Primerjava hitrosti simetričnih bločnih šifer

Marko Kompara[1], Tomi Jerenko[1], Marko Hölbl[1]
[1] Univeza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, Koroška cesta 46, Maribor, Slovenija
marko.kompara@um.si, jerenkotomi@gmail.com, marko.holbl@um.si

Izvleček

Zaupnost podatkov je ena od osnovnih zahtev varovanja podatkov, ki je danes najpogosteje zagotovljena z uporabo simetričnih bločnih šifer. Najpogosteje uporabljena simetrična bločna šifra je AES. Njegova prilagodljivost z različnimi načini delovanja, možnostjo hitre programske in strojne implementacije, ter podpora s strani proizvajalcev opreme omogočajo, da je AES prisoten skoraj povsod, kjer je potrebno šifriranje podatkov. V prispevku je hitrost delovanja AES primerjana s šiframi Camellia, DES, 3DES, Serpent in Twofish. Poleg tega je v analizo vključena tudi strojna implementacija algoritma AES. Zanima nas, kakšen je dejanski učinek strojne implementacije algoritma na hitrost njegovega delovanja. Rezultati raziskave kažejo, da je AES najhitrejši algoritem med primerjanimi algoritmi. Razlika v hitrosti pa se v strojno pospešenem načinu izboljša za več kot cel red velikosti.

Ključne besede: AES, AES-NI, Camellia, DES, 3DES, Serpent, Twofish, primerjava hitrosti delovanja, simetrične bločne šifre.

Abstract

Confidentiality is one of the underlying data protection requirements. In modern cryptography, this is generally provided for by symmetric block ciphers. The most frequently used such cipher is AES. The cipher's modularity achieved with the help of the encryption modes, fast software and hardware implementation, and good support from many equipment manufacturers and software developers have allowed AES to be present wherever encryption is needed. In the paper, the performance of the AES algorithm is compared to Camellia, DES, 3DES, Serpent and Twofish. The hardware implementation of the AES is also included in the comparison. We are interested to see how much hardware implementation increases the algorithm's performance. Results show that AES is the fastest algorithm of those compared. Furthermore, the hardware-implemented version is more than an order of magnitude faster.

Keywords: AES, AES-NI, Camellia, DES, 3DES, Serpent, Twofish, speed comparison, symmetric block ciphers.

1 UVOD

Zagotavljanje varnosti in zasebnosti v digitalnem svetu je postalo zelo pomembno. Zgodovinsko je bila zaupnost podatkov zanimiva predvsem za državne organizacije. S širjenjem digitalizacije je ta postala pomembna za podjetja in kar nekaj časa je minilo, preden je zagotavljanje zaupnosti podatkov postalo pomembno tudi za navadnega uporabnika. Zasebnost uporabnikov na spletu je problematika, ki se je začela sistematično naslavljati še kasneje in je v veliki meri tudi še vedno aktualna. Šifrirni algoritmi so glavni način zagotavljanja zaupnosti in so pomembni tudi pri zagotavljanju zasebnosti. Poznamo več vrst šifrirnih algoritmov. Simetrični bločni algoritmi so najpogosteje uporabljena oblika šifrirnih algoritmov in med temi je šifra AES (Advanced Encryption Standard) najbolj razširjena. Natančnih vrednosti o deležih uporabe ni mogoče pridobiti, ampak konser-

vativna ocena je, da se AES uporablja v več kot 50% šifriranja vseh podatkov [1].

Algoritem AES je nastal leta 2001, kot rezultat NIST (National Institute of Standards and Technology) natečaja [2], v katerem so izbirali novo simetrično bločno šifro, ki bi postala standardizirana oblika šifriranja in tako nadomestila obstoječi algoritem DES (Data Encryption Standard) oziroma 3DES (Triple DES). AES je torej več kot dvajset let star algoritem. V tem prispevku želimo preveriti hitrost tega algoritma proti nekaterim drugim dobro poznanim algoritmom. V dvajsetih letih se je v veliki meri izboljšala tudi procesorska moč naprav, zato nas zanima, v kakšni meri strojna pospešitev še vpliva na hitrost delovanja algoritma AES.

V raziskavo smo vključili popularne knjižnice Crypto++, Libgcrypt, Nettle in OpenSSL. Izkazalo se je, da različne knjižnice razen osnovnega nabora tipično ne implementirajo velikega števila šifrirnih algoritmov. Zato smo se tudi omejili na šifrirne algoritme, ki so bili zastopani v vseh knjižnicah: AES, AES-NI, Camellia, DES, 3DES, Serpent in Twofish. Edina izjema v tem naboru je OpenSSL, ki ne implementira Serpent in Twofish, vendar ker je to zelo pogosto uporabljena knjižnica, smo se določili, da jo kljub temu vključimo. Čeprav je osnovni namen uporabe več različnih knjižnic preprečiti vpliv na rezultate hitrosti, ki bi lahko nastali kot posledica različnih kakovosti implementacij šifer, bodo rezultati te raziskave prikazali tudi razlike med delovanjem knjižnic in izpostavili, katere knjižnice vključujejo najbolj učinkovito implementacijo posamezne šifre.

V nadaljevanju tega prispevka bodo najprej na kratko predstavljeni vsi šifrirni algoritmi, ki so primerjani v tem delu. Zatem bomo kratko poglavje namenili predstavitvi okoliščin eksperimenta, katere omejitve smo pri raziskavi upoštevali in kako so meritve izvedene. V sledečem poglavju bomo predstavili in analizirali rezultate primerjave delovanja šifer in različnih knjižnic. V zadnjem poglavju bomo povzeli rezultate in zaključili prispevek.

2 SIMETRIČNE BLOČNE ŠIFRE

Simetrično bločno šifriranje [3] je postopek, v katerem se šifriranje in dešifriranje izvede na podlagi enakega ključa. Takšne šifre lahko istočasno obdelujejo samo podatke določene velikosti, ki jo imenujemo blok. Večje količine podatkov se razdelijo v več blokov, od katerih je vsak posamezno obdelan. Simetrične bločne šifre so najpogostejši način šifriranja podatkov (npr. na disku ali med prenosom), medtem ko se druge oblike šifriranja (asimetrične šifre in tokovne simetrične šifre) uporabljajo za bolj specializirane naloge. V nadaljevanju tega poglavja bomo na kratko predstavili simetrične bločne šifre, katerih hitrosti bomo merili v tem prispevku. Tabela 1 vključuje podatke o šifrah, ki smo jih uporabili v meritvah in knjižnicah, ki jih implementirajo.

2.1 Šifra AES

Standard AES (Advanced Encryption Standard) [4] je svojo pot začel kot algoritem Rijndael, ki je delo dveh belgijskih kriptografov (Vincent Rijmen in Joan Daemen) [5]. Rijndael je bil eden od 15 kandidatov in končni zmagovalec NIST (National Institute of Standards and Technology) natečaja za novo standardizirano simetrično bločno šifro, ki je potekal med 1997 in 2000. Rijndael je bil leta 2001 standardiziran kot AES v nekoliko omejenem delovanju, tako da deluje

Tabela 1: **Lastnosti izbranih šifer in njihova implementacija v knjižnicah.**

	Lastnost šifre		**Knjižnica**			
Šifre	**Blok (biti)**	**Testiran ključ (biti)**	**Crzpto++**	**Libgcrzpt**	**Nettle**	**OpenSSL**
AES	128	128	✓	✓	✓	✓
AES-NI	128	128	✓	✓	✓	✓
Camellia	128	128	✓	✓	✓	✓
DES	64	56	✓	✓	✓	✓
3DES	64	168	✓	✓	✓	✓
Serpent	128	128	✓	✓	✓	✓
Twofish	128	128	✓	✓	✓	✓

s 128 bitnimi bloki in s 128, 192 in 256 bitov dolgimi ključi. AES deluje po principu substitucijsko-permutacijskega omrežja (angl. substitution–permutation network).

2.2 Implementacija AES-NI

AES-NI [6, 7] je strojno pospešena implementacija algoritma AES, ki dobi svoje ime po Intel AES novih ukazih (angl. Intel Advanced Encryption Standard New Instructions). To je bila prva množična implementacija takšnega delovanja šifre. Danes s tem imenom poimenujemo tudi stojno pospešene implementacije drugih proizvajalcev (npr. na AMD in ARM procesorjih). AES-NI vsebuje šest ukazov, ki omogočajo strojno izvajanje celotnega algoritma AES. Štirje ukazi so namenjeni šifriranju in dešifriranju, dva ukaza pa sta namenjena generiranju vmesnih ključev algoritma. Ti ukazi so fizično vgrajeni v centralno procesno enoto, zaradi česar je njihovo izvajanje tudi toliko hitrejše. Ti ukazi so neposredno dostopni iz uporabniškega prostora, kar omogoča tudi enostavnejšo implementacijo AES šifre. Neposredno izvajanje šifriranja in dešifriranja na procesorju tudi izniči potrebo po hranjenju iskalnih tabel v predpomnilniku in zagotavlja časovno zakasnitev, ki ni odvisna od obdelovanih podatkov. Takšno delovanje preprečuje nekatere najbolj razširjene in najbolj nevarne napade po stranskem kanalu (angl. side channel attack).

2.3 Šifra Camellia

Camellia je najnovejša med primerjanimi šiframi. Razvili so jo v Mitsubishi Electric in Nippon Telegraph and Telephone leta 2000 na Japonskem [8]. Camellia deluje po načinu Feistelove mreže in uporablja 128 bitne bloke z možnostjo 128, 192 in 256 bitnih ključev. Te postavke so bile namenoma izbrane, da je šifra primerljiva s šiframi, ki so sodelovale na natečaju AES. Za šifro so pokazali, da je primerljiva v hitrosti in nivoju varnosti s kandidati natečaja, njeno prednost pa predstavlja kompaktna strojna implementacija [9]. Deloma se Camellia lahko izvede tudi z ukazi AES-NI (substitucijski del v S-škatlah), kar omogoča njeno pospešeno delovanje na obstoječi strojni opremi [8]. V tem prispevku so merjene zgolj programske implementacije te šifre (ne izkorišča se pospešitev, ki bi jo prinesla uporaba AES-NI). Camellia je vključena tudi v ISO/IEC standard in je bila potrjena za uporabo v projektu Evropske Unije NESSIE (New European Schemes for Signatures, Integrity and Encryption) in japonskem projektu CRYPTOREC (Cryptography Research and Evaluation Committees).

2.4 Šifri DES in 3DES

DES (Data Encryption Standard) je standardiziran algoritem DEA (Data Encryption Algorithm), ki ga je razvil IBM [10]. Algoritem je bil leta 1977 standardiziran s strani NBS (National Bureau of Standards), ki je s časom postal današnji NIST. To je vodilo k hitremu sprejemu šifre v praksi, vendar so se zelo hitro pojavili tudi pomisleki glede njene varnosti. Največja pomanjkljivost je njena dolžina ključa, ki je znašala samo 56 bitov. DES je bil dokončno opuščen kot standard leta 2005 in ni več primeren za varovanje podatkov, ker ne zadošča modernim minimalnim standardom dolžine ključa [11]. DES je zasnovan na Feistelovi mreži s 64 bitnim ključem, pri čemer se 8 bitov uporablja za zaznavanje napak, in 64 bitnim blokom.

TDES (Triple Data Encryption Standard) oz. 3DES je standard algoritma TDEA, ki je bil ustvarjen leta 1995 in je v svojem delovanju enak standardu DES, le da se izvede trikrat ob uporabi treh različnih ključev (možne so tudi kombinacije ponovne uporabe enakega ključa, vendar se ti načini ne smatrajo več kot varni) [11]. Posledično je ta sestavljen iz 168 naključnih bitov in 24 paritetnih bitov. Čeprav izgleda, kot da 3DES zagotavlja najvišji nivo varnosti s 168 bitnim ključe, je standard ranljiv na napad s srečanjem v sredini (angl. meet-in-the-middle attack), ki zniža varnost na 112 bitov [12]. Posledično je 3DES poleg DES najmanj varna šifra od primerjanega nabora algoritmov.

2.5 Šifra Serpent

Serpent izhaja iz leta 1998 in je druga šifra v našem naboru, ki je bila med kandidati NIST natečaja in glede na rezultate natečaja tudi druga najboljša [13]. Čeprav je Serpent veljal za bolj varen algoritem kot Rijndael, je bila njegova pomanjkljivost predvsem počasnejše delovanje v programskih implementacijah. Tako kot AES deluje s 128 bitnimi bloki in omogoča uporabo 128, 192 ali 256 bitnega ključa, čeprav sam algoritem vedno uporabi 256 bitno vrednost, ki je v primeru manjšega posredovanega ključa razširjena na to velikost. Kot je že bilo omenjeno smo se v tej raziskavi omejili na 128 bitne ključe (ki bodo kot del algoritma nato avtomatsko razširjeni na 256 bitov). Tako kot AES je tudi Serpent zgrajen iz substitucijsko-permutacijskih gradnikov.

2.6 Šifra Twofish

Twofish [14] je še ena šifra iz leta 1998 in je tretja šifra (poleg Rijndael in Serpent) v tem seznamu, ki je bila udeležena na natečaju, na katerem je bil izbran AES. Za svoje delovanje uporablja Feistelovo mrežo. Podatki se obdelujejo v 128 bitov velikih blokih in možna je uporaba ključev velikosti 128, 192 ali 256 bitov, čeprav algoritem omogoča tudi vnos drugih vrednosti, ki pa so nato dopolnjene z ničlami do prvega od teh mejnikov.

3 ŠIFRA AES

Standard AES (Advanced Encryption Standard) [4] je svojo pot začel kot algoritem Rijndael, ki je delo dveh belgijskih kriptografov (Vincent Rijmen in Joan Daemen) [5]. Rijndael je bil eden od 15 kandidatov in končni zmagovalec NIST (National Institute of Standards and Technology) natečaja za novo standardizirano simetrično bločno šifro, ki je potekal med 1997 in 2000. Rijndael je bil leta 2001 standardiziran kot AES v nekoliko omejenem delovanju, tako da deluje

Tabela 2: **Povprečne meritve hitrosti v ciklih/zlog, za vse kombinacije šifer, knjižnic, velikosti podatkov in operacijo.**

		16B		32B		512B		10MB	
		Šif	**Deš**	**Šif**	**Deš**	**Šif**	**Deš**	**Šif**	**Deš**
Crypto++	AES	17,051	19,856	14,326	19,047	11,488	18,556	11,358	18,512
	AES-NI	6,250	6	3,653	3,967	0,819	0,814	0,732	0,732
	Camellia	23,32	23,215	22,675	22,861	22,17	22,232	22,183	22,26
	DES	43,484	43,388	42,526	42,584	41,652	41,672	41,874	41,872
	3DES	117,094	117,143	116,539	116,56	115,059	115,091	115,275	115,274
	Serpent	37,788	33,413	37,019	33,238	36,228	32,704	36,302	32,782
	Twofish	19,6	19,468	18,699	18,899	18,31	18,207	18,265	18,215
Libgcrypt	AES	13,75	16,247	11,878	14,158	9,952	12,321	9,692	12,17
	AES-NI	8,553	8,371	4,75	4,684	1,672	1,773	1,534	1,577
	Camellia	26,16	25,423	22,795	22,71	19,99	20,088	19,885	19,881
	DES	45,386	44,93	42,513	42,361	40,067	40,104	40,188	40,177
	3DES	96,858	96,646	93,799	93,395	91,323	90,794	91,318	91,008
	Serpent	41,467	38,019	38,609	34,855	36,217	32,875	36,213	32,549
	Twofish	21,873	21,638	19,312	19,137	16,799	16,87	16,733	16,827
Nettle	AES	12	12,117	11,689	11,791	11,49	11,45	11,545	11,529
	AES-NI	4,554	4,5	2,875	2,779	1,352	1,288	1,286	1,238
	Camellia	19,988	19,986	19,851	19,837	19,503	19,475	19,608	19,611
	DES	41,856	42,197	40,803	41,201	39,983	40,58	40,339	40,823
	3DES	124,36	124,666	122,789	123,516	120,573	121,525	121,462	121,917
	Serpent	38,325	32,396	38,169	32,174	14,115	13,107	14,21	13,186
	Twofish	20,685	18,642	20,416	18,505	20,226	18,33	19,818	18,323
OpenSSL	AES	11,873	15,067	10,555	13,434	9,492	11,836	9,268	11,666
	AES-NI	6,681	7,642	3,456	3,937	0,753	0,799	0,677	0,677
	Camellia	21,027	21,964	20,384	20,766	18,622	18,855	18,689	18,9
	DES	47,983	47,263	46,493	45,326	45,132	43,516	45,207	43,827
	3DES	127,068	125,512	124,848	123,495	123,225	121,698	124,715	123,989
	Serpent	Knjižnica ne implementira šifre Serpent.							
	Twofish	Knjižnica ne implementira šifre Twofish							

s 128 bitnimi bloki in s 128, 192 in 256 bitov dolgimi ključi. AES deluje po principu substitucijsko-permutacijskega omrežja (angl. substitution–permutation network).

4 IMPLEMENTACIJA AES-NI

AES-NI [6, 7] je strojno pospešena implementacija algoritma AES, ki dobi svoje ime po Intel AES novih ukazih (angl. Intel Advanced Encryption Standard New Instructions). To je bila prva množična implementacija takšnega delovanja šifre. Danes s tem imenom poimenujemo tudi stojno pospešene implementacije drugih proizvajalcev (npr. na AMD in ARM procesorjih). AES-NI vsebuje šest ukazov, ki omogočajo strojno izvajanje celotnega algoritma AES. Štirje ukazi so namenjeni šifriranju in dešifriranju, dva ukaza pa sta namenjena generiranju vmesnih ključev algoritma. Ti ukazi so fizično vgrajeni v centralno procesno enoto, zaradi česar je njihovo izvajanje tudi toliko hitrejše. Ti ukazi so neposredno dostopni iz uporabniškega prostora, kar omogoča tudi enostavnejšo implementacijo AES šifre. Neposredno izvajanje šifriranja in dešifriranja na procesorju tudi izniči potrebo po hranjenju iskalnih tabel v predpomnilniku in zagotavlja časovno zakasnitev, ki ni odvisna od obdelovanih podatkov. Takšno delovanje preprečuje nekatere najbolj razširjene in najbolj nevarne napade po stranskem kanalu (angl. side channel attack).

4.1 Analiza rezultatov

Analizo hitrosti različnih simetričnih bločnih šifrirnih algoritmov smo začeli s statistično analizo. Podatki pridobljeni v eksperimentu so bili pretežno nenormalno porazdeljeni, zato smo se odločili za uporabo neparametričnih testov. Za vsako knjižnico smo posebej izvedli *Kruskal-Wallis test* [18], ki pokaže, ali med algoritmi (znotraj posamezne knjižnice) obstajajo signifikantne razlike. Rezultat je bil v vseh primerih pozitiven, zato smo nadaljevali z *Mann-Whitney U testi* [19], s katerimi med vsakim parom algoritmov preizkusimo signifikantnost razlik (v tem primeru hitrosti) med njima. Kljub uporabi *Bonferronijevega popravka* [20] so statistični testi za vse parne primerjave pokazali, da med algoritmi obstajajo signifikantne razlike. P-vrednosti so bile tudi po zaokroženju na več deset decimalnih mest še vedno 0. Zato smo se odločili, da teh rezultatov ne vključimo, temveč raje nadaljujemo z analizo velikosti vpliva (angl. effect size). Velikost vpliva je metrika, ki nam pove, kako velika oz. pomembna je razlika, ki so jo statistični testi pokazali. V tej nalogi smo uporabili *Cohenov d* za izračun velikosti učinka, ki je definiran v enačbi (1), kjer sta $\bar{A}$ in $\bar{B}$ povprečni vrednosti meritev algoritmov A in B, in σ je standardni odklon.

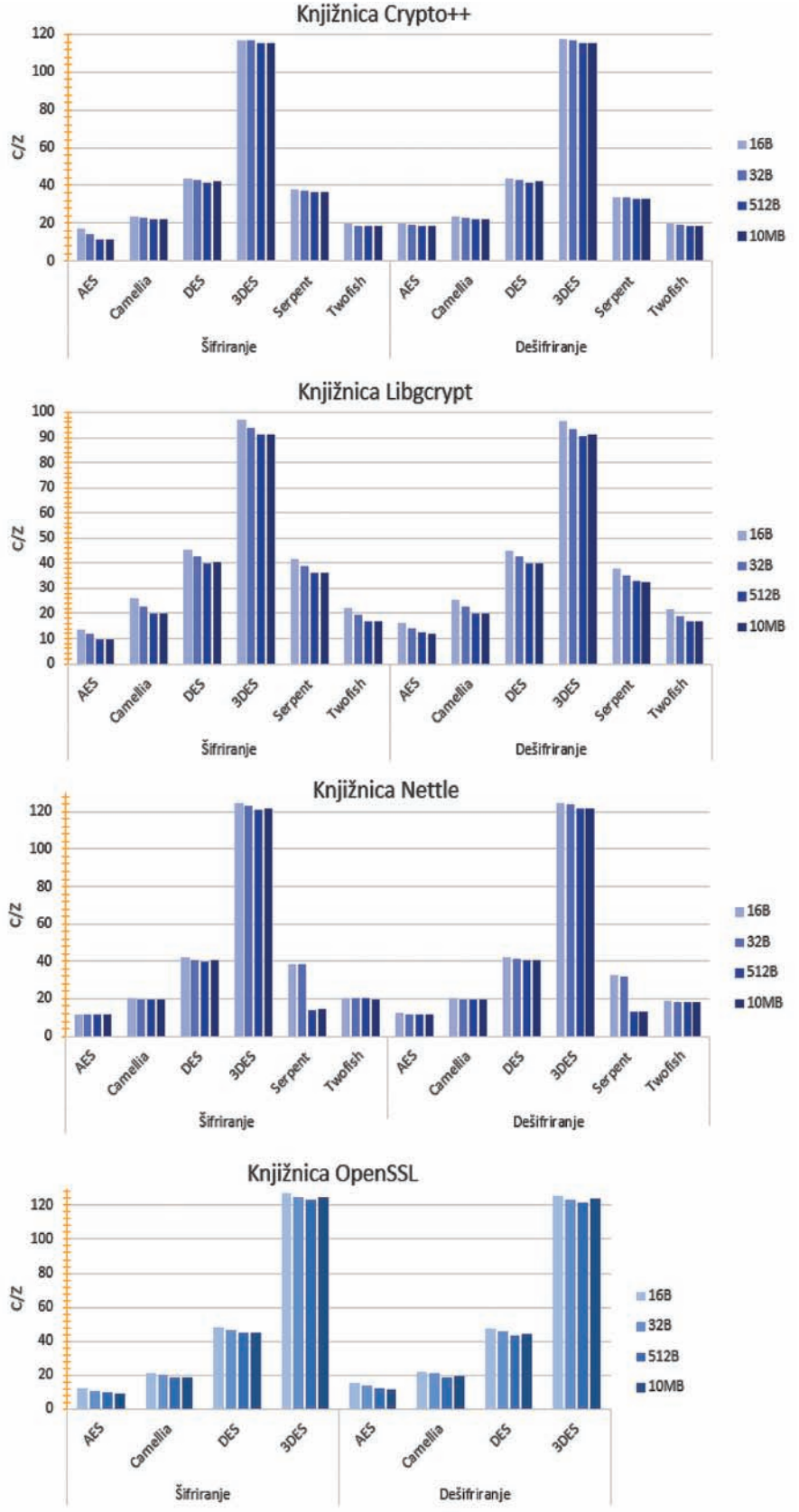
$$d = \frac{\bar{A} - \bar{B}}{\sqrt{\frac{\sigma_A^2 + \sigma_B^2}{2}}} \qquad (1)$$

Rezultati velikosti učinka so zaradi zelo majhnih standardnih odklonov računalniško generiranih podatkov zelo veliki v primerjavi z lestvico za interpretacijo rezultatov, ki jo je predlagal Cohen [21]. Ne glede na to so med velikostmi učinka za vsak par šifer (znotraj iste knjižnice) opazne razlike. Rezultati kažejo, da velikost učinka raste z večanje velikost vhodnih podatkov. Iz tega lahko razberemo, da z večanjem količine šifriranih podatkov, raste tudi pomembnost razlik med algoritmi. Rezultati nakazujejo, da so si med seboj po hitrosti delovanja (vsaj pri manjši količini podatkov) najbolj podobne šifre AES, Camellia in Twofish ter DES in Serpent. Iz rezultatov je tudi hitro razvidno, da se algoritma AES-NI in 3DES v hitrosti izvajanja najbolj razlikujeta od ostalih.

Rezultati statistične analize nakazujejo, da so med algoritmi pomembne razlike v hitrosti delovanja, vendar na podlagi teh testov nismo dobili praktične predstave, kako velike so te razlike. Zato bomo v nadaljevanju razlike v hitrosti delovanja različnih simetričnih bločnih šifer opisali in analizirali še s pomočjo opisne statistike.

4.2 Primerjava hitrosti delovanja simetričnih bločnih šifer

Na sliki 1 so prikazani grafikoni z rezultati hitrosti v ciklih na zlog (C/Z) delovanja programsko implementiranih šifer v različnih knjižnicah. Najbolj očiten rezultat je zagotovo počasno delovanje algoritma 3DES ne glede na uporabljeno knjižnico. Druga najpočasnejša šifra je DES. Ti dve sta tudi najmanj varni od vseh primerjanih šifer, zato je uporaba teh šifer v kakršenkoli drug namen, kot zaradi združljivosti za nazaj (angl. backward compatible) nerazumljiva. Za slabe rezultate lahko v majhni meri vpliva tudi izbira velikosti šifriranih in dešifriranih podatkov, ki so večkratniki 128 bitov, kar pomeni, da morata algoritma DES in 3DES procesirati večje število blokov, ker



Slika 1: **Grafikoni primerjave šifrirnih algoritmov po knjižnicah.**

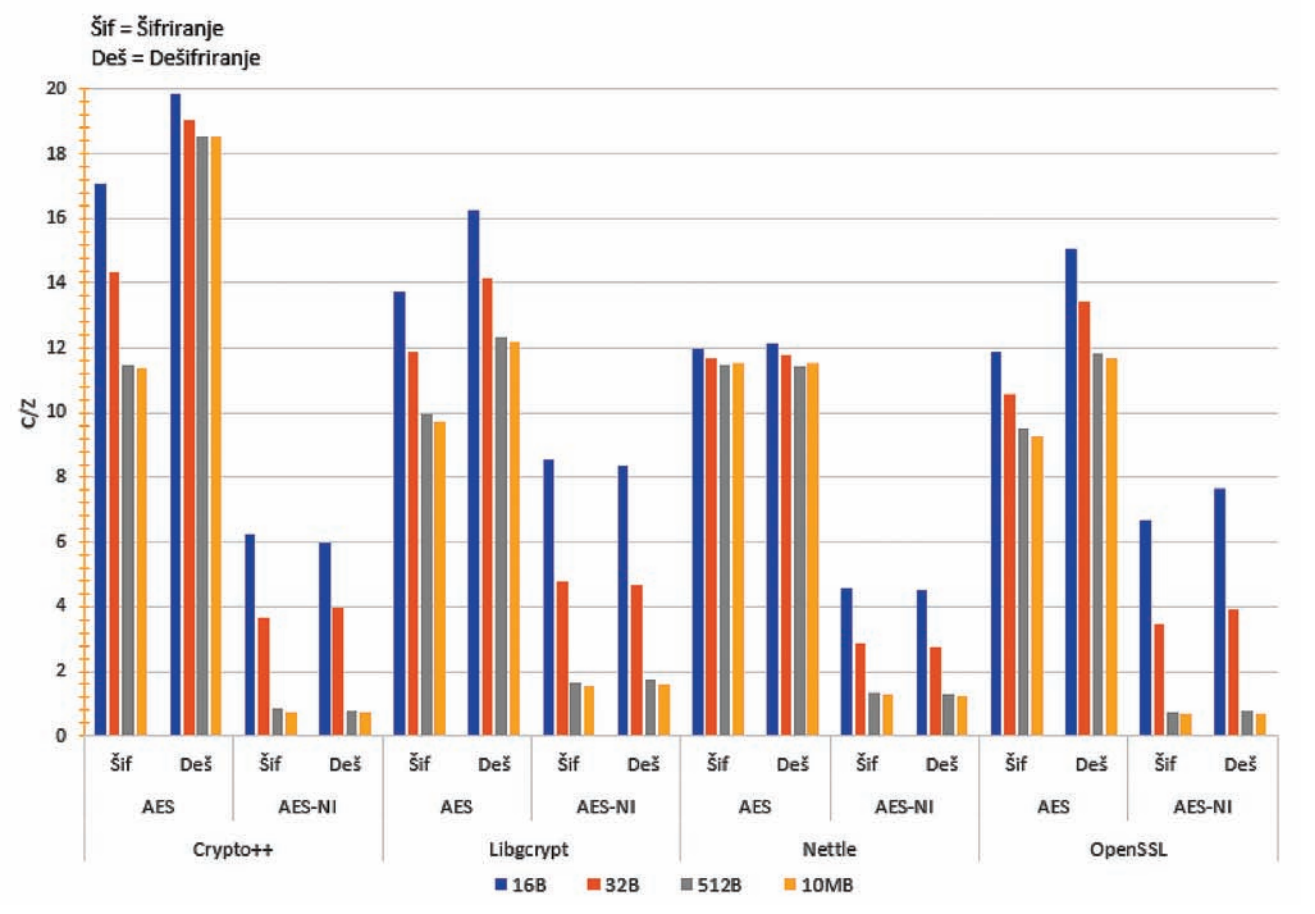
procesirata vhodne podatke v blokih velikosti 64 bitov. Tretja najpočasnejša šifra v večini knjižnic je Serpent. Glede na to, da smo že v opisu šifre omenili, da velja za bolj kompleksen algoritem, ki zagotavlja višji nivo varnosti, je takšen rezultat pričakovan. Camellia in Twofish sta bila primerljivo hitra z zelo majhnimi razlikami med vsemi knjižnicami. Ne glede na to je bila šifra Twofish v povprečju malenkostno hitrejša. Pomanjkljivost te šifre je predvsem to, da ni implementirana v knjižnici OpenSSL, ki je zelo pogosto uporabljena knjižnica [22].

Med vsemi programsko implementiranimi algoritmi je bil brez dvoma najhitrejši algoritem AES. Na podlagi tega lahko sklepamo, da je najpogosteje uporabljen šifrirni algoritem tudi najhitrejši. Med knjižnicami se je za izvajanje AES in Camellia najboljše odrezala knjižnica OpenSSL, Serpent je bil najhitrejši v knjižnici Nettle, medtem ko je za preostale algoritme bila Libgcrypt najboljša izbira. Predvsem v knjižnici Nettle, v določeni meri pa tudi pri Crypto++, je mogoče opaziti, da z večanjem velikosti šifriranih oz. dešifriranih podatkov hitrost ne narašča, kot bi pričakovali zaradi vedno manjšega deleža režijskih stroškov. Če torej pogosto šifriramo zelo majhne količine podatkov, bi ti dve knjižnici mogoče (odvisno od dane šifre) bili boljša izbira kot knjižnici OpenSSL in Libgcrypt.

4.3 Primerjava delovanja AES in AES-NI

V prejšnjem poglavju smo pokazali, da je AES najhitrejša programsko implementirana šifra v naboru algoritmov, ki smo jih primerjali. Velik delež modernih procesorskih enot dodatno podpira tudi strojno pospešitev tega algoritma. Na sliki 2 je prikazana primerjava hitrosti delovanja šifre AES v programski in strojno pospešeni implementaciji (AES-NI) v različnih knjižnicah.

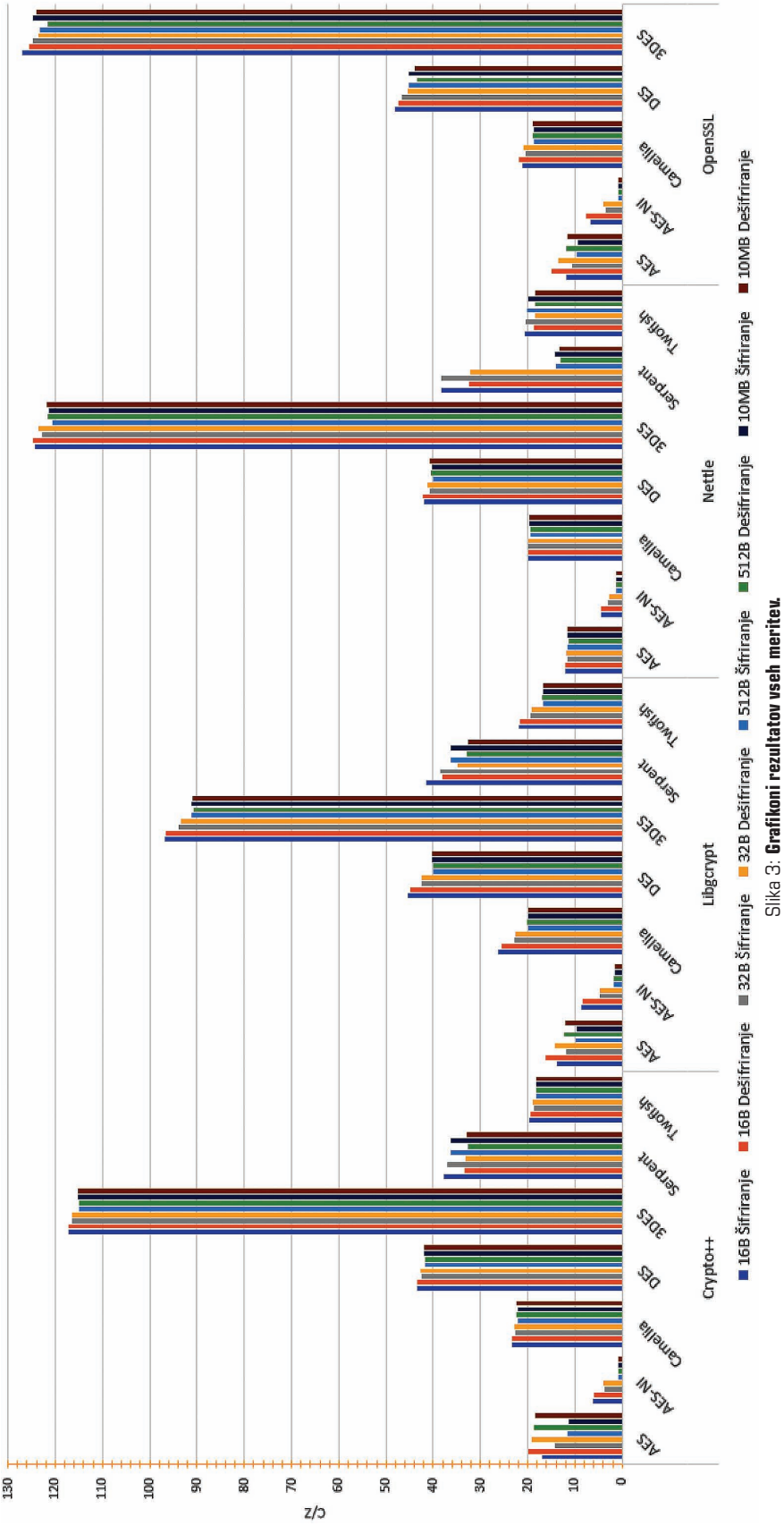
Razlike v hitrosti delovanja med AES in AES-NI so takoj očitne, ne glede na uporabljeno knjižnico. AES-NI je vedno signifikantno hitrejši. Razlike v primeru manjših velikosti podatkov so nekoliko manjše zaradi režijskega dela, ki je potreben ne glede na



Slika 2: **Grafikoni primerjave AES in AES-NI hitrosti delovanja.**

tip implementacije. Za podatke velikosti 16B je AES-NI med 1,6 (Libgcrypt) in 2,7 (Crypto++) krat hitrejši pri šifriranju in med 1,9 (Libgcrypt) ter 3,3 (Crypto++) krat hitrejši pri dešifriranju podatkov. Z večanjem količine podatkov hitrost izvajanja proporcionalno na količino podatkov narašča, predvsem v AES-NI. Tu vidimo, da je signifikantna razlika med meritvami do velikosti podatkov 512B. Med 512B in 10MB je izboljšanje komaj opazno, kar pomeni, da smo dosegli zgornji prag hitrosti izvajanja. Za podatke velikosti 10MB je AES-NI med 6,3 (Libgcrypt) in 15,5 (Crypto++) krat hitrejši pri šifriranju in med 7,7 (Libgcrypt) ter 25,3 (Crypto++) krat hitrejši pri dešifriranju podatkov. Opazimo tudi, da je med programsko in strojno pospešeno implementacijo vedno do najmanjšega izboljšanja prišlo v knjižnici Libgcrypt in do največjega v Crypto++. Absolutno najhitrejše delovanje za majhne količine podatkov pa dosegla knjižnica Nettle (4,5 C/Z za 16B podatkov pri šifriranju in dešifriranju), medtem, ko je najhitrejše, za vse razen najmanjših podatkov, delovala implementacija knjižnice OpenSSL (0,7 C/Z za 10MB podatkov pri šifriranju in dešifriranju). Knjižnica OpenSSL se je v splošnem najbolje odrezala tudi pri izvajanju programske implementacije šifre AES.

Iz slike 2 je mogoče razbrati še eno lastnost programske implementacije AES. V vseh knjižnicah je v programskih implementacijah dešifriranje počasnejše od šifriranja. Ta lastnost je značilna za AES in v neki meri tudi za Serpent, ker imata podobno struk-



Slika 3: **Grafikoni rezultatov vseh meritev.**

turo delovanja (substitucijsko-permutacijsko omrežje). Kot bomo videli v nadaljevanju to ni značilno za druge primerjane šifre. Te razlike nastanejo zaradi zelo majhnih razlik med operacijami, ki se uporabljajo za šifriranje in dešifriranje in so opazne samo v programski implementaciji. Kot lahko vidimo AES-NI izvaja šifriranje in dešifriranje skoraj brez razlike v hitrosti med obema operacijama. To je tudi razlog, da so izboljšave med hitrostjo delovanja programske implementacije in strojne implementacije toliko večje pri dešifriranju.

4.4 Razprava

Na sliki 3 so povzeti vsi rezultati te raziskave. Iz grafikona lahko razberemo razlike med šifriranjem in dešifriranjem, ki smo jih omenili na koncu prejšnjega poglavja. Programsko implementiran AES in Serpent imata zobčaste vrhove stolpičnih grafikonov. Ta oblika nakazuje razlike med hitrostjo šifriranja in dešifriranja s temi šiframi, medtem ko so sosedni stolpiči pri ostalih šifrah, ki so osnovane na Feistelovi mreži, na isti višini. To je posledica delovanja Feistelove mreže, v kateri sta operaciji šifriranja in dešifriranja identični in posledično tudi enako hitri pri enaki velikosti vhodnih podatkov.

Med vsemi šiframi je AES-NI ne glede na izbrano knjižnico vedno najhitrejša izbira algoritma. Prikaz hitrosti takšnega delovanja v primerjavi z ostalimi šiframi (Slika 3), pokaže, kako velika je dejansko ta razlika. Algoritmi AES, AES-NI in Camellia se najhitreje izvajajo s knjižnico OpenSSL. Šifra Serpent je najhitrejša v knjižnici Nettle, medtem ko knjižnica Libgcrypt v primerjavi z drugimi knjižnicami, najhitreje izvaja šifre DES, 3DES in Twofish. Čeprav je knjižnica Crypto++ ena najbolj znanih knjižnic, namenjenih kriptografiji, izvajanje s to knjižnico ni za noben algoritem najhitrejše, vendar pa ne glede na okoliščine zagotavlja solidno hitrost delovanja vseh primerjanih šifer.

5 ZAKLJUČEK

V prispevku smo primerjali hitrosti šifriranja in dešifriranja različnih šifrirnih algoritmov v več boljše poznanih in sprejetih knjižnicah. Za daleč najhitrejše se je izkazalo strojno pospešeno izvajanje algoritma AES, ki je signifikantno hitrejše kot programska implementacija - v povprečju je AES-NI pri 16B podatkov 2,5 krat in pri 10MB podatkov 14,9 krat hitrejši kot osnoven AES. Tudi med izključno programskimi šiframi se je AES izkazal za najhitrejšega med vsemi primerjanimi algoritmi. Za drugi in tretji najhitrejši algoritem sta se izkazala Twofish in Camellia z zelo primerljivimi rezultati. Tem so sledili še Serpent, DES in 3DES. 3DES je bil občutno najpočasnejši algoritem v naboru primerjanih šifer. Rezultati so pokazali, da se v praksi dejansko uporablja najbolj učinkovit algoritem in čeprav je mogoče intuitivno mišljenje, da so starejši algoritmi, ki niso več varni bolj preprosti za izvajanje, rezultati jasno pokažejo, da temu ni tako in so dejansko najslabši v naboru primerjanih algoritmov.

Zaključimo lahko torej, da je AES še vedno najhitrejši algoritem in da strojna implementacija v veliki meri izboljša učinkovitost tega algoritma. Raziskava tudi nakazuje, da je za najhitrejšo možno implementacijo tega algoritma potrebno uporabiti knjižnico OpenSSL.

Afiliacije

Ta raziskava je nastala ob podpori raziskovalnega programa št. P2-0057, katerega je sofinancirala Javna agencija za raziskovalno dejavnost Republike Slovenije iz državnega proračuna ter projekta Cyber Security Network of Competence Centres for Europe (CyberSec4Europe), katerega je financirala EU iz okvirnega programa EU za raziskave in inovacije – Obzorje H2020.

LITERATURA

[1] C. Paar, Introduction to Cryptography by Christoff Paar: Lecture 8: Advanced Encryption Standard (AES), (2014). https://www.youtube.com/watch?v=NHuibtoL_qk.

[2] Crypto competitions - AES: the Advanced Encryption Standard, (n.d.). https://competitions.cr.yp.to/aes.html (dostopano 21 april 2020).

[3] D. Altermatt, Symmetric Encryption – An Introduction, (2019). https://www.scip.ch/en/?labs.20190815 (dostopano 17 junij 2020).

[4] M.J. Dworkin, E.B. Barker, J. R. Nechvatal, J. Foti, L. E. Bassham, E. Roback, J. F. Dray Jr., Advanced Encryption Standard (AES), (2001). doi:10.6028/NIST.FIPS.197.

[5] J. Daemen, V. Rijmen, The Block Cipher Rijndael, in: Int. Conf. Smart Card Res. Adv. Appl., Springer Verlag, 2000: pp. 277–284. doi:10.1007/10721064_26.

[6] J. Rott, Intel® Advanced Encryption Standard Instructions (AES-NI), (2012). https://software.intel.com/en-us/articles/intel-advanced-encryption-standard-instructions-aes-ni (dostopano 6 februar 2020).

[7] S. Gueron, Intel® Advanced Encryption Standard (Intel® AES) Instructions Set - Rev 3.01, 2012. https://software.intel.com/en-us/articles/intel-advanced-encryption-standard-aes-instructions-set (dostopano 21 april 2020).

[8] J. Kivilinna, Block Ciphers: Fast Implementations on x86-64 Architecture, 2013.

[9] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, T. Tokita, Camellia: A 128-Bit block cipher suitable for multiple platforms – Design and analysis, in: Lect. Notes Comput. Sci., Springer Verlag, 2001: pp. 39–56. doi:10.1007/3-540-44983-3_4.

[10] W. Stallings, Cryptography and Network Security, 4th ed., Prentice Hall, 2005. http://www.inf.ufsc.br/~bosco.sobral/ensino/ine5680/material-cripto-seg/2014-1/Stallings/Stallings_Cryptography_and_Network_Security.pdf.

[11] E. Barker, Recommendation for Key Management Part 1: General, NIST Spec. Publ. 800-57 Part 1 Revis. 4. (2016). https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf.

[12] W. Diffie, M.E. Hellman, Exhaustive Cryptanalysis of the NBS Data Encryption Standard, Computer (Long. Beach. Calif). 10 (1977) 74–84. doi:10.1109/C-M.1977.217750.

[13] J. Sugier, Implementing AES and Serpent Ciphers in New Generation of Low-Cost FPGA Devices, in: Complex Syst. Dependability, 2013: pp. 273–287.

[14] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson, Twofish: A 128Bit Block Cipher, 1998. https://www.researchgate.net/publication/245272403_Twofish_A_128Bit_Block_Cipher (dostopano 5 februar 2020).

[15] Intel® CoreTM i5-4200M Processor Product Specifications, (2013). https://ark.intel.com/content/www/us/en/ark/products/76348/intel-core-i5-4200m-processor-3m-cache-up-to-3--10-ghz.html (dostopano 21 april 2020).

[16] G. Paoloni, How to Benchmark Code Execution Times on Intel IA-32 and IA-64 Instruction Set Architectures, (2010). https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/ia-32-ia-64-benchmark-code-execution-paper.pdf.

[17] M. Dworkin, Recommendation for block cipher modes of operation: Methods and Techniques, 2001. doi:10.6028/NIST.SP.800-38a.

[18] A.C. Leon, Kruskal–Wallis test, in: Compr. Clin. Psychol., Elsevier, 1998: pp. 243–285. doi:10.1016/b0080-4270(73)00264-9.

[19] N.R. Smalheiser, The Mann-Whitney U Test, in: Data Lit. How to Make Your Exp. Robust Reprod., 2017. https://www.sciencedirect.com/book/9780128113066/data-literacy.

[20] E.W. Weisstein, Bonferroni Correction, (n.d.). https://mathworld.wolfram.com/BonferroniCorrection.html (dostopano 21 april 2020).

[21] J. Cohen, Statistical power analysis for the behavioral sciences, 1977.

[22] Cryptography and Encryption Libraries, LibHunt. (2020). https://cpp.libhunt.com/categories/661-cryptography (dostopano 22 april 2020).

▪

Dr. Marko Kompara je raziskovalec in asistent na Fakulteti za elektrotehniko, računalništvo in informatiko, Univerze v Mariboru. Doktorat iz računalništva in informatike je pridobil leta 2019. Tematika doktorske disertacije se je nanašala na protokole vzpostavitve ključa v strojno omejenih okoljih. Trenutno deluje kot raziskovalec na Horizon 2020 projektu CyberSec4Europe. Njegova raziskovalna področja vključujejo: zasebnost, kriptografija, zaščita brezžične komunikacije in zaščita informacijskih sistemov.

▪

Tomi Jerenko, magister inženir informatike in tehnologij komuniciranja, je pridobil svoj naziv na Fakulteti za elektrotehniko, računalništvo in informatiko v Mariboru leta 2019. Trenutno je zaposlen v Berlinu pri največjem evropskem ponudniku oblačne infrastrukture, imenovanem 1&1 IONOS, kjer sodeluje v razvoju z ekipama za aplikacijsko varnost in zagotavljanje kakovosti. Hkrati je tudi magistrski kandidat za naziv magister znanosti na nemški fakulteti Brandenburgische Technische Universität Cottbus-Senftenberg iz področja kibernetske varnosti.

▪

Dr. Marko Hölbl je docent na Fakulteti za elektrotehniko, računalništvo in informatiko, Univerze v Mariboru. V preteklosti je uspešno sodeloval v več nacionalnih in mednarodnih projektih trenutno pa je med drugim tudi lokalni koordinator Horizon 2020 projekta CyberSec4Europe. Deluje tudi kot generalni tajnik CEPIS LSI in EAEEIE, član ECSO WG 6 in član izvršnega odbora SDI. Njegova raziskovalna področja vključujejo: kriptografija, zaščita informacijskih sistemov, varnost na spletu, zaznavanje varnosti in zasebnosti na spletu, podatkovne baze in analiza podatkov.

.