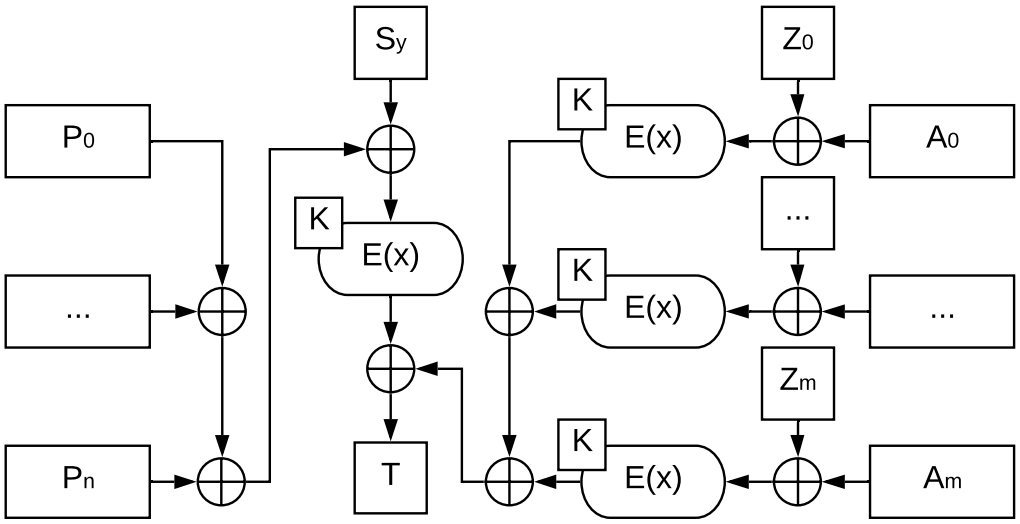
Slika 4.15: Delovanje OCB-šifriranja zadnjega bloka [33]

Pri overjanju, prikazanem na sliki 4.16, sta overjena čistopis P in povezani podatki A. Iz čistopisa se pridobi kontrolno vsoto na način: $KVsota = P_0 \oplus P_1 \oplus \cdots \oplus P_n$. Nato se iz A izračuna avtentikacijska značka: $Auth = E(A_0 \oplus Z_0) \oplus \cdots \oplus E(A_m \oplus Z_m)$, pri čemer je števec Z_0 blok ničlnih bitov, naslednji števci pa so povečava svojega predhodnega števca. V primeru, da zadnji blok povezanih podatkov ni enak dolžini bloka šifre, se ga najprej zapolni z enim bitom vrednosti 1 in nato s preostalimi ničlnimi biti, da se doseže dolžina bloka. Overitvena značka T je na koncu izračunana na način: $T = E(KVsota \oplus S_y) \oplus Auth$, pri tem pa je S_y povečan zadnji števec šifriranja S_n [33].

Dešifriranje kriptograma se izvede na enak način kot šifriranje (sliki 4.14 in 4.15), le da so bloki čistopisa P zamenjani z bloki kriptograma C: $P_i = E(C_i \oplus S_i) \oplus S_i$ in $P_n = E(S_n) \oplus C_n$ [33].

Iz dešifriranega kriptograma se na enak način kot pri fazi šifriranja (slika 4.16) izračuna

overitvena značka $T' = E(KVsota \oplus S_y) \oplus Auth$. V primeru, da $T' \neq T$, algoritem javi napako [33].

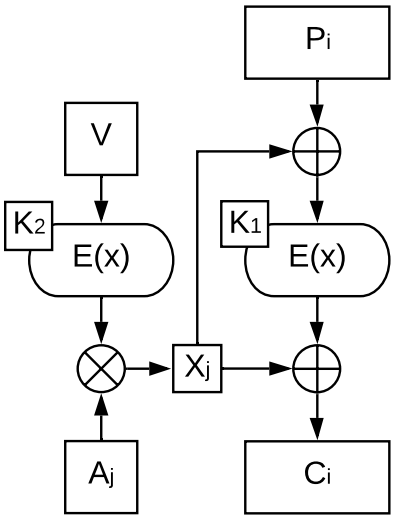


Slika 4.16: Delovanje overjanja z načinom OCB [33]

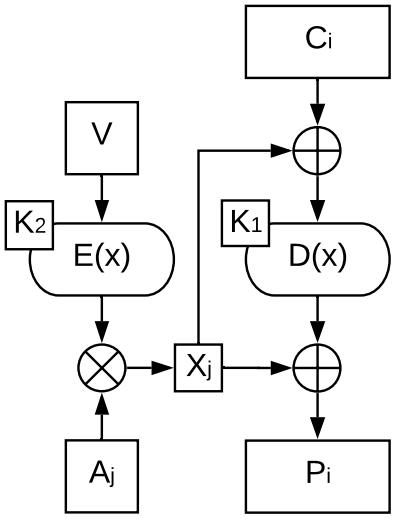
4.7 XTS

XEX-temelječa prirejena elektronska knjiga s krajo šifriranega teksta (angl. XEX-Based Tweaked-Codebook Mode With Ciphertext Stealing, v nadaljevanju XTS) je način šifriranja, namenjen šifriranju podatkov, hranjenih na napravah za shranjevanje podatkov. Sestavljata jo način XOR-šifriraj-XOR (angl. XOR-encrypt-XOR, v nadaljevanju XEX) in kraja šifriranega teksta (angl. Ciphertext stealing, v nadaljevanju CTS). XTS uporablja dva ključa K_1 in K_2; K_1 je uporabljen za šifriranje oz. dešifriranje in K_2 za izračun vmesnih vrednosti [29].

Pri šifriranju, prikazanem na sliki 4.17, je najprej izračunana vmesna vrednost na način: $X_j = E_{K_2}(V) \otimes A_j$, kjer V predstavlja začetno vrednost tweak (npr. število sektorja), A_j primitiven element (npr. število bloka v sektorju) in $\otimes$ definirano množenje elementov. Bloki čistopisa so nato šifrirani z načinom XEX: $C_i = E_{K_1}(X_j \oplus P_i) \oplus X_j$. Dešifriranje, prikazano na sliki 4.18, je izvedeno na enak način kot šifriranje, le da se pri šifriranju z XEX uporabi šifrina funkcija dešifriranja $D(x)$, v katero so kot vhodni bloki podani bloki kriptograma C_i [29].

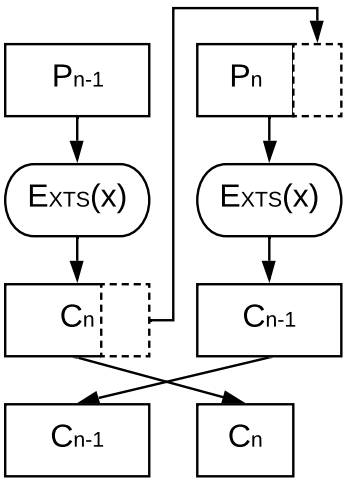


Slika 4.17: Delovanje dešifriranja z načinom XTS [29]

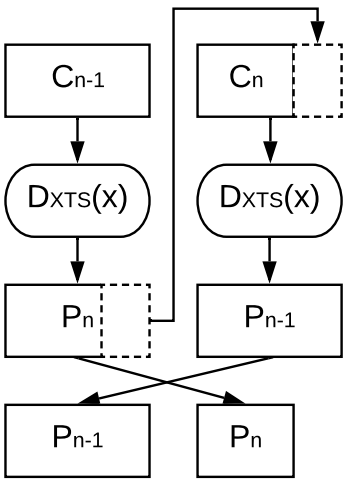


Slika 4.18: Delovanje dešifriranja z načinom XTS [29]

V primeru, da zadnji blok čistopisa ni poln blok, se njegova zadnja dva bloka šifrira z uporabo metode CTS (slika 4.19). Predzadnji blok čistopisa P_{n-1} je šifriran na način: $C_n = E_{XTS}(P_{n-1})$, pri čemer E_{XTS} predstavlja šifriranje, definirano v prejšnjem odstavku (slika 4.17). Število bitov bloka P_n, ki manjkajo do polnega bloka, je odvzeto šifriranemu bloku C_n in dodano bloku P_n, ki je nato šifriran: $C_{n-1} = E_{XTS}(P_n)$. Blok C_{n-1} postane predzadnji blok kriptograma, blok C_n z odvzetimi biti pa zadnji blok kriptograma. Kriptogram se dešifrira na enak način, le da je uporabljena funkcija dešifriranja $D_{XTS}(x)$, njeni vhodni bloki pa so bloki kriptograma (slika 4.20). Funkcija dešifriranja $D_{XTS}(x)$ je prikazana na sliki 4.18 [29].



Slika 4.19: Šifriranje nepolnega bloka z metodo CTS [29]



Slika 4.20: Dešifriranje nepolnega bloka z metodo CTS [29]

5 Zasnova eksperimenta

Teste za preizkušanje zmogljivosti smo izvedli na prenosniku Lenovo B5400 s procesorjem Intel Core i5-4200M 2,50 GHz, operacijskim sistemom Ubuntu 18.04.3 LTS bionic in J7ET61WW (2.06) osnovnim vhodno/izhodnim sistemom (angl. Basic Input/Output System - BIOS).

5.1 Priprava preskušene naprave

Da bi bile meritve izmerjene čim natančneje, smo svojo napravo najprej vizualno pregledali in z nje očistili prah, da bi se izognili vplivu termalnega zaviranja CPE (angl. CPU thermal throttling), zaradi česar se lahko izvajanje naše kode upočasni. Dodatno smo še v nastavitvah BIOS izklopili naslednje nastavitve procesorja in napajanja:

- Core Multi-Processing - izklopimo večjedrno procesiranje,
- Intel (R) Hyper-Threading Technology - izklopimo večnitno procesiranje,
- Intel (R) SpeedStep technology - izklopimo avtomatsko prehajanje med zmogljivostnim in varčevalnim načinom,
- CPU Power Management - izklopimo avtomatsko izklapljanje procesorske ure v času sistemske neaktivnosti.

5.2 Programski jezik, prevajalnik, knjižnice

Ker izbira prevajalnika vpliva na hitrost izvajanja procesa [14], smo za prevajanje kode izbrali prevajalnik G++, s katerim je mogoče prevajati kodo, napisano v programskih

jezikih C in C++ hkrati. Na podlagi tega so bile izbrane knjižnice OpenSSL (C), Nettle (C), Libgcrypt (C), Crypto++ (C++).

5.3 Program za preizkušanje zmogljivosti

Namen programa za preizkušanje zmogljivosti je, da šifrira oz. dešifrira podan čistopis oz. kriptogram z implementacijo šifer in načinov šifriranja, ki jih knjižnice ponujajo, in izmeri njihovo hitrost v procesorskih ciklih. Da lahko izmerimo število procesorskih ciklov, potrebnih za izvajanje kode, je treba tik pred pričetkom in takoj po koncu izvajanja kode iz procesorja prebrati števec časovnih žigov (angl. Time Stamp Counter - TSC) in izračunati razliko.

Intelovi procesorji vsebujejo števec, ki iterira za vsak izveden procesorski cikel. Do njega dostopamo z RDTSC- in RDTSCP-ukazoma za zbirni jezik (angl. Assembly). Ukaza RDTSC in RDTSCP delujeta tako, da prebereta števec časovnih žigov in ga po polovicah zapišeta v EDX- in EAX-procesorska registra. Ukaz RDTSCP še dodatno prebere identifikacijsko vrednost CPE in jo shrani v register ECX. Ukaz RDTSCP v nasprotju z RDTSC počaka, da se vsi ukazi pred njegovim klicem izvedejo do konca, in pridobi časovni register, vendar ne jamči, da se izvedba ukazov za njim počaka. V ta namen se pred klicem ukaza RDTSC in za klicem ukaza RDTSCP uporabi ukaz CPUID, ki zapiše identifikacijske podatke procesorja v EAX-, EBX-, ECX- in EDX-registre glede na vrednost, predhodno vnešeno v register EAX, in tako služi kot izolacija klicev ukazov. Ker klic ukazov RDTSC in RDTSCP procesorju predstavlja dodatno delo, ki porabi nekaj procesorskih ciklov, je treba to dodatno delo predhodno izmeriti in ga odšteti od končnih meritev [32].

Izvorna koda 5.1 prikazuje strukturo programa za preizkušanje zmogljivosti, katere glavni del je zanka med vrsticami 5-22. Pred zanko se izvedejo inicializacije in deklaracije spremenljivk, kot so inicializacija ključa, IV, čistopisa idr. Število iteracij zanke $SteviloMeritev$ je sestavljeno iz števila segrevalnih iteracij in števila meritvenih iteracij. Koda zbirnega jezika med vrsticami 7-10 z ukazom RDTSC pridobi števec trenutnega

stanja procesorskih ciklov in ga shrani v spremenljivki *cycles_high* in *cycles_low*. Nato se kliče funkcija šifriranja oz. dešifriranja, ki jo želimo izmeriti. Takoj zatem se s kodo zbirnega jezika zopet pridobi števec trenutnega stanja procesorskih ciklov, tokrat z ukazom RDTSCP, in ga shrani v spremenljivki *cycles_high*1 in *cycles_low*1. V vrsticah 19-20 se pridobljene vrednosti stanja števcev pretvorijo v cela števila in shranijo v spremenljivki *Zacetek* in *Konec*, ki predstavljata vrednost števca procesorskih ciklov pred klicem in po klicu funkcije. V vrstici 21 se izračuna njuna razlika in zapiše v polje *Meritve*. V nadaljevanju se izmerjene vrednosti zapišejo v datoteko, pri čemer se prve vrednosti polja (število segrevalnih iteracij) ignorira.

Kot smo omenili, klici ukazov RDTSC in RDTSCP povzročajo dodatno delo, ki vpliva na končne meritve. Zato smo izvorno kodo 5.1 testno pognali brez klicev funkcij šifriranja ali dešifriranja v vrstici 12. Ugotovili smo, da merjenje hitrosti 'ničesar' na omenjeni napravi potrebuje 32 procesorskih ciklov. Da lahko natančno merimo hitrost izvajanja za nadaljnje teste, to vrednost končnemu rezultatu v vrstici 21 odštejemo.

Izsek programske kode 5.1: Program za preizkušanje zmogljivosti šifriranja in dešifriranja [32]

```
1  void main()
2  {
3      //Deklaracija in inicializacija spremenljivk
4
5      for(int i = 0; i < SteviloMeritev; i++)
6      {
7          asm volatile ("CPUID\n\t"
8              "RDTSC\n\t"
9              "mov %%edx, %0\n\t"
10             "mov %%eax, %1\n\t": "=r" (cycles_high), "=r" (cycles_low
                   ):: "%rax", "%rbx", "%rcx", "%rdx");
11
12         //Klic funkcije za sifriranje oz. desifriranje
13
14         asm volatile("RDTSCP\n\t"
15             "mov %%edx, %0\n\t"
16             "mov %%eax, %1\n\t"
```

```
17              "CPUID\n\t": "=r" (cycles_high1), "=r" (cycles_low1):: "%
                    rax", "%rbx", "%rcx", "%rdx");
18
19          Zacetek = (((uint64_t)cycles_high << 32) | cycles_low);
20          Konec = (((uint64_t)cycles_high1 << 32) | cycles_low1);
21          Meritve[i] = (uint64_t)(Konec - Zacetek) - 32;
22      }
23
24      //Shranjevanje meritev v datoteko
25  }
```

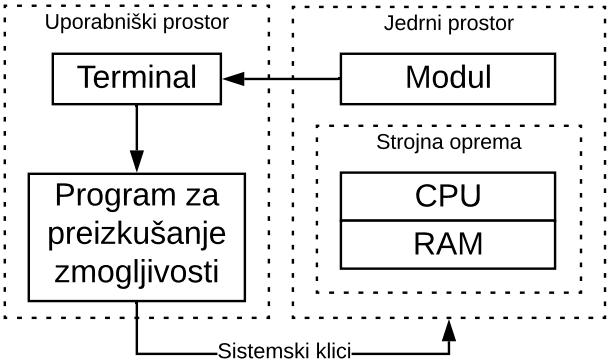
5.4 Poganjanje testov iz jedrnega prostora

Ker na izvajanje kode lahko vplivajo prekinitve časovnega razporejevalniga (angl. scheduler), ki izvaja 'pre-emption', je treba našo kodo pognati v jedrnem modulu, pri čemer lahko zagotovimo, da se med izvajanjem naše kode druga opravila ne izvajajo [32].

Operacijski sistem Linux zaradi varnosti in stabilnosti loči sistemski pomnilnik na dve regiji - jedrni prostor (angl. Kernel space) in uporabniški prostor (angl. User space). V jedrnem prostoru se izvaja jedro operacijskega sistema, ki nadzoruje celotno napravo in preko vmesnika ponuja uporabniškemu prostoru dostop do svojih storitev - npr. ustvarjanje novih procesov, komuniciranje z vhodno-izhodnimi napravami, rezervacijo prostora na pomnilniku idr. Uporabniški prostor lahko do teh storitev dostopa le preko sistemskih zahtev [6]. Na drugi strani je uporabniški prostor, pri katerem tečejo uporabniški procesi. Uporabniški procesi so instance vseh programov, razen jedrnih. Ko se uporabniški program požene, je iz shrambe naložen v uporabniški prostor, da lahko do njega z visoko hitrostjo dostopa CPE [7].

Ker programi iz uporabniškega prostora ne morejo dostopati do jedrnega prostora in posledično nimajo nadzora nad strojno opremo, je treba naš program za preizkušanje zmogljivosti pognati iz jedrnega prostora. To lahko storimo z jedrnim modulom (v jedrni prostor naložena koda, ki ima dostop do vseh funkcij jedra operacijskega sistema Linux), iz katerega lahko dostopamo do uporabniškega prostora z 'usermode-helper'

vmesnikom [2]. Na sliki 5.1 je prikazana struktura poganjanja testov iz jedrnega prostora. Iz naloženega modula z vmesnikom 'usermode-helper' dostopamo do terminala, preko katerega poženemo program za preizkušanje zmogljivosti, ta pa preko sistemskih klicev dostopa do CPE in bralno-pisalnega pomnilnika RAM.



Slika 5.1: Proces poganjanja testov iz jedrnega prostora [2][6][7]

Izvorna koda 5.2 predstavlja jedrni modul. V vrsticah 3-8 se izvedeta deklaracija in inicializacija spremenljivk, pri tem spremenljivka *argv* v vrstici 4 predstavlja argument pot do programa za preizkušanje zmogljivosti in spremenljivka *envp* pot do terminala. V vrstici 10 se kliče metoda *preempt_disable*(), ki izključi 'pre-emption' CPE, in takoj za njo v vrstici 11 *raw_local_irq_save*(), ki izključi trde prekinitve CPE (angl. Hard interrupts) in shrani trenutne nastavitve procesorja v spremenljivko *flags*. V vrstici 13, v kateri imamo popoln nadzor nad CPE, se kliče funkcija *call_usermodehelper*(), ki zažene test za preizkušanje zmogljivosti. S parametrom *UMH_WAIT_PROC* ji sporočimo, da je treba počakati, da se pred nadaljevanjem klicani program izvede do konca. Na koncu v vrsticah 15 in 16 povrnemo nastavitve procesorja v prejšnje stanje (vklopimo trde prekinitve in pre-emption).

Izsek programske kode 5.2: Jedrni modul za poganjanje programa za preizkušanje zmogljivosti [2][32]

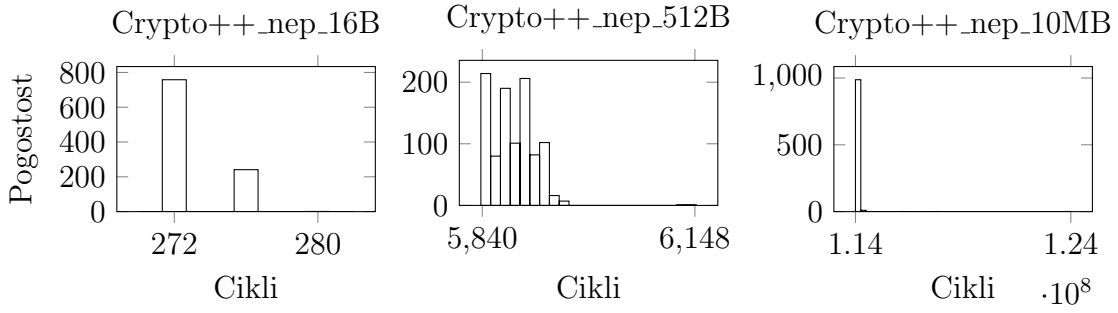
```
1 static int __init hello_start(void)
2 {
3     unsigned long flags;
4     char *argv[] = { "/pot/do/datoteke", NULL };
5     char *envp[] = {
6         "HOME=/",
```

```
7           "TERM=linux",
8           "PATH=/sbin:/bin:/usr/sbin:/usr/bin", NULL };
9
10      preempt_disable();
11      raw_local_irq_save(flags);
12
13      call_usermodehelper(argv[0], argv, envp, UMH_WAIT_PROC);
14
15      raw_local_irq_restore(flags);
16      preempt_enable();
17
18      return 0;
19  }
20
21  static void __exit hello_end(void){}
22
23  module_init(hello_start);
24  module_exit(hello_end);
```

6 Rezultati

Teste za preizkušanje zmogljivosti smo najprej izvajali za strojno pospešen in nepospešen algoritem AES v načinu ECB. Prav tako smo jih izvajali za strojno nepospešene algoritme Camellia, TwoFish, Serpent, DES in 3DES, pri tem pa smo se omejili samo na način ECB. Nazadnje smo testirali tudi CBC-, CTR-, GCM-, CCM-, OCB- in XTS-načine šifriranja pri strojno pospešenem izvajanju algoritma AES. Pri načinih šifriranja AEAD povezanih podatkov nismo vključili. Za algoritme AES, Camellia, TwoFish in Serpent smo uporabili 128-bitne ključe, za algoritem DES 56-bitni ključ in za algoritem 3DES 168-bitni ključ. Teste smo izvajali za knjižnice Libgcrypt, OpenSSL, Nettle in Crypto++ za čistopise in kriptograme velikosti 16 bajtov, 32 bajtov, 512 bajtov in 10 megabajtov. Za vsako kombinacijo je bilo izmerjenih 1000 meritev.

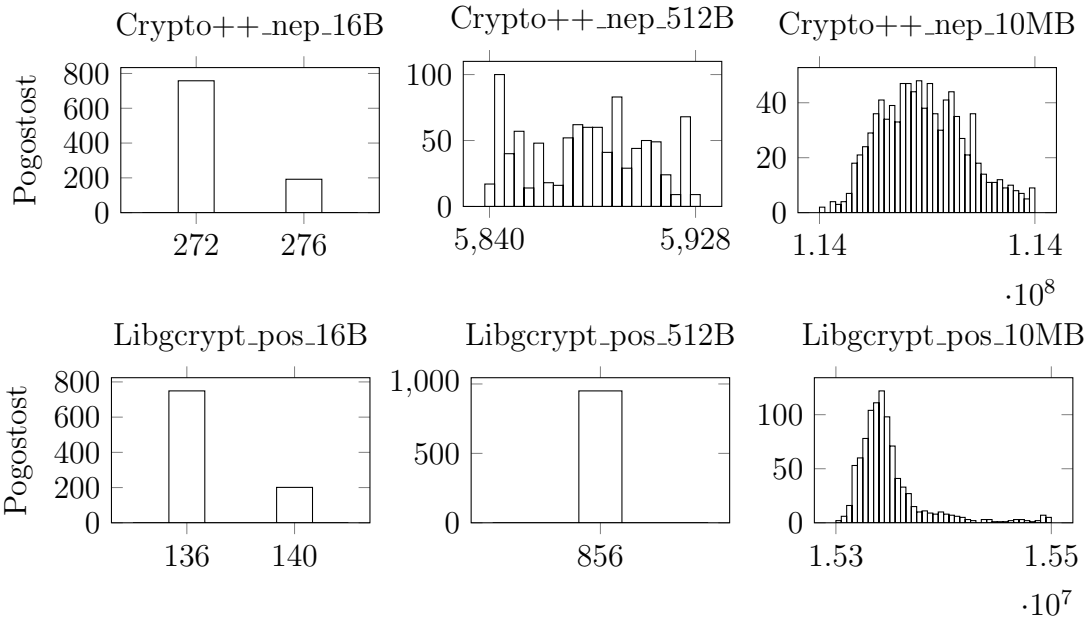
Graf 6.1 prikazuje histograme vrednosti izmerjenih hitrosti šifriranja čistopisa velikosti 16 B, 512 B in 10 MB z algoritmom AES (knjižnica Crypto++), pri čemer Y-os predstavlja pogostost izmerjenih vrednosti, X-os pa število procesorskih ciklov. Za poimenovanje histogramov je uporabljena notacija imeKnjižnice_načinIzvajanja_velikostČistopisa (način izvajanja predstavlja pospešeno - pos ali nepospešeno - nep).



Graf 6.1: Histogrami meritev šifriranja z osamelci za algoritem AES

Po vizualnem pregledu histogramov smo ugotovili, da vzorci na desni strani vsebujejo dolg rep osamelcev (graf 6.1). Ker se osamelci pojavljajo samo enostransko, smo vre-

dnosti vseh vzorcev razvrstili naraščajoče in odstranili 5 % največjih. Graf 6.2 prikazuje histrograme vzorcev z odstranjenimi osamelci (ostane 950 meritev na vzorec).



Graf 6.2: Histogrami meritev šifriranja za algoritem AES

6.1 Metodologija

Histogrami vzorcev nakazujejo, da vrednosti meritev šifriranja oz. dešifriranja čistopisov oz. kriptogramov niso porazdeljene normalno. Z večanjem velikosti čistopisa oz. kriptograma meritve težijo k normalni porazdelitvi. Zato smo za statistično analizo izbrali statistične teste, ki ne zahtevajo normalne porazdeljenosti podatkov. Izbrana statistična testa sta bila Kruskall-Wallisov H-test in Mann-Whitneyjev U-test, predstavljena v podpoglavjih 6.1.1 in 6.1.2, ki ju bomo uporabili pri potrjevanju hipotez H1, H2 in H3 na način:

- $\mathbf{H_1}$: Uporabili bomo Mann-Whitneyjev U-test, da ugotovimo, ali obstaja signifikantna razlika med pospešenim in nepospešenim izvajanjem algoritma AES v načinu ECB. Primerjali bomo vzorece meritev pospešenega v primerjavi z vzorcem nepospešenega šifriranja in dešifriranja za različne velikosti čistopisov oz. kriptogramov in različne knjižnice.

- **H_2**: Uporabili bomo Kruskall-Wallisov H-test, da ugotovimo, ali obstaja signifikantna razlika med vzorci nepospešenega izvajanja algoritmov AES, Camellia, TwoFish, Serpent, DES in 3DES v načinu ECB. V primerjavi bomo upoštevali tudi različne velikosti čistopisov in kriptogramov ter različne knjižnice. V primeru signifikantnih razlik bomo s post-hoc Mann-Whitneyjevimi U-testi ugotavljali signifikantne razlike med posameznimi vzorci omenjenih kombinacij.
- **H_3**: Uporabili bomo Kruskall-Wallisov H-test, da ugotovimo, ali obstaja signifikanta razlika med meritvami pospešenega izvajanja algoritma AES z različnimi načini šifriranja za različne velikosti čistopisov oz. kriptogramov za različne knjižnice. V primeru signifikantnih razlik bomo s post-hoc Mann-Whitneyjevimi U-testi ugotavljali signifikantne razlike med posameznimi načini šifriranja.

6.1.1 Kruskall-Wallisov H-test

Kruskall-Wallisov H-test je neparametriziran, na rangih temelječ statistični test, ki se uporablja kot pokazatelj signifikantnih razlik med dvema ali več neodvisnimi skupinami intervalnih ali ordinalnih odvisnih spremenljivk. Je razširitev statističnega Mann-Whitneyjevega U-testa, ki ga bomo predstavili v podpoglavju 6.1.2. Statistični test nam pokaže, ali je prišlo do signifikantnih razlik med dvema ali več neodvisnimi skupinami, ne pokaže pa, med katerimi. Zato je potrebno post hoc testiranje. Hipotezi H_0 in H_A definiramo [4]:

- **H_0**: med skupinami ni signifikantnih razlik,
- **H_A**: med skupinami so signifikantne razlike.

Pri uporabi Kruskall-Wallisovega H-testa ni treba, da so testirani podatki normalno porazdeljeni, morajo pa zadovoljevati naslednje štiri predpostavke [4]:

1. odvisna spremenljivka mora biti merjena ordinalno ali intervalno,
2. neodvisna spremenljivka mora biti sestavljena iz najmanj dveh kategoričnih neodvisnih skupin,

3. meritve morajo biti medsebojno neodvisne med skupinami in v skupinah,

4. v primeru, da imajo podatki enako variabilnost, se primerja mediano, v nasprotnem primeru pa povprečne range.

6.1.2 Mann-Whitneyjev U-test

Mann-Whitneyjev U-test se uporablja za primerjavo razlik med dvema neodvisnima skupinama ordinalnih ali intervalnih odvisnih spremenljivk. Hipotezi H_0 in H_A sta [3]:

- $\mathbf{H_0}$: med skupinama ni signifikante razlike,
- $\mathbf{H_A}$: med skupinama je signifikantna razlika.

Za podatke ni treba, da so normalno porazdeljeni, morajo pa zadoščati naslednjim štirim predpostavkam [3]:

1. odvisna spremenljivka mora biti merjena ordinalno ali intervalno,

2. neodvisno spremenljivko morata sestavljati dve kategorični neodvisni skupini,

3. meritve morajo biti medsebojno neodvisne med skupinami in v skupinah,

4. v primeru, da imajo podatki enako variabilnost, se primerja mediano, v nasprotnem primeru pa povprečne range.

6.2 Analiza

Kruskall-Wallisove H- in Mann-Whitneyjeve U-teste smo izvedli s pomočjo knjižnice SciPy [8], implementirane v programskem jeziku Python. Rezultate obeh testov interpretiramo tako, da vrednost p primerjamo s stopnjo značilnosti α. V primeru $p > \alpha$ sprejmemo hipotezo H_0, v primeru $p \leq \alpha$ pa jo zavrnemo in posledično sprejmemo hipotezo H_A [9]. V primerih, ko je vrednost $p = 0$, pomeni, da je vrednost p tako majhna,

da je ni bilo mogoče prikazati s tipom števila float. U- in H-vrednosti v našem primeru za ugotavljanje signifikantnih razlik niso tako pomembne in jih lahko ignoriramo. Za stopnjo značilnosti smo izbrali vrednost $\alpha = 0.05$, ki smo jo za primerjavo med več skupinami primerjanih spremenljivk sproti prilagodili z Bonferronijevim popravkom.

6.2.1 Primerjava hitrosti strojno pospešenega in nepospešenega algoritma AES

Tabela 6.1 vsebuje izračunane povprečne vrednosti porabljenih procesorskih ciklov za šifriranje oz. dešifriranje različnih velikosti čistopisov oz. kriptogramov za strojno pospešen in nepospešen algoritm AES v načinu ECB in hitrosti, izračunane kot število porabljenih procesorskih ciklov na bajt CpB. Povprečne vrednosti procesorskih ciklov so izračunane iz vzorcev velikosti $n = 950$, vrednosti hitrosti pa z enačbo 6.1.

$$CpB = \frac{PovprecjeProcesorskihCiklov}{VelikostCistopisaAliKriptograma} \tag{6.1}$$

Za vzorce, katerih povprečne vrednosti so v tabeli 6.1, smo v tabeli 6.2 s statističnim Mann-Whitneyjevim U-testom izračunali U- in p-vrednosti. Primerjali smo vzorce strojno pospešenega šifriranja z nepospešenim šifriranjem in pospešenega dešifriranja z nepospešenim. Primerjave smo izvedli za različne velikosti čistopisov oz. kriptogramov in različne knjižnice. Po primerjavi rezultatov s stopnjo značilnosti $\alpha = 0.05$ smo ugotovili, da v vseh primerih prihaja do signifikantnih razlik, zato smo vse hipoteze H_0 zavrnili in v nadaljevanju primerjali povprečna števila procesorskih ciklov za šifriranje in dešifriranje različnih velikosti čistopisov in kriptogramov.

Tabela 6.1: Povprečne vrednosti meritev strojno pospešenega in nepospešenega algoritma AES v načinu ECB

n = 950	Šifriranje				Dešifriranje			
	Nepospešeno		Pospešeno		Nepospešeno		Pospešeno	
Knjižnica	Povprečje	CpB	Povprečje	CpB	Povprečje	CpB	Povprečje	CpB
Velikost čistopisa/kriptograma: 16 bajtov								
OpenSSL	189,971	11,873	106,901	6,681	241,065	15,067	122,265	7,642
Nettle	192,0	12,0	72,859	4,554	193,874	12,117	72,0	4,5
Libgcrypt	220,0	13,75	136,846	8,553	259,949	16,247	133,941	8,371
Crypto++	272,808	17,05	100,0	6,25	317,693	19,856	96,0	6,0
Velikost čistopisa/kriptograma: 32 bajtov								
OpenSSL	337,773	10,555	110,606	3,456	429,903	13,434	125,987	3,937
Nettle	374,051	11,689	92,0	2,875	377,326	11,791	88,939	2,779
Libgcrypt	380,101	11,878	152,0	4,75	453,053	14,158	149,891	4,684
Crypto++	458,438	14,326	116,905	3,653	609,491	19,047	126,943	3,967
Velikost čistopisa/kriptograma: 512 bajtov								
OpenSSL	4859,663	9,492	385,693	0,753	6060,088	11,836	409,225	0,799
Nettle	5882,737	11,49	692,0	1,352	5862,299	11,45	659,495	1,288
Libgcrypt	5095,625	9,952	856,0	1,672	6308,526	12,321	907,899	1,773
Crypto++	5881,903	11,488	419,331	0,819	9500,669	18,556	416,535	0,814
Velikost čistopisa/kriptograma: 10 000 000 bajtov								
OpenSSL	92676103,68	9,268	6772415,992	0,677	116660502,674	11,666	6774777,263	0,677
Nettle	115452521,478	11,545	12859427,486	1,286	115294557,478	11,529	12382681,368	1,238
Libgcrypt	96922697,461	9,692	15344504,463	1,534	121695525,406	12,17	15769204,021	1,577
Crypto++	113579884,771	11,358	7317001,326	0,732	185116331,053	18,512	7320588,392	0,732

Tabela 6.2: U- in p-vrednosti Mann-Whitneyjevih U-testov strojno pospešenega in nepospešenega algoritma AES v načinu ECB

n = 950	Šifriranje		Dešifriranje		Šifriranje		Dešifriranje	
Velikost	U	p	U	p	U	p	U	p
	OpenSSL				Libgcrypt			
16B	902500,0	0,0	902500,0	0,0	902500,0	0,0	902500,0	0,0
32B	902500,0	0,0	902500,0	0,0	902500,0	0,0	902500,0	0,0
512B	902500,0	0,0	902500,0	0,0	902500,0	0,0	902500,0	0,0
10MB	902500,0	0,0	902500,0	0,0	902500,0	0,0	902500,0	0,0
	Nettle				Crypto++			
16B	902500,0	0,0	902500,0	0,0	902500,0	0,0	902500,0	0,0
32B	902500,0	0,0	902500,0	0,0	902500,0	0,0	902500,0	0,0
512B	902500,0	0,0	902500,0	0,0	902500,0	0,0	902500,0	0,0
10MB	902500,0	0,0	902500,0	0,0	902500,0	0,0	902500,0	0,0

Po primerjavi povprečnih vrednosti meritev tabele 6.1 smo ugotovili:

- V knjižnici OpenSSL je strojno pospešeno šifriranje 1,8-krat, 3,1-krat, 12,6-krat in 13,7-krat hitrejše kot nepospešeno za velikosti čistopisov 16 B, 32 B, 512 B in 10 MB. Strojno pospešeno dešifriranje je v primerjavi z nepospešenim hitrejše 2-krat, 3,4-krat, 14,8-krat in 17,2-krat za velikosti kriptogramov 16 B, 32 B, 512 B in 10 MB.

- V knjižnici Nettle je strojno pospešeno šifriranje 2,6-krat, 4,1-krat, 8,5-krat in 9-krat hitrejše kot nepospešeno za velikosti čistopisov 16 B, 32 B, 512 B in 10 MB. Strojno pospešeno dešifriranje je v primerjavi z nepospešenim hitrejše 2,7-krat, 4,2-krat, 8,9-krat in 9,3-krat za velikosti kriptogramov 16 B, 32 B, 512 B in 10 MB.

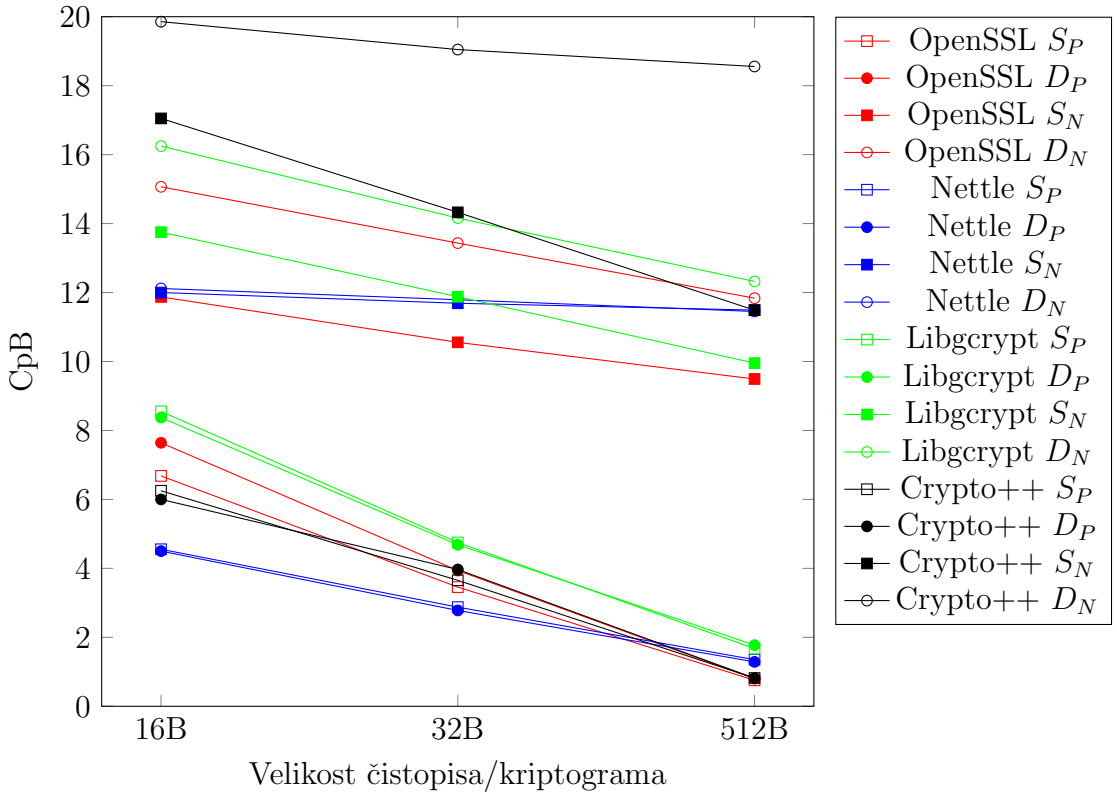
- V knjižnici Libgcrypt je strojno pospešeno šifriranje 1,6-krat, 2,5-krat, 6-krat in 6,3-krat hitrejše kot nepospešeno za velikosti čistopisov 16 B, 32 B, 512 B in 10 MB. Strojno pospešeno dešifriranje je v primerjavi z nepospešenim hitrejše 1,9-krat, 3-krat, 7-krat in 7,7-krat za velikosti kriptogramov 16 B, 32 B, 512 B in 10 MB.

- V knjižnici Crypto++ je strojno pospešeno šifriranje 2,7-krat, 3,9-krat, 14-krat

in 15,5-krat hitrejše kot nepospešeno za velikosti čistopisov 16 B, 32 B, 512 B in 10 MB. Strojno pospešeno dešifriranje je v primerjavi z nepospešenim hitrejše 3,3-krat, 4,8-krat, 22,8-krat in 25,3-krat za velikosti kriptogramov 16 B, 32 B, 512 B in 10 MB.

Primerjave smo zaključili z ugotovitvijo, da je pospešeno šifriranje v primerjavi z nepospešenim v povprečju hitrejše 2,2-krat, 3,4-krat, 10,3-krat in 11,1-krat za velikosti čistopisov 16 B, 32 B, 512 B in 10 MB. Pospešeno dešifriranje je 2,5-krat, 3,9-krat, 13,4-krat in 14,9-krat hitrejše kot nepospešeno za velikosti kriptogramov 16 B, 32 B, 512 B in 10 MB.

Na grafu 6.3 so prikazane hitrosti pospešenega in nepospešenega algoritma AES v načinu ECB za različne velikosti čistopisa oz. kriptograma. Oznake v legendi predstavljajo: S_P pospešeno šifriranje, D_P pospešeno dešifriranje, S_N nepospešeno šifriranje in D_N nepospešeno dešifriranje.



Graf 6.3: Primerjava hitrosti strojno pospešenega in nepospešenega algoritma AES v načinu ECB

6.2.2 Primerjava hitrosti strojno nepospešenih šifer

Tabeli 6.3 in 6.4 vsebujeta izračunane povprečne vrednosti vzorcev velikosti $n = 950$ in hitrosti, preračunane v CpB za nepospešeno šifriranje, in dešifriranje različnih velikosti čistopisov oz. kriptogramov z implementacijami algoritmov AES, Camellia, DES, 3DES, TwoFish in Serpent v načinu ECB za različne knjižnice.

Za vzorce, katerih povprečja so podana v tabelah 6.3 in 6.4, smo v tabeli 6.5 s statističnim Kruskall-Wallisovim H-testom izračunali H- in p-vrednosti. Primerjali smo vzorce nepospešenega šifriranja in dešifriranja z različnimi algoritmi in jih med seboj primerjali za različne velikosti čistopisov oz. kriptogramov in različne knjižnice. Po primerjavi rezultatov s stopnjo značilnosti $\alpha = 0.05$ smo ugotovili, da je v vseh primerih prišlo do signifikantnih razlik, zato smo vse hipoteze H_0 zavrnili.

Ker so statistični testi pokazali signifikantne razlike med vzorci, smo naknadno izvedli še post hoc Mann-Whitneyjeve U-teste, s katerimi smo vzorce primerjali paroma. Rezultati statističnih Mann-Whitneyjevih U-testov so na voljo v prilogi A v tabelah A.1, A.2, A.3 in A.4 za knjižnice OpenSSL, Nettle, Libgcrypt in Crypto++. V tabeli so po vrsticah primerjane vse kombinacije algoritmov (algoritem X proti algoritmu Y), zraven pa rezultati Mann-Whitneyjevih U-testov za šifriranje in dešifriranje. Primerjave so izvedene za različne velikosti čistopisa/kriptograma.

Za post hoc teste smo prilagodili stopnjo značilnosti, zaokroženo na $\alpha = 0.001$ (48 primerjav) za tabelo A.1 in $\alpha = 0.0008$ (60 primerjav) za tabele A.2, A.3 in A.4. Po primerjavi stopenj značilnosti s p-vrednostmi smo ugotovili, da v vseh primerih prihaja do signifikantnih razlik. Zato smo v nadaljevanju primerjali povprečna števila procesorskih ciklov za šifriranje in dešifriranje različnih velikosti čistopisov in kriptogramov.

Tabela 6.3: Povprečne vrednosti meritev strojno nepospešenih algoritmov v načinu ECB za knjižnici OpenSSL in Nettle

n = 950	OpenSSL				Nettle			
	Šifriranje		Dešifriranje		Šifriranje		Dešifriranje	
Alg.	Povprečje	CpB	Povprečje	CpB	Povprečje	CpB	Povprečje	CpB
Velikost čistopisa/kriptograma: 16 bajtov								
AES	189,971	11,873	241,065	15,067	192,0	12,0	193,874	12,117
Camellia	336,425	21,027	351,419	21,964	319,815	19,988	319,773	19,986
DES	767,735	47,983	756,211	47,263	669,697	41,856	675,158	42,197
3DES	2033,095	127,068	2008,194	125,512	1989,764	124,36	1994,661	124,666
TwoFish	/	/	/	/	330,96	20,685	298,265	18,642
Serpent	/	/	/	/	613,196	38,325	518,333	32,396
Velikost čistopisa/kriptograma: 32 bajtov								
AES	337,773	10,555	429,903	13,434	374,051	11,689	377,326	11,791
Camellia	652,295	20,384	664,501	20,766	635,221	19,851	634,796	19,837
DES	1487,781	46,493	1450,442	45,326	1305,701	40,803	1318,442	41,201
3DES	3995,149	124,848	3951,836	123,495	3929,263	122,789	3952,501	123,516
TwoFish	/	/	/	/	653,309	20,416	592,147	18,505
Serpent	/	/	/	/	1221,423	38,169	1029,558	32,174
Velikost čistopisa/kriptograma: 512 bajtov								
AES	4859,663	9,492	6060,088	11,836	5882,737	11,49	5862,299	11,45
Camellia	9534,716	18,622	9653,848	18,855	9985,642	19,503	9971,347	19,475
DES	23107,781	45,132	22279,945	43,516	20471,52	39,983	20776,952	40,58
3DES	63091,011	123,225	62309,566	121,698	61733,389	120,573	62220,568	121,525
TwoFish	/	/	/	/	10355,592	20,226	9385,057	18,33
Serpent	/	/	/	/	7227,015	14,115	6710,968	13,107
Velikost čistopisa/kriptograma: 10 000 000 bajtov								
AES	92676103,68	9,268	116660502,674	11,666	115452521,478	11,545	115294557,478	11,529
Camellia	186892253,933	18,689	189003458,286	18,9	196084080,143	19,608	196108824,863	19,611

Nadaljevanje na naslednji strani

Tabela 6.3 – *Nadaljevanje s prejšnje strani*

n = 950	OpenSSL				Nettle			
	Šifriranje		Dešifriranje		Šifriranje		Dešifriranje	
Alg.	Povprečje	CpB	Povprečje	CpB	Povprečje	CpB	Povprečje	CpB
DES	452066770,328	45,207	438266891,002	43,827	403387987,966	40,339	408229637,971	40,823
3DES	1247151880,434	124,715	1239886295,76	123,989	1214615872,779	121,462	1219174613,718	121,917
TwoFish	/	/	/	/	198178345,646	19,818	183225480,644	18,323
Serpent	/	/	/	/	142096565,297	14,21	131864382,383	13,186

Tabela 6.4: Povprečne vrednosti meritev strojno nepospešenih algoritmov v načinu ECB za knjižnici Libgcrypt in Crypto++

n = 950	Libcrypt				Crypto++			
	Šifriranje		Dešifriranje		Šifriranje		Dešifriranje	
Alg.	Povprečje	CpB	Povprečje	CpB	Povprečje	CpB	Povprečje	CpB
Velikost čistopisa/kriptograma: 16 bajtov								
AES	220,0	13,75	259,949	16,247	272,808	17,05	317,693	19,856
Camellia	418,556	26,16	406,766	25,423	373,124	23,32	371,444	23,215
DES	726,181	45,386	718,884	44,93	695,739	43,484	694,211	43,388
3DES	1549,735	96,858	1546,337	96,646	1873,499	117,094	1874,286	117,143
TwoFish	349,966	21,873	346,202	21,638	313,604	19,6	311,486	19,468
Serpent	663,469	41,467	608,312	38,02	604,615	37,788	534,615	33,413
Velikost čistopisa/kriptograma: 32 bajtov								
AES	380,101	11,878	453,053	14,158	458,438	14,326	609,491	19,047
Camellia	729,448	22,795	726,728	22,71	725,587	22,675	731,549	22,861
DES	1360,404	42,513	1355,558	42,361	1360,821	42,526	1362,682	42,584
3DES	3001,566	93,799	2988,653	93,395	3729,259	116,539	3729,92	116,56
TwoFish	617,983	19,312	612,387	19,137	598,358	18,699	604,775	18,899
Serpent	1235,474	38,609	1115,352	34,855	1184,619	37,019	1063,604	33,238

Nadaljevanje na naslednji strani

Tabela 6.4 – *Nadaljevanje s prejšnje strani*

n = 950	Libcrypt				Crypto++			
	Šifriranje		Dešifriranje		Šifriranje		Dešifriranje	
Alg.	Povprečje	CpB	Povprečje	CpB	Povprečje	CpB	Povprečje	CpB
Velikost čistopisa/kriptograma: 512 bajtov								
AES	5095,625	9,952	6308,526	12,321	5881,903	11,488	9500,669	18,556
Camellia	10235,091	19,99	10285,242	20,088	11350,872	22,17	11383,002	22,232
DES	20514,404	40,067	20533,048	40,104	21325,718	41,652	21336,093	41,672
3DES	46757,579	91,323	46486,577	90,794	58910,404	115,059	58926,375	115,091
TwoFish	8601,246	16,799	8637,613	16,87	9374,497	18,31	9322,139	18,207
Serpent	18543,297	36,217	16832,105	32,875	18548,968	36,228	16744,232	32,704
Velikost čistopisa/kriptograma: 10 000 000 bajtov								
AES	96922697,461	9,692	121695525,406	12,17	113579884,771	11,358	185116331,053	18,512
Camellia	198853363,389	19,885	198813767,735	19,881	221832938,16	22,183	222596231,558	22,26
DES	401882197,541	40,188	401774600,497	40,177	418741015,68	41,874	418715389,983	41,872
3DES	913179130,446	91,318	910082554,299	91,008	1152749754,766	115,275	1152737492,173	115,274
TwoFish	167333491,975	16,733	168265179,613	16,827	182651562,408	18,265	182148173,528	18,215
Serpent	362125915,364	36,213	325488755,259	32,549	363018876,08	36,302	327816316,004	32,782

Tabela 6.5: H- in p-vrednosti Kruskall-Wallisovih H-testov strojno nepospešenih algoritmov v načinu ECB

n = 950	Šifriranje		Dešifriranje		Šifriranje		Dešifriranje	
Veli. Č/K	H	p	H	p	H	p	H	p
	OpenSSL				Libgcrypt			
16B	3582,072	0,0	3582,376	0,0	5589,643	0,0	5568,86	0,0
32B	3591,097	0,0	3576,499	0,0	5557,801	0,0	5568,941	0,0
512B	3564,037	0,0	3563,026	0,0	5543,655	0,0	5544,091	0,0
10MB	3561,563	0,0	3561,563	0,0	5540,695	0,0	5540,695	0,0
	Nettle				Crypto++			
16B	5599,184	0,0	5576,936	0,0	5581,878	0,0	5523,085	0,0
32B	5572,923	0,0	5573,784	0,0	5566,731	0,0	5446,796	0,0
512B	5544,457	0,0	5552,988	0,0	5544,889	0,0	5546,394	0,0
10MB	5540,695	0,0	5540,695	0,0	5540,695	0,0	5540,695	0,0

Tabela 6.6 vsebuje razvrstitev algoritmov po hitrosti padajoče (za prvi algoritem je navedena njegova hitrost, za ostale pa zaostanek za njo). Ugotovimo, da je v primeru šifriranja algoritem AES v vseh primerih najhitrejši, DES in 3DES pa zmeraj zavzemata predzadnje in zadnje mesto. V knjižnici Nettle algoritem Camellia zaseda drugo mesto, algoritem TwoFish tretje in Serpent četrto. Pri velikosti čistopisa 512 B in 10 MB se algoritem Serpent prebije na drugo mesto. V knjižnicah Libgcrypt in Crypto++ za vse velikosti čistopisa algoritem TwoFish zasede drugo, Camellia tretje in Serpent četrto mesto.

Tabela 6.6: Razvrstitev hitrosti strojno nepospešenega šifriranja z algoritmi v načinu ECB

	OpenSSL		Nettle		Libgcrypt		Crypto++	
#	Alg.	CpB	Alg.	CpB	Alg.	CpB	Alg.	CpB
Velikost čistopisa/kriptograma: 16 bajtov								
1	AES	11,873	AES	12,0	AES	13,75	AES	17,05
2	Came	+9,154	Came	+7,988	TwoF	+8,123	TwoF	+2,55
3	DES	+36,11	TwoF	+8,685	Came	+12,41	Came	+6,27
4	3DES	+115,195	Serp	+26,325	Serp	+27,717	Serp	+20,738
5	/	/	DES	+29,856	DES	+31,636	DES	+26,434
6	/	/	3DES	+112,36	3DES	+83,108	3DES	+100,044
Velikost čistopisa/kriptograma: 32 bajtov								
1	AES	10,555	AES	11,689	AES	11,878	AES	14,326
2	Came	+9,829	Came	+8,162	TwoF	+7,434	TwoF	+4,373
3	DES	+35,938	TwoF	+8,727	Came	+10,917	Came	+8,349

Nadaljevanje na naslednji strani

Tabela 6.6 – *Nadaljevanje s prejšnje strani*

	OpenSSL		Nettle		Libgcrypt		Crypto++	
#	Alg.	CpB	Alg.	CpB	Alg.	CpB	Alg.	CpB
4	3DES	+114,293	Serp	+26,48	Serp	+26,731	Serp	+22,693
5	/	/	DES	+29,114	DES	+30,635	DES	+28,2
6	/	/	3DES	+111,1	3DES	+81,921	3DES	+102,213
Velikost čistopisa/kriptograma: 512 bajtov								
1	AES	9,492	AES	11,49	AES	9,952	AES	11,488
2	Came	+9,13	Serp	+2,625	TwoF	+6,847	TwoF	+6,822
3	DES	+35,64	Came	+8,013	Came	+10,038	Came	+10,682
4	3DES	+113,733	TwoF	+8,736	Serp	+26,265	Serp	+24,74
5	/	/	DES	+28,493	DES	+30,115	DES	+30,164
6	/	/	3DES	+109,083	3DES	+81,371	3DES	+103,571
Velikost čistopisa/kriptograma: 10 000 000 bajtov								
1	AES	9,268	AES	11,545	AES	9,692	AES	11,358
2	Came	+9,421	Serp	+2,665	TwoF	+7,041	TwoF	+6,907
3	DES	+35,939	Came	+8,063	Came	+10,193	Came	+10,825
4	3DES	+115,447	TwoF	+8,273	Serp	+26,521	Serp	+24,944
5	/	/	DES	+28,794	DES	+30,496	DES	+30,516
6	/	/	3DES	+109,917	3DES	+81,626	3DES	+103,917

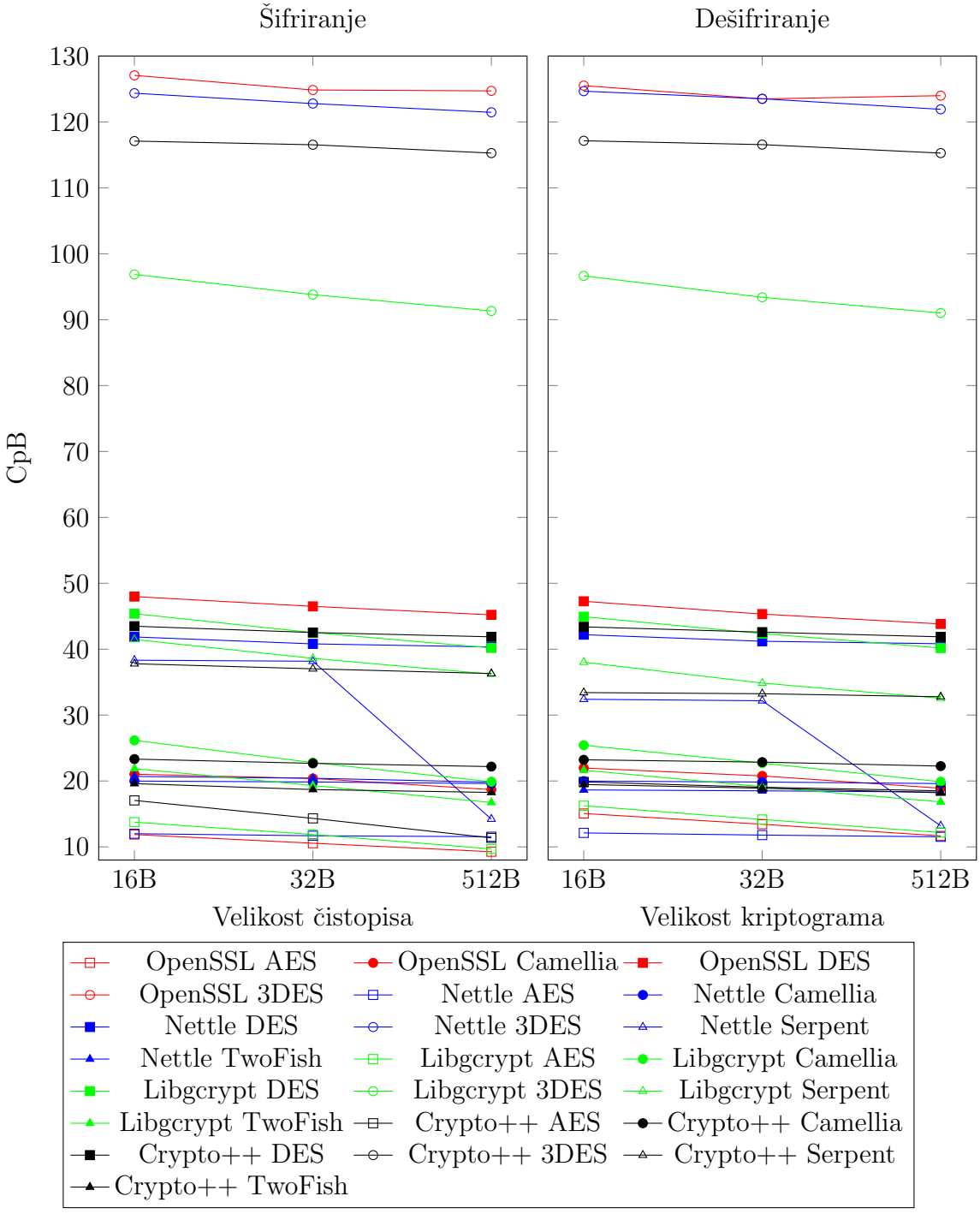
Tabela 6.7 vsebuje vrednosti hitrosti dešifiranja. Iz nje razberemo, da algoritem AES zaseda prvo mesto za vse velikosti kriptogramov v knjižnicah OpenSSL, Nettle in Libgcrypt, v knjižnici Crypto++ pa takoj za algoritmom TwoFish drugo mesto. Algoritma DES in 3DES, tako kot pri šifriranju, v vseh primerih zavzemata predzadnje in zadnje mesto. V knjižnicah OpenSSL, Libgcrypt in Crypto++ (z omenjeno razliko) je razvrstitev enaka kot pri šifriranju. Dešifriranje z algoritmom TwoFish je v knjižnici Nettle v vseh primerih hitrejše od dešifriranja z algoritmom Camellia, prav tako pa ju algoritem Serpent tako kot pri šifriranju prehiti tudi pri velikostih kriptograma 512 B in 10 MB.

Tabela 6.7: Razvrstitev hitrosti strojno nepospešenega dešifriranja z algoritmi v načinu ECB

	OpenSSL		Nettle		Libgcrypt		Crypto++	
#	Alg.	CpB	Alg.	CpB	Alg.	CpB	Alg.	CpB
Velikost čistopisa/kriptograma: 16 bajtov								
1	AES	15,067	AES	12,117	AES	16,247	TwoF	19,468
2	Came	+6,897	TwoF	+6,525	TwoF	+5,391	AES	+0,388
3	DES	+32,196	Came	+7,869	Came	+9,176	Came	+3,747
4	3DES	+110,445	Serp	+20,279	Serp	+21,773	Serp	+13,945
5	/	/	DES	+30,08	DES	+28,683	DES	+23,92
6	/	/	3DES	+112,549	3DES	+80,399	3DES	+97,675
Velikost čistopisa/kriptograma: 32 bajtov								
1	AES	13,434	AES	11,791	AES	14,158	TwoF	18,899
2	Came	+7,332	TwoF	+6,714	TwoF	+4,979	AES	+0,148
3	DES	+31,892	Came	+8,046	Came	+8,552	Came	+3,962
4	3DES	+110,061	Serp	+20,383	Serp	+20,697	Serp	+14,339
5	/	/	DES	+29,41	DES	+28,203	DES	+23,685
6	/	/	3DES	+111,725	3DES	+79,237	3DES	+97,661
Velikost čistopisa/kriptograma: 512 bajtov								
1	AES	11,836	AES	11,45	AES	12,321	TwoF	18,207
2	Came	+7,019	Serp	+1,657	TwoF	+4,549	AES	+0,349
3	DES	+31,68	TwoF	+6,88	Came	+7,767	Came	+4,025
4	3DES	+109,862	Came	+8,025	Serp	+20,554	Serp	+14,497
5	/	/	DES	+29,13	DES	+27,783	DES	+23,465
6	/	/	3DES	+110,075	3DES	+78,473	3DES	+96,884
Velikost čistopisa/kriptograma: 10 000 000 bajtov								
1	AES	11,666	AES	11,529	AES	12,17	TwoF	18,215
2	Came	+7,234	Serp	+1,657	TwoF	+4,657	AES	+0,297
3	DES	+32,161	TwoF	+6,794	Came	+7,711	Came	+4,045
4	3DES	+112,323	Came	+8,082	Serp	+20,379	Serp	+14,567
5	/	/	DES	+29,294	DES	+28,007	DES	+23,657
6	/	/	3DES	+110,388	3DES	+78,838	3DES	+97,059

Primerjave hitrosti šifriranj in dešifriranj v tabelah 6.6 in 6.7 zaključimo z ugotovitvijo, da je nepospešeni algoritem AES najhitrejši v vseh primerih šifriranja in dešifriranja, razen v knjižnici Crypto++ pri dešifriranju.

Na grafu 6.4 je prikazana primerjava hitrosti nepospešenega šifriranja in dešifriranja z različnimi algoritmi za različne knjižnice. Opazimo rahel trend naraščanja hitrosti, do katerega prihaja zaradi različnih inicializacijskih časov knjižnic, katerih vpliv na hitrost se z večanjem velikosti čistopisa oz. kriptograma zmanjšuje.



Graf 6.4: Primerjava hitrosti strojno nepospešenih algoritmov v načinu ECB

6.2.3 Primerjava hitrosti strojno pospešenih načinov šifriranja

Tabeli 6.9 in 6.10 vsebujeta izračunane povprečne vrednosti vzorcev velikosti $n = 950$ in hitrosti, preračunane v CpB pri strojno pospešenem šifriranju in dešifriranju različnih

velikosti čistopisov oz. kriptogramov z algoritmom AES v ECB-, CBC-, CTR-, CCM-, GCM-, OCB- in XTS-načinih šifriranja za različne knjižnice.

Za vzorce, katerih povprečja so podana v tabelah 6.9 in 6.10, smo v tabeli 6.8 s statističnim Kruskall-Wallisovim H-testom za šifriranje in dešifriranje z načini šifriranja v različnih knjižnicah izračunali H- in p-vrednosti. Primerjali smo vzorce pospešenega šifriranja in dešifriranja različnih načinov šifriranja pri različnih velikostih čistopisov oz. kriptogramov za različne knjižnice. Za primerjavo smo uporabili stopnjo značilnosti $\alpha = 0.05$. Po primerjavi rezultatov smo ugotovili, da je v vseh primerih prišlo do signifikantnih razlik, zato smo vse hipoteze H_0 zavrnili.

Ker so statistični testi pokazali signifikantne razlike, smo naknadno izvedli še post hoc Mann-Whitneyjeve U-teste, s katerimi smo paroma primerjali vzorce in jih navedli v prilogi B v tabelah B.1, B.2, B.3 in B.4 za knjižnice OpenSSL, Nettle, Libgcrypt in Crypto++. V tabeli so po vrsticah primerjane vse kombinacije načinov šifriranja s pospešenim algoritmom AES (način X proti načinu Y), zraven pa so rezultati Mann-Whitneyjevih U-testov za šifriranje in dešifriranje. Za post hoc teste smo prilagodili stopnjo značilnosti, zaokroženo na $\alpha = 0.00029$ (168 primerjav) za tabeli B.1 in B.3, in $\alpha = 0.0006$ (80 primerjav) za tabeli B.2 in B.4. Po primerjavi stopenj značilnosti z vrednostmi p smo ugotovili, da je v vseh post hoc primerjavah prišlo do signifikantnih razlik.

Tabela 6.8: H- in p-vrednosti Kruskall-Wallisovih H-testov strojno pospešenega algoritma AES z različnimi načini šifriranja

n = 950	Šifriranje		Dešifriranje		Šifriranje		Dešifriranje	
Veli. Č/K	H	p	H	p	H	p	H	p
	OpenSSL				Libgcrypt			
16B	6424,079	0,0	6397,642	0,0	6463,351	0,0	6575,891	0,0
32B	6532,739	0,0	6532,616	0,0	6559,807	0,0	6472,316	0,0
512B	6529,57	0,0	6492,581	0,0	6584,978	0,0	6553,299	0,0
10MB	6406,041	0,0	6508,436	0,0	6513,306	0,0	6513,294	0,0
	Nettle				Crypto++			
16B	4645,832	0,0	4645,165	0,0	4633,15	0,0	4594,575	0,0
32B	4637,629	0,0	4633,26	0,0	4608,14	0,0	4595,957	0,0
512B	4681,308	0,0	4635,612	0,0	4578,689	0,0	4529,987	0,0
10MB	4559,04	0,0	4559,04	0,0	4559,04	0,0	4559,04	0,0

Tabela 6.9: Povprečne vrednosti meritev strojno pospešenega algoritma AES z različnimi načini šifriranja za knjižnici OpenSSL in Libgcrypt

n = 950	OpenSSL				Libgcrypt			
	Šifriranje		Dešifriranje		Šifriranje		Dešifriranje	
Način	Povprečje	CpB	Povprečje	CpB	Povprečje	CpB	Povprečje	CpB
Velikost čistopisa/kriptograma: 16 bajtov								
ECB	106,901	6,681	122,265	7,642	136,846	8,553	133,941	8,371
CBC	113,592	7,1	126,585	7,912	155,794	9,737	152,0	9,5
CTR	151,714	9,482	166,08	10,38	153,874	9,617	148,0	9,25
GCM	351,137	21,946	365,528	22,846	391,448	24,466	418,973	26,186
CCM	497,865	31,117	585,634	36,602	300,636	18,79	344,181	21,511
OCB	349,213	21,826	368,531	23,033	359,002	22,438	381,823	23,864
XTS	206,771	12,923	223,806	13,988	225,444	14,09	223,339	13,959
Velikost čistopisa/kriptograma: 32 bajtov								
ECB	110,606	3,456	125,987	3,937	152,0	4,75	149,891	4,684
CBC	196,189	6,131	143,726	4,491	230,257	7,196	160,838	5,026
CTR	172,16	5,38	186,632	5,832	165,28	5,165	158,131	4,942
GCM	410,922	12,841	395,309	12,353	423,971	13,249	460,88	14,402
CCM	535,116	16,722	651,204	20,35	395,486	12,359	417,293	13,04
OCB	352,636	11,02	368,636	11,52	372,846	11,651	392,653	12,27
XTS	209,36	6,542	228,526	7,141	325,078	10,159	321,592	10,05
Velikost čistopisa/kriptograma: 512 bajtov								
ECB	385,693	0,753	409,225	0,799	856,0	1,672	907,899	1,773
CBC	2313,693	4,519	397,474	0,776	2392,0	4,672	456,316	0,891
CTR	432,046	0,844	446,733	0,873	478,964	0,935	477,853	0,933
GCM	941,322	1,839	928,737	1,814	1129,958	2,207	1153,642	2,253
CCM	2665,549	5,206	2790,64	5,45	2868,051	5,602	2896,109	5,656
OCB	802,842	1,568	814,547	1,591	706,547	1,38	734,303	1,434

Nadaljevanje na naslednji strani

Tabela 6.9 – *Nadaljevanje s prejšnje strani*

n = 950	OpenSSL				Libgcrypt			
	Šifriranje		Dešifriranje		Šifriranje		Dešifriranje	
Način	Povprečje	CpB	Povprečje	CpB	Povprečje	CpB	Povprečje	CpB
XTS	543,739	1,062	558,8	1,091	3295,954	6,437	3291,221	6,428
Velikost čistopisa/kriptograma: 10 000 000 bajtov								
ECB	6772415,992	0,677	6774777,263	0,677	15344504,463	1,534	15769204,021	1,577
CBC	44640748,135	4,464	6524859,179	0,652	45278127,124	4,528	7192993,406	0,719
CTR	6620587,634	0,662	6610300,257	0,661	7439279,335	0,744	7447869,823	0,745
GCM	10288819,684	1,029	10293126,749	1,029	16207750,964	1,621	16174113,061	1,617
CCM	44840933,731	4,484	44811908,328	4,481	52782478,32	5,278	52751670,32	5,275
OCB	7183977,764	0,718	7265547,768	0,727	7989388,013	0,799	8212218,985	0,821
XTS	7172139,469	0,717	7158975,158	0,716	64381144,762	6,438	64454151,651	6,445

Tabela 6.10: Povprečne vrednosti meritev strojno pospešenega algoritma AES z različnimi načini šifriranja za knjižnici Nettle in Crypto++

n = 950	Nettle				Crypto++			
	Šifriranje		Dešifriranje		Šifriranje		Dešifriranje	
Način	Povprečje	CpB	Povprečje	CpB	Povprečje	CpB	Povprečje	CpB
Velikost čistopisa/kriptograma: 16 bajtov								
ECB	72,859	4,554	72,0	4,5	100,0	6,25	96,0	6,0
CBC	131,992	8,25	122,754	7,672	119,693	7,481	124,943	7,809
CTR	118,333	7,396	115,373	7,211	207,549	12,972	198,459	12,404
GCM	490,067	30,629	494,518	30,907	1102,278	68,892	1196,417	74,776
CCM	307,705	19,232	289,128	18,07	1140,488	71,28	1233,28	77,08
Velikost čistopisa/kriptograma: 32 bajtov								
ECB	92,0	2,875	88,939	2,779	116,905	3,653	126,943	3,967

Nadaljevanje na naslednji strani

Tabela 6.10 – *Nadaljevanje s prejšnje strani*

n = 950	Nettle				Crypto++			
	Šifriranje		Dešifriranje		Šifriranje		Dešifriranje	
Način	Povprečje	CpB	Povprečje	CpB	Povprečje	CpB	Povprečje	CpB
CBC	213,886	6,684	141,874	4,434	191,402	5,981	174,876	5,465
CTR	135,078	4,221	135,183	4,224	253,806	7,931	231,377	7,231
GCM	627,617	19,613	697,819	21,807	1140,749	35,648	1239,347	38,73
CCM	445,12	13,91	408,636	12,77	1292,333	40,385	1329,903	41,559
Velikost čistopisa/kriptograma: 512 bajtov								
ECB	692,0	1,352	659,495	1,288	419,331	0,819	416,535	0,814
CBC	2944,0	5,75	756,156	1,477	2529,92	4,941	573,208	1,12
CTR	1268,463	2,477	1267,924	2,476	582,417	1,138	570,227	1,114
GCM	6534,029	12,762	6604,472	12,899	1945,684	3,8	2039,979	3,984
CCM	4136,286	8,079	4105,571	8,019	3938,522	7,692	4029,629	7,87
Velikost čistopisa/kriptograma: 10 000 000 bajtov								
ECB	12859427,486	1,286	12382681,368	1,238	7317001,326	0,732	7320588,392	0,732
CBC	57199503,217	5,72	16620002,661	1,662	49048187,044	4,905	7943470,387	0,794
CTR	30771709,486	3,077	30764926,686	3,076	8214138,686	0,821	8165429,267	0,817
GCM	124293212,825	12,429	124262314,114	12,426	14845469,613	1,485	15056748,585	1,506
CCM	83580838,417	8,358	83500824,497	8,35	57217913,213	5,722	57160512,236	5,716

Tabela 6.11 vsebuje hitrosti šifriranj s strojno pospešenimi načini šifriranja z algoritmom AES, razvrščene padajoče, za različne velikosti čistopisov in različne knjižnice. Osnovni načini šifriranja (ECB, CBC in CTR) pri čistopisih velikosti 16 B in 32 B zmeraj zasedajo prva 3 mesta, pri čemer je ECB zmeraj najhitrejši, načina CBC in CTR pa sta glede na knjižnico na 2. ali 3. mestu. Pri velikostih čistopisa 512 B in 10 MB je moč opaziti pohitritev načina CTR, ki se v knjižnici Libgcrypt pri čistopisu 512 B in 10 B tako kot tudi v knjižnici OpenSSL pri 10 MB prebije na prvo mesto in tako prehiti način ECB. Zaostanek CBC se v splošnem v primerjavi z ECB in CTR povečuje.

Načini AEAD (GCM, OCB in CCM) so v splošnem počasnejši, saj morajo dodatno še izračunati overitveno značko čistopisa, zato se posledično pri velikostih čistopisa 16 B in 32 B uvrstijo na zadnja mesta. Njihova medsebojna razvrstitev je pri omenjenih velikostih drugačna glede na knjižnico. Pri čistopisih 512 B in 10 MB se njihova hitrost šifriranja znatno poveča, njihov vrstni red pa je v knjižnicah OpenSSL, Libgcrypt in Crypto++ sledeč: OCB je najhitrejši, sledi GCM, CCM pa je najpočasnejši. V knjižnici Nettle je CCM zmeraj hitrejši od GCM-načina šifriranja. Način XTS, ki se ne klasificira med načine AEAD, se v knjižnici OpenSSL zmeraj umesti nekje vmes med osnovnimi in AEAD-načini šifriranja, v knjižnici Libgcrypt pa najprej vmes, pri velikostih 512 B in 10 MB pa na zadnje mesto.

V splošnem je šifriranje majhnih velikosti čistopisov mnogo počasnejše v primerjavi z velikim. Največji preskok je moč opaziti med 16 B in 32 B. Narastek hitrosti je v povprečju 1,8-kraten (za način ECB), z nadaljnjim večanjem velikosti čistopisa pa upada in se postopoma ustali. Način ECB je za velikost čistopisov 16 B in 32 B najhitrejši, pri velikosti 512 B in 10 MB pa mu, glede na knjižnico, konkurira način CTR, ki zavzema prvo ali drugo mesto. V primeru AEAD-načinov je za večje čistopise najhitrejši OCB, ki je zelo blizu načinu CTR. Hitrost šifriranja v vseh primerih narašča logaritmično.

Tabela 6.11: Razvrstitev hitrosti strojno pospešenega šifriranja za različne načine šifriranja z algoritmom AES

	OpenSSL		Nettle		Libgcrypt		Crypto++	
#	Nač.	CpB	Nač.	CpB	Nač.	CpB	Nač.	CpB
Velikost čistopisa/kriptograma: 16 bajtov								
1	ECB	6,681	ECB	4,554	ECB	8,553	ECB	6,25
2	CBC	+0,419	CTR	+2,842	CTR	+1,064	CBC	+1,231
3	CTR	+2,801	CBC	+3,696	CBC	+1,184	CTR	+6,722
4	XTS	+6,242	CCM	+14,678	XTS	+5,537	GCM	+62,642
5	OCB	+15,145	GCM	+26,075	CCM	+10,237	CCM	+65,03
6	GCM	+15,265	/	/	OCB	+13,885	/	/
7	CCM	+24,436	/	/	GCM	+15,913	/	/
Velikost čistopisa/kriptograma: 32 bajtov								
1	ECB	3,456	ECB	2,875	ECB	4,75	ECB	3,653
2	CTR	+1,924	CTR	+1,346	CTR	+0,415	CBC	+2,328
3	CBC	+2,675	CBC	+3,809	CBC	+2,446	CTR	+4,278
4	XTS	+3,086	CCM	+11,035	XTS	+5,409	GCM	+31,995
5	OCB	+7,564	GCM	+16,738	OCB	+6,901	CCM	+36,732
6	GCM	+9,385	/	/	CCM	+7,609	/	/
7	CCM	+13,266	/	/	GCM	+8,499	/	/
Velikost čistopisa/kriptograma: 512 bajtov								
1	ECB	0,753	ECB	1,352	CTR	0,935	ECB	0,819
2	CTR	+0,091	CTR	+1,125	OCB	+0,445	CTR	+0,319
3	XTS	+0,309	CBC	+4,398	ECB	+0,737	GCM	+2,981
4	OCB	+0,815	CCM	+6,727	GCM	+1,272	CBC	+4,122
5	GCM	+1,086	GCM	+11,41	CBC	+3,737	CCM	+6,873
6	CBC	+3,766	/	/	CCM	+4,667	/	/
7	CCM	+4,453	/	/	XTS	+5,502	/	/
Velikost čistopisa/kriptograma: 10 000 000 bajtov								
1	CTR	0,662	ECB	1,286	CTR	0,744	ECB	0,732
2	ECB	+0,015	CTR	+1,791	OCB	+0,055	CTR	+0,089
3	XTS	+0,055	CBC	+4,434	ECB	+0,79	GCM	+0,753
4	OCB	+0,056	CCM	+7,072	GCM	+0,877	CBC	+4,173
5	GCM	+0,367	GCM	+11,143	CBC	+3,784	CCM	+4,99
6	CBC	+3,802	/	/	CCM	+4,534	/	/
7	CCM	+3,822	/	/	XTS	+5,694	/	/

Tabela 6.12 vsebuje hitrosti dešifriranj s strojno pospešenimi načini šifriranja z algoritmom AES, razvrščene padajoče, za različne velikosti kriptogramov in različne knjižnice. Osnovni načini šifriranja (ECB, CBC in CTR) zavzemajo pri vseh velikostih kriptogramov prva 3 mesta, razen v primeru 512 B in 10 MB v knjižnici Libgcrypt. V knjižnicah OpenSSL in Libgcrypt način CBC pri 10 MB zavzame prvo mesto, razen v

knjižnicah Nettle in Crypto++, kjer način ECB ostane na prvem mestu. V primerjavi dešifriranja s šifriranjem je CBC ostalim osnovnim načinom konkurenčen. Način CTR glede na knjižnico v vseh primerih zavzema drugo ali tretje mesto.

Načini AEAD zmeraj zavzemajo zadnja mesta. Njihov vrstni red, razen v knjižnici Nettle, je sledeč: OCB je zmeraj najhitrejši, GCM drugi in CCM najpočasnejši. V knjižnici Nettle je način CCM tako kot pri šifriranju hitrejši od načina GCM. Način XTS je v knjižnici OpenSSL zmeraj uvrščen med osnovnimi in AEAD-načini, v knjižnici Libgcrypt pa je najprej vmes, nato pa na zadnjem mestu.

V splošnem je dešifriranje majhnih velikosti čistopisov tako kot pri šifriranju mnogo počasneje kot dešifriranje velikih. Največji preskok se zgodi med 16 B in 32 B, v povprečju 1,7-kratni narastek hitrosti (za način ECB), ki z večanjem čistopisa pada in se postopoma ustali. Način ECB je v knjižnicah Nettle in Crypto++ zmeraj najhitrejši. V knjižnici OpenSSL ga pri velikosti kriptograma 10 MB prehitita načina CBC in CTR, v knjižnici Libgcrypt pa pri velikosti 512 B in 10 MB. Način CBC je v primeru dolžine kriptogramov 512 B in 10 MB hitrejši od načina CTR, razen v knjižnici Crypto++ pri 512 B.

Tabela 6.12: Razvrstitev hitrosti strojno pospešenega dešifriranja za različne načine šifriranja z algoritmom AES

	OpenSSL		Nettle		Libgcrypt		Crypto++	
#	Nač.	CpB	Nač.	CpB	Nač.	CpB	Nač.	CpB
Velikost čistopisa/kriptograma: 16 bajtov								
1	ECB	7,642	ECB	4,5	ECB	8,371	ECB	6,0
2	CBC	+0,27	CTR	+2,711	CTR	+0,879	CBC	+1,809
3	CTR	+2,738	CBC	+3,172	CBC	+1,129	CTR	+6,404
4	XTS	+6,346	CCM	+13,57	XTS	+5,588	GCM	+68,776
5	GCM	+15,204	GCM	+26,407	CCM	+13,14	CCM	+71,08
6	OCB	+15,391	/	/	OCB	+15,493	/	/
7	CCM	+28,96	/	/	GCM	+17,815	/	/
Velikost čistopisa/kriptograma: 32 bajtov								
1	ECB	3,937	ECB	2,779	ECB	4,684	ECB	3,967
2	CBC	+0,554	CTR	+1,445	CTR	+0,258	CBC	+1,498
3	CTR	+1,895	CBC	+1,655	CBC	+0,342	CTR	+3,264
4	XTS	+3,204	CCM	+9,991	XTS	+5,366	GCM	+34,763
5	OCB	+7,583	GCM	+19,028	OCB	+7,586	CCM	+37,592
6	GCM	+8,416	/	/	CCM	+8,356	/	/

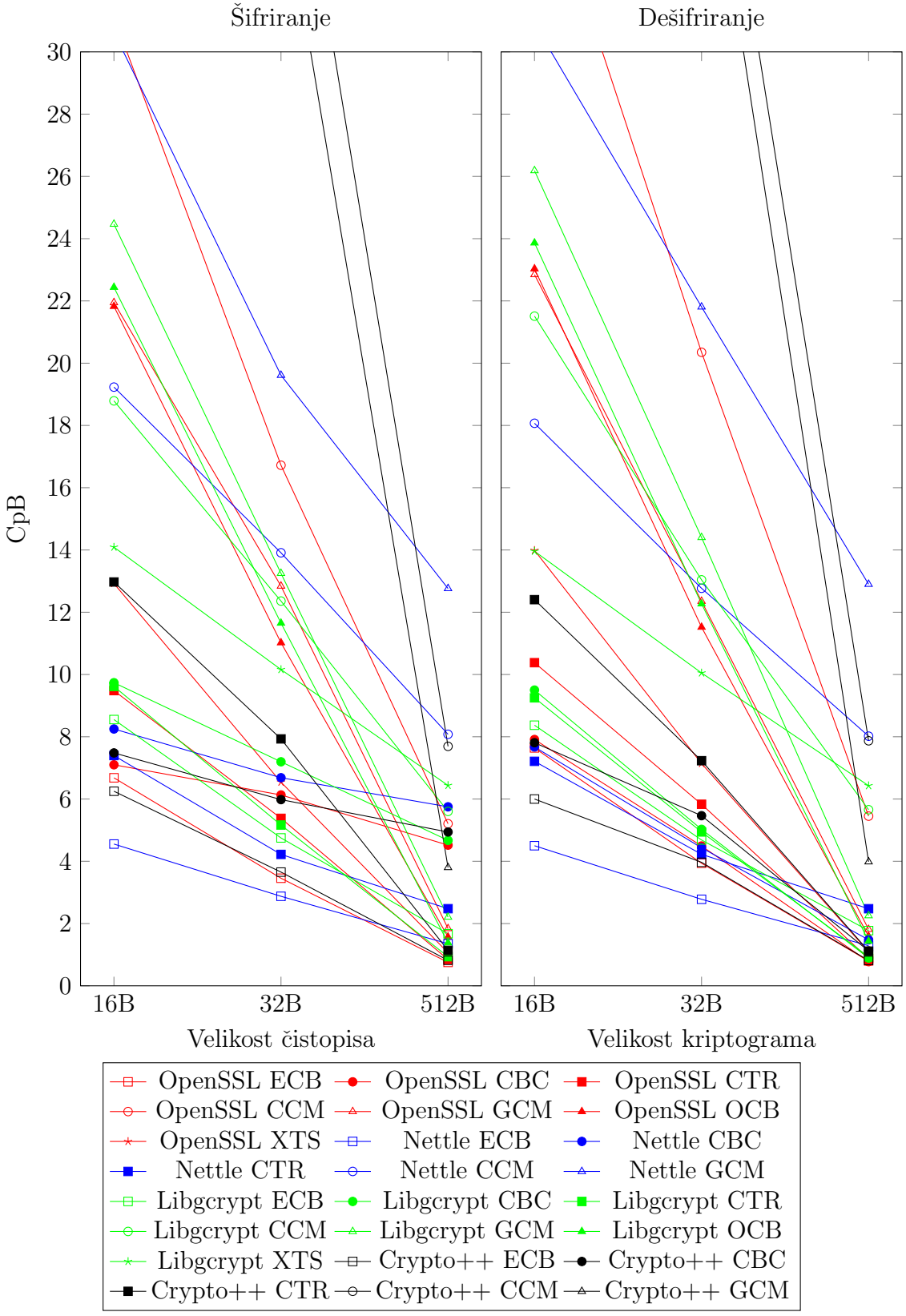
Nadaljevanje na naslednji strani

Tabela 6.12 – *Nadaljevanje s prejšnje strani*

	OpenSSL		Nettle		Libgcrypt		Crypto++	
#	Nač.	CpB	Nač.	CpB	Nač.	CpB	Nač.	CpB
7	CCM	+16,413	/	/	GCM	+9,718	/	/
Velikost čistopisa/kriptograma: 512 bajtov								
1	CBC	0,776	ECB	1,288	CBC	0,891	ECB	0,814
2	ECB	+0,023	CBC	+0,189	CTR	+0,042	CTR	+0,3
3	CTR	+0,097	CTR	+1,188	OCB	+0,543	CBC	+0,306
4	XTS	+0,315	CCM	+6,731	ECB	+0,882	GCM	+3,17
5	OCB	+0,815	GCM	+11,611	GCM	+1,362	CCM	+7,056
6	GCM	+1,038	/	/	CCM	+4,765	/	/
7	CCM	+4,674	/	/	XTS	+5,537	/	/
Velikost čistopisa/kriptograma: 10 000 000 bajtov								
1	CBC	0,652	ECB	1,238	CBC	0,719	ECB	0,732
2	CTR	+0,009	CBC	+0,424	CTR	+0,026	CBC	+0,062
3	ECB	+0,025	CTR	+1,838	OCB	+0,102	CTR	+0,085
4	XTS	+0,064	CCM	+7,112	ECB	+0,858	GCM	+0,774
5	OCB	+0,075	GCM	+11,188	GCM	+0,898	CCM	+4,984
6	GCM	+0,377	/	/	CCM	+4,556	/	/
7	CCM	+3,829	/	/	XTS	+5,726	/	/

Iz primerjav hitrosti šifriranj in dešifriranj v tabelah 6.11 in 6.12 smo ugotovili, da je izmed osnovnih načinov v primeru šifriranja najhitrejši način CTR, v primeru šifriranja z AEAD pa način OCB (če primerjamo najboljše hitrosti). V primeru dešifriranja je med osnovnimi načini najhitrejši CBC, v primeru AEAD pa OCB (način ECB ignoriramo, saj predstavlja samo uporabo šifrirne oz. dešifrirne funkcije algoritma). Način XTS, ki ne spada v nobeno izmed omenjenih kategorij, se v knjižnici OpenSSL uvršča nekje vmes med osnovne in AEAD-načine, v knjižnici Libgcrypt pa na zadnje mesto.

Graf 6.5 prikazuje primerjavo hitrosti pospešenega šifriranja in dešifriranja z algoritmom AES za različne knjižnice. Z večanjem velikosti čistopisa oz. kriptograma opazimo znaten trend naraščanja hitrosti, do katerega prihaja zaradi načinov šifriranja, strojno pospešenega izvajanja in vpliva inicializacijskih časov knjižnic. AEAD-načini šifriranja so v splošnem počasnejši od osnovnih. Imajo daljše inicializacijske čase in manjše končne hitrosti, saj morajo dodatno še računati overitvene značke in preverjati njihovo skladnost. Strmina naraščanja hitrosti se z večanjem velikosti čistopisa oz. kriptograma ustali.



Graf 6.5: Primerjava hitrosti različnih strojno pospešenih načinov šifriranja z algoritmom AES

7 Sklep

Namen magistrskega dela je bil primerjati hitrosti šifer in načinov šifriranja z uporabo raziskovalno-znanstvenih metod. Kot osrednji algoritem smo uporabili algoritem AES, saj je standardiziran in trenutno najbolj uporabljan. Hitrost njegovega izvajanja smo najprej primerjali pospešeno z nepospešenim. Nato smo nepospešeno izvajanje primerjali z ostalimi nepospešenimi algoritmi. Na koncu smo primerjali še hitrosti pospešenega izvajanja med različnimi načini šifriranja.

Implementacij algoritmov in načinov šifriranja se nismo lotili sami, saj implementacije algoritmov že obstajajo v različnih kriptografskih knjižnicah, ki so testirane in uveljavljene. Da bi bila primerjava algoritmov čim natančnejša in čim bolj neodvisna od ene same knjižnice, smo enake primerjave izvedli v več knjižnicah.

Ugotovili smo, da je nepospešeno izvajanje algoritma AES še vedno hitrejše kot ostale primerljive šifre. Pokazali smo, da je pospešeno šifriranje z algoritmom AES hitrejše kot nepospešeno (pri 16 B je v povprečju 2,5-krat in pri 10 MB 14,9-krat). Prav tako način šifriranja vpliva na hitrost izvajanja algoritma AES, tako da algoritem upočasni. V nekaterih primerih je prišlo do pohitritve algoritma zaradi različnih stopenj optimizacij implementacij. V primeru šifriranja se je za najhitrejši način izkazal način CTR, med AEAD-načini pa OCB. V primeru dešifriranja je najhitrejši način CBC, najhitrejši AEAD-način pa prav tako OCB.

Kode knjižnic ni bilo mogoče neposredno pognati v jedrnem modulu, v katerem smo imeli nadzor nad procesorjem in smo lahko izklopili 'preemption' in prekinitve. Zato smo se morali poslužiti tehnike, s katero smo lahko teste za preizkušanje zmogljivosti pognali iz jedrnega prostora preko terminala v uporabniškem prostoru. Kljub temu da smo 'preemption' in prekinitve izklopili, so vzorci vsebovali repe, zato smo vsem vzorcem morali odstraniti 5 % največjih meritev. Ker smo uporabljali implementa-

cije algoritmov v različnih knjižnicah, ki so različno optimizirane, so se meritve med različnimi knjižnicami razlikovale.

V nadaljevanju dela bi bilo treba testirati optimizirane implementacije algoritmov, brez uporabe knjižnic, in narediti sistematsko primerjavo. Dodatno bi lahko vključili tudi tokovne simetrične šifre in primerjali njihovo hitrost s pospešenim oz. nepospešenim izvajanjem algoritma AES z različnimi načini šifriranja in z ostalimi algoritmi. Prav tako bi bilo treba raziskati stopnjo paralelizacije, ki jo algoritmi in načini teoretično ponujajo. Za še obsežnejšo primerjavo bi bilo primerno kodo prevajati z različnimi prevajalniki in teste izvajati na različnih procesorjih.

Literatura

[1] Internet of Things forecast – Ericsson Mobility Report. Dostopno: https://www.ericsson.com/en/mobility-report/internet-of-things-forecast. [Dostopano: 15. julij 2019].

[2] Invoking user-space applications from the kernel. Dostopno: https://developer.ibm.com/articles/l-user-space-apps, februar 2010. [Dostopano: 9. september 2019].

[3] Mann-Whitney U Test in SPSS Statistics | Setup, Procedure & Interpretation | Laerd Statistics. Dostopno: https://statistics.laerd.com/spss-tutorials/mann-whitney-u-test-using-spss-statistics.php, september 2019. [Dostopano: 9. september 2019].

[4] Kruskal-Wallis H Test in SPSS Statistics | Procedure, output and interpretation of the output using a relevant example. Dostopno: https://statistics.laerd.com/spss-tutorials/kruskal-wallis-h-test-using-spss-statistics.php. [Dostopano: 9. september 2019].

[5] News Release 050710. Dostopno: https://www.ntt.co.jp/news/news05e/0507/050720.html, julij 2005. [Dostopano: 20. september 2019].

[6] Kernel Space Definition. Dostopno: http://www.linfo.org/kernel_space.html, februar 2005. [Dostopano: 24. september 2019].

[7] User Space Definition. Dostopno: http://www.linfo.org/user_space.html, februar 2005. [Dostopano: 24. september 2019].

[8] SciPy.org. Dostopno: https://www.scipy.org. [Dostopano: 5. oktober 2019].

[9] How to Calculate Nonparametric Statistical Hypothesis Tests in Python. Dostopno: https://machinelearningmastery.com/nonparametric-statistical-significance-tests-in-python, maj 2018. [Dostopano: 5. oktober 2019].

[10] 2019 Cyberthreat Defense Report | Illusive Networks. Dostopno: https://go.illusivenetworks.com/2019-cyberthreat-defense-report. [Dostopano: 15. julij 2019].

[11] Block Ciphers and Modes of Operation | CommonLounge. Dostopno: https://www.commonlounge.com/discussion/6747358d828a45c99f61f4c09ff2f371, november 2019. [Dostopano: 1. november 2019].

[12] Crypto-Gram: October 15, 1998 - Schneier on Security. Dostopno: https://www.schneier.com/crypto-gram/archives/1998/1015.html#cipherdesign, julij 2019. [Dostopano: 1. november 2019].

[13] What is Pre-Emption? - Definition from Techopedia. Dostopno: https://www.techopedia.com/definition/27086/pre-emption. [Dostopano: 16. julij 2019].

[14] Speedtest and Comparsion of Open-Source Cryptography Libraries and Compiler Flags - panthema.net. Dostopno: https://panthema.net/2008/0714-cryptography-speedtest-comparison, julij 2008. [Dostopano: 17. julij 2019].

[15] Discrete Logarithm Problem. Dostopno: https://www.doc.ic.ac.uk/~mrh/330tutor/ch06s02.html. [Dostopano: 18. julij 2019].

[16] What is a Feistel Network? - Definition from Techopedia. Dostopno: https://www.techopedia.com/definition/27121/feistel-network. [Dostopano: 21. julij 2019].

[17] Intel® Advanced Encryption Standard Instructions (AES-NI). Dostopno: https://software.intel.com/en-us/articles/intel-advanced-encryption-standard-instructions-aes-ni, februar 2012. [Dostopano: 27. julij 2019].

[18] OCB - An Authenticated-Encryption Scheme - Background - Rogaway. Dostopno: https://web.cs.ucdavis.edu/~rogaway/ocb/ocb-faq.htm. [Dostopano: 8. avgust 2019].

[19] Kazumaro Aoki, Tetsuya Ichikawa, Masayuki Kanda, Mitsuru Matsui, Shiho Moriai, Junko Nakajima, and Toshio Tokita. Specification of camellia-a 128-bit block cipher. *Specification Version*, 2, 2000.

[20] Elaine Barker. Recommendation for Key Management, Part 1: General. *CSRC | NIST*, januar 2016.

[21] Gilbert Baumslag, Benjamin Fine, Martin Kreuzer, and Gerhard Rosenberger. *A Course in Mathematical Cryptography*. De Gruyter, junij 2015.

[22] Eli Biham, Ross Anderson, and Lars Knudsen. Serpent: A new block cipher proposal. In *International Workshop on Fast Software Encryption*, pages 222–238. Springer, 1998.

[23] Daniel Brecht. Tales from the Crypt: Hardware vs Software. *Infosecurity Magazine*, junij 2015.

[24] Joan Daemen and Vincent Rijmen. The Block Cipher Rijndael. *Lect. Notes Comput. Sci.*, 1820:277–284, januar 1998.

[25] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: Methods and Techniques. *CSRC | NIST*, december 2001.

[26] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. *CSRC | NIST*, november 2007.

[27] Morris Dworkin. Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality. *CSRC | NIST*, julij 2007.

[28] Damien Giry. Keylength - NIST Report on Cryptographic Key Length and Cryptoperiod (2016). Dostopno: https://www.keylength.com/en/4, junij 2018. [Dostopano: 19. julij 2019].

[29] IEEE. Ieee standard for cryptographic protection of data on block-oriented storage devices. *IEEE Std 1619-2018 (Revision of IEEE Std 1619-2007)*, pages 1–41, januar 2019.

[30] Omar Farook Mohammad, Mohd Shafry Mohd Rahim, Subhi R. M. Zeebaree, and Falah Y. H. Ahmed. A Survey and Analysis of the Image Encryption Methods. *International Journal of Applied Engineering Research*, 12(23):13265–13280, december 2017.

[31] National Institute of Standards and Technology. Advanced Encryption Standard (AES). *CSRC | NIST*, november 2001.

[32] Gabriele Paoloni. How to benchmark code execution times on intel ia-32 and ia-64 instruction set architectures. *Intel Corporation*, 123, 2010.

[33] Phillip Rogaway and Ted Krovetz. The Software Performance of Authenticated-Encryption Modes. *Fast Software Encryption 2011 (FSE 2011)*, marec 2011.

[34] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. Twofish: A 128-bit block cipher. *NIST AES Proposal*, 15(1):23–91, 1998.

[35] National Institute Of Standards and Technology. Data Encryption Standard (DES). *CSRC | NIST*, oktober 1999.

A Rezultati statističnih testov primerjave šifrirnih algoritmov

Tabela A.1: U- in p-vrednosti statističnih Mann-Whitneyjevih U-testov strojno nepospešenih algoritmov v načinu ECB za knjižnico OpenSSL

n = 950		Šifriranje		Dešifriranje	
X	Y	U	p	U	p
Velikost čistopisa/kriptograma: 16 bajtov					
AES	Camellia	0,0	0,0	0,0	0,0
AES	DES	0,0	0,0	0,0	0,0
AES	3DES	0,0	0,0	0,0	0,0
Camellia	DES	0,0	0,0	0,0	0,0
Camellia	3DES	0,0	0,0	0,0	0,0
DES	3DES	0,0	0,0	0,0	0,0
Velikost čistopisa/kriptograma: 32 bajtov					
AES	Camellia	0,0	0,0	0,0	0,0
AES	DES	0,0	0,0	0,0	0,0
AES	3DES	0,0	0,0	0,0	0,0
Camellia	DES	0,0	0,0	0,0	0,0
Camellia	3DES	0,0	0,0	0,0	0,0
DES	3DES	0,0	0,0	0,0	0,0
Velikost čistopisa/kriptograma: 512 bajtov					
AES	Camellia	0,0	0,0	0,0	0,0
AES	DES	0,0	0,0	0,0	0,0
AES	3DES	0,0	0,0	0,0	0,0
Camellia	DES	0,0	0,0	0,0	0,0
Camellia	3DES	0,0	0,0	0,0	0,0
DES	3DES	0,0	0,0	0,0	0,0
Velikost čistopisa/kriptograma: 10 000 000 bajtov					
AES	Camellia	0,0	0,0	0,0	0,0
AES	DES	0,0	0,0	0,0	0,0
AES	3DES	0,0	0,0	0,0	0,0
Camellia	DES	0,0	0,0	0,0	0,0

Nadaljevanje na naslednji strani

Tabela A.1 – *Nadaljevanje s prejšnje strani*

n = 950		Šifriranje		Dešifriranje	
X	Y	U	p	U	p
Camellia	3DES	0,0	0,0	0,0	0,0
DES	3DES	0,0	0,0	0,0	0,0

Tabela A.2: U- in p-vrednosti statističnih Mann-Whitneyjevih U-testov strojno nepospešenih algoritmov v načinu ECB za knjižnico Nettle

n = 950		Šifriranje		Dešifriranje	
X	Y	U	p	U	p
Velikost čistopisa/kriptograma: 16 bajtov					
AES	Camellia	0,0	0,0	0,0	0,0
AES	DES	0,0	0,0	0,0	0,0
AES	3DES	0,0	0,0	0,0	0,0
AES	Serpent	0,0	0,0	0,0	0,0
AES	TwoFish	0,0	0,0	0,0	0,0
Camellia	DES	0,0	0,0	0,0	0,0
Camellia	3DES	0,0	0,0	0,0	0,0
Camellia	Serpent	0,0	0,0	0,0	0,0
Camellia	TwoFish	0,0	0,0	902500,0	0,0
DES	3DES	0,0	0,0	0,0	0,0
DES	Serpent	902500,0	0,0	902500,0	0,0
DES	TwoFish	902500,0	0,0	902500,0	0,0
3DES	Serpent	902500,0	0,0	902500,0	0,0
3DES	TwoFish	902500,0	0,0	902500,0	0,0
Serpent	TwoFish	902500,0	0,0	902500,0	0,0
Velikost čistopisa/kriptograma: 32 bajtov					
AES	Camellia	0,0	0,0	0,0	0,0
AES	DES	0,0	0,0	0,0	0,0
AES	3DES	0,0	0,0	0,0	0,0
AES	Serpent	0,0	0,0	0,0	0,0
AES	TwoFish	0,0	0,0	0,0	0,0
Camellia	DES	0,0	0,0	0,0	0,0
Camellia	3DES	0,0	0,0	0,0	0,0
Camellia	Serpent	0,0	0,0	0,0	0,0
Camellia	TwoFish	4680,0	0,0	902500,0	0,0
DES	3DES	0,0	0,0	0,0	0,0
DES	Serpent	902500,0	0,0	902500,0	0,0

Nadaljevanje na naslednji strani

Tabela A.2 – *Nadaljevanje s prejšnje strani*

n = 950		Šifriranje		Dešifriranje	
X	Y	U	p	U	p
DES	TwoFish	902500,0	0,0	902500,0	0,0
3DES	Serpent	902500,0	0,0	902500,0	0,0
3DES	TwoFish	902500,0	0,0	902500,0	0,0
Serpent	TwoFish	902500,0	0,0	902500,0	0,0
Velikost čistopisa/kriptograma: 512 bajtov					
AES	Camellia	0,0	0,0	0,0	0,0
AES	DES	0,0	0,0	0,0	0,0
AES	3DES	0,0	0,0	0,0	0,0
AES	Serpent	0,0	0,0	0,0	0,0
AES	TwoFish	0,0	0,0	0,0	0,0
Camellia	DES	0,0	0,0	0,0	0,0
Camellia	3DES	0,0	0,0	0,0	0,0
Camellia	Serpent	902500,0	0,0	902500,0	0,0
Camellia	TwoFish	0,0	0,0	902500,0	0,0
DES	3DES	0,0	0,0	0,0	0,0
DES	Serpent	902500,0	0,0	902500,0	0,0
DES	TwoFish	902500,0	0,0	902500,0	0,0
3DES	Serpent	902500,0	0,0	902500,0	0,0
3DES	TwoFish	902500,0	0,0	902500,0	0,0
Serpent	TwoFish	0,0	0,0	0,0	0,0
Velikost čistopisa/kriptograma: 10 000 000 bajtov					
AES	Camellia	0,0	0,0	0,0	0,0
AES	DES	0,0	0,0	0,0	0,0
AES	3DES	0,0	0,0	0,0	0,0
AES	Serpent	0,0	0,0	0,0	0,0
AES	TwoFish	0,0	0,0	0,0	0,0
Camellia	DES	0,0	0,0	0,0	0,0
Camellia	3DES	0,0	0,0	0,0	0,0
Camellia	Serpent	902500,0	0,0	902500,0	0,0
Camellia	TwoFish	0,0	0,0	902500,0	0,0
DES	3DES	0,0	0,0	0,0	0,0
DES	Serpent	902500,0	0,0	902500,0	0,0
DES	TwoFish	902500,0	0,0	902500,0	0,0
3DES	Serpent	902500,0	0,0	902500,0	0,0
3DES	TwoFish	902500,0	0,0	902500,0	0,0
Serpent	TwoFish	0,0	0,0	0,0	0,0

Tabela A.3: U- in p-vrednosti statističnih Mann-Whitneyjevih U-testov strojno nepospešenih algoritmov v načinu ECB za knjižnico Libgcrypt

n = 950		Šifriranje		Dešifriranje	
X	Y	U	p	U	p
Velikost čistopisa/kriptograma: 16 bajtov					
AES	Camellia	0,0	0,0	0,0	0,0
AES	DES	0,0	0,0	0,0	0,0
AES	3DES	0,0	0,0	0,0	0,0
AES	Serpent	0,0	0,0	0,0	0,0
AES	TwoFish	0,0	0,0	0,0	0,0
Camellia	DES	0,0	0,0	0,0	0,0
Camellia	3DES	0,0	0,0	0,0	0,0
Camellia	Serpent	0,0	0,0	0,0	0,0
Camellia	TwoFish	902500,0	0,0	902500,0	0,0
DES	3DES	0,0	0,0	0,0	0,0
DES	Serpent	902500,0	0,0	902500,0	0,0
DES	TwoFish	902500,0	0,0	902500,0	0,0
3DES	Serpent	902500,0	0,0	902500,0	0,0
3DES	TwoFish	902500,0	0,0	902500,0	0,0
Serpent	TwoFish	902500,0	0,0	902500,0	0,0
Velikost čistopisa/kriptograma: 32 bajtov					
AES	Camellia	0,0	0,0	0,0	0,0
AES	DES	0,0	0,0	0,0	0,0
AES	3DES	0,0	0,0	0,0	0,0
AES	Serpent	0,0	0,0	0,0	0,0
AES	TwoFish	0,0	0,0	0,0	0,0
Camellia	DES	0,0	0,0	0,0	0,0
Camellia	3DES	0,0	0,0	0,0	0,0
Camellia	Serpent	0,0	0,0	0,0	0,0
Camellia	TwoFish	902500,0	0,0	902500,0	0,0
DES	3DES	0,0	0,0	0,0	0,0
DES	Serpent	902500,0	0,0	902500,0	0,0
DES	TwoFish	902500,0	0,0	902500,0	0,0
3DES	Serpent	902500,0	0,0	902500,0	0,0
3DES	TwoFish	902500,0	0,0	902500,0	0,0
Serpent	TwoFish	902500,0	0,0	902500,0	0,0
Velikost čistopisa/kriptograma: 512 bajtov					
AES	Camellia	0,0	0,0	0,0	0,0
AES	DES	0,0	0,0	0,0	0,0
AES	3DES	0,0	0,0	0,0	0,0
AES	Serpent	0,0	0,0	0,0	0,0

Nadaljevanje na naslednji strani

Tabela A.3 – *Nadaljevanje s prejšnje strani*

n = 950		Šifriranje		Dešifriranje	
X	Y	U	p	U	p
AES	TwoFish	0,0	0,0	0,0	0,0
Camellia	DES	0,0	0,0	0,0	0,0
Camellia	3DES	0,0	0,0	0,0	0,0
Camellia	Serpent	0,0	0,0	0,0	0,0
Camellia	TwoFish	902500,0	0,0	902500,0	0,0
DES	3DES	0,0	0,0	0,0	0,0
DES	Serpent	902500,0	0,0	902500,0	0,0
DES	TwoFish	902500,0	0,0	902500,0	0,0
3DES	Serpent	902500,0	0,0	902500,0	0,0
3DES	TwoFish	902500,0	0,0	902500,0	0,0
Serpent	TwoFish	902500,0	0,0	902500,0	0,0
Velikost čistopisa/kriptograma: 10 000 000 bajtov					
AES	Camellia	0,0	0,0	0,0	0,0
AES	DES	0,0	0,0	0,0	0,0
AES	3DES	0,0	0,0	0,0	0,0
AES	Serpent	0,0	0,0	0,0	0,0
AES	TwoFish	0,0	0,0	0,0	0,0
Camellia	DES	0,0	0,0	0,0	0,0
Camellia	3DES	0,0	0,0	0,0	0,0
Camellia	Serpent	0,0	0,0	0,0	0,0
Camellia	TwoFish	902500,0	0,0	902500,0	0,0
DES	3DES	0,0	0,0	0,0	0,0
DES	Serpent	902500,0	0,0	902500,0	0,0
DES	TwoFish	902500,0	0,0	902500,0	0,0
3DES	Serpent	902500,0	0,0	902500,0	0,0
3DES	TwoFish	902500,0	0,0	902500,0	0,0
Serpent	TwoFish	902500,0	0,0	902500,0	0,0

Tabela A.4: U- in p-vrednosti statističnih Mann-Whitneyjevih U-testov strojno nepospešenih algoritmov v načinu ECB za knjižnico Crypto++

n = 950		Šifriranje		Dešifriranje	
X	Y	U	p	U	p
Velikost čistopisa/kriptograma: 16 bajtov					
AES	Camellia	0,0	0,0	0,0	0,0
AES	DES	0,0	0,0	0,0	0,0

Nadaljevanje na naslednji strani

Tabela A.4 – *Nadaljevanje s prejšnje strani*

n = 950		Šifriranje		Dešifriranje	
X	Y	U	p	U	p
AES	3DES	0,0	0,0	0,0	0,0
AES	Serpent	0,0	0,0	0,0	0,0
AES	TwoFish	0,0	0,0	823344,0	8,313881184556997e-232
Camellia	DES	0,0	0,0	0,0	0,0
Camellia	3DES	0,0	0,0	0,0	0,0
Camellia	Serpent	0,0	0,0	0,0	0,0
Camellia	TwoFish	902500,0	0,0	902500,0	0,0
DES	3DES	0,0	0,0	0,0	0,0
DES	Serpent	902500,0	0,0	902500,0	0,0
DES	TwoFish	902500,0	0,0	902500,0	0,0
3DES	Serpent	902500,0	0,0	902500,0	0,0
3DES	TwoFish	902500,0	0,0	902500,0	0,0
Serpent	TwoFish	902500,0	0,0	902500,0	0,0
Velikost čistopisa/kriptograma: 32 bajtov					
AES	Camellia	0,0	0,0	0,0	0,0
AES	DES	0,0	0,0	0,0	0,0
AES	3DES	0,0	0,0	0,0	0,0
AES	Serpent	0,0	0,0	0,0	0,0
AES	TwoFish	0,0	0,0	635364,0	7,423189783352645e-63
Camellia	DES	0,0	0,0	0,0	0,0
Camellia	3DES	0,0	0,0	0,0	0,0
Camellia	Serpent	0,0	0,0	0,0	0,0
Camellia	TwoFish	902500,0	0,0	902500,0	0,0
DES	3DES	0,0	0,0	0,0	0,0
DES	Serpent	902500,0	0,0	902500,0	0,0
DES	TwoFish	902500,0	0,0	902500,0	0,0
3DES	Serpent	902500,0	0,0	902500,0	0,0
3DES	TwoFish	902500,0	0,0	902500,0	0,0
Serpent	TwoFish	902500,0	0,0	902500,0	0,0
Velikost čistopisa/kriptograma: 512 bajtov					
AES	Camellia	0,0	0,0	0,0	0,0
AES	DES	0,0	0,0	0,0	0,0
AES	3DES	0,0	0,0	0,0	0,0
AES	Serpent	0,0	0,0	0,0	0,0
AES	TwoFish	0,0	0,0	902500,0	0,0
Camellia	DES	0,0	0,0	0,0	0,0
Camellia	3DES	0,0	0,0	0,0	0,0
Camellia	Serpent	0,0	0,0	0,0	0,0
Camellia	TwoFish	902500,0	0,0	902500,0	0,0

Nadaljevanje na naslednji strani

Tabela A.4 – *Nadaljevanje s prejšnje strani*

n = 950		Šifriranje		Dešifriranje	
X	Y	U	p	U	p
DES	3DES	0,0	0,0	0,0	0,0
DES	Serpent	902500,0	0,0	902500,0	0,0
DES	TwoFish	902500,0	0,0	902500,0	0,0
3DES	Serpent	902500,0	0,0	902500,0	0,0
3DES	TwoFish	902500,0	0,0	902500,0	0,0
Serpent	TwoFish	902500,0	0,0	902500,0	0,0
Velikost čistopisa/kriptograma: 10 000 000 bajtov					
AES	Camellia	0,0	0,0	0,0	0,0
AES	DES	0,0	0,0	0,0	0,0
AES	3DES	0,0	0,0	0,0	0,0
AES	Serpent	0,0	0,0	0,0	0,0
AES	TwoFish	0,0	0,0	902500,0	0,0
Camellia	DES	0,0	0,0	0,0	0,0
Camellia	3DES	0,0	0,0	0,0	0,0
Camellia	Serpent	0,0	0,0	0,0	0,0
Camellia	TwoFish	902500,0	0,0	902500,0	0,0
DES	3DES	0,0	0,0	0,0	0,0
DES	Serpent	902500,0	0,0	902500,0	0,0
DES	TwoFish	902500,0	0,0	902500,0	0,0
3DES	Serpent	902500,0	0,0	902500,0	0,0
3DES	TwoFish	902500,0	0,0	902500,0	0,0
Serpent	TwoFish	902500,0	0,0	902500,0	0,0

B Rezultati statističnih testov primerjave načinov šifriranja

Tabela B.1: U- in p-vrednosti statističnih Mann-Whitneyjevih U-testov strojno pospešenega algoritma AES z različnimi načini šifriranja za knjižnico OpenSSL

n = 950		Šifriranje		Dešifriranje	
X	Y	U	p	U	p
Velikost čistopisa/kriptograma: 16 bajtov					
ECB	CBC	29694,5	7,04062499817663e-294	143319,0	2,0494909906340163e-163
ECB	CTR	0,0	0,0	0,0	0,0
ECB	CCM	0,0	0,0	0,0	0,0
ECB	GCM	0,0	0,0	0,0	0,0
ECB	OCB	0,0	0,0	0,0	0,0
ECB	XTS	0,0	0,0	0,0	0,0
CBC	CTR	0,0	0,0	0,0	0,0
CBC	CCM	0,0	0,0	0,0	0,0
CBC	GCM	0,0	0,0	0,0	0,0
CBC	OCB	0,0	0,0	0,0	0,0
CBC	XTS	0,0	0,0	0,0	0,0
CTR	CCM	0,0	0,0	0,0	0,0
CTR	GCM	0,0	0,0	0,0	0,0
CTR	OCB	0,0	0,0	0,0	0,0
CTR	XTS	0,0	0,0	0,0	0,0
CCM	GCM	902500,0	0,0	902500,0	0,0
CCM	OCB	902500,0	0,0	902500,0	0,0
CCM	XTS	902500,0	0,0	902500,0	0,0
GCM	OCB	569488,5	6,05890820871376e-26	254986,0	1,88693254084167e-67
GCM	XTS	902500,0	0,0	902500,0	0,0
OCB	XTS	902500,0	0,0	902500,0	0,0
Velikost čistopisa/kriptograma: 32 bajtov					
ECB	CBC	0,0	0,0	10,5	0,0
ECB	CTR	0,0	0,0	0,0	0,0
ECB	CCM	0,0	0,0	0,0	0,0

Nadaljevanje na naslednji strani

Tabela B.1 – *Nadaljevanje s prejšnje strani*

n = 950		Šifriranje		Dešifriranje	
X	Y	U	p	U	p
ECB	GCM	0,0	0,0	0,0	0,0
ECB	OCB	0,0	0,0	0,0	0,0
ECB	XTS	0,0	0,0	0,0	0,0
CBC	CTR	902500,0	0,0	0,0	0,0
CBC	CCM	0,0	0,0	0,0	0,0
CBC	GCM	0,0	0,0	0,0	0,0
CBC	OCB	0,0	0,0	0,0	0,0
CBC	XTS	2472,0	0,0	0,0	0,0
CTR	CCM	0,0	0,0	0,0	0,0
CTR	GCM	0,0	0,0	0,0	0,0
CTR	OCB	0,0	0,0	0,0	0,0
CTR	XTS	0,0	0,0	0,0	0,0
CCM	GCM	902500,0	0,0	902500,0	0,0
CCM	OCB	902500,0	0,0	902500,0	0,0
CCM	XTS	902500,0	0,0	902500,0	0,0
GCM	OCB	902500,0	0,0	902500,0	0,0
GCM	XTS	902500,0	0,0	902500,0	0,0
OCB	XTS	902500,0	0,0	902500,0	0,0
Velikost čistopisa/kriptograma: 512 bajtov					
ECB	CBC	0,0	0,0	821812,0	3,6224091336497774e-221
ECB	CTR	0,0	0,0	0,0	0,0
ECB	CCM	0,0	0,0	0,0	0,0
ECB	GCM	0,0	0,0	0,0	0,0
ECB	OCB	0,0	0,0	0,0	0,0
ECB	XTS	0,0	0,0	0,0	0,0
CBC	CTR	902500,0	0,0	0,0	0,0
CBC	CCM	0,0	0,0	0,0	0,0
CBC	GCM	902500,0	0,0	0,0	0,0
CBC	OCB	902500,0	0,0	0,0	0,0
CBC	XTS	902500,0	0,0	0,0	0,0
CTR	CCM	0,0	0,0	0,0	0,0
CTR	GCM	0,0	0,0	0,0	0,0
CTR	OCB	0,0	0,0	0,0	0,0
CTR	XTS	0,0	0,0	0,0	0,0
CCM	GCM	902500,0	0,0	902500,0	0,0
CCM	OCB	902500,0	0,0	902500,0	0,0
CCM	XTS	902500,0	0,0	902500,0	0,0
GCM	OCB	902500,0	0,0	902500,0	0,0
GCM	XTS	902500,0	0,0	902500,0	0,0

Nadaljevanje na naslednji strani

Tabela B.1 – *Nadaljevanje s prejšnje strani*

n = 950		Šifriranje		Dešifriranje	
X	Y	U	p	U	p
OCB	XTS	902500,0	0,0	902500,0	0,0
Velikost čistopisa/kriptograma: 10000000 bajtov					
ECB	CBC	0,0	0,0	902500,0	0,0
ECB	CTR	902500,0	0,0	902500,0	0,0
ECB	CCM	0,0	0,0	0,0	0,0
ECB	GCM	0,0	0,0	0,0	0,0
ECB	OCB	0,0	0,0	0,0	0,0
ECB	XTS	0,0	0,0	0,0	0,0
CBC	CTR	902500,0	0,0	9257,5	4,1865950142649464e-299
CBC	CCM	3,0	0,0	0,0	0,0
CBC	GCM	902500,0	0,0	0,0	0,0
CBC	OCB	902500,0	0,0	0,0	0,0
CBC	XTS	902500,0	0,0	0,0	0,0
CTR	CCM	0,0	0,0	0,0	0,0
CTR	GCM	0,0	0,0	0,0	0,0
CTR	OCB	0,0	0,0	0,0	0,0
CTR	XTS	0,0	0,0	0,0	0,0
CCM	GCM	902500,0	0,0	902500,0	0,0
CCM	OCB	902500,0	0,0	902500,0	0,0
CCM	XTS	902500,0	0,0	902500,0	0,0
GCM	OCB	902500,0	0,0	902500,0	0,0
GCM	XTS	902500,0	0,0	902500,0	0,0
OCB	XTS	577088,0	6,687889256140992e-26	902215,0	0,0

Tabela B.2: U- in p-vrednosti statističnih Mann-Whitnjevih U-testov strojno pospešenega algoritma AES z različnimi načini šifriranja za knjižnico Nettle

n = 950		Šifriranje		Dešifriranje	
X	Y	U	p	U	p
Velikost čistopisa/kriptograma: 16 bajtov					
ECB	CBC	0,0	0,0	0,0	0,0
ECB	CTR	0,0	0,0	0,0	0,0
ECB	CCM	0,0	0,0	0,0	0,0
ECB	GCM	0,0	0,0	0,0	0,0
CBC	CTR	902500,0	0,0	902500,0	0,0
CBC	CCM	0,0	0,0	0,0	0,0

Nadaljevanje na naslednji strani

Tabela B.2 – *Nadaljevanje s prejšnje strani*

n = 950		Šifriranje		Dešifriranje	
X	Y	U	p	U	p
CBC	GCM	0,0	0,0	0,0	0,0
CTR	CCM	0,0	0,0	0,0	0,0
CTR	GCM	0,0	0,0	0,0	0,0
CCM	GCM	0,0	0,0	0,0	0,0
Velikost čistopisa/kriptograma: 32 bajtov					
ECB	CBC	0,0	0,0	0,0	0,0
ECB	CTR	0,0	0,0	0,0	0,0
ECB	CCM	0,0	0,0	0,0	0,0
ECB	GCM	0,0	0,0	0,0	0,0
CBC	CTR	902500,0	0,0	902500,0	0,0
CBC	CCM	0,0	0,0	0,0	0,0
CBC	GCM	0,0	0,0	0,0	0,0
CTR	CCM	0,0	0,0	0,0	0,0
CTR	GCM	0,0	0,0	0,0	0,0
CCM	GCM	0,0	0,0	0,0	0,0
Velikost čistopisa/kriptograma: 512 bajtov					
ECB	CBC	0,0	0,0	0,0	0,0
ECB	CTR	0,0	0,0	0,0	0,0
ECB	CCM	0,0	0,0	0,0	0,0
ECB	GCM	0,0	0,0	0,0	0,0
CBC	CTR	902500,0	0,0	0,0	0,0
CBC	CCM	0,0	0,0	0,0	0,0
CBC	GCM	0,0	0,0	0,0	0,0
CTR	CCM	0,0	0,0	0,0	0,0
CTR	GCM	0,0	0,0	0,0	0,0
CCM	GCM	0,0	0,0	0,0	0,0
Velikost čistopisa/kriptograma: 10000000 bajtov					
ECB	CBC	0,0	0,0	0,0	0,0
ECB	CTR	0,0	0,0	0,0	0,0
ECB	CCM	0,0	0,0	0,0	0,0
ECB	GCM	0,0	0,0	0,0	0,0
CBC	CTR	902500,0	0,0	0,0	0,0
CBC	CCM	0,0	0,0	0,0	0,0
CBC	GCM	0,0	0,0	0,0	0,0
CTR	CCM	0,0	0,0	0,0	0,0
CTR	GCM	0,0	0,0	0,0	0,0
CCM	GCM	0,0	0,0	0,0	0,0

Tabela B.3: U- in p-vrednosti statističnih Mann-Whitneyjevih U-testov strojno pospešenega algoritma AES z različnimi načini šifriranja za knjižnico Libgcrypt

n = 950		Šifriranje		Dešifriranje	
X	Y	U	p	U	p
Velikost čistopisa/kriptograma: 16 bajtov					
ECB	CBC	0,0	0,0	0,0	0,0
ECB	CTR	0,0	0,0	0,0	0,0
ECB	CCM	0,0	0,0	0,0	0,0
ECB	GCM	0,0	0,0	0,0	0,0
ECB	OCB	0,0	0,0	0,0	0,0
ECB	XTS	0,0	0,0	0,0	0,0
CBC	CTR	616845,0	3,506166971763445e-53	902500,0	0,0
CBC	CCM	0,0	0,0	0,0	0,0
CBC	GCM	0,0	0,0	0,0	0,0
CBC	OCB	0,0	0,0	0,0	0,0
CBC	XTS	0,0	0,0	0,0	0,0
CTR	CCM	0,0	0,0	0,0	0,0
CTR	GCM	0,0	0,0	0,0	0,0
CTR	OCB	0,0	0,0	0,0	0,0
CTR	XTS	0,0	0,0	0,0	0,0
CCM	GCM	0,0	0,0	0,0	0,0
CCM	OCB	0,0	0,0	0,0	0,0
CCM	XTS	902500,0	0,0	902500,0	0,0
GCM	OCB	902500,0	0,0	902500,0	0,0
GCM	XTS	902500,0	0,0	902500,0	0,0
OCB	XTS	902500,0	0,0	902500,0	0,0
Velikost čistopisa/kriptograma: 32 bajtov					
ECB	CBC	0,0	0,0	0,0	0,0
ECB	CTR	0,0	0,0	0,0	0,0
ECB	CCM	0,0	0,0	0,0	0,0
ECB	GCM	0,0	0,0	0,0	0,0
ECB	OCB	0,0	0,0	0,0	0,0
ECB	XTS	0,0	0,0	0,0	0,0
CBC	CTR	902500,0	0,0	683415,0	1,4150445043035178e-101
CBC	CCM	0,0	0,0	0,0	0,0
CBC	GCM	0,0	0,0	0,0	0,0
CBC	OCB	0,0	0,0	0,0	0,0
CBC	XTS	0,0	0,0	0,0	0,0
CTR	CCM	0,0	0,0	0,0	0,0
CTR	GCM	0,0	0,0	0,0	0,0
CTR	OCB	0,0	0,0	0,0	0,0

Nadaljevanje na naslednji strani

Tabela B.3 – *Nadaljevanje s prejšnje strani*

n = 950		Šifriranje		Dešifriranje	
X	Y	U	p	U	p
CTR	XTS	0,0	0,0	0,0	0,0
CCM	GCM	0,0	0,0	0,0	0,0
CCM	OCB	902500,0	0,0	902500,0	0,0
CCM	XTS	902500,0	0,0	902500,0	0,0
GCM	OCB	902500,0	0,0	902500,0	0,0
GCM	XTS	902500,0	0,0	902500,0	0,0
OCB	XTS	902500,0	0,0	902500,0	0,0
Velikost čistopisa/kriptograma: 512 bajtov					
ECB	CBC	0,0	0,0	902500,0	0,0
ECB	CTR	902500,0	0,0	902500,0	0,0
ECB	CCM	0,0	0,0	0,0	0,0
ECB	GCM	0,0	0,0	0,0	0,0
ECB	OCB	902500,0	0,0	902500,0	0,0
ECB	XTS	0,0	0,0	0,0	0,0
CBC	CTR	902500,0	0,0	0,0	0,0
CBC	CCM	0,0	0,0	0,0	0,0
CBC	GCM	902500,0	0,0	0,0	0,0
CBC	OCB	902500,0	0,0	0,0	0,0
CBC	XTS	0,0	0,0	0,0	0,0
CTR	CCM	0,0	0,0	0,0	0,0
CTR	GCM	0,0	0,0	0,0	0,0
CTR	OCB	0,0	0,0	0,0	0,0
CTR	XTS	0,0	0,0	0,0	0,0
CCM	GCM	902500,0	0,0	902500,0	0,0
CCM	OCB	902500,0	0,0	902500,0	0,0
CCM	XTS	0,0	0,0	0,0	0,0
GCM	OCB	902500,0	0,0	902500,0	0,0
GCM	XTS	0,0	0,0	0,0	0,0
OCB	XTS	0,0	0,0	0,0	0,0
Velikost čistopisa/kriptograma: 10000000 bajtov					
ECB	CBC	0,0	0,0	902500,0	0,0
ECB	CTR	902500,0	0,0	902500,0	0,0
ECB	CCM	0,0	0,0	0,0	0,0
ECB	GCM	0,0	0,0	23,0	0,0
ECB	OCB	902500,0	0,0	902500,0	0,0
ECB	XTS	0,0	0,0	0,0	0,0
CBC	CTR	902500,0	0,0	0,0	0,0
CBC	CCM	0,0	0,0	0,0	0,0
CBC	GCM	902500,0	0,0	0,0	0,0

Nadaljevanje na naslednji strani

Tabela B.3 – *Nadaljevanje s prejšnje strani*

n = 950		Šifriranje		Dešifriranje	
X	Y	U	p	U	p
CBC	OCB	902500,0	0,0	0,0	0,0
CBC	XTS	0,0	0,0	0,0	0,0
CTR	CCM	0,0	0,0	0,0	0,0
CTR	GCM	0,0	0,0	0,0	0,0
CTR	OCB	0,0	0,0	0,0	0,0
CTR	XTS	0,0	0,0	0,0	0,0
CCM	GCM	902500,0	0,0	902500,0	0,0
CCM	OCB	902500,0	0,0	902500,0	0,0
CCM	XTS	0,0	0,0	0,0	0,0
GCM	OCB	902500,0	0,0	902500,0	0,0
GCM	XTS	0,0	0,0	0,0	0,0
OCB	XTS	0,0	0,0	0,0	0,0

Tabela B.4: U- in p-vrednosti statističnih Mann-Whitneyjevih U-testov strojno pospešenega algoritma AES z različnimi načini šifriranja za knjižnico Crypto++

n = 950		Šifriranje		Dešifriranje	
X	Y	U	p	U	p
Velikost čistopisa/kriptograma: 16 bajtov					
ECB	CBC	0,0	0,0	0,0	0,0
ECB	CTR	0,0	0,0	0,0	0,0
ECB	CCM	0,0	0,0	0,0	0,0
ECB	GCM	0,0	0,0	0,0	0,0
CBC	CTR	0,0	0,0	0,0	0,0
CBC	CCM	0,0	0,0	0,0	0,0
CBC	GCM	0,0	0,0	0,0	0,0
CTR	CCM	0,0	0,0	0,0	0,0
CTR	GCM	0,0	0,0	0,0	0,0
CCM	GCM	893311,0	1,2677067526195803e-300	872007,0	3,4270005782359216e-272
Velikost čistopisa/kriptograma: 32 bajtov					
ECB	CBC	0,0	0,0	0,0	0,0
ECB	CTR	0,0	0,0	0,0	0,0
ECB	CCM	0,0	0,0	0,0	0,0
ECB	GCM	0,0	0,0	0,0	0,0
CBC	CTR	0,0	0,0	0,0	0,0
CBC	CCM	0,0	0,0	0,0	0,0

Nadaljevanje na naslednji strani

Tabela B.4 – *Nadaljevanje s prejšnje strani*

n = 950		Šifriranje		Dešifriranje	
X	Y	U	p	U	p
CBC	GCM	0,0	0,0	0,0	0,0
CTR	CCM	0,0	0,0	0,0	0,0
CTR	GCM	0,0	0,0	0,0	0,0
CCM	GCM	902500,0	0,0	902480,5	0,0
Velikost čistopisa/kriptograma: 512 bajtov					
ECB	CBC	0,0	0,0	0,0	0,0
ECB	CTR	0,0	0,0	0,0	0,0
ECB	CCM	0,0	0,0	0,0	0,0
ECB	GCM	0,0	0,0	0,0	0,0
CBC	CTR	902500,0	0,0	831250,0	7,513846349736774e-232
CBC	CCM	0,0	0,0	0,0	0,0
CBC	GCM	902500,0	0,0	0,0	0,0
CTR	CCM	0,0	0,0	0,0	0,0
CTR	GCM	0,0	0,0	0,0	0,0
CCM	GCM	902500,0	0,0	902500,0	0,0
Velikost čistopisa/kriptograma: 10000000 bajtov					
ECB	CBC	0,0	0,0	0,0	0,0
ECB	CTR	0,0	0,0	0,0	0,0
ECB	CCM	0,0	0,0	0,0	0,0
ECB	GCM	0,0	0,0	0,0	0,0
CBC	CTR	902500,0	0,0	0,0	0,0
CBC	CCM	0,0	0,0	0,0	0,0
CBC	GCM	902500,0	0,0	0,0	0,0
CTR	CCM	0,0	0,0	0,0	0,0
CTR	GCM	0,0	0,0	0,0	0,0
CCM	GCM	902500,0	0,0	902500,0	0,0