# Kriptografski algoritmi za uređaje s ograničenim resursima

**Rusan, Brigita**

**Master's thesis / Diplomski rad**

**2025**

*Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj:* **University of Zagreb, Faculty of Science / Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet**

*Permanent link / Trajna poveznica:* https://urn.nsk.hr/urn:nbn:hr:217:857265

**SVEUČILIŠTE U ZAGREBU**

**PRIRODOSLOVNO–MATEMATIČKI FAKULTET**

**MATEMATIČKI ODSJEK**

Brigita Rusan

# KRIPTOGRAFSKI ALGORITMI ZA UREĐAJE S OGRANIČENIM RESURSIMA

Diplomski rad

Voditelj rada:
prof. dr. sc. Andrej Dujella

Zagreb, 2025.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

*Hvala svima koji su bili dio mog života tijekom studiranja – od početka, tijekom, na kraju, a najviše onima koji su bili uz mene od početka do samog kraja (takvih je najviše). Hvala svima koji su me tijekom ovog dugog perioda podržavali s ohrabrujućim riječima: „Možeš ti to!", „Uspjet ćeš!", „Stisni to i izguraj!", kao i onima koji su imali komentare poput „Odustani ti Brigita od faksa, nađi si posao, ovo više nema smisla". Zahvaljujem svom mentoru, prof. dr. sc. Andrej Dujelli, na stručnim savjetima, pomoći, smjernicama i vodstvu tijekom izrade ovog diplomskog rada. I za kraj, želim zahvaliti sebi na upornosti i tvrdoglavosti koja me dogurala do kraja.*

# Sadržaj

# Uvod

Kriptografija predstavlja ključnu disciplinu u području informacijske sigurnosti, omogućujući zaštitu podataka kroz različite metode šifriranja i autentifikacije. Ovaj rad istražuje kriptografske algoritme namijenjene uređajima s ograničenim resursima, s posebnim naglaskom na algoritme obrađene u trećem poglavlju rada. Također, u uvodnim poglavljima pruža pregled novih horizonata u kriptografiji te značaja „lake"(lightweight) kriptografije u kontekstu Interneta stvari (IoT).

Prvo poglavlje bavi se suvremenim trendovima i inovacijama u području kriptografije. Razmatrane su kvantna kriptografija, koja koristi principe kvantne mehanike za osiguranje komunikacije, kaotična kriptografija, koja se oslanja na teoriju kaosa za generiranje sigurnih ključeva, te blockchain kriptografija, koja osigurava siguran i transparentan način vođenja digitalnih transakcija. Poseban naglasak stavljen je i na DNA kriptografiju, koja koristi biološke principe za šifriranje podataka. Glavna literatura korištena u ovom poglavlju je [15].

Drugo poglavlje, Lightweight kriptografija, fokusira se na ulogu kriptografije u kontekstu Interneta stvari (IoT). Razmatraju se specifični izazovi kriptografije u IoT okruženju, gdje su uređaji često ograničeni po pitanju memorije, procesorske snage i energetske učinkovitosti. U ovom poglavlju korištena je literatura [24].

Treće poglavlje, Kriptografski algoritmi za uređaje lakih resursa, detaljno obrađuje specifične algoritme koji su razvijeni za rad na uređajima s ograničenim resursima. Analizirani su algoritmi Present, Simon, Speck, Twine, Prince i Rectangle, pri čemu su dani njihovi pseudokodovi. Glavna literatura korištena u ovom poglavlju uključuje izvore [6, 27, 23, 5, 2]. Cilj ovog rada je opisati kriptografske algoritme obrađene u trećem poglavlju, analizirati njihove performanse i sigurnosne karakteristike te istražiti njihovu primjenjivost u okruženjima s ograničenim resursima.

Diplomski rad napravljen je u sklopu aktivnosti Projekta PK.1.1.02.0004 - Znanstveni centar izvrsnosti za kvantne i kompleksne sustave te reprezentacije Liejevih algebri.

# Poglavlje 1

# Novi horizonti u kriptografiji

Ovo poglavlje opisuje nove i napredne pristupe u oblasti kriptografije, koji se razvijaju u skladu s modernim izazovima i tehnologijama. Razmatraju se različite oblasti kao što su kvantna kriptografija, kaotična kriptografija, lightweight kriptografija, blockchain kriptografija i DNA kriptografija. Svaka od ovih oblasti donosi inovativne metode zaštite podataka, koje ne samo da proširuju granice tradicionalnih kriptografskih tehnika, već i nude nove mogućnosti u osiguravanju privatnosti i sigurnosti u digitalnom okruženju. Više o temi može se pročitati i u [15].

## 1.1 Kvantna kriptografija

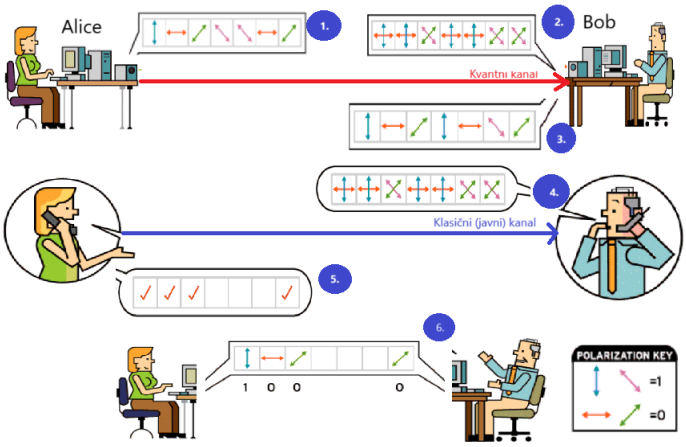
Kvantna računala predstavljaju novu vrstu računala koja mogu riješiti određene probleme, složene za klasična računala (npr. eksponencijalno zahtjevne probleme), u mnogo kraćem vremenu, ponekad čak i s linearnom vremenskom složenošću. Naime, s napretkom tehnologije klasična kriptografija postati će vrlo nesigurna. Danas RSA algoritam nije moguće razbiti. Međutim, teoretski je ranjiv ako se otkrije brz algoritam za faktorizaciju brojeva koji su umnožak dva ili više prostih brojeva. Stoga je kvantno računanje prijetnja za RSA enkripciju. Dva osnovna algoritma kvantne kriptografije su Shorov algoritam i Groverov algoritam. Riječ kvantna sama po sebi odnosi se na najosnovnija svojstva najmanjih čestica materije i energije. Kvantna kriptografija razlikuje se od tradicionalnih kriptografskih sustava po tome što se više oslanja na fiziku, a ne na matematiku. Kvantna kriptografija počiva na dva stupa kvantne mehanike:

- Heisenbergovo načelo neodređenosti: tvrdi da je nemoguće istovremeno znati točan položaj i moment čestice. Budući da nije moguće mjeriti kvantno stanje bilo kojeg sustava bez ometanja tog sustava, polarizacija fotona ili svjetlosne čestice može se znati samo u trenutku kada se mjeri.

- Kvantna superpozicija: Foton može biti u više stanja (usmjereni ili polarizirani u specifične smjerove) dok se ne mjeri, a mjerenje ga „kolapsira” u jedno od mogućih stanja. Štoviše, fotonski filter s ispravnom polarizacijom može detektirati samo polarizirani foton, inače će foton biti uništen.

U kvantnoj kriptografiji, informacije se prenose kvantnim bitom, također poznatim kao qubit, koji je zapravo pojedinačna čestica fotona. Običan bit je tranzistor koji registrira ili visoki ili niski napon, što odgovara 1 ili 0, respektivno. Mnoge stvari mogu se koristiti kao qubitovi, poput horizontalne i vertikalne polarizacije fotona ili spina prema gore ili prema dolje elektrona. C.H.Bennett (IBM) i G. Brassard (U.Montreal) objavili su 1984. godine prvi bezuvjetno siguran protokol za generiranje tajnog ključa između dvije strane koje ne dijele nikakovu prethodnu tajnu, baziran na zakonima kvantne fizike. Taj je protokol poznat kao BB84. Dvije strane vezane su s dva kanala: kvantnim kanalom (kojim se prenose fotoni) i klasičnim (javnim) kanalom (kojim se prenose poruke).

Kvantni kanal najčešće se realizira pomoću fotona dobro određene polarizacije. Odsutnost šuma u određenoj situaciji podrazumijeva da se kvantno stanje čestica ne mijenja duž kvantnog kanala.




Slika 1.1: BB84 [22]

Suština BB84 protokola opisana je na slici 1.1. Jedan od korisnika (Alice) nasumično odabere niz bitova i niz osnova, a zatim pošalje korisniku (Bob) niz fotona (1), pri čemu za svaki bit Alice određuje kako će biti predstavljen polarizacijom fotona, prema odabranoj bazi. Dakle svaki foton nosi informaciju o bitu, kodiranu u polarizaciji, koristeći bazu koju je Alice odabrala za taj bit. Prilikom primanja fotona, Bob nasumično odabire jedan od dva tipa detektora, jedan detektira fotone s horizontalnom/vertikalnom polarizacijom, a

drugi one polarizirane pod +/- 45 stupnjeva za svaki foton (2) i neovisno o Alice analogno interpretira rezultat svog mjerenja za svaki foton na dva načina, kao nulu ili jedan. Njegova polarizacija se pretvara u horizontalnu ili vertikalnu liniju i obrnuto, s nasumičnim rezultatima (3). Na taj način, Bob dobiva rezultate koji se podudaraju sa stanjem fotona koji su poslani u otprilike polovici slučajeva (50%), točnije kada točno pogodi osnovu. No zakoni kvantne mehanike dopuštaju da i fotoni koji ne odgovaraju orijentaciji detektora svejedno mogu biti detektirani kao oni koji odgovaraju, zato Bob ne zna koje je bitove primio ispravno. Sljedeća faza protokola ostvaruje se putem javnog kanala (4), kroz koji Alice i Bob mogu međusobno otvoreno prenositi klasične informacije. Za početak, Alice i Bob određuju (putem javnog kanala) koji su fotoni uspješno primljeni od strane Boba i koji su od njih mjereni u ispravnoj osnovi (5). Nakon toga, Alice i Bob imaju iste bitne vrijednosti kodirane u tim fotonima (6), bez obzira na činjenicu da ove informacije nikada nisu bile uspostavljene u otvorenom komunikacijskom kanalu. Kvantna kriptografija mogla bi biti praktična, osim možda za vrlo ograničene uređaje Interneta stvari (IoT) i radio uređaje. Također neki od nedostataka su: skupo (potrebna je specijalizirana oprema), kompleksno, teško za implementaciju na velikoj udaljenosti, kvantni algoritmi su uglavnom probabilistički: To znači da kvantno računalo u jednoj operaciji vraća mnogo rješenja, pri čemu je samo jedno ispravno.

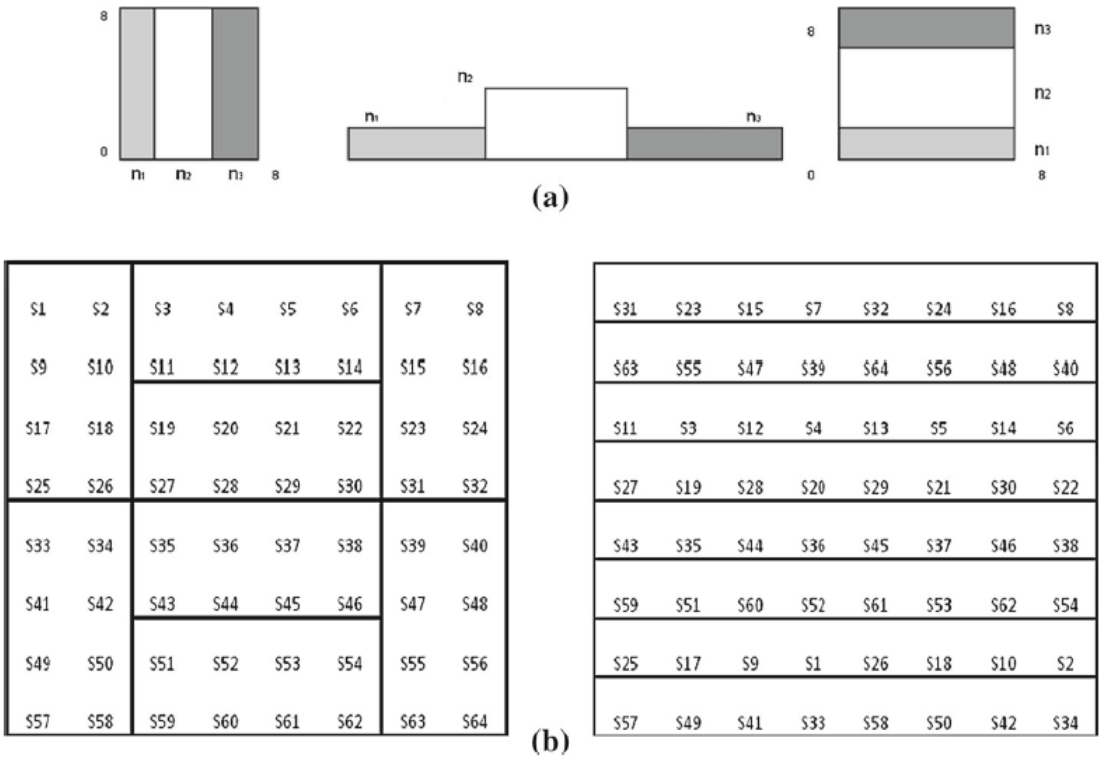
## 1.2 Kaotična kriptografija

Mnogi snažni kriptosustavi, poput DES-a, AES-a i RSA-a, koriste se u raznim sustavima i industrijama za zaštitu podataka, šifriranje komunikacija i osiguranje digitalnih transakcija, čime su postali neizostavni dio moderne tehnologije. Međutim, većina njih ne može se izravno koristiti za šifriranje ugrađenih sustava u stvarnom vremenu jer njihova brzina šifriranja nije dovoljno brza i jer zahtijevaju veliku računalnu snagu. Stoga u ovom radu predstavljamo brzi algoritam za šifriranje temeljen na kaotičnim sustavima, koji je prikladan za ugrađene sustave u stvarnom vremenu s obzirom na izvedbu, prostornu i energetsku učinkovitost. Postoji mnogo shema izmjenjivanja podataka (Chaotic Interleaving schemes) koje su predložene kako bi se smanjile pogreške tijekom prijenosa podataka. 2D kaotična Bakerova mapa u svojoj diskretiziranoj verziji dobar je kandidat za ovu svrhu. Kao referencu navedenog šifriranja navodimo sljedeće izvore [1, 8, 9]. U predloženom sustavu modela uzorci signala mogu se rasporediti u kvadratnu matricu, a zatim nasumično rasporediti pomoću kaotične Bakerove mape. Kaotično izmjenjivanje N $\times$ N kvadratne matrice može se izvoditi na sljedeći način kao Algoritam 1.

Slika 1.2 prikazuje primjer kaotičnog šifriranja kvadratne matrice dimenzija 8$\times$8. Tajni ključ $S_{key}(n_1, n_2, n_3)$=(2,4,2).

**Algoritam 1** Chaotic Interleaving Algorithm

1) Kvadratna matrica dimenzija $N \times N$ podijeljena je na k vertikalnih pravokutnika visine N i širine $n_i$, tako da vrijedi $n_1 + n_2 + \cdots + n_k = N$.
2) Ti vertikalni pravokutnici rastežu se u horizontalnom smjeru i skupljaju vertikalno kako bi se dobili horizontalni pravokutnici dimenzija $n_i \times N$.
3) Horizontalni pravokutnici slažu se tako da se lijevi pravokutnik stavlja na dno, a desni na vrh.
4) Svaki vertikalni pravokutnik dimenzija $n_i \times N$ podijeljen je na $n_i$ okvira dimenzija $\frac{N}{n_i} \times n_i$, pri čemu svaki okvir sadrži točno N točaka.
5) Svaki od tih okvira mapira se stupac po stupac u jedan redak podataka. Unutar svakog pravokutnika, skeniranje započinje iz donjeg lijevog kuta prema gornjim elementima.




Slika 1.2: Kaotično izmjenjivaje (a) Diskretizirana Bakerova mapa.
(b) Stvaranje kaotične mape $2 \times 4 \times 2$ i šifrirana matrica [1].

## 1.3 Lightweight kriptografija - „Laka” kriptografija

Laka kriptografija je grana kriptografije koje se bavi razvojem kriptografskih algoritama za uređaje s ograničenim resursima poput mobilnih telefona, senzora ili IoT uređaja, zahtijevajući minimalne resurse. Tradicionalni kriptografski algoritmi često balansiraju sigurnost, performanse i troškove, što ih čini neprikladnima za takve uređaje. Laki algoritmi, poput L-AES, Simon, Speck i HMAC, optimizirani su za učinkovitost, smanjujući potrošnju energije i poboljšavajući performanse. Ovi algoritmi koriste jednostavne operacije i manje složene matematičke funkcije, što ih čini prikladnim za uređaje s niskim kapacitetima. S porastom IoT uređaja, potrebni su sigurni i niskobudžetni hardverski sustavi, jer IoT povezuje objekte i olakšava svakodnevni život. Primjeri nekih IoT uređaja su prikazani na slici 1.3. Njih je potrebno zaštiti jer su to uređaji koji se povezuju na internet te su tako nezaštićeni od napada.


Slika 1.3: Pametni sat, pametna sigurnosna kamera, pametni termostat

Glavni cilj lake kriptografije je osigurati sigurnost podataka bez velikih zahtjeva za procesorsku snagu, memoriju i energiju. Laka kriptografija obuhvaća i prilagodbu postojećih algoritama, kao i dizajn novih, specifično za takve uređaje. Osigurava osnovne funkcionalnosti poput šifriranja, autentifikacije i integriteta podataka. Laka kriptografija postaje ključna za aplikacije u mobilnim mrežama, IoT-u, pametnim uređajima, senzorima i industriji. U obzir se uzimaju i specifičnosti uređaja poput brzine procesora i kapaciteta memorije. Razvijaju se i standardi koji omogućuju sigurno i učinkovito korištenje u uvjetima s ograničenim resursima. Više o temi u Poglavlju 2 gdje ćemo detaljnije objasniti motivaciju za uvođenje lake kriptografije te opisati uređaje koji koriste algoritme lake kriptografije.

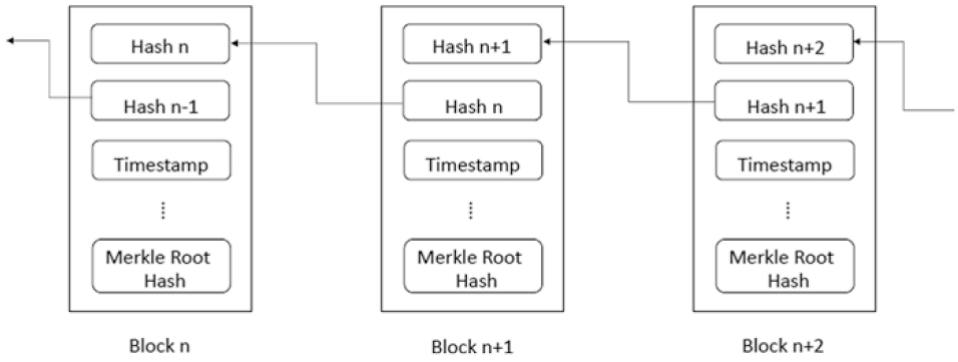
## 1.4 Blockchain kriptografija

Blockchain (hrv. lanac blokova) je distribuirana baza podataka koja omogućuje izravne transakcije između dviju strana bez potrebe za autoritativnim posrednikom. Blockchain tehnologija ima zanimljivu povijest koja seže sve do 1982. godine, kada je David Chaum u svojoj je disertaciji „Uspostavljeni i održavani računalni sustavi kojima vjeruju međusobno nepovjerljive grupe" prvi put opisao temelje decentralizirane baze podataka. Iako nije bio usmjeren na digitalne valute, njegov rad postavio je temelje za buduće inovacije. Chaum je 1989. pokrenuo tvrtku pod nazivom DigiCash, a tvrtka je predstavila kriptovalutu različitih imena – digicash, eCash i cyberbucks [12]. DigiCash je obećao mnoge značajke modernih kriptovaluta, uključujući anonimnost i šifrirane prijenose koje ni vlada nije mogla dešifrirati. Međutim, Chaum nije uspio uvjeriti banke da podrže projekt, a bez odgovarajuće internetske infrastrukture za peer-to-peer transakcije, DigiCash je propao i tvrtka je bankrotirala. Prvi pravi proboj dogodio se 2008. godine, kada je Satoshi Nakamoto predstavio Bitcoin i njegov blockchain, revolucionirajući način na koji mislimo o novcu i sigurnosti podataka. Nekoliko godina kasnije, Ethereum je proširio mogućnosti blockchaina, omogućujući razvoj decentraliziranih aplikacija. Danas blockchain nije samo za kriptovalute njegova primjena raste u industrijama poput zdravstva, upravljanja identitetom i lanaca opskrbe, a istraživači širom svijeta nastavljaju otkrivati nove mogućnosti ove tehnologije. Blockchain se sastoji od niza blokova koji se pohranjuju i kopiraju na javno dostupnim serverima, čime se osigurava sigurnost i transparentnost podataka. Svaki blok sastoji se od četiri osnovna elementa:

- Hash blok
- Hash prethodnog bloka
- Sadržaj podataka bloka
- Nonce koji se koristi za oblikovanje hasha.

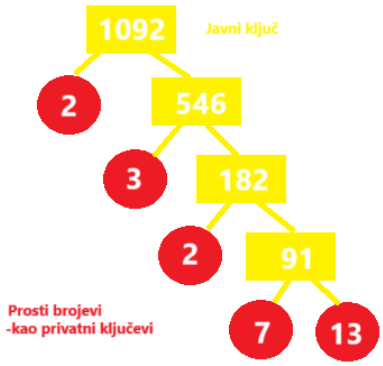
„Nonce" je skraćenica za izraz „number used once" (broj koji se koristi samo jednom). Nasumični broj koji se koristi u procesu rudarenja za pronalaženje valjanog hasha za blok. Blockchain i njegovi elementi prikazani su na slici 1.4.

Uključivanjem hasha prethodnog bloka, svaki blok potvrđuje autentičnost prethodnog. Hash svakog bloka je jedinstven i identificira sadržaj bloka, pa svaka promjena unutar bloka mijenja njegov hash. Blokovi na početku lanca ne mogu se mijenjati bez promjene svih sljedećih blokova, što uzrokuje nesklad u hash-evima. Slično tome, dodavanje podataka u blok i njihovo hashiranje čini podatke nepromjenjivima, jer bilo kakva izmjena podataka mijenja njihov hash, što odmah narušava integritet cijelog lanca blokova i osigurava konzistentnost sekvence blokova. Ako napadač promijeni podatke u bloku, hash bloka se mijenja, a sljedeći blokovi postaju nevažeći jer sadrže stari hash. U blockchain kriptografiji koristi se matematički princip faktorizacije na proste faktore, gdje je lako generirati




Slika 1.4: Osnovna struktura blockchaina

broj množenjem prostih brojeva, ali je teško obrnuti proces i otkriti koji su prosti brojevi korišteni. Ovaj proces je poznat kao faktorizacija prostih brojeva, a javni i privatni ključ povezani su putem ove operacije. (Slika 1.5).




Slika 1.5: Faktorizacija

Prosti faktori javnog ključa čine privatni ključ. Kriptografija blockchaina oslanja se na funkcije s trapdoorom, koje su matematički problemi koji se lako rješavaju u jednom smjeru, ali je gotovo nemoguće obrnuti ih. Ova karakteristika omogućuje sigurnost i autentifikaciju jer funkcije s trapdoorom otežavaju krivotvorenje podataka. U RSA kriptografiji, jednosmjerna funkcija temelji se na faktorizaciji velikih brojeva. Pomnožiti dva velika broja je jednostavno, no gotovo je nemoguće vratiti se na ta dva broja samo iz njihovog produkta, osim ako se ne posjeduje specifičan tajni podatak – trapdoor. U ovom slučaju,

trapdoor je privatni ključ, koji omogućuje dešifriranje podataka ili vraćanje izvornog broja pomoću specifičnih matematičkih svojstava, dok je za svakog drugog korisnika gotovo nemoguće obaviti taj postupak. Ukratko, trapdoor omogućuje da se jednosmjerne funkcije koriste za sigurno šifriranje i autentifikaciju podataka. On omogućuje dešifriranje podataka (ili obrnuti postupak) samo onima koji posjeduju tajni ključ, dok svima koji ga nemaju, bez tog ključa, ostaje gotovo nemoguće razbiti šifru i rekonstruirati izvorne informacije.

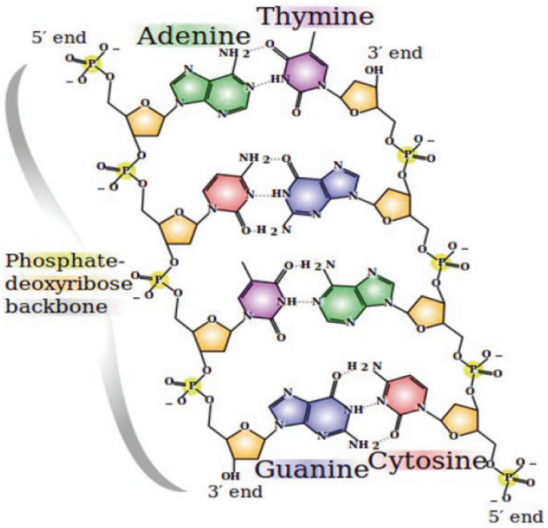
U tablici 1.1 možemo vidjeti prednosti i mane ove kriptografije.

| **Prednosti** | **Izazovi** |
|---|---|
| **Otpornost:** Lanac i dalje upravlja većinom čvorova u slučaju masovnog napada na sustav. | **Viši troškovi:** Čvorovi traže veće nagrade za završavanje transakcija u poslovanjima koja rade prema principu ponude i potražnje. |
| **Smanjenje vremena:** blockchain može igrati ključnu ulogu omogućujući brzinu jer ne zahtijeva dugotrajan proces verifikacije, poravnanja i obrade. | **Sporije transakcije:** Čvorovi daju prioritet transakcijama s višim nagradama, što dovodi do stvaranja zaostataka u transakcijama. |
| **Pouzdanost:** Blockchain potvrđuje i verificira identitete zainteresiranih strana pa to uklanja duple zapise. | **Manje registara:** Nije moguće imati punu kopiju blockchaina, što potencijalno može utjecati na nepromjenjivost, konsenzus i druge aspekte. |
| **Nepromjenjive transakcije:** Registriranjem transakcija u kronološkom redu, blockchain jamči nepromjenjivost svih operacija, što znači da kada se novi blok doda u lanac registara, ne može biti uklonjen ili izmijenjen. | **Troškovi transakcija i brzina mreže:** Trošak transakcija u Bitcoin-u postao je prilično visok nakon što je bio promoviran kao „skoro besplatan” tijekom prvih nekoliko godina. |
| **Prevencija prijevara:** Koncepti zajedničkih informacija i konsenzusa sprječavaju moguće gubitke zbog prijevara ili malverzacija. Blockchain kao mehanizam praćenja djeluje na smanjenje troškova. | **Rizik od pogreške:** Uvijek postoji rizik od pogreške dokle god je ljudski faktor uključen. Ako blockchain služi kao baza podataka, svi dolazni podaci moraju biti visoke kvalitete. |

Tablica 1.1: Prednosti i mane korištenja Blockchain kriptografije

## 1.5 DNA kriptografija

DNA kriptografija spaja kriptologiju i modernu biotehnologiju te donosi nadu u osnivanje algoritama koji su otporni na napade. DNA, odnosno deoksiribonukleinska kiselina, sastoji se od četiri osnovne nukleinske kiseline: adenin (A), citozin (C), gvanin (G) i timin (T), pri čemu se parovi (A, T) i (C, G) međusobno dopunjuju. Ovi simboli mogu se lako povezati s binarnim vrijednostima (A-00, C-01, G-10, T-11). Prema ovom kodiranju postoje 24 moguće metode kodiranja, ali samo osam poštuje princip komplementarnosti, jer su binarni brojevi ”0” i ”1” te parovi ”00” i ”11” i ”01” i ”10” komplementarni. Slika 1.6 prikazuje osnove DNA bloka te se na njoj vide parovi nuklenskih kiselina.




Slika 1.6: Osnove DNA bloka [15]

Autori Bonny B. Raj, J. Frank Vijay i T. Mahalakshmi su predložili jednostavnu i osnovnu substitucijsku šifru [18] koja je inspirirana strukturom i sekvencom DNA. Za šifriranje pomoću DNA, pošiljatelj stvara DNA tablicu za kodiranje, dok primatelj generira svoju tablicu koristeći istu tehniku te šalje trag pošiljatelju za lokalnu rekonstrukciju. Tekst se dijeli na dva jednaka dijela, a ako nije paran, koristi se nasumično popunjavanje dodavanjem znaka. Jedan dio pretvara se u DNA sekvencu pomoću tablice pošiljatelja, a drugi pomoću tablice primatelja. Šifriranje pomoću DNA provodi se sljedećim koracima:

- Pretvorba običnog teksta u ASCII format, a zatim u binarni kod od 8 bitova Slika 1.7

- Kodiranje binarnih podataka u DNA formu (A-00, C-01, G-10, T-11) kao DNA sekvencu
- Primjena pravila komplementarnosti ($A \rightarrow C$, $C \rightarrow G$, $G \rightarrow T$, $T \rightarrow A$)
- Generiranje nasumičnog ključa (broj između 1 i 256), pretvorba u DNA sekvencu i binarni format. Ključ određuje permutaciju A, T, G i C. Ako je nasumični ključ 1, postoji tablica za pretvorbu ASCII koda u nukleotidne sekvence, a ako je 2, koristi se druga tablica i tako dalje.
- XOR operacija između ključa i podataka

| S | DNAseq | S | DNAseq | S | DNAseq | S | DNAseq | S | DNAseq | S | DNAseq |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ' ' | ACAT | 0 | CAAA | @ | CCAC | P | TTCA | ‘ | TCCG | p | GACA |
| ! | AGGT | 1 | CACC | A | TACT | Q | TTTA | a | GAGC | q | GACT |
| " | AAAG | 2 | CCGT | B | TCCT | R | TAGA | b | GTGC | r | GGAT |
| # | AGAC | 3 | CGAG | C | TACG | S | TGAG | c | GACG | s | GGTG |
| $ | AAGC | 4 | CCTT | D | TGCC | T | TAAA | d | GTAA | t | GCTT |
| % | AACT | 5 | CCGT | E | TCTA | U | TGAC | e | GTAC | u | GACC |
| & | AGAA | 6 | CTGT | F | TAGT | V | TGAG | f | GCCT | v | GACT |
| ' | AATC | 7 | CTCT | G | TTAA | W | TAAC | g | GCTA | w | GCCC |
| ( | ATTG | 8 | CCGT | H | TGGC | X | TCCT | h | GAGT | x | GATC |
| ) | AATT | 9 | CTCA | I | TGTT | Y | TGAA | i | GATG | y | GTCG |
| * | AATG | : | CTAG | J | TTCC | Z | TAAG | j | GATT | z | GTGA |
| + | AAGA | ; | CCGC | K | TACT | [ | TCAT | k | GGGC | { | GGCT |
| , | AGAG | < | CACA | L | TATG | \\ | TAAG | l | GTTG | \| | GGTG |
| - | AAGC | = | CATA | M | TAGT | ] | TCCA | m | GTGA | } | GAAC |
| . | ACAC | > | CTAC | N | TGTC | ^ | TGTT | n | GACT | ~ | GATG |
| / | ACGT | ? | CCAG | O | TATT | _ | TCCG | o | GCCG | \x7f | GAGT |

Slika 1.7: ASCII simboli i njihove odgovarajuće DNA sekvence

DNA kriptografija ima velik potencijal za široku primjenu u različitim područjima poput mobilnih mreža, cloud računalstva, IoT uređaja, aplikacija u stvarnom vremenu, Interneta i multicast aplikacija za zaštitu običnih poruka, slika, videa i poslužitelja. Prednosti DNA računala u odnosu na tradicionalna računala su sljedeće:

- Brzina: Tradicionalna računala izvršavaju oko $10^8$ instrukcija u sekundi, dok kombiniranje DNA lanaca može omogućiti računanja ekvivalentna $10^9$ instrukcija
- Pohrana: DNA pohranjuje memoriju brzinom od $1bit/nm^3$, dok konvencionalni mediji pohranjuju $1bit/10^{12}\ nm^3$

- Potrebe za energijom: DNA računalo ne zahtijeva energiju tijekom izračuna, jer kemijske reakcije koje stvaraju gradivne blokove DNA ne koriste vanjski izvor energije.

Detaljnije o temi može se pročitati u [11].

# Poglavlje 2

# Lightweigth kriptografija

Internet stvari (IoT) skraćeno od Internet of Things, označava mrežu fizičkih uređaja, vozila, kućanskih aparata i drugih predmeta opremljenih senzorima, softverom i povezivošću koji im omogućuju međusobnu razmjenu podataka preko interneta. IoT uređaji mogu se svrstati u dvije glavne skupine: uređaje s bogatim resursima te uređaje s ograničenim resursima. Upravo ova druga kategorija, koja često ima ograničenja u procesorskoj snazi, memoriji i energetskoj potrošnji, predstavlja izazov za osiguranje podataka. Kako bi se zadovoljile potrebe ovih resursno ograničenih uređaja, razvijena je laka kriptografija (eng. lightweight cryptography). Ova specijalizirana grana kriptografije fokusira se na dizajn algoritama koji nude visoku razinu sigurnosti uz minimalnu potrošnju resursa. U ovom poglavlju uvodimo pojam IoT te opisujemo kriptografiju za takve uređaje ograničenih resursa.

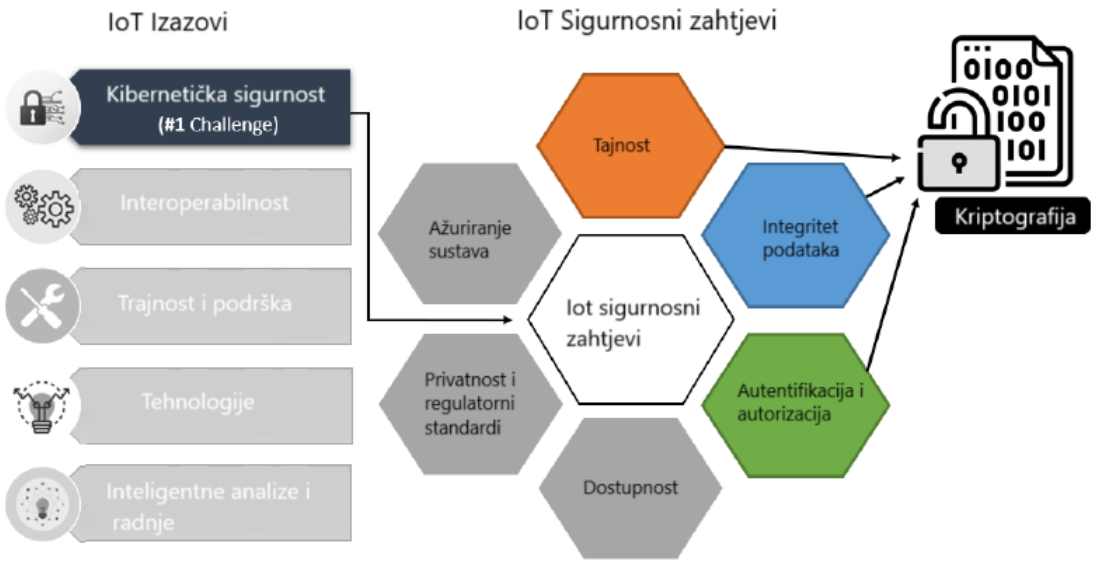
## 2.1 IoT - Internet stvari

Internet stvari (IoT) jedno je od ključnih područja suvremenog istraživanja zahvaljujući svojoj širokoj primjeni u raznim sektorima, uključujući pametni transport i logistiku, pametno zdravstvo, zaštitu okoliša, pametnu infrastrukturu (poput pametnih gradova, domova, ureda i trgovačkih centara), pametnu poljoprivredu i mnoge druge sfere. IoT uređaji pružaju revolucionarne mogućnosti integracijom fizičkog svijeta s digitalnim, omogućujući učinkovito prikupljanje podataka i automatizirano upravljanje sustavima. Ovi uređaji dijele se u sljedeće dvije glavne kategorije [24], a primjere uređaja vidimo na slici 2.1:

- uređaje s bogatim resursima: poslužitelji, osobna računala, tableti i pametni telefoni...
- uređaje s ograničenim resursima: industrijski senzori, senzorski čvorovi, RFID oznake i aktuatori...


Slika 2.1: Klasifikacija IoT uređaja [24]

Upravo zbog ograničenih kapaciteta uređaja u drugoj skupini, poput ograničene procesorske snage, memorije i energetske autonomije, osiguravanje njihove sigurnosti predstavlja poseban izazov. Slika 2.2 prikazuje izazove i zahtjeve IoT uređaja.

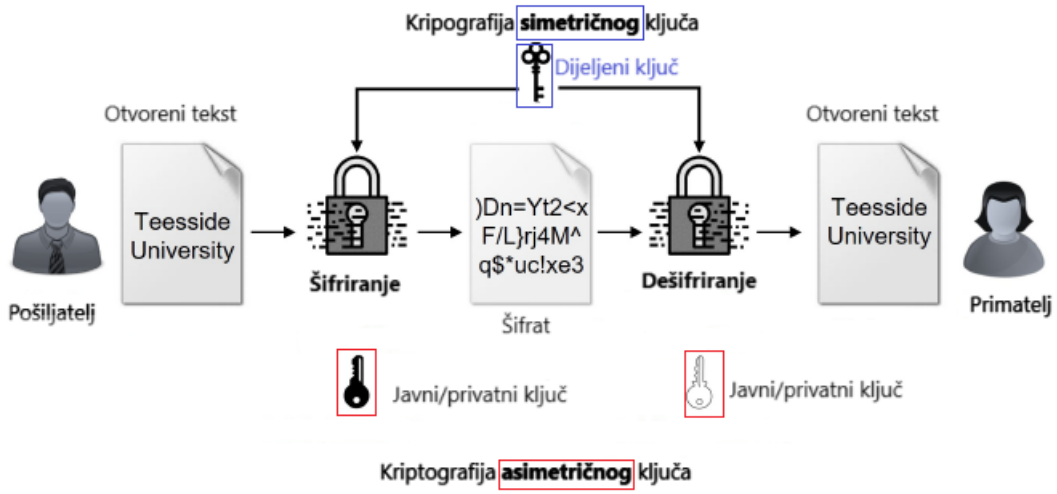



Slika 2.2: IoT sigurnosni zahtjevi [24]

IoT uređaji su lako dostupni i izravno komuniciraju s fizičkim svijetom kako bi prikupljali povjerljive podatke ili upravljali varijablama fizičkog okruženja. Ova povezanost s okolinom i razmjena podataka čine ih ranjivima na različite vrste kibernetičkih prijetnji. Posljedice kompromitacije IoT sustava mogu biti ozbiljne, od krađe osjetljivih podataka do ugrožavanja fizičke sigurnosti sustava. Kako bi se osigurala pouzdanost i sigurnost IoT ekosustava, ključni izazovi uključuju zaštitu povjerljivosti i integriteta podataka, implementaciju robusnih mehanizama autentifikacije i autorizacije, održavanje dostupnosti sustava, očuvanje privatnosti korisnika, usklađenost s regulatornim standardima i omogućavanje redovitih sigurnosnih ažuriranja. Rješavanje ovih izazova od presudne je važnosti za širu primjenu IoT tehnologije i povjerenje korisnika.

## 2.2 Kriptografija za IoT

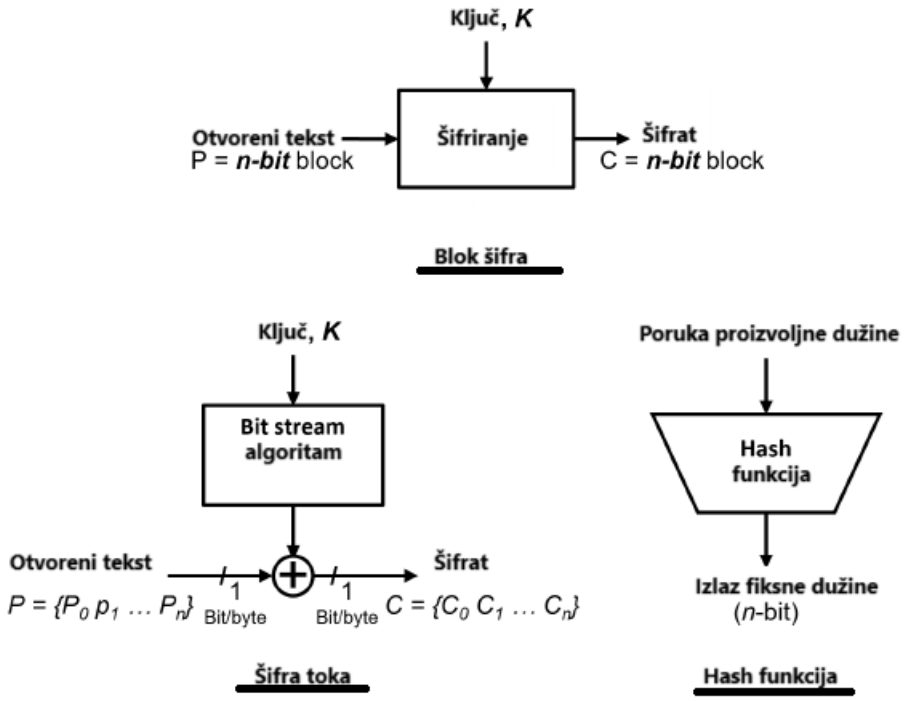
Laka kriptografija (LWC) je lakša verzija kriptografije koja osigurava sigurnost komunikacije u okruženju s ograničenim resursima, posebno za IoT uređaje s ograničenim resursima, s vrlo malim zahtjevima kao što su mala memorija, niska računalna snaga, mala fizička površina i niska baterijska snaga. Dvije glavne vrste algoritama za kriptografiju su simetrična kriptografija i asimetrična kriptografija. Na slici 2.3 vidimo tipove kriptografije s obzirom na vrstu ključa prilikom šifriranja i dešifriranja. Simetrična kriptografija koristi




Slika 2.3: Glavni tipovi kriptografije [24]

jedan ključ za šifriranje i dešifriranje podataka, dok asimetrična kriptografija koristi dva različita ključa za šifriranje i dešifriranje podataka. Simetrična kriptografija je sigurna i

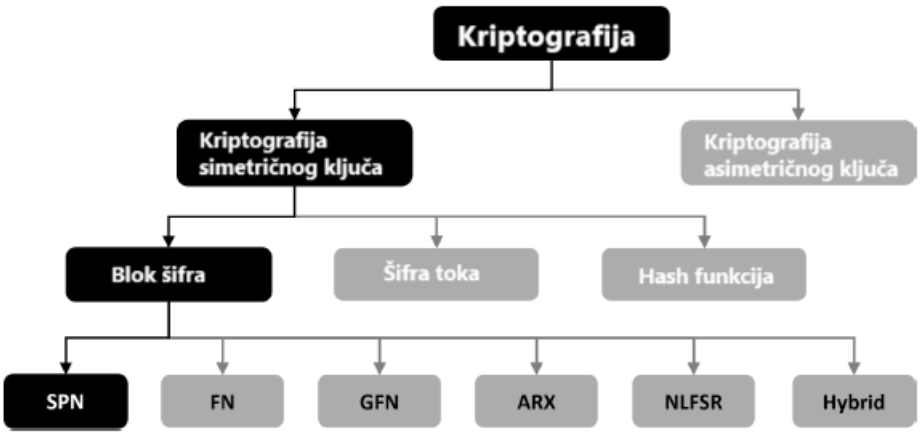
relativno brza, a jedini nedostatak simetričnog šifriranja je dijeljenje ključa između komunikacijskih strana bez ugrožavanja njegove sigurnosti. Nedostatak asimetričnog šifriranja je veliki ključ, što povećava složenost i usporava proces. Zbog gore navedenih razloga, simetrična kriptografija najbolje odgovara IoT uređajima u okruženju s ograničenim resursima. Simetrična kriptografija ključa može se dalje klasificirati u tri kategorije, ovisno o načinu na koji funkcionira: blok šifra, protočna šifra i hash funkcije (Slika 2.4).




Slika 2.4: Blok šifra, šifra toka i hash funkcija

Kriptografija se oslanja na dva ključna svojstva: konfuziju i difuziju, koja su uvedena kako bi se ojačala šifra. Iako šifra toka koristi samo konfuziju, dok blokovska šifra koristi oba svojstva, blokovska šifra u usporedbi sa šifrom toka ima puno jednostavniji dizajn. Osim toga, proces vraćanja izvornog teksta u blokovskoj šifri je složeniji. Šifra toka koristi XOR funkciju koja omogućuje lako vraćanje podataka u izvorni oblik. S druge strane, hash funkcija je jednsmjerna i transformira podatke proizvoljne dužine u kratki bitni niz fiksne dužine. Iako se hash funkcije široko koriste za pohranu lozinki i provjeru integriteta podataka, nisu prikladne za IoT aplikacije gdje je povratak izvornog poruke ključan.

Iz tog razloga, blokovska šifra, koja nudi veću sigurnost i omogućuje vraćanje izvornog teksta, preferira se u IoT uređajima s ograničenim resursima u odnosu na šifre toka i hash funkcije. Ovisno o unutarnjoj strukturi (funkcijama) koje se koriste za stvaranje šifre, blokovska šifra se klasificira u šest sljedećih vrsta. Slika 2.5 prikazuje kompletnu klasifikaciju kriptografije.




Slika 2.5: Glavni tipovi kriptografije

- Substitucijsko-permutacijska mreža (SPN) - eng. Substitution-Permutation Network: mijenja podatke pomoću skupa substitucijskih kutija i permutacijskih tablica te ih priprema za sljedeći krug.
- Feistelova mreža (FN) - eng. Feistel Network: dijeli ulazni blok na jednake polovice i primjenjuje difuziju u svakom krugu samo na jednu polovicu. Osim toga, na početku svakog kruga dolazi do zamjene dviju polovica.
- Opća Feistelova mreža (GFN) - eng. General Feistel Network: proširena je verzija klasične Feistelove mreže. Ona dijeli ulazni blok na više podblokova i primjenjuje Feistelove funkcije na svaki par podblokova, nakon čega slijedi cikličko pomicanje proporcionalno broju podblokova.
- Zbroj-rotacija-XOR (ARX) - eng. Add-Rotate-XOR: obavlja šifriranje-dešifriranje koristeći operacije zbrajanja, rotacije i XOR bez upotrebe S-kutija. Implementacija ARX-a je brza i kompaktna, ali ograničena u sigurnosnim svojstvima u usporedbi sa SPN i Feistelovim šiframa.

- Nelinearni pomični registarski povratni sustav (NLFSR) - eng. NonLinear-Feedback Shift Register: primjenjuje se na šifre toka i blokovne šifre, Koristi gradivne blokove šifre toka čije trenutno stanje proizlazi iz njihovog posljednjeg stanja, koje je nelinearna povratna vrijednost.

- Hibrid - eng. Hybrid: Hibridna šifra kombinira bilo koja tri tipa (SPN, FN, GFN, ARX, NLFSR) ili čak miješa svojstva blokovnih i šifri toka kako bi poboljšala specifična svojstva ovisno o zahtjevima primjene.

Slika 2.6 prikazuje koje strukture su određeni algoritmi lake kriptografije. Algoritme koji su označeni detaljnije ćemo opisati u poglavlju 3.

| Vrsta strukture | Algoritmi |
|---|---|
| SPN | AES, Present, GIFT, SKINNY, Rectangle, Midori, mCrypton, Noekeon, Iceberg, Puffin-2, Prince, Pride, Print, Klein, Led, Picaro, Zorro, EPCBC, I-Present, ASCON, SHAMASH, PRIMATE, ICEPOLE |
| FN | DESL/DESXL, TEA/XTEA/XXTEA, Camellia, Simon, SEA, KASUMI, MIBS, LBlock, ITUbee, FeW, GOST, Robin, Fantomas |
| GFN | CLEFIA, Piccolo, Twis, Twine, HISEC |
| ARX | Speck, IDEA, HIGHT, BEST-1, LEA |
| NLFSR | KeeLoq, KATAN/KTANTAN, Halka |
| Hybrid | Hummingbird, Hummingbird-2, Present-GRP |

Slika 2.6: Struktura lakih algoritama [24]

Od ovih struktura, SPN i FN su najpopularniji izbor zbog svoje fleksibilnosti u implementaciji, ovisno o zahtjevima aplikacije. Iako se Feistelove strukture lako mogu integrirati u hardver s niskom prosječnom potrošnjom energije (zbog izostanka funkcije runde u jednoj polovici stanja), obično zahtijevaju više funkcija rundi u usporedbi s SPN strukturama iz sigurnosnih razloga. Stoga, kada postoji izbor između manjeg broja SPN funkcijskih rundi i većeg broja Feistelovih funkcijskih rundi, SPN funkcija bi mogla biti bolja. SPN pruža isti nivo sigurnosti uz slične ili niže troškove energije u odnosu na Feistelovu mrežu. Objašnjenje funkcija rundi i drugih pojmova vezanih uz strukture može se pronaći u poglavlju 3 gdje je svaka od struktura objašnjena na primjeru algoritma.

## 2.3 Laka kriptografija - lightweight kriptografija

Laka kriptografija usmjerena je na razvijanje algoritama koji, uz zadržavanje sigurnosti, minimiziraju vrijeme izvršavanja, korištenje memorije i potrošnju energije. Takvi su algoritmi prikladni za male ugrađene sustave, poput onih koji se široko koriste u Internetu stvari (IoT). Rad na lakoj kriptografiji gotovo je isključivo posvećen simetričnim (s tajnim ključem) algoritmima i kriptografskim hash funkcijama. Pojam ugrađeni sustav odnosi se na korištenje elektronike i softvera unutar proizvoda koji ima određenu funkciju ili skup funkcija, za razliku od računala opće namjene, poput prijenosnih ili stolnih sustava. Ugrađeni sustav možemo također definirati kao bilo koji uređaj koji sadrži računalni čip, ali nije radna stanica, stolno ili prijenosno računalo opće namjene. Danas mnogi, a možda i većina uređaja koji koriste električnu energiju, imaju ugrađeni računalni sustav. Neki od uređaja s ugrađenim sustavima su mobilni telefoni, digitalni fotoaparati, video kamere, kalkulatori, mikrovalne pećnice, sustavi za kućnu sigurnost, perilice rublja, rasvjetni sustavi, termostati, pisači, razni automobilski sustavi (npr. upravljanje prijenosom, ubrizgavanje goriva, sustavi protiv blokiranja kočnica), teniski reketi, četkice za zube te brojni tipovi senzora i aktuatora u automatiziranim sustavima. Na slici 2.7 je jedan mikrokontroler.


Slika 2.7: Mikrokontroler

Mikrokontroler je čip koji sadrži procesor, trajnu memoriju za program (ROM ili flash), hlapljivu memoriju za ulaz i izlaz (RAM), sat i upravljačku jedinicu za I/O. Naziva se i „računalo na čipu”. Obično se koristi kao namjenski procesor za specifične zadatke, poput automatizacije. Mikrokontroleri omogućuju upravljanje strojevima, uključivanje ventilatora, otvaranje ventila i slično. Iako su sporiji od mikroprocesora (rade u MHz rasponu naspram GHz brzina mikroprocesora), vrlo su jeftini i učinkoviti za izradu složenih uređaja. Programirani su za specifične zadatke i obavljaju ih prema potrebi. Ograničeni uređaji imaju ograničenu memoriju, procesorsku snagu i nisku brzinu prijenosa podataka. Mnogi IoT uređaji spadaju u ovu kategoriju, posebno oni manji i prisutni u većem broju. Tipično su opremljeni 8- ili 16-bitnim mikrokontrolerima s malim kapacitetom RAM-a i pohrane.

Tri klase ograničenih uređaja razlikuju se prema resursima i sposobnostima (Slika 2.8) [21]:

1. Klasa 0: Vrlo ograničeni uređaji, poput senzora (motes ili pametna prašina), koji prikupljaju podatke (npr. temperatura, vlažnost) i prosljeđuju ih do središnjeg čvora. Ovi uređaji nisu sigurni niti lako upravljivi, a rijetko se rekonfiguriraju.

2. Klasa 1: Ovi uređaji imaju ograničene resurse, ali mogu koristiti protokole dizajnirane za ograničene čvorove i komunicirati bez posredničkog čvora (gateway). Imaju osnovne komunikacijske sposobnosti. Obično uključuju manji broj perifernih uređaja i ograničenu povezanost (primjer: pametni termostat).

3. Klasa 2: Manje ograničeni uređaji koji mogu podržavati većinu protokola kao računala, ali zahtijevaju lake, energetski učinkovite protokole i nisku količinu prijenosa podataka zbog ograničenja u resursima. Mogu obavljati složenije zadatke i omogućiti veću interakciju s korisnikom, iako su i dalje ograničeni u odnosu na opća računala (primjer: pametna kamera).

| Class | Data Size (RAM) | Code Size (flash, ROM) |
|---|---|---|
| Class 0 | « 10 kB | « 100 kB |
| Class 1 | ~ 10 kB | ~ 100 kB |
| Class 2 | ~ 50 kB | ~ 250 kB |

Slika 2.8: Klase uređaja s ograničenim resursima [21]

Iako se tradicionalni algoritmi mogu lako implementirati u IoT uređaje s bogatim resursima (poput računala, poslužitelja, pametnih telefona itd.), oni nisu prikladni za IoT uređaje s ograničenim resursima. Dajemo ključne izazove pri implementaciji tradicionalne kriptografije u IoT uređaje s ograničenim resursima:

- Površina čipa: Kada se kriptografski algoritam implementira u hardveru, vrlo mali uređaji (poput senzora) imaju ograničen prostor na čipu za sigurnost. Površina čipa izražava se u ekvivalentima vrata (GEs), koji se računaju prema površini integriranog kruga i površini NAND vrata.

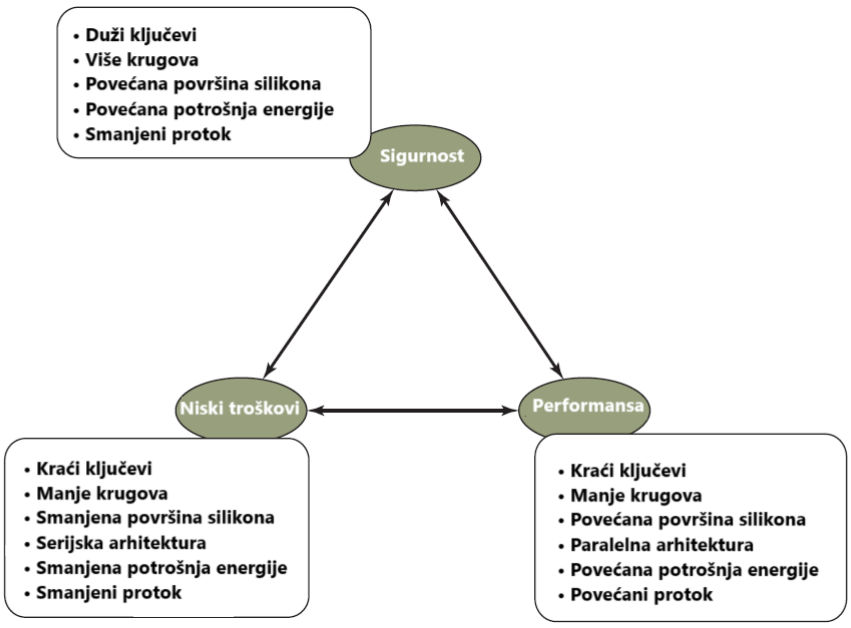
- Potrošnja energije: Mnogi ograničeni uređaji rade s malim baterijama ili od energije iz dolaznog signala. Algoritmi moraju biti dizajnirani za minimalnu potrošnju energije, uzimajući u obzir vrijeme obrade, površinu čipa, radnu frekvenciju i količinu podataka koja se prenosi (posebno u bežičnim prijenosima).

- Veličina programske memorije i RAM-a: Ovi uređaji imaju vrlo ograničen prostor za programski kod (ROM) i RAM za izvršavanje. Kriptografski algoritmi moraju biti kompaktni i koristiti minimalnu količinu RAM-a.

- Brzina prijenosa podataka: Ograničeni uređaji, poput senzora i RFID oznaka, imaju vrlo nisku brzinu prijenosa podataka. Stoga, količina podataka vezanih uz sigurnost, poput kodova za autentifikaciju poruka i razmjene ključeva, mora biti minimalna.

- Vrijeme izvršavanja: Za uređaje poput beskontaktnih kartica i RFID oznaka, vrijeme izvršavanja ograničeno je vremenom koje uređaj provodi u komunikacijskoj zoni.

Slika 2.9 prikazuje kompromis između sigurnosti, cijene i performansi pri dizajniranju lakih kriptografskih algoritama. Općenito, za bilo koji algoritam, što je ključ duži i što ima više krugova (rundi), to je veća sigurnost.




Slika 2.9: Kompromisi u lakoj kriptografiji

Više od pedeset simetričnih LWC algoritama predložili su različiti akademski, privatni i vladini subjekti s ciljem smanjenja troškova te poboljšanja hardverskih i softverskih performansi. Većina algoritama simetrične blokovske kriptografije izvedena je iz DES-a ili AES-a, koji su temeljni stupovi kriptografskih algoritama. PRESENT je inspiriran AES-om, temelj je lake kriptografije i osnova mnogih novih LWC algoritama. Ipak, većina njih

ne usredotočuje se istovremeno na sva tri temeljna svojstva (trošak, performanse i sigurnost), već se fokusira na jedno ili najviše dva od njih. Slika 2.10 pokazuje da Print, Simon i Speck zahtijevaju minimalno područje za implementaciju. Također Rectangle i Present pokazuju konkurenciju u ovoj utrci hardverskih performansi. Međutim, Speck i Simon su najbolji u softverskim performansama, dokumentirajući najveću softversku učinkovitost, 3511,19 Kbps/KB i 1900 Kbps/KB, redom. Osim toga, poznati su i po najnižem kašnjenju, 408 ciklusa/bloku i 594 ciklusa/bloku, redom.

| LWC Algorithm | Hardware Implementation | | | | | | | | Software Implementation | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Key Size | Block Size | Tech ($\mu m$) | Area (GE) | Power ($\mu W$) | Energy ($\mu J/bit$) | Throughput @100KHz (Kbps) | Hardware Efficiency (Kbps/KGE) | Key Size | Block Size | ROM (byte) | RAM (byte) | Latency (Cycles/ block) | Energy ($\mu J/bit$) | Throughput @4MHz (Kbps) | Software Efficiency (Kbps/KB) |
| AES* | 128 | 128 | 0.13 | 2400 | 2.4 | 42.38 | 56.64 | 23.6 | 128 | 128 | 918 | 0 | 4192 | 16.7 | 122 | 132.9 |
| PRESENT* | 80 | 64 | 0.18 | 1570 | 2.35 | 11.77 | 200 | 127.38 | 128 | 64 | 660 | 0 | 10792 | 43.1 | 23.7 | 35.91 |
| RECTANGLE | 80 | 64 | 0.13 | 1467 | 1.46 | 5.96 | 246 | 167.68 | - | - | - | - | - | - | - | - |
| PRINCE* | 128 | 64 | 0.13 | 2953 | 2.95 | 5.53 | 533.3 | 180.59 | 128 | 64 | 1108 | 0 | 3614 | 14.4 | 70.8 | 63.9 |
| SIMON* | 96 | 48 | 0.13 | 763 | 0.76 | 48.32 | 15.8 | 20.7 | 96 | 48 | 170 | 0 | 594 | 2.3 | 323 | 1900 |
| TWINE# | 80 | 64 | 0.09 | 1503 | 1.05 | 5.91 | 178 | 118.42 | 80 | 64 | 1180 | 140 | 20505 | - | 12.48 | 10.58 |
| SPECK* | 96 | 48 | 0.13 | 884 | 0.88 | 73.67 | 12 | 13.57 | 96 | 48 | 134 | 0 | 408 | 1.6 | 470.5 | 3511.19 |

Slika 2.10: Hardverske i softverske performanse LWC algoritama [24]

Sigurnost algoritama lake kriptografije procjenjuje se kroz kriptoanalizu, koja identificira ranjivosti pokušajima različitih napada. Ključne vrste kriptoanalize uključuju:

- nelinearnost - eng. nonlinearity
- linearnu aproksimaciju - eng. linear approximation probability (LP)
- diferencijalnu aproksimaciju - eng. differential approximate probability (DAP)
- efekt lavine - eng. degree of avalanche effect (SAC)
- neovisnost bitova - eng. bit independence criteria (BIC)
- algebarske napade - eng. algebraic attacks

Diferencijalna kriptoanaliza (DAP) analizira izlaze u odnosu na ulaze i uključuje napade poput višeg reda, skraćenih, nemogućih i „boomerang” napada. Linearna kriptoanaliza (LP) koristi „piling-up lemma” za aproksimaciju odnosa između otvorenog teksta, šifriranog teksta i ključa. Nelinearnost, mjerena Hammingovom udaljenošću ili Walshovom matricom, povećava sigurnost algoritma. Algebarska kriptoanaliza rješava jednadžbe, što ju čini učinkovitom za jednostavnije algoritme. Napadi se provode na temelju šifriranog ili poznatog teksta, te koriste metode poput MITM (Man-in-the-Middle), brute force i bočnih

kanala. Diferencijalni napadi na greške ciljaju unutarnju strukturu algoritma. Slika 2.11 prikazuje analizu sigurnosti različitih postojećih algoritama lake kriptografije u mrežnom formatu. Na slici su prikazane ranjivosti algoritama na napade. Ako je algoritam označen kvačicom, to znači da je ranjiv na taj napad.

| LWC Algorithm | Differential Cryptanalysis* | Linear Crypt-analysis | Integral/Square/ Saturation Cryptanalysis | Algebraic/Cube Cryptanalysis | MITM/ Biclique | Related Key attack | Side-Channel/ Differential fault attacks |
|---|---|---|---|---|---|---|---|
| AES | ✓ | - | - | - | ✓ | ✓ | ✓ |
| PRESENT | ✓ | - | - | - | ✓ | ✓ | ✓ |
| GIFT | ✓ | - | ✓ | - | ✓ | ✓ | ✓ |
| SKINNY | ✓ | - | - | - | - | ✓ | ✓ |
| RECTANGLE | - | - | ✓ | - | - | ✓ | ✓ |
| PRINCE | ✓ | - | - | - | - | - | - |
| SIMON | ✓ | - | - | ✓ | - | ✓ | - |
| TWINE | - | - | ✓ | - | ✓ | - | - |
| SPECK | ✓ | - | - | - | - | ✓ | - |

Slika 2.11: Sigurnosna analiza LWC algoritama [24]

Analiza pokazuje da gotovo sva postojeća rješenja lake blokovske šifre pate od više vrsta napada, osobito napada s povezanim ključem, zatim raznih diferencijalnih i MITM napada. Štoviše, lakše verzije (s reduciranim brojem rundi) ranjivije su na brojne napade u usporedbi sa standardnim verzijama. Suprotno tome, većina algoritama pokazuje dobar otpor prema linearnim i algebarskim svojstvima. Poglavlje 3 opisuje neke od algoritama koji su namjenjeni za uređaje lakih resursa te detaljno opisuje strukture tih algoritama.

# Poglavlje 3

# Kriptografski algoritmi za uređaje lakih resursa

U prethodnom poglavlju opisali smo klasifikaciju blokovskih šifri na: substitucijsko-permutacijsku mrežu (SPN), Feistelovu mrežu (FN), opću Feistelovu mrežu (GFN), zbroj-rotacija-XOR (ARX), nelinearni pomični registarski povratni sustav (NLFSR) i hibrid. U ovom poglavlju dajemo kriptografske algoritme za prve 4 vrste s obzirom na tablicu na slici 2.6.

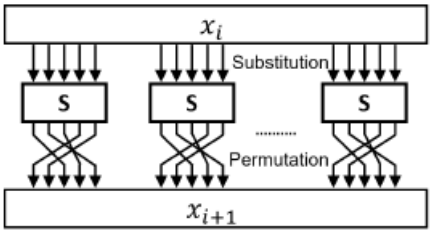
## 3.1 Present

U ovom potpoglavlju upoznat ćemo supstitucijsko-permutacijsku mrežu (SPN), koncept u modernoj kriptografiji koji kombinira operacije supstitucije i permutacije za visoku sigurnost. Predstavit ćemo osnovne značajke algoritma Present, objasniti njegovu strukturu i funkcionalnost, te prikazati pseudokod za implementaciju.

### 3.1.1 Supstitucijsko permutacijska mreža (SPN)

Supstitucijsko permutacijska mreža (SPN) je jedna vrsta iteriranog blokovskog šifriranja. Kao što ime sugerira, svaki krug šifre sastoji se od primjene dviju osnovnih komponenti, supstitucije i permutacije. Sloj supstitucije obično se dizajnira kombiniranjem niza malih nelinearnih operacija poznatih kao S-kutije (Substitucijske kutije). S-kutije moraju biti bijektivne kako bi se osigurala inverzija za proces dešifriranja. Permutacija je linearni sloj koji osigurava difuziju. Potključevi se kombiniraju sa stanjima pomoću grupnih operacija poput zbrajanja ili XOR-a [20]. Algoritmi koji koriste mrežu supstitucije primjenjuju nekoliko naizmjeničnih slojeva kutija za supstituciju, koji čine nelinearni sloj, nakon čega slijedi linearni sloj koji izvodi permutaciju (Slika 3.1). Prednosti ove strukture uključuju

paralelizam, što omogućuje vrlo brzo i učinkovito izvođenje. Neki poznati algoritmi koji koriste mreže supstitucije i permutacije su AES, Present, Prince, Piccolo, Led, Klein, Print, Skinny i Gift. AES pripada ovoj kategoriji te se temelji na mreži supstitucije i permutacije, a brz je u izvedbi na softverskoj i hardverskoj razini [10]. Present je predložen na konferenciji CHES'2007, te je privukao veliku pažnju kriptografskih istraživača zbog svoje jednostavnosti, impresivnih hardverskih performansi i jake sigurnosti. Dizajn Presenta je izuzetno učinkovit u pogledu hardverske implementacije, budući da koristi permutaciju bitova kao svoj difuzijski sloj, što je dizajn koji je jednostavan za implementaciju u fizičkom hardveru. Godine 2012., Present je prihvaćen kao ISO/IEC standard za laku kriptografiju.




Slika 3.1: SPN [24]

### 3.1.2 Općenito o Present algoritmu

Present je primjer SPN-mreže (supstitucijsko permutacijske mreže) i sastoji se od 31 runde u kojim je u svaki krug uvedena operacija XOR s rundnim ključem. Duljina ulaznog teksta (plaintexta) iznosi 64 bita, a podržane su dvije duljine ključa: 80 i 128 bita. S obzirom na namjene koje imamo na umu, preporučujemo verziju s ključevima duljine 80 bita [5]. Razlika između 80-bitnog i 128-bitnog ključa u SPN mreži leži u duljini ključa, 128-bitni ključ nudi bolju sigurnost s više mogućih kombinacija, dok je 80-bitni ključ brži i zahtijeva manje resursa. Međutim, 80-bitni ključ je manje siguran i podložniji napadima, dok 128-bitni ključ pruža veću sigurnost, ali je resursno zahtjevniji. Present se sastoji od linearne transformacije nazvane permutacija i nelinearne transformacije nazvane supstitucija [15]. Djelovanje S-boxa (nelinearne transformacije) u heksadecimalnom zapisu vidimo u sljedećoj tablici na slici 3.2 dok je permutacija bitova (linearna transformacija) korištena u Present šifri prikazana na slici 3.3

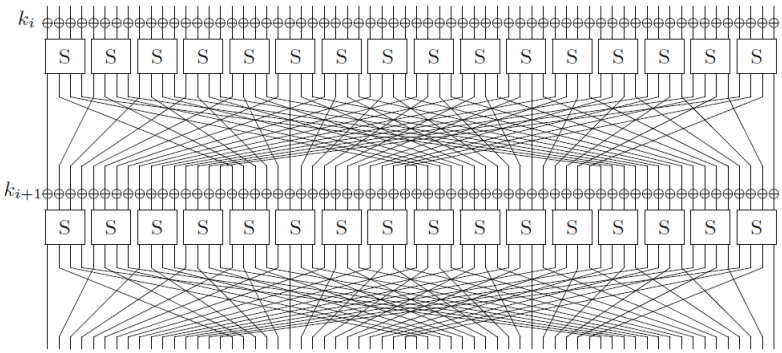
Supstitucija i permutacija izvode se jednom u svakoj od 31 runde, a supstitucijsko permutacijsku mrežu možemo vidjeti na slici 3.4

| $x$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S[x]$ | C | 5 | 6 | B | 9 | 0 | A | D | 3 | E | F | 8 | 4 | 7 | 1 | 2 |

Slika 3.2: Substitucija [15]

| $i$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $P(i)$ | 0 | 16 | 32 | 48 | 1 | 17 | 33 | 49 | 2 | 18 | 34 | 50 | 3 | 19 | 35 | 51 |
| $i$ | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| $P(i)$ | 4 | 20 | 36 | 52 | 5 | 21 | 37 | 53 | 6 | 22 | 38 | 54 | 7 | 23 | 39 | 55 |
| $i$ | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| $P(i)$ | 8 | 24 | 40 | 56 | 9 | 25 | 41 | 57 | 10 | 26 | 42 | 58 | 11 | 27 | 43 | 59 |
| $i$ | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| $P(i)$ | 12 | 28 | 44 | 60 | 13 | 29 | 45 | 61 | 14 | 30 | 46 | 62 | 15 | 31 | 47 | 63 |

Slika 3.3: Permutacija [15]




Slika 3.4: SP mreža za Present [5]

Iz ulaznog ključa generira se novi ključ za svaki krug i taj ključ nazivamo rundni ključ ili round key, a za šifriranje postoje četiri glavna bloka koji se izvode za svaku rundu Presenta: AddRoundKey, S-box, P-layer i raspored ključeva.

### 3.1.3 Pseudokod

U ovom pododijeljku opisat ćemo pseudokod za Present algoritam. Kao ulaz prima otvoreni tekst veličine 64 bita i ključ veličine 80 bita, dok je izlaz šifrat od 64 bita. Algoritam se izvodi kroz 30 rundi, a u svakoj rundi izvode se sljedeća četiri koraka [15]:

1. Generira se 80-bitni ključ trenutne runde i pohranjuje u varijablu RoundKey.

2. Izvršava se XOR operacija između sadržaja varijable state i RoundKey.
3. Provodi se supstitucija sadržaja varijable state primjenom S-box funkcije (SBoxLayer).
4. Provodi se permutacija sadržaja varijable state prema unaprijed definiranim pravilima (PermutationLayer).

Nakon što se petlja završi, generira se ključ za posljednju rundu (31. rundu). Na kraju se u šifrat sprema rezultat XOR operacije između trenutnog sadržaja varijable state i zadnje generiranog ključa. Algoritam 2 prikazuje psudokod za naš algoritam, a nakon algoritma dajemo objašnjenje svakog sloja (layera).

**Algoritam 2** Present encryption algorithm [15]

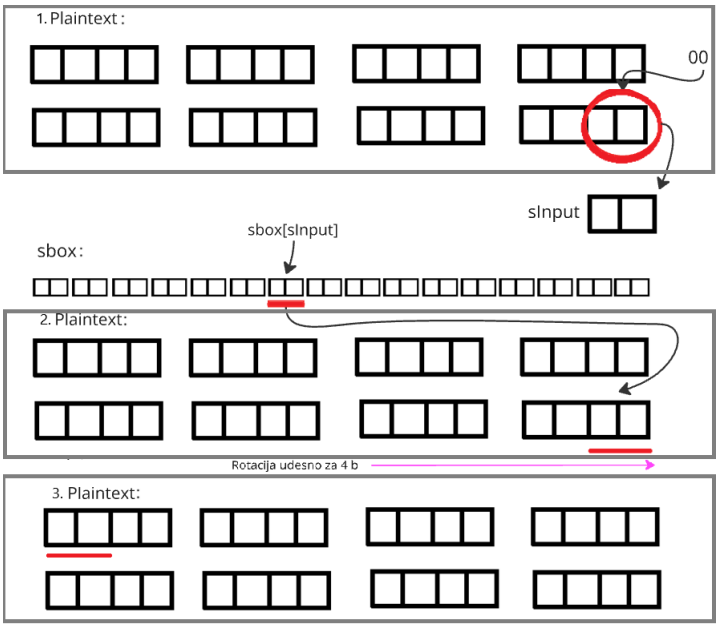
**Require:** Plaintext[64], Key[80]
**Ensure:** Ciphertext[64]
$R \leftarrow 1$
$State \leftarrow Plaintext$
**while** $R < 31$ **do**
  $RoundKey_R \leftarrow PRESENTgenerateKey_{80}(Key, R)$
  $State \leftarrow State \oplus RoundKey_R$
  $State \leftarrow SBoxLayer(State)$
  $State \leftarrow PermutationLayer(State)$
  $R \leftarrow R + 1$
**end while**
$lastRoundKey \leftarrow generateKey(Key, R)$
$State \leftarrow State \oplus lastRoundKey$
$Ciphertext \leftarrow State$

Slika 3.5 prikazuje postupak supstitucije SBoxLayer i njegovu primjenu u prvoj rundi na otvoreni tekst (plaintext). Otvoreni tekst od 64 bita pohranjuje se u varijablu state, a cilj je analizirati što se događa s tekstom tijekom prve primjene funkcije na state tj. SBoxLayer(state). U prvoj rundi u varijabli state spremljen je plaintext s toga prikazujemo što se događa u prvoj rundi s otvorenim tekstom, a analogno se postupak odvija u svakoj rundi za varijablu state. Supstitucija se odvija na način da se prvo iz ulaznog podatka, tj. otvorenog teksta (64 bita) izdvoje posljednja 4 bita, koji se zatim pohranjuju u varijablu sInput. Nakon što su izdvojeni, posljednjih 4 bita otvorenog teksta maskiraju se na 0, čime se pripremaju za zamjenu. Zamjena se vrši korištenjem unaprijed definiranog S-boxa, koji u algoritmu izgleda ovako:

$$sbox[] = \{0xC, 0x5, 0x6, 0xB, 0x9, 0x0, 0xA, 0xD, 0x3, 0xE, 0xF, 0x8, 0x4, 0x7, 0x1, 0x2\}.$$

Vrijednost varijable sInput koristi se kao indeks za pristup S-boxu, a vrijednost na toj poziciji u S-boxu zamjenjuje prethodnih 4 bita iz state-a tj. u našem slučaju otvorenog

teksta. Ova nova vrijednost sBox[sInput] postaje posljednjih 4 bita u otvorenom tekstu. Nakon što se izvrši zamjena, cijeli tekst se rotira udesno za 4 bita. Bit koji „ispadne" s kraja tekstualnog niza prebacuje se na početak, čime se postiže kružna rotacija. Svaki segment se transformira pomoću S-boxa i ažurira unutar varijable state, čime se dobiva novo stanje koje je spremno za sljedeći korak algoritma.



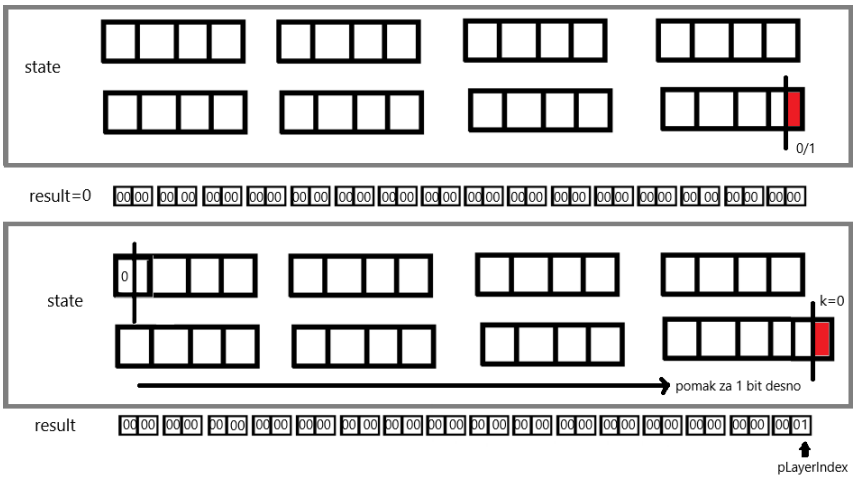

Slika 3.5: SBoxLayer

Na slici 3.6 prikazan je postupak permutacije PermutationLayer u Present algoritmu. Permutacija se koristi kako bi se bitovi unutar stanja (state) premjestili na nove pozicije prema unaprijed definiranom uzorku, čime se povećava difuzija podataka. Prvo se inicijalizira varijabla result s vrijednošću 0. Ova varijabla postupno se gradi kako bi sadržavala rezultat permutacije. Zatim se prolazi kroz svih 64 bita trenutnog stanja (state) koristeći petlju. Unutar petlje, izdvoji se najmanje značajni bit iz trenutnog stanja (označeno crveno). Taj bit predstavlja trenutno obrađivani bit te nakon što se bit izdvoji, stanje se pomiče udesno za jedan bit kako bi sljedeći bit postao najmanje značajni u idućoj iteraciji. Ako je izdvojeni bit različit od nule, potrebno je odrediti njegovu novu poziciju u rezultatu. Nova pozicija računa se kao

$$pLayerIndex = (16 * k)\%63,$$

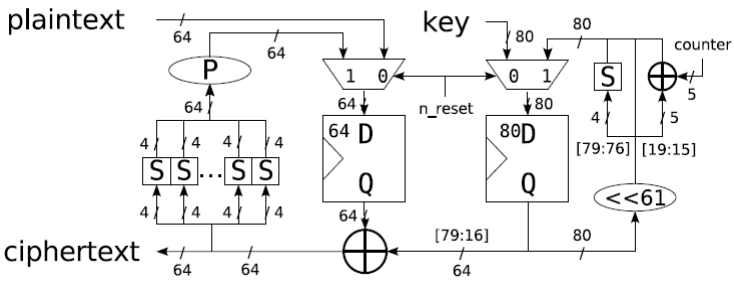
gdje je $k$ (od 0 do 63) trenutni indeks u petlji, tj. $k$-ti najmanje značajni bit iz state-a. Ako je trenutni bit posljednji ($k = 63$), njegova nova pozicija se ručno postavlja na 63 kako bi se

izbjegla pogreška u permutaciji. Nakon što se izračuna nova pozicija, taj bit se premješta na novu poziciju unutar varijable result. Za $k = 0$ ta pozicija je 0 i zapisuje se bit na tu poziciju. Na kraju petlje, kada su svi bitovi premješteni na svoje nove pozicije, sadržaj varijable state koji je tada 0 zamjenjuje se sadržajem varijable result, čime se permutacija završava. Ovaj proces osigurava da se bitovi unutar stanja redistribuiraju na način koji povećava složenost analize šifriranog teksta.




Slika 3.6: PermutationLayer

Cijeli postupak algoritma od otvorenog teksta (plaintext) kroz permutaciju i supstituciju te xor stanja s ključem kroz svaku rundu do šifrata (ciphertext) može se vidjeti i na slici 3.7.




Slika 3.7: Present [5]

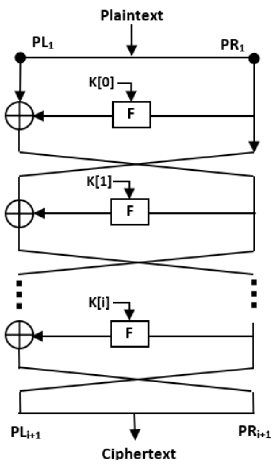
## 3.2 Simon

U ovom potpoglavlju detaljno ćemo objasniti Feistelovu mrežu, koja predstavlja temelj mnogih modernih šifrirajućih algoritama, uključujući i Simon šifru. Zatim ćemo se usredotočiti na strukturu i način rada Simon šifre, analizirajući ključne operacije koje koristi (kružne rotacije, bitovni XOR i AND) kako bi se postigla sigurnost uz minimalne hardverske zahtjeve. Na kraju, u dijelu posvećenom pseudokodu, prikazat ćemo algoritam Simon šifre kroz njegovu formalnu specifikaciju i opisati implementacijske korake.

### 3.2.1 Feistel mreža (FN)

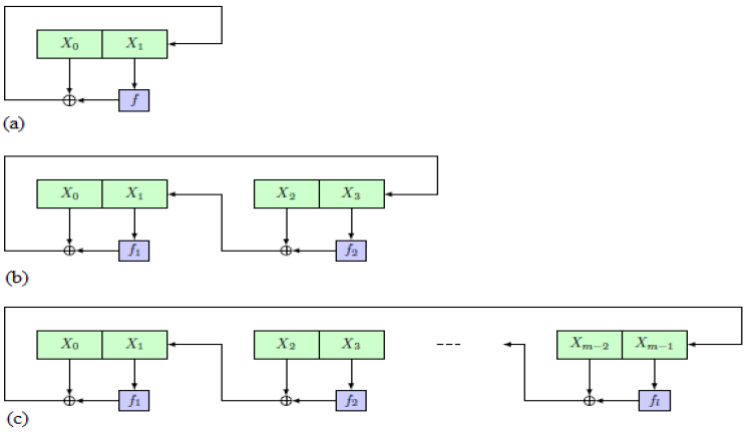
Feistel šifra je iterativna struktura u kojoj se stanje dijeli na dva dijela od po $n/2$ bita. Svaka runda koristi rundnu funkciju koja, uz pomoć $k$-bitnog podključa, transformira jedan dio stanja, a zatim se rezultat XOR-a s drugim dijelom. Nakon toga, dijelovi se zamjenjuju, osim u posljednjoj rundi. Ključna prednost Feistel strukture je što rundna funkcija ne mora biti invertibilna, a dešifriranje se jednostavno izvodi korištenjem istih operacija u obrnutom redoslijedu. Jedan od najpoznatijih primjera Feistel šifre je DES, koji je 1976. postao američki standard za enkripciju [20]. Feistelova mreža je simetrična struktura koja se koristi u konstrukciji blok šifara. Ime je dobila po kriptografu Horstu Feistelu, njemačko-američkom kriptografu koji je razvio temeljne principe modernih blokovskih šifri. Njegov rad je doveo do razvoja Feistelove mreže, koja se koristi u mnogim poznatim šiframa, uključujući DES. Feistelova šifra funkcionira na sljedeći način [17]:

1. Ulaz (podaci koje treba šifrirati) podijeljen je na dvije polovice jednake veličine. Nazovimo ih lijeva polovica i desna polovica.
2. Jezgra Feistelove mreže je funkcija koja se često naziva Feistelova funkcija. Potrebna su mu dva ulaza: blok podataka (polovica podataka) i ključ (dio ključa za šifriranje) i proizvodi izlaz koji se miješa s drugom polovicom podataka.
3. Feistelova mreža primjenjuje ovu funkciju u više krugova. U svakom krugu, desna polovica prolazi kroz Feistelovu funkciju zajedno s ključem specifičnim za krug, a na izlazu se zatim vrši XOR s lijevom polovicom.
4. Nakon operacije XOR, dvije se polovice zamijene. Desna polovica postaje nova lijeva polovica, a XOR lijeve polovice postaje nova desna polovica (Slika 3.8).
5. Ovaj proces se ponavlja u nekoliko krugova. Broj rundi može varirati ovisno o specifičnom dizajnu šifre. Više krugova općenito znači veću sigurnost, ali također oduzimaju više vremena.
6. Nakon posljednjeg kruga, dvije se polovice kombiniraju kako bi se proizveo konačni šifrirani blok podataka. Za dešifriranje isti se postupak primjenjuje obrnutim redoslijedom, s rundnim ključevima koji se koriste suprotnim redoslijedom.




Slika 3.8: Feistel struktura [14]

Feistelove strukture, potekle iz DES-a, predstavljaju nelinearne pomične registre s povratnom informacijom. Osnovna struktura primjenjuje funkciju na jednu polovicu bloka, koja se zatim kombinira s drugom polovicom i pomiče poput NLFSR-a. Primjeri tih šifri su DES, Katan, Simon i Simeck. Ključna prednost Feistelove strukture je što operacije šifriranja i dešifriranja mogu biti gotovo identične, što omogućuje učinkovitu implementaciju [10]. Najpoznatija Feistelova struktura je ona korištena u DES-u, prikazana na Slici 3.9(a). Simon je uravnotežena Feistelova šifra s $n$-bitnom riječju, gdje je duljina bloka $2n$,




Slika 3.9: Feistelova struktura s različitim duljinama [10]

a konstrukcija je prikazana na Slici 3.9-(a). Duljina ključa je višekratnik od $n$, pomnožen s

2, 3 ili 4, što predstavlja vrijednost $m$. Ova obitelj blok šifri dizajnirana je za optimizaciju hardverskih implementacija [10].

### 3.2.2 Općenito o Simon algoritmu

Simon šifra je idealna za sustave poput RFID-a i Internet stvari (IoT) zbog niskog broja tranzistora i male složenosti, što je ključno u pasivno napajanim sustavima s ograničenim prostorom i potrošnjom energije. Šifra koristi uravnoteženu Feistelovu mrežu s pomakom, XOR-om i AND-om. Simon primjenjuje složen raspored ključeva kroz nekoliko rundi i podržava različite konfiguracije veličina blokova, ključeva i broja rundi, kako je prikazano na slici 3.10 [7]:

| Veličina bloka (bitovi) | Veličina ključa (bitovi) | runde |
|---|---|---|
| 32 | 64 | 32 |
| 48 | 72 | 36 |
| | 96 | 36 |
| 64 | 96 | 42 |
| | 128 | 44 |
| 96 | 96 | 52 |
| | 144 | 54 |
| 128 | 128 | 68 |
| | 192 | 69 |
| | 256 | 72 |

Slika 3.10: Simon [25]

Funkcija runde Simon algoritma koristi sljedeće operacije nad $n$-bitnim riječima [3]:

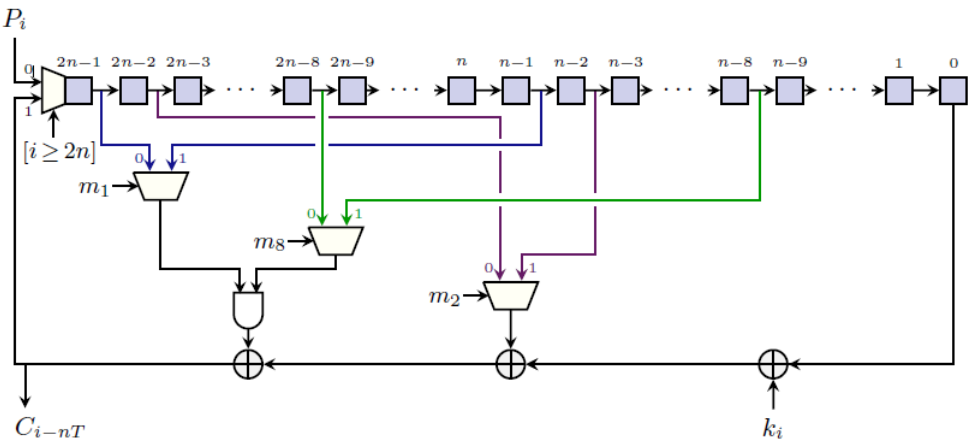
- bitovni XOR $\oplus$,
- bitovni AND &,
- lijeva kružna rotacija $S^j$, za $j$ bita,
- desna kružna rotacija $S^{-j}$, za $j$ bita,
- modularno zbrajanje +.

Simon šifra je simetrična šifra dizajnirana za energetski ograničena okruženja. Koristi matematički jednostavnu, ali sigurnu rundnu funkciju koja štiti od tradicionalnih napada.

Iako nije zamjena za AES, fleksibilne veličine ključeva i blokova omogućuju bolju implementaciju u ograničenim uvjetima. Implementacija ne zahtijeva SRAM za međustanja jer se stanje šifre čuva unutar samih rundnih bitova, čime se smanjuje potrošnja energije, a također zahtijeva manje logičkih vrata u odnosu na AES [7]. Simon šifra je simetrična šifra niske složenosti, dizajnirana za aplikacije poput RFID-a i IoT-a. Objavljena je 2013. godine kao dio „Simon i Speck obitelji laganih blok šifri" od strane NSA [7], do tad nije postojala implementacija koja zadovoljava specifične zahtjeve za nisku potrošnju energije i ograničene hardverske resurse. Simon blok šifra koristi uravnoteženu Feistelovu strukturu s $n$-bitnim riječima, gdje je duljina bloka $2n$. Na primjer, Simon64/128 označava šifru koja koristi 64-bitni blok i 128-bitni ključ. Logika generiranja ključeva ovisi o broju ključeva (2, 3 ili 4) u implementaciji. Ovdje ćemo priložiti pseudokod za Simon64/96 s 42 runde.

### 3.2.3 Pseudokod

Pseudokod Simon algoritma prikazan na slici 3.11 ilustrira proces serijalizacije, gdje se šifriranje odvija jedan bit po ciklusu. U svakom ciklusu, bitovi otvorenog teksta i rundni ključ se obrađuju, dok kontrolni bitovi upravljaju odabirom između različitih opcija u multiplexerima (sklop koji omogućava odabir između različitih ulaza na temelju kontrolnih bitova) na temelju trenutnog ciklusa. Šifrirani bitovi izlaze tijekom posljednjih $2n$ ciklusa, nakon što su svi bitovi prošli kroz cijeli proces obrade.




Slika 3.11: Serijalizacija Simon funkcije runde, jedan bit u isto vrijeme [4]

Kao što je navedeno u potpoglavlju 3.2.1, otvoreni tekst dijeli se u dvije polovice: lijevu ($plain_1$) i desnu ($plain_2$). Funkcija Simon64/96KeySchedule generira ključeve za

**Algoritam 3** Simon Encryption Algorithm [13]

**Require:** PlainText[64], Keys[96]
**Ensure:** CipherText[64]
  $plain_1 \leftarrow PlainText[0]$
  $plain_2 \leftarrow PlainText[1]$
  $roundKeys \leftarrow simon64/96KeySchedule[keys]$
  **for** $i \leftarrow 0$ **to** 41 **step** 2 **do**
    $plain_1 \leftarrow plain_1 \oplus roundKeys[i] \oplus f(plain_2)$
    $plain_2 \leftarrow plain_2 \oplus roundKeys[i+1] \oplus f(plain_1)$
  **end for**
  $CipherText[0] \leftarrow plain_1$
  $CipherText[1] \leftarrow plain_2$

svaku rundu koristeći desnu rotaciju *ror* i XOR operaciju. Arhitektura rasporeda ključeva dana je jednadžbom na slici 3.12, a broj ključnih riječi $m$ određuje širinu ključa $m \times n$. Šifriranje se temelji na rundnoj funkciji [4]:

$$R_k(x, y) = (y \oplus f(x) \oplus k, x)$$

gdje je $k$ rundni ključ, $x$ i $y$ su lijeva i desna polovica, a Feistelova funkcija definirana je kao:

$$f(x) = (Sx \& S^8 x) \oplus S^2 x.$$

Ključevi rundi $k_0, \ldots, k_{m-1}$ jednaki su početnim $m$ riječima ključa, dok se za $0 \leq i < T - m$ ključ $k_{i+m}$ definira prema slici 3.12.

$$k_{i+m} = \begin{cases} c \oplus (z_j)_i \oplus k_i \oplus \left(I \oplus S^{-1}\right)\left(S^{-3} k_{i+1}\right), & m = 2 \\ c \oplus (z_j)_i \oplus k_i \oplus \left(I \oplus S^{-1}\right)\left(S^{-3} k_{i+2}\right), & m = 3 \\ c \oplus (z_j)_i \oplus k_i \oplus \left(I \oplus S^{-1}\right)\left(S^{-3} k_{i+3} \oplus k_{i+1}\right), & m = 4 \end{cases}$$

Slika 3.12: Matematički opis generiranja ključeva [7]

Sekvenca $zx$ uklanja klizna svojstva, dok XOR maska s „1” (označena kao $c$) utječe na najniža dva bita ključa. Za 2- i 3-riječne ključeve logika je ista, dok 4-riječni ključ uključuje dodatne XOR operacije. **Generacija** $zx$**:** Konstantna sekvenca $zx$ generira se pomoću LFSR-a (Linear Feedback Shift Register), koji proizvodi bitne konstante ovisno o veličini ključa i bloka. Selekcijski kriteriji prikazani su u tablici na slici 3.13 [4].

Feistelova funkcija $f(x)$ kombinira tri kružne rotacije te koristi bitovni AND i XOR kako bi generirala novu 32-bitnu vrijednost:

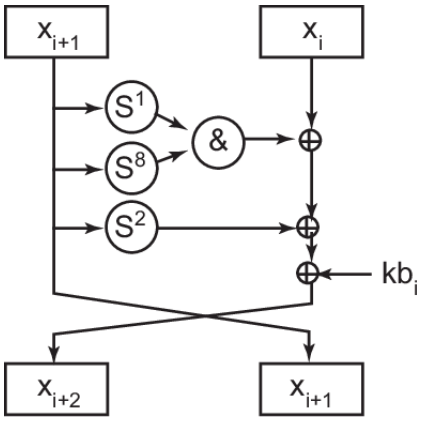
$$f(x) = (rol_{32}(x, 1) \& rol_{32}(x, 8)) \oplus rol_{32}(x, 2)$$

| block size ($2n$) | key size ($mn$) | word bit size ($n$) | key word count ($m$) | const. seq. | rounds T |
|---|---|---|---|---|---|
| 32 | 64 | 16 | 4 | $z_0$ | 32 |
| 48 | 72 | 24 | 3 | $z_0$ | 36 |
| | 96 | | 4 | $z_1$ | 36 |
| 64 | 96 | 32 | 3 | $z_2$ | 42 |
| | 128 | | 4 | $z_3$ | 44 |
| 96 | 96 | 48 | 2 | $z_2$ | 52 |
| | 144 | | 3 | $z_3$ | 54 |
| 128 | 128 | 64 | 2 | $z_2$ | 68 |
| | 192 | | 3 | $z_3$ | 69 |
| | 256 | | 4 | $z_4$ | 72 |

$z_0$ = 11111010001001010110000111001101111101000100101011000011100110
$z_1$ = 10001110111110010011000010110101000111011111001001100001011010
$z_2$ = 10101111011100000011010010011000101000010001111110010110110011
$z_3$ = 11011011101011000110010111100000010010001010011100110100001111
$z_4$ = 11010001111001101011011000100000010111000011001010010011101111

Slika 3.13: Selekcijski kriterij [7]

Oznake $rol_{32}(x, 1)$, $rol_{32}(x, 2)$ i $rol_{32}(x, 8)$ (na slici 3.14 kao $S^1, S^2, S^8$) označavaju lijevu kružnu rotaciju 32-bitne vrijednosti za 1, 2 i 8 bita. $rol_{32}(x, 1)$ & $rol_{32}(x, 8)$ koristi bitovni AND između dvije kružno rotirane verzije teksta. Slika 3.14 dosta jednostavno prikazuje što se događa s polovicama kad na njih djeluje Feistelova funkcija.




Slika 3.14: Feistelova funkcija [7]

## 3.3 Speck

Speck je lagana blokovska šifra koju je NSA razvila 2013. godine, zajedno sa šifrom Simon, za optimalnu izvedbu na uređajima s ograničenim resursima. Za razliku od Simon šifre, koja koristi Feistelovu mrežu, Speck se temelji na ARX strukturi (Addition-Rotation-XOR) kombinirajući modularno zbrajanje, kružne rotacije i XOR operacije. Ova arhitektura omogućuje visoku fleksibilnost i efikasnost, posebno u softverskim implementacijama. U nastavku detaljno opisujemo ARX strukturu te prikazujemo pseudokod algoritma Speck.

### 3.3.1 Zbroj-rotacija-XOR (ARX)

ARX šifre koriste jednostavne operacije poput modularnog zbrajanja, rotacije bitova, pomaka i XOR-a kako bi osigurale difuziju i konfuziju, predstavljajući učinkovitu alternativu S-box šiframa. Njihova prednost leži u brzoj i jeftinoj implementaciji u softveru i hardveru te otpornosti na napade temeljene na analizi vremena izvršavanja. Međutim, sigurnosna svojstva kombinacije ovih operacija nisu u potpunosti istražena, posebno zbrajanje modulo $n$, što otežava predviđanje njihovih utjecaja na nasumičnost rezultata. Također, zbog sporije difuzije, ARX šifre zahtijevaju veći broj rundi za otpornost na diferencijalne i linearne napade [20]. ARX operacije označavaju tri ključne operacije:

- Addition (zbrajanje): zbrajanje s modulom, što je temeljna operacija u šifri.
- Rotation (rotacija): rotacije bitova, specifično desnu i lijevu rotaciju (ROR i ROL), kako bi pomaknuo bitove u bloku, povećavajući difuziju.
- XOR (bitovni ekskluzivni ili): XOR se koristi za postizanje konfuzije, čineći vezu između ključeva i podataka teže predvidljivom.

Ove tri operacije kombinirane su u svakoj rundi, dajući Speck njegov ARX karakter. ARX mreže imaju nekoliko ključnih svojstava koja ih čine pogodnima za primjenu u kriptografiji. Prvo, jednostavne bitovne operacije omogućuju visoku brzinu izvršavanja, osobito na procesorima koji podržavaju efikasne implementacije modularnog zbrajanja i rotacija. Drugo, ARX šifre ne zahtijevaju S-box tablice, što smanjuje potrebu za memorijom i omogućuje lakšu implementaciju na ugrađenim sustavima. Još jedna važna karakteristika ARX mreža je njihova otpornost na diferencijalne i linearne napade zbog načina na koji kombiniraju zbrajanje i XOR operacije. Također, zbog modularnog zbrajanja, male promjene u ulaznim podacima uzrokuju nelinearne promjene u izlazu, što povećava sigurnost protiv analitičkih napada. Iako ARX struktura pruža visoku efikasnost, njezina glavna slabost je sporija difuzija u odnosu na metode koje koriste S-boxove. Zbog toga ARX šifre obično zahtijevaju veći broj rundi kako bi postigle odgovarajuću sigurnost. Unatoč tome, zbog svoje jednostavnosti, brzine i prilagodljivosti, ARX mreže ostaju popularan izbor za moderne blok šifre poput Speck-a, ChaCha20 i Blake hash funkcija.

### 3.3.2 Općenito o Speck algoritmu

Iako Speck koristi Feistelov stil, gdje se tekst dijeli na dvije riječi i obrađuje kroz više rundi, ono što ga čini ARX algoritmom je primjena ARX operacija – zbrajanje (Addition), rotacije (Rotation) i XOR (XOR). Speck podržava različite veličine blokova i ključeva. Blok uvijek sadrži dvije riječi veličine 16, 24, 32, 48 ili 64 bita, dok ključ može imati 2, 3 ili 4 riječi. Svaka runda uključuje dvije rotacije, modularno zbrajanje desne i lijeve riječi, XOR ključa s lijevom riječi te završni XOR lijeve i desne riječi. Ove operacije omogućuju brzu i učinkovitu softversku implementaciju [10]. Speck je optimiziran za softver i podržava 22 do 34 iteracije, ovisno o veličini bloka i ključa. Najkompaktnija hardverska implementacija koristi 48-bitni blok i 96-bitni ključ, dok najefikasnija softverska implementacija zahtijeva 599 ciklusa i 186 bajtova ROM-a za 64-bitni blok s 128-bitnim ključem [24]. Među laganim blokovskim šiframa, Speck postiže najbolje performanse za sve podržane konfiguracije, dok Simon zauzima drugo mjesto, osim za 128-bitne blokove [3]. Jedna od prednosti Specka je njegova skalabilnost – algoritam se može prilagoditi različitim veličinama blokova i ključeva, omogućujući primjenu na širokom spektru uređaja, od mikrokontrolera do snažnijih procesora. Speck je posebno pogodan za ugradbene sustave i IoT uređaje zbog svoje niske memorijske potrošnje i brzine izvođenja. Unatoč visokoj učinkovitosti, Speck je izazvao kontroverze zbog svog podrijetla iz NSA-e, što je dovelo do njegove isključenosti iz standardizacijskih procesa poput ISO/IEC 29192-2 [16]. Ipak, brojna istraživanja pokazala su da algoritam nema poznate ranjivosti u realističnim uvjetima napada, a njegova jednostavna struktura omogućuje lako razumijevanje i analizu sigurnosti. Speck koristi jednostavnu rundnu funkciju, koja osigurava dovoljno difuzije kroz niz iteracija. Raspored ključeva (key schedule) temelji se na ARX operacijama, što omogućava brzo generiranje podključeva uz minimalne resurse. Dodatna sigurnosna analiza pokazuje da Speck pruža otpornost na klasične kriptografske napade, uključujući diferencijalnu i linearnu kriptanalizu, ali zahtijeva veći broj rundi kako bi osigurao visoku razinu sigurnosti. Speckova jednostavnost, visok stupanj optimizacije za softver i prilagodljivost različitim platformama čine ga jednim od najučinkovitijih laganih blokovskih šifriranih algoritama danas.

### 3.3.3 Pseudokod

Algoritam iako nije Feistel mreže, prvo dijeli otvoreni tekst na dvije polovice. SpeckKeySchedule postavlja početne vrijednosti rundi iz originalnog ključa te koristi operacije rotacije (ror32 i rol32), zbrajanje i XOR za generiranje novih ključeva. Za kraj daje rezultat koji je niz roundKeys za daljnje šifriranje. Enkripcija se provodi kroz 25 rundi, pri čemu svaka runda uključuje desnu rotaciju prve riječi, modularno zbrajanje s drugom riječi i XOR operaciju s rundnim ključem. Nakon toga, druga riječ prolazi kroz lijevu rotaciju i XOR operaciju s prvom riječi. Nakon završetka svih rundi, dobiveni šifrat (CipherText)

**Algoritam 4** Speck Encryption Algorithm [13]

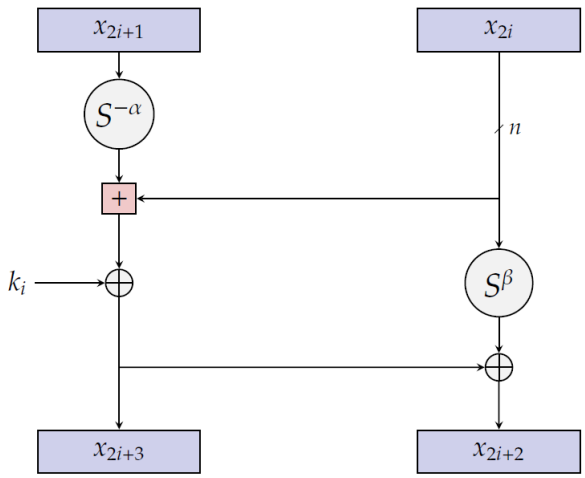
**Require:** PlainText[64], Keys[96]
**Ensure:** CipherText[64]

$plain_1 \leftarrow PlainText[0]$
$plain_2 \leftarrow PlainText[1]$
$roundKeys \leftarrow speckKeySchedule_{96}(Keys)$

**for** $i \leftarrow 0$ **to** 24 **do**
  $plain_1 \leftarrow rotateRight(plain_1, 8)$
  $plain_1 \leftarrow plain_1 + plain_2$
  $plain_1 \leftarrow plain_1 \oplus roundKeys[i]$
  $plain_2 \leftarrow rotateLeft(plain_2, 3)$
  $plain_2 \leftarrow plain_2 \oplus plain_1$
**end for**

$CipherText[0] \leftarrow plain_1$
$CipherText[1] \leftarrow plain_2$

sadrži enkriptirane vrijednosti $plain_1$ i $plain_2$. Postupak speck algoritma kroz rundu vidimo i na slici 3.15.




Slika 3.15: Struktura Speck algoritma [2]

## 3.4 Twine

Twine je simetrična blok šifra dizajnirana za visoku sigurnost uz minimalne računalne resurse, čineći je pogodnom za uređaje s ograničenim resursima poput IoT uređaja. Njegova jednostavna struktura, efikasnost i otpornost na napade poput linearnih i diferencijalnih čine ga posebnim. U ovom poglavlju opisat ćemo strukturu algoritma, generiranje rundnih ključeva i implementaciju pseudokoda koji prikazuje osnovne korake šifriranja.

### 3.4.1 Opća Feistelova mreža (GFN)

Opća Feistelova mreža (GFN) je proširena verzija klasične Feistelove mreže, koja se koristi u mnogim modernim šiframa zbog svoje fleksibilnosti i sigurnosti. Za razliku od klasične mreže, koja dijeli ulazni blok na dvije polovice, GFN omogućuje podjelu bloka na više manjih dijelova, obično 4-bitne ili 8-bitne podblokove. Twine algoritam koristi 8-blokovski GFN sa 36 rundi, a svaki par podblokova prolazi kroz operacije poput S-box transformacija i permutacija, čime se postiže snažna difuzija podataka. Jedna od ključnih karakteristika GFN-a je uporaba složenih permutacija i rotacija između podblokova, što povećava složenost algoritma i otežava napadima analizu promjena u šifriranom tekstu. Za razliku od klasične Feistelove mreže, koja koristi XOR operacije i rotacije, GFN može koristiti ciklične pomake unutar bloka, čime se postiže veća difuzija podataka. Zahvaljujući svojoj modularnoj strukturi, GFN je prilagodljiv za različite uređaje, od onih s velikim računalnim resursima do uređaja s ograničenim resursima poput ugradbenih sustava i IoT uređaja. Broj podblokova, krugova i vrsta funkcija mogu se mijenjati ovisno o potrebama sigurnosti i efikasnosti. Rotacije i ciklični pomaci omogućuju širenje informacija kroz cijeli blok, a složene permutacije dodatno otežavaju napade, čineći GFN snažnijim i sigurnijim od klasične Feistelove mreže.

### 3.4.2 Općenito o Twine algoritmu

U Twine algoritmu ulazni blok se dijeli na više dijelova, u ovom slučaju na 8 dijelova, pri čemu svi dijelovi prolaze kroz modificirane Feistel operacije. Twine je 64-bitna blok šifra koja koristi ključ duljine 80 ili 128 bitova. Ovisno o duljini ključa, koristimo oznake Twine-80 ili Twine-128. Struktura Twinea je varijanta GFN-a s 16 4-bitnih podblokova. Rundna funkcija sastoji se od nelinearnog sloja koji koristi 4-bitne S-kutije, kao i sloja difuzije u obliku 4-bitne permutacije koja je jednostavan (lijevi ili desni) ciklični pomak [23]. Stoga je dizajn algoritma prilično minimalistički, ali istovremeno učinkovit. Twine algoritam radi na sljedeći način:

- **Podjela bloka:** Ulazni blok od 64 bita dijeli se na 8 dijelova, svaki po 8 bita.

- **Runde:** Twine koristi ukupno 36 rundi. Svaka runda uključuje primjenu S-box funkcije (substitucije) i permutacije (mreže za miješanje dijelova podataka).
- **Ključ:** Ključ se koristi u svakoj rundi algoritma, kroz funkciju koja je specifična za GFN strukturu.
- **Krajnji izlaz:** Na kraju, dobiva se šifrirani tekst.

Rekli smo da Twine koristi 8-blokovski GFN sa 36 runde to znači da algoritam kao ulaz prima 64-bitni ulazni blok, koji je podijeljen na 8 podblokova na sljedeći način:

$$X = (X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7)$$

Svaki podblok $X_i$ predstavlja 8-bitnu vrijednost. Za svaku rundu koristi se odgovarajući podključ $RK$, koji se generira iz rasporeda ključeva (key schedule). Svaki drugi podblok koristit će se kasnije za XOR s pripadajućim dijelom podključa: $X_1 \oplus RK_1, X_3 \oplus RK_3, X_5 \oplus RK_5, X_7 \oplus RK_7$. Twine koristi fiksni 4-bitni S-box, koji se primjenjuje na podblokove $X_1, X_3, X_5, X_7$, uvodeći nelinearnost u algoritam. Nakon toga, podblokovi se reorganiziraju prema unaprijed definiranoj permutacijskoj funkciji. Ovi koraci ponavljaju se ukupno 36 puta kako bi se postigla puna difuzija i sigurnost algoritma. Slika 3.16 prikazuje substituciju koja se odvija pomoću S-kutije i permutaciju koja se odvija reorganizacijom prema unaprijed definiranim pravilima.

| $x$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S(x)$ | C | 0 | F | A | 2 | B | 9 | 5 | 8 | 3 | D | 7 | 1 | E | 6 | 4 |

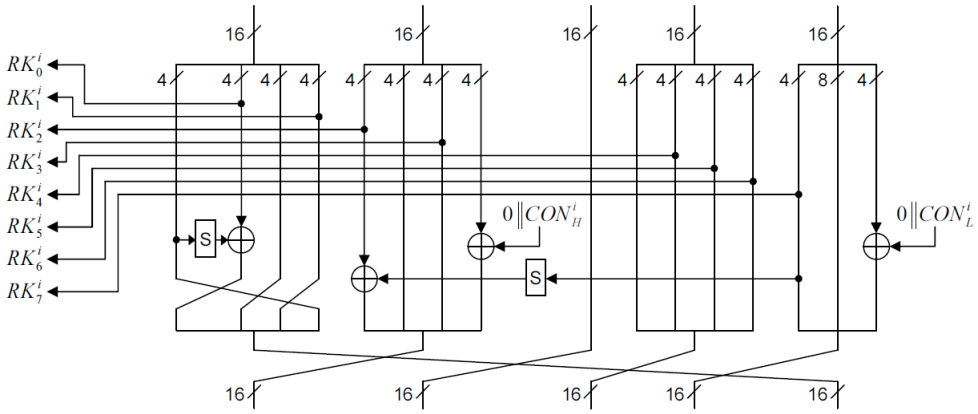
| $h$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\pi[h]$ | 5 | 0 | 1 | 4 | 7 | 12 | 3 | 8 | 13 | 6 | 9 | 2 | 15 | 10 | 11 | 14 |
| $\pi^{-1}[h]$ | 1 | 2 | 11 | 6 | 3 | 0 | 9 | 4 | 7 | 10 | 13 | 14 | 5 | 8 | 15 | 12 |

Slika 3.16: Enkripcija i dekripcija sa S-box S i permutacijom nibbla (polubajta) [23]

U konstrukciji rundnih ključeva koristimo raspored ključeva (key schedule) na sljedeći način:

- Podijeli ključ u 4-bitne blokove.
- Generiraj round ključeve za 36 krugova.
- Transformiraj blokove ključeva koristeći S-box, konstante i rotacije.
- Vraća niz round ključeva koji će biti korišteni u šifriranju.

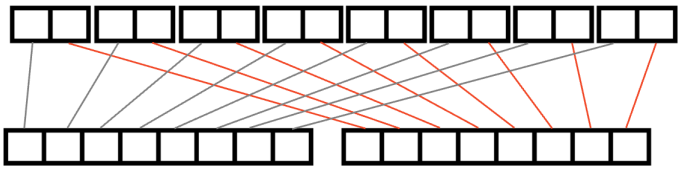
Na slici 3.17 vidimo key schedule 80-bitnog ključa za Twine algoritam.




Slika 3.17: Twine Key Schedule [23]

### 3.4.3 Pseudokod

Slijedi pseudokod za algoritam 5. Otvoreni tekst od 64 bita podijeljen je na 8 podblokova, svaki od 8 bita. Svaki od tih 8-bitnih podblokova dalje se dijeli na lijevu i desnu polovicu, pri čemu svaka polovica ima po 4 bita. Od lijevih i desnih polovica podblokova formira se lijeva i desna polovica cijelog otvorenog teksta, kako je prikazano na slici 3.18. Svaka druga polovica podbloka postaje desna polovica otvorenog teksta (u pseudokodu označeno kao right), dok ostale idu u lijevu polovicu (u pseudokodu označeno kao left). Zatim je potrebno generirati round ključeve prije for petlje, to se odvija u funkciji key schedule koja je već opisana na slici 3.17. Algoritam ima 36 runde te u svakoj se odvija takozvana rundna funkcija koja se sastoji od toga da lijevi dio supstituiramo, permutiramo i zatim napravimo XOR s rundnim ključem. Na kraju runde imamo XOR vrijednosti rundne funkcije sa desnim dijelom, te zamjenu novodobivene lijeve polovice s desnom. Postupak koji se odvija u rundi može se vidjeti i na slici 3.19.



Slika 3.18: Podjela otvorenog teksta prije rundne funkcije

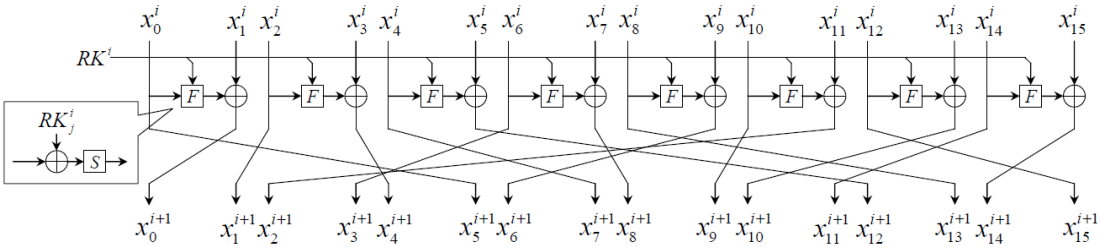
**Algoritam 5** Twine Encryption Algorithm [26]

**Require:** PlainText[64], Key[80]
**Ensure:** CipherText[64]

```
for i ← 0 to plainText_length step 2 do
    left ← PlainText[i]
    right ← PlainText[i + 1]
end for
roundKeys ← KeySchedule_80(Key)
for i ← 0 to 35 do
    temp ← left

    roundF ← substitute(left)
    roundF ← permute(roundF)
    roundF ← roundF ⊕ roundKeys[i]

    left ← right ⊕ roundF
    right ← temp
end for
CipherText[0] ← left
CipherText[1] ← right
```




Slika 3.19: Rundna funkcija Twine algoritma [23]

## 3.5 Prince

Prince je ultralagani blokovski algoritam zasnovan na SP mreži, optimiziran za visoku brzinu i nisku latenciju. Za razliku od Presenta i Rectanglea, omogućuje gotovo trenutnu enkripciju i dekripciju zahvaljujući $\alpha$-refleksiji, koja eliminira potrebu za klasičnim generiranjem ključeva (key scheduleom) pri šifriranju. Njegova jednostavna hardverska implementacija i energetska učinkovitost čine ga idealnim za real-time i sigurnosno osjetljive sustave.

### 3.5.1 Općenito o Prince algoritmu

Sve više ugrađenih sustava zahtijeva šifriranje s niskom latencijom, posebno aplikacije sa zahtjevima u stvarnom vremenu. Za sve te slučajeve želimo imati simetrične šifre koje mogu odmah šifrirati zadani otvoreni tekst. Prince omogućuje šifriranje podataka unutar jednog takta s vrlo konkurentnom veličinom čipa u usporedbi s poznatim rješenjima. Broj rundi mora biti umjeren, a runde moraju imati kratka kašnjenja u hardveru. Prince je 64-bitni blokovna šifra s 128-bitnim ključem, a ključ se dijeli na dva dijela od po 64 bita. Pri dizajniraju Princea cilj je bio dizajnirati novu blokovsku šifru optimiziranu prema sljedećim kriterijima pri implementaciji u hardveru [6]:

1. Šifra može izvršiti trenutno šifriranje, pri čemu se šifrirani tekst izračunava unutar jednog takta. Nema faze zagrijavanja.
2. Ako se implementira u modernoj čip tehnologiji, mogu se postići niska kašnjenja koja rezultiraju umjereno visokim taktovima.
3. Troškovi hardvera su umjereni (tj. znatno niži od potpuno razmotanih verzija AES-a ili Presenta).
4. Šifriranje i dešifriranje trebaju biti mogući uz niske troškove i minimalni dodatni teret.

Kad kažemo potpuno razmotane mislimo na verzije gdje su svi dijelovi algoritma razrađeni unaprijed, čime se smanjuje potreba za ponovnim iteriranjem kroz iste operacije, tj. nema potrebe za ponovnim izvođenjem petlji. Postojeće lake šifre, poput Presenta, ne ispunjavaju kriterije niskog kašnjenja i male površine zbog velikog broja rundi. Prince, koji je optimiziran prema tim kriterijima, nudi nekoliko inovacija. Prvo, eliminira potrebu za iterativnim rundama, omogućujući učinkovitiju implementaciju gdje dešifriranje s jednim ključem odgovara šifriranju s povezanim ključem. Ovo svojstvo, poznato kao $\alpha$-refleksija, poboljšava sigurnost. Razmotana šifra unaprijed „razmotava” sve runde šifriranja, pojednostavljujući i ubrzavajući proces jer spaja operacije u jednu funkciju. Druga značajka je balansiranje troškova S-box sloja i linearnog sloja. Optimizacija S-box funkcije utječe na ukupni trošak, a Prince omogućuje odabir S-boxa iz skupa koji ispunjava specifične kriterije, čime se povećava učinkovitost. Kao rezultat, Prince koristi 6-7 puta manje površine od Present-80 i 14-15 puta manje od AES-128 za iste vremenske uvjete i tehnologije. Za razliku od klasičnog key schedule-a u šiframa poput Presenta ili AES-a, Prince eliminira potrebu za generiranjem novih ključeva za svaku rundu zahvaljujući $\alpha$-refleksiji. Ovo značajno ubrzava proces dešifriranja, omogućujući istovremenu obradu enkripcije i dekripcije. Ova svojstva omogućuju optimizaciju algoritama, smanjenje hardverskih zahtjeva i pojednostavljenje operacija šifriranja i dešifriranja.

### 3.5.2 Pseudokod

Prince je 64-bitna blokovna šifra s 128-bitnim ključem, a ključ se dijeli na dva dijela od po 64 bita $k = k_0||k_1$ i proširuje se na 192 bita pomoću sljedećeg mapiranja [6]:

$$(k_0||k_1) \rightarrow (k_0||k_0'||k_1) := (k_0||(k_0 >>> 1) \oplus (k_0 >> 63)||k_1). \tag{3.1}$$

Prva dva podključa, $k_0$ i $k_0'$, koriste se za maskiranje podataka (whitening keys), dok je $k_1$ 64-bitni ključ za osnovnu 12-rundnu blokovnu šifru, nazvanu $PRINCE_{core}$. Ovdje dajemo pregled ključnih elemenata korištenih u algoritmu, uključujući maskiranje, te opisujemo cijelu konstrukciju prije nego što predstavimo pseudokod i njegovo objašnjenje. FX konstrukcija je dizajn blokovnih šifri u kojoj se koriste funkcija (F) i XOR operacija (X) za miješanje podataka i ključeva. U Prince šifri, $k_0$ i $k_0'$ služe za whitening (Slika 3.20), dok $k_1$ pokreće osnovnu šifru $PRINCE_{core}$.


Slika 3.20: $Prince_{core}$

Svaka runda sastoji se od dodavanja ključa (**k-add**), S-box sloja (**S-Layer**), linearnog sloja (**linear M/M' Layer**) i dodavanja konstantne vrijednosti runde (**RC-add**).

**k-add:** XOR 64-bitnog stanja s 64-bitnim podključem.

**S-layer:** Šifra koristi jednu 4-bitnu S-kutiju (S-box). Djelovanje S-kutije u heksadecimalnoj notaciji dana je sljedećom tablicom (slika 3.5.2).

| $x$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S[x]$ | B | F | 3 | 2 | A | C | 9 | 1 | 6 | 7 | 8 | 0 | E | 5 | D | 4 |

Slika 3.21: S-box

**linear layer:** U M i M' sloju 64-bitno stanje množi se s $64 \times 64$ matricom M (tj M')

**RC-add:** U koraku RC-add, 64-bitna konstantna vrijednost runde u heksadecimalnoj notaciji (Slika 3.22) miješa se sa stanjem pomoću XOR operacije.

Prije nego što opišemo pseudokod, potrebno je objasniti linearne slojeve $M$, $M'$ i $M^{-1}$.

| | |
|---|---|
| $RC_0$ | 0000000000000000 |
| $RC_1$ | 13198a2e03707344 |
| $RC_2$ | a4093822299f31d0 |
| $RC_3$ | 082efa98ec4e6c89 |
| $RC_4$ | 452821e638d01377 |
| $RC_5$ | be5466cf34e90c6c |
| $RC_6$ | 7ef84f78fd955cb1 |
| $RC_7$ | 85840851f1ac43aa |
| $RC_8$ | c882d32f25323c54 |
| $RC_9$ | 64a51195e0e3610d |
| $RC_{10}$ | d3b5a399ca0c2399 |
| $RC_{11}$ | c0ac29b7c97c50dd |

Slika 3.22: RC Add [19]

U $M$ i $M'$ slojevima, 64-bitno stanje množi se s $64 \times 64$ matricom $M$ (odnosno $M'$). Za ove slojeve imamo različite zahtjeve: $M'$ sloj se koristi samo u srednjoj rundi i mora biti inverzan kako bi osigurao $\alpha$-refleksiju. Ovaj zahtjev ne vrijedi za $M$ sloj, koji osigurava potpunu difuziju nakon dvije runde. Da bismo to postigli, kombiniramo $M'$ preslikavanje s primjenom matrice SR, koja permutira 16 polubajtova (nibbleova) na način sličan ShiftRows u AES-u (Slika 3.23).

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | $\longrightarrow$ | 0 | 5 | 10 | 15 | 4 | 9 | 14 | 3 | 8 | 13 | 2 | 7 | 12 | 1 | 6 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Slika 3.23: ShiftRows [6]

Dakle, $M = SR \circ M$. Troškovi implementacije trebaju biti minimalizirani, što zahtijeva smanjenje broja jedinica u matricama M' i M. Istovremeno, potrebno je osigurati da najmanje 16 S-kutija bude aktivno u 4 uzastopne runde. Da bi se to postiglo, svaki izlazni bit iz S-kutije mora utjecati na 3 S-kutije u sljedećoj rundi, što zahtijeva minimalno 3 jedinice po retku i stupcu. Stoga, kao osnovne građevne blokove za M'-sloj koristimo četiri $4 \times 4$ matrice kao sa slike 3.24.

U sljedećem koraku generiramo $4 \times 4$ blok-matricu $\hat{M}$, čiji su redci i stupci permutacije matrica $M_0, \ldots, M_3$. Permutacije se biraju tako da matrica bude simetrična. Ovaj odabir osigurava da rezultirajuća $16 \times 16$ matrica bude involucija.

Algoritam 6 započinje mapiranjem $k_0, k_1, k_0'$ prema već zadanoj formuli 3.1. Kao što smo opisali algoritam ima 3 dijela, maskiranje s $k_0$, $Prince_{core}$ te maskiranje s $k_0'$. Ta ma-

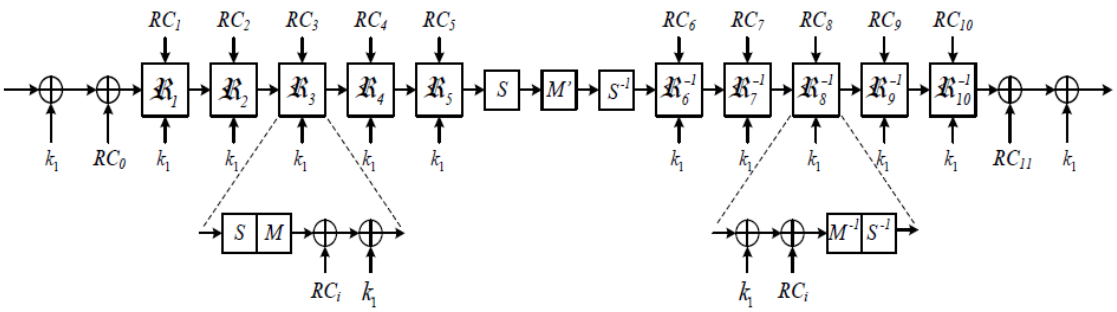
$$M_0 = \begin{pmatrix} 0000 \\ 0100 \\ 0010 \\ 0001 \end{pmatrix}, \; M_1 = \begin{pmatrix} 1000 \\ 0000 \\ 0010 \\ 0001 \end{pmatrix}, \; M_2 = \begin{pmatrix} 1000 \\ 0100 \\ 0000 \\ 0001 \end{pmatrix}, \; M_3 = \begin{pmatrix} 1000 \\ 0100 \\ 0010 \\ 0000 \end{pmatrix}$$

Slika 3.24: matrice M [6]

$$\hat{M}^{(0)} = \begin{pmatrix} M_0 & M_1 & M_2 & M_3 \\ M_1 & M_2 & M_3 & M_0 \\ M_2 & M_3 & M_0 & M_1 \\ M_3 & M_0 & M_1 & M_2 \end{pmatrix} \quad \hat{M}^{(1)} = \begin{pmatrix} M_1 & M_2 & M_3 & M_0 \\ M_2 & M_3 & M_0 & M_1 \\ M_3 & M_0 & M_1 & M_2 \\ M_0 & M_1 & M_2 & M_3 \end{pmatrix}$$

Slika 3.25: matrice [6]

skiranja u pseudokodu smo nazvali pre-masking i post-masking, a između se nalazi glavni dio (core). $Prince_{core}$ odvija se kroz 12 rundi. Cijeli proces enkripcije $PRINCE_{core}$ opisan je i na slici 3.26.




Slika 3.26: $PRINCE_{core}$ [6]

Prva runda je zapravo u našem slučaju nulta runda, imamo XOR stanja šifre s ključem $k_1$ te nultim rundnim ključem. Nakon te runde imamo petlju za sljedećih 5 rundi u kojoj imamo SBox, linearni sloj M (MPrime, ShiftRows-nonInverse), RC-add i k-add. Zatim jedna runda je samo S-box, linearni sloj i inverzni S-box. Slijedi 5 rundi u kojima imamo RC-add, k-add, linearan sloj M' (MPrime, ShiftRows-Inverse) te inverzni S-box, a redosljed izvršavanja obrnut je od prethodnih 5 rundi for petlje. Core završava sa zadnjom (12.)

rundom u kojoj slično kao i u nultoj rundi imamo XOR stanja šifre s ključem $k_1$ te završnim rundnim ključem.

**Algoritam 6** Prince Encryption Algorithm [19]

**Require:** Plaintext[64], Key[128]
**Ensure:** Ciphertext[64]
$k_0 \leftarrow$ ExtractFirst64Bits($Key$)
$k_1 \leftarrow$ ExtractSecond64Bits($Key$)
$k_0' \leftarrow$ RotateRight($k_0, 1$) $\oplus$ ($k_0 \gg 63$)
$CipherState \leftarrow Plaintext \oplus k_0$ **# Pre-masking**
$CipherState \leftarrow CipherState \oplus k_1 \oplus RC[0]$
**for** $Round \leftarrow$ 1 to 5 **do**
  $CipherState \leftarrow$ Sbox($CipherState, S$)
  $CipherState \leftarrow$ MPrime($CipherState$)
  $CipherState \leftarrow$ ShiftRows($CipherState$, inverse = False)
  $CipherState \leftarrow CipherState \oplus k_1 \oplus RC[Round]$
**end for**

$CipherState \leftarrow$ Sbox($CipherState, S$)
$CipherState \leftarrow$ MPrime($CipherState$)
$CipherState \leftarrow$ Sbox($CipherState, S^{-1}$)

**for** $Round \leftarrow$ 6 to 10 **do**
  $CipherState \leftarrow CipherState \oplus k_1 \oplus RC[Round]$
  $CipherState \leftarrow$ ShiftRows($CipherState$, inverse = True)
  $CipherState \leftarrow$ MPrime($CipherState$)
  $CipherState \leftarrow$ Sbox($CipherState, S^{-1}$)
**end for**

$CipherState \leftarrow CipherState \oplus k_1 \oplus RC[11]$
$CipherState \leftarrow CipherState \oplus k_0'$ **# Post-masking**

$Ciphertext \leftarrow CipherState$

## 3.6 Rectangle

U ovom potpoglavlju upoznat ćemo se s algoritmom Rectangle, također primjerom SP-mreže, koji je temeljni koncept u modernoj kriptografiji koji kombinira operacije supstitucije i permutacije. Rectangle je zanimljiv za proučavanje zbog svoje inovativne arhitekture

koja omogućava visoku sigurnost i izuzetnu učinkovitost u hardverskim i softverskim implementacijama. Ovaj algoritam pruža dobar omjer između sigurnosti i performansi, čineći ga pogodnim za širok spektar primjena. U ovom dijelu predstavit ćemo osnovne značajke Rectangle-a, objasniti njegovu strukturu i funkcionalnost, te također pružiti pseudokod koji ilustrira njegovu implementaciju.

### 3.6.1 Općenito o Rectangle algoritmu

Rectangle je ultra lagani blok šifrirani algoritam koji se može koristiti u raznim aplikacijama. S manjim promjenama u SPN strukturi, broj rundi je smanjen na 25 (u usporedbi s 31 rundom u Presentu) kako bi se zadovoljili zahtjevi konkurentnog okruženja [24]. Rectangle je iterativna blokovska šifra s duljinom bloka od 64 bita, duljina ključa može biti 80 ili 128 bita, a sloj supstitucije sastoji se od 16 S-kutija dimenzija $4 \times 4$ koje rade paralelno. Sloj permutacije sastoji se od 3 rotacije. Rectangle je dizajniran korištenjem bit-slice tehnike. S obzirom na sigurnost algoritama Present i Rectangle, diferencijalna i linearna kriptoanaliza najučinkovitije su metode [28]. Rectangle nudi izvrsne performanse u hardverskim i softverskim okruženjima, pružajući dovoljno fleksibilnosti za različite scenarije primjene. Slijede tri glavne prednosti Rectanglea [27]:

- Dobar je za hardverske implementacije jer omogućuje visoku brzinu, energetsku učinkovitost i malu potrošnju prostora na čipu, što ga čini pogodnim za uređaje s ograničenim resursima.
- Učinkovit je u softverskim implementacijama zahvaljujući „bit-slice” pristupu, koji omogućuje paralelnu obradu podataka na razini bita, optimizirajući brzinu šifriranja i omogućujući visoke performanse.
- Uvodi nove kriterije za dizajn S-kutije. Pažljiv odabir S-kutije i asimetrični dizajn permutacijskog sloja osiguravaju dobar balans između sigurnosti i performansi, s visokom otpornošću čak i uz veći broj rundi.

Bit-slice je metoda obrade podataka u kriptografiji i računarstvu koja omogućuje paralelnu obradu bitova unutar više registara ili procesnih jedinica. Umjesto da se podaci obrađuju bajt po bajt (ili blok po blok), bit-slice pristup tretira svaki bit kao zasebnu jedinicu i paralelno obrađuje odgovarajuće bitove iz različitih podataka. Bit-slice pristup dijeli blok podataka na bitove koji se paralelno obrađuju koristeći SIMD instrukcije, čime se postiže učinkovitije iskorištavanje hardvera, brža obrada i bolja energetska učinkovitost. Prednosti bit-slice metode uključuju veću paralelizaciju koja omogućuje istovremenu obradu više podataka, jednostavniji dizajn koji olakšava implementaciju složenih operacija te poboljšanu sigurnost smanjenjem rizika od vremenskih i drugih napada zahvaljujući konzistentnosti operacija u vremenskom trajanju.

### 3.6.2 Pseudokod

Prije nego što damo pseudokod objasnit ćemo neke elemente koji se koriste u pseudokodu. Otvoreni tekst od 64 bita, međurezultat od 64 bita ili šifrirani tekst od 64 bita zajednički se nazivaju stanjem šifre. Stanje šifre može se prikazati kao pravokutna matrica dimenzija $4 \times 16$ bita, što je i razlog za naziv šifre Rectangle. Neka je $W = w_{63}||w_{62}||\cdots||w_2||w_1||w_0$ stanje šifre. Prvih 16 bitova $w_{15}||\cdots||w_1||w_0$ raspoređeno je u redak 0, sljedećih 16 bitova $w_{31}||\cdots||w_{17}||w_{16}$ u redak 1, i tako dalje, kao što je prikazano na slici 3.27 Takav proces raspoređivanja otvorenog teksta u matricu u pseudokodu nazvat ćemo LoadPlaintext.

$$\begin{bmatrix} w_{15} & \cdots & w_2 & w_1 & w_0 \\ w_{31} & \cdots & w_{18} & w_{17} & w_{16} \\ w_{47} & \cdots & w_{34} & w_{33} & w_{32} \\ w_{63} & \cdots & w_{50} & w_{49} & w_{48} \end{bmatrix}$$

Slika 3.27: Stanje šifre - CipherState [27]

Stanje podključa od 64 bita prikazuje se kao pravokutna matrica dimenzija $4 \times 16$ bita, a ovaj postupak odvija se u KeySchedule-u prije generiranja rundnih ključeva. Rectangle je šifra SP-mreže s 25 rundi, pri čemu svaka runda uključuje tri koraka:

- AddRoundkey,
- SubColumn,
- ShiftRow.

Nakon posljednje runde izvršava se završni AddRoundKey. AddRoundKey je jednostavan bitovni XOR podključa s međustanjem. SubColumn primjenjuje paralelno S-kutije na 4 bita u istom stupcu, dok je stanje ključa prikazano u dvodimenzionalnom obliku na slici 3.28. Ulazni podaci na koje primjenjujemo S-box je stupac $a_{3,j}||a_{2,j}||a_{1,j}||a_{0,j}$. Imamo

$$\begin{bmatrix} a_{0,15} & \cdots & a_{0,2} & a_{0,1} & a_{0,0} \\ a_{1,15} & \cdots & a_{1,2} & a_{1,1} & a_{1,0} \\ a_{2,15} & \cdots & a_{2,2} & a_{2,1} & a_{2,0} \\ a_{3,15} & \cdots & a_{3,2} & a_{3,1} & a_{3,0} \end{bmatrix}$$

Slika 3.28: 2D način zapisa stanje šifre [27]

S( $a_{3,j}||a_{2,j}||a_{1,j}||a_{0,j}$) = $b_{3,j}||b_{2,j}||b_{1,j}||b_{0,j}$ za $0 \leq j \leq 15$ (Slika 3.29). S-box korišten u

$$\begin{array}{ccccc}
\begin{pmatrix} a_{0,15} \\ a_{1,15} \\ a_{2,15} \\ a_{3,15} \end{pmatrix} & \cdots & \begin{pmatrix} a_{0,2} \\ a_{1,2} \\ a_{2,2} \\ a_{3,2} \end{pmatrix} & \begin{pmatrix} a_{0,1} \\ a_{1,1} \\ a_{2,1} \\ a_{3,1} \end{pmatrix} & \begin{pmatrix} a_{0,0} \\ a_{1,0} \\ a_{2,0} \\ a_{3,0} \end{pmatrix} \\
\downarrow S & \cdots & \downarrow S & \downarrow S & \downarrow S \\
\begin{pmatrix} b_{0,15} \\ b_{1,15} \\ b_{2,15} \\ b_{3,15} \end{pmatrix} & \cdots & \begin{pmatrix} b_{0,2} \\ b_{1,2} \\ b_{2,2} \\ b_{3,2} \end{pmatrix} & \begin{pmatrix} b_{0,1} \\ b_{1,1} \\ b_{2,1} \\ b_{3,1} \end{pmatrix} & \begin{pmatrix} b_{0,0} \\ b_{1,0} \\ b_{2,0} \\ b_{3,0} \end{pmatrix}
\end{array}$$

Slika 3.29: Djelovanje SubColumn na stupce stanja [27]

| $x$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S(x)$ | 6 | 5 | C | A | 1 | E | 7 | 9 | B | 0 | 3 | D | 8 | F | 4 | 2 |

Slika 3.30: sBox [27]

Rectangle-u je 4-bitni prema 4-bitnom S-boxu $S : F_2^4 \rightarrow F_2^4$. Djelovanje S-boxa u heksadecimalnoj notaciji prikazano je na slici 3.30. ShiftRow je proces u kojem se odvija lijeva rotacija svakog reda za različite pomake. Redak 0 se ne rotira, redak 1 se rotira ulijevo za 1 bit, redak 2 se rotira ulijevo za 12 bitova, a redak 3 se rotira ulijevo za 13 bitova. Neka $<<< x$ označava lijevu rotaciju za x bitova, operacija ShiftRow ilustrirana je na slici 3.31. U zadnjoj rundi imamo još samo jedan XOR podključa zadnje runde s međustanjem.

$$\begin{aligned}
&(a_{0,15} \cdots a_{0,2}\ a_{0,1}\ a_{0,0}) \xrightarrow{\lll 0} (a_{0,15} \cdots a_{0,2}\ a_{0,1}\ a_{0,0}) \\
&(a_{1,15} \cdots a_{1,2}\ a_{1,1}\ a_{1,0}) \xrightarrow{\lll 1} (a_{1,14} \cdots a_{1,1}\ a_{1,0}\ a_{1,15}) \\
&(a_{2,15} \cdots a_{2,2}\ a_{2,1}\ a_{2,0}) \xrightarrow{\lll 12} (a_{2,3} \cdots a_{2,6}\ a_{2,5}\ a_{2,4}) \\
&(a_{3,15} \cdots a_{3,2}\ a_{3,1}\ a_{3,0}) \xrightarrow{\lll 13} (a_{3,2} \cdots a_{3,5}\ a_{3,4}\ a_{3,3})
\end{aligned}$$

Slika 3.31: ShiftRow [27]

Sada dajemo pregled pseudokoda za Rectangle algoritam 7, a na slici 3.32 vidimo kako se odvija postupak šifriranja za algoritam od otvorenog teksta do šifrata.

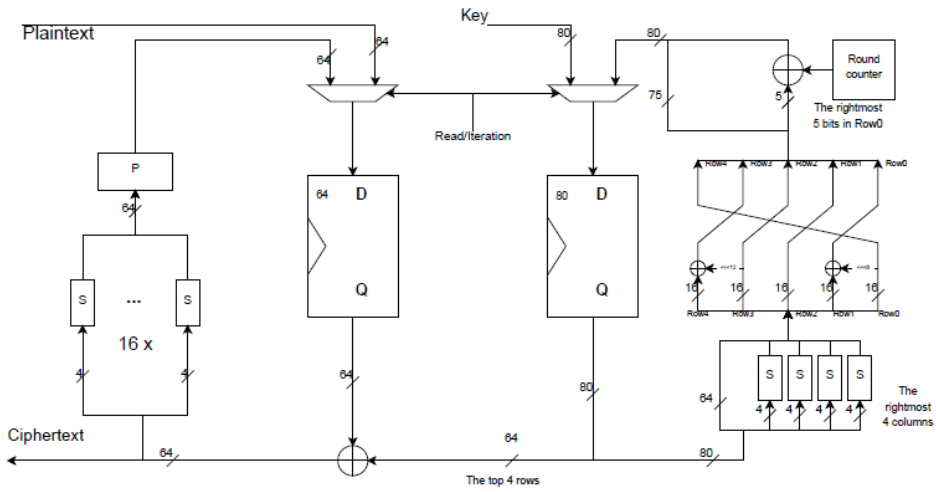
**Algoritam 7** Rectangle Encryption Algorithm [13]

**Require:** Plaintext[64], Key[80]
**Ensure:** Ciphertext[64]

$Round \leftarrow 1$
$RoundKeys \leftarrow$ KeySchedule($Key$)
$CipherState \leftarrow$ LoadPlaintext($Plaintext$)

**while** $Round < 25$ **do**
  $RoundKey \leftarrow$ LoadRoundKey($RoundKeys, Round$)
  $CipherState \leftarrow CipherState \oplus RoundKey$
  $CipherState \leftarrow$ SubColumn($CipherState$)
  $CipherState \leftarrow$ ShiftRow($CipherState$)
  $Round \leftarrow Round + 1$
**end while**

$LastRoundKey \leftarrow$ LoadRoundKey($RoundKeys, 25$)
$CipherState \leftarrow CipherState \oplus LastRoundKey$
$Ciphertext \leftarrow CipherState$




Slika 3.32: Rectangle-80 [27]

# Bibliografija


[1] Zahraa Abdel Hamid, Moataz Samir, Saied M. Abd El-atty, Adel El-Hennawy, Hamed Shenawy, Saleh Abudullah Alshebeili i Fathi E. Abd El-Samie, *On the performance of FFT/DWT/DCT-based OFDM systems with chaotic interleaving and channel estimation algorithms*, Wireless Pers Commun 78 (2014), 1495–1510.

[2] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks i Louis Wingers, *The Simon and Speck families of Lightweight block ciphers*, National Security Agency, USA, 2013.

[3] ———, *The Simon and Speck Block Ciphers on AVR 8-Bit Microcontrollers*, Lecture Notes in Computer Science 8898, Springer, Cham, 2015.

[4] ———, *Implementation and Performance of the Simon and Speck Lightweight Block Ciphers on ASICs*, Crypto-Design Office National Security Agency, 2016, `https://api.semanticscholar.org/CorpusID:39039629`.

[5] Andrey Bogdanov, Lars Ramkilde Knudsen, Gregor Leanderand, Christof Paar, Alex Poschmann, Matthew J.B. Robshaw, Yannick Seurin i C. Vikkelsoe, *PRESENT: An ultra-lightweight block cipher*, Lecture Notes in Computer Science 4727, Springer, Berlin, Heidelberg, 2007.

[6] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knežević, Lars R. Knudsen, Gregor Leande, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen i Tolga Yalçın, *PRINCE: A Low-latency Block Cipher for Pervasive Computing Applications*, Lecture Notes in Computer Science 7658, Springer, Berlin, Heidelberg, 2012.

[7] Brian Degnan i Gregory Durgin, *Simontool: Simulation Support for the Simon Cipher*, IEEE Journal of Radio Frequency Identification, vol. 1, no. 2 (2017), 195–201.

[8] Mohsen A. M. M. El-Bendarya, Atef Abou El-Azmb, Nawal El-Fishawy, Farid S. M. Al-Hosary, Mostafa A. R. Eltokhy, Fathi E. Abd El-Samie i Hasssan B. Kazemian, *An Efficient Chaotic Interleaver for Image Transmission over IEEE 802.15. 4 Zigbee*

*Network*, Journal of Telecommunications and Information Technology, no. 2 (2011), 67–73.

[9] Emad S. Hassan, Xu Zhu, Said E. El-Khamy, Moawad Ibrahim Dessouky, Sami A. El-Dolil i Fathi E. Abd El-Samie, *A Chaotic Interleaving Scheme for the Continuous Phase Modulation Based Single-Carrier Frequency-Domain Equalization System*, Wireless Pers Commun 62 (2012), 183–199.

[10] Morgan He, *Multi-Purpose Designs in Lightweight Cryptography*, Master Thesis, University of Waterloo, 2019.

[11] Antonio Kovačić, *DNA kriptografija*, Diplomski rad, Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet, 2014.

[12] Kriptomat, *Kratka povijest blockchain tehnologije koju bi svatko trebao pročitati*, n.d., `https://kriptomat.io/hr/blockchain/povijest-blockchaina/`, Posjećeno: 18. prosinca 2024.

[13] Ping Luo, *Block ciphers*, 2025, `https://github.com/openluopworld/block-ciphers`, Accessed: 2025-01-14.

[14] Hakeem Imad Mhaibes, May Abood i Alaa Farhan, *Simple Lightweight Cryptographic Algorithm to Secure Imbedded IoT Devices*, International Journal of Interactive Mobile Technologies, vol. 16, no. 20 (2022), 98–113.

[15] Khaled Salah Mohamed, *New Frontiers in Cryptography: Quantum, Blockchain, Lightweight,Chaotic and DNA*, Springer, Cham, 2020.

[16] Tech Monitor, *NSA Ciphers Rejected by ISO Over Trust Issues*, 2017, `https://www.techmonitor.ai/technology/cybersecurity/nsa-ciphers-iso`, Accessed: 2025-02-06.

[17] NordVPN, *Feistel Network - Cybersecurity Glossary*, 2024, `https://nordvpn.com/cybersecurity/glossary/feistel-network/`, Accessed: 2025-02-03.

[18] Bonny B. Raj, J. Frank Vijay i T. Mahalakshmi, *Secure Data Transfer through DNA Cryptography using Symmetric Algorithm*, International Journal of Computer Applications, vol. 133, no. 2 (2016), 19–23.

[19] Vijay Bhaskar Rameshbabu, *Prince*, 2015, `https://github.com/vijaybhaskarrameshbabu/prince`, Accessed: 2025-01-25.

[20] Hadi Soleimany, *Studies in Lightweight Cryptography*, Aalto University publication series, Helsinki, 2014.

[21] Williams Stallings, *Cryptography and network security: principles and practice*, Pearson, Harlow, 2016.

[22] Mario Stipčević, *Kvantna kriptografija*, PowerPoint presentation, 2003, https://www2.irb.hr/korisnici/stipcevi/download/fer171203.pdf, Accessed: 2024-12-07.

[23] Tomoyasu Suzaki, Kazuhiko Minematsu, Sumio Morioka i Eita Kobayashi, *TWINE: A Lightweight Block Cipher for Multiple Platforms*, Lecture Notes in Computer Science 7707, Springer, Berlin, Heidelberg, 2012.

[24] Vishalkumar Arjunsinh Thakor, *Lightweight Cryptography for Resource Constrained IoT devices*, PhD Thesis, School of Computing, Engineering and Digital Technologies Teesside University, UK, 2022.

[25] Wikipedia contributors, *Simon (cipher)*, 2024, https://en.wikipedia.org/wiki/Simon_(cipher), Accessed: 2024-02-03.

[26] Yonatankinfe, *Twine Cipher Java*, 2024, https://github.com/Yonatankinfe/Twine-cipher-java, Accessed: 2025-02-06.

[27] Wentao Zhang, Zhenzhen Bao, Dongdai Lin, Vincent Rijmen, Bohan Yang i Ingrid Verbauwhede, *RECTANGLE: A Bit-slice Lightweight Block Cipher Suitable for Multiple Platforms*, Science China Information Sciences, vol. 58 (2015), 1–15.

[28] Wentao Zhang, Zhenzhen Bao, Vincent Rijmen i Meicheng Liu, *A New Classification of 4-bit Optimal S-boxes and its Application to PRESENT, RECTANGLE and SPONGENT*, Lecture Notes in Computer Science 9054, Springer, Berlin, Heidelberg, 2015.

# Sažetak

Ovaj rad proučava kriptografske algoritme namijenjene uređajima s ograničenim resursima, s posebnim naglaskom na algoritme prikladne za korištenje u kontekstu Interneta stvari (IoT). Kroz analizu kriptografskih algoritama poput Present, Simon, Speck, Twine, Prince i Rectangle, može se zaključiti da svaki od njih ima svoje prednosti i nedostatke za uređaje s ograničenim resursima, posebno u IoT okruženju. Algoritmi kao što su Present i Speck, zbog svoje jednostavne strukture i optimiziranih tablica, idealni su za uređaje s ograničenim procesorskim kapacitetom i memorijom, nudeći dobar balans između sigurnosti i performansi. Algoritmi poput Simona i Twinea također pružaju dobar kompromis, no njihova implementacija može biti složenija, što može utjecati na učinkovitost u IoT uređajima s ograničenim resursima. Algoritmi Prince i Rectangle, koji naglašavaju energetsku učinkovitost, prikladni su za IoT uređaje koji zahtijevaju visoku energetsku učinkovitost, ali njihova složenost može otežati implementaciju. Pseudokodovi analiziranih algoritama omogućuju fleksibilnost u primjeni i prilagodbi za specifične IoT scenarije, a algoritmi poput Specka omogućuju prilagodbu ključeva i broja operacija, što ih čini prilagodljivima različitim sigurnosnim zahtjevima. Zaključno, izbor kriptografskog algoritma za IoT sustave ovisi o specifičnim sigurnosnim zahtjevima, performansama uređaja i energetskim potrebama, a rezultati istraživanja pomažu u boljem odabiru algoritama za konkretne primjene.

# Summary

This paper studies cryptographic algorithms intended for resource-constrained devices, with a particular focus on algorithms suitable for use in the context of the Internet of Things (IoT). Through the analysis of cryptographic algorithms such as Present, Simon, Speck, Twine, Prince, and Rectangle, it can be concluded that each of them has its advantages and disadvantages for devices with limited resources, especially in IoT environments. Algorithms such as Present and Speck, due to their simple structure and optimized tables, are ideal for devices with limited processing power and memory, offering a good balance between security and performance. Algorithms like Simon and Twine also provide a good compromise, but their implementation may be more complex, which can affect efficiency in IoT devices with limited resources. Algorithms like Prince and Rectangle, which emphasize energy efficiency, are suitable for IoT devices that require high energy efficiency, but their complexity may make implementation more difficult. The pseudocodes of the analyzed algorithms provide flexibility in application and adaptation to specific IoT scenarios, and algorithms like Speck allow for the adjustment of key lengths and the number of operations, making them adaptable to different security requirements. In conclusion, the choice of cryptographic algorithm for IoT systems depends on specific security requirements, device performance, and energy needs, and the results of this research help in better selecting algorithms for specific applications.

# Životopis

Rođena 8. rujna 1996. godine u Zagrebu. Nakon završetka osnovne škole srednjoškolsko obrazovanje započinje u Zagrebu u III.gimnaziji. Nakon prirodoslovno matematičkog smjera u gimnaziji odlučuje se za upis Prirodoslovno matematičkog fakulteta u Zagrebu, smjer matematika. Po završetku preddiplomskog studija uz razne računarske predmete odlučuje se za smjer Računarstvo i matematika na Prirodoslovno matematičkom fakultetu u Zagrebu. Tijekom studija, osim što je radila na svojim akademskim obvezama, aktivno je radila na različitim studentskim poslovima. Stekla je iskustvo u raznim sektorima, od rada u trgovinama, preko rada u centru za financijska vještačenja, do poslova u struci vezanih uz programiranje. Ova iskustva su joj pomogla u stjecanju praktičnih vještina te boljoj pripremi za izazove na tržištu rada.