Kriptiranje CAN komunikacije na ugradbenoj računalnoj platformi

Horvat, Iva

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij elektrotehnike

KRIPTIRANJE CAN KOMUNIKACIJE NA UGRADBENOJ RAČUNALNOJ PLATFORMI

Diplomski rad

Iva Horvat

Osijek, 2023. godina

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 15.09.2023.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime Pristupnika:	Iva Horvat
Studij, smjer:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. Pristupnika, godina upisa:	D-1365, 07.10.2021.
OIB studenta:	86031782704
Mentor:	izv. prof. dr. sc. Ratko Grbić
Sumentor:	,
Sumentor iz tvrtke:	Zvonimir Kaprocki
Predsjednik Povjerenstva:	prof. dr. sc. Mario Vranješ
Član Povjerenstva 1:	izv. prof. dr. sc. Ratko Grbić
Član Povjerenstva 2:	prof. dr. sc. Marijan Herceg
Naslov diplomskog rada:	Kriptiranje CAN komunikacije na ugradbenoj računalnoj platformi
Znanstvena grana diplomskog rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak diplomskog rada:	Controller Area Network (CAN) protokol danas predstavlja standardni način razmjene podataka unutar vozila između elektroničkih upravljačkih jedinica (eng. Electronic Control Unit). Međutim, s porastom broja ECU-a raste i mogućnost napada na takve sustave. Naime, postojeće ugrađene sigurnosne značajke CAN sabirnice su primarno dizajnirane za osiguranje pouzdane komunikacije, a ne za sigurnost. U okviru ovog diplomskog rada potrebno je analizirati moguće fizičke napade na CAN sabirnicu te predložiti načine zaštite, prvenstveno u obliku enkripcije podataka. Predloženi način zaštite demonstrirati implementacijom algoritama za enkripciju na ugradbenu računalnu platformu koja podržava CAN komunikaciju (npr. STM32).
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	15.09.2023.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 27.09.2023.

Ime i prezime studenta:	Iva Horvat
Studij:	Diplomski sveučilišni studij Elektrotehnika, smjer Komunikacije i informatika'
Mat. br. studenta, godina upisa:	D-1365, 07.10.2021.
Turnitin podudaranje [%]:	3

Ovom izjavom izjavljujem da je rad pod nazivom**: Kriptiranje CAN komunikacije na ugradbenoj računalnoj platformi**

izrađen pod vodstvom mentora izv. prof. dr. sc. Ratko Grbić

i sumentora ,

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. UVOD 2
2. PROBLEM SIGURNOSTI CAN KOMUNIKACIJE 4
2.1. Opis fizičkog i podatkovnog sloja CAN specifikacije 4
2.2. Nedostatci CAN komunikacije i mogući napadi 10
2.3. Pregled postojećih rješenja za ostvarenje sigurne CAN komunikacije 14
3. PRIJEDLOG ZAŠTITE CAN KOMUNIKACIJE NA UGRADBENOJ RAČUNALNOJ PLATFORMI 19
3.1. *Data Encryption Standard* 19
3.2. *Advanced Encryption Standard* 25
3.3. Implementacija algoritma kriptiranja zasnovanog na *Data Encryption Standardu* . 30
3.4. Implementacija algoritma kriptiranja zasnovanog na *Advanced Encryption Standardu* 35
3.5. Implementacija kriptirane CAN komunikacije na ugradbenoj računalnoj platformi 41
4. VERIFIKACIJA I EVALUACIJA PREDLOŽENOG RJEŠENJA ZAŠTITE CAN KOMUNIKACIJE NA UGRADBENOJ RAČUNALNOJ PLATFORMI 46
4.1. Verifikacije predloženog rješenja zaštite CAN komunikacije na osobnom računalu i na ugradbenoj računalnoj platformi 46
4.2. Evaluacija predloženog rješenja zaštite CAN komunikacije na ugradbenoj računalnoj platformi 52
5. ZAKLJUČAK 60
LITERATURA 61
SAŽETAK 64
ABSTRACT 65
ŽIVOTOPIS 66
PRILOZI 67

1. UVOD

Tehnološki napredak razvoja elektroničkih komponenti omogućio je široku primjenu elektronike u različitim industrijama. Upotreba elektronike u auto industriji omogućila je povećanje učinkovitosti, sigurnosti i udobnosti vožnje. Može se reći kako je elektronika srce modernih automobila. Naime, 1980. godine je cijena elektronike iznosila svega 10% ukupne cijene automobila. U 2010. godini taj se postotak popeo na 35%, a predviđa se kako će se u 2030. godini čak 50% ukupne cijene automobila odnositi na elektroniku [1]. Sustavom ubrizgavanja goriva, sustavom upravljanja rad motora, sustavom kočenja, sustavom mjerenja emisije plinova, sustavom zaključavanja i mnogim drugim sustavima upravljaju senzori i elektroničke upravljačke jedinice, ECU (engl. *Electronic Control Unit*). To su ugradbene računalne platforme koje obavljaju različite zadatke kako bi se osigurao pravilan rad različitih sustava u vozilu.

Za ispravan rad međusobno odvojenih podsustava, potrebno je ostvariti komunikaciju između različitih ECU-a. Dvosmjernu komunikaciju je moguće ostvariti različitim protokolima: LIN (engl. *Local Interconnect Network*) [2], CAN (engl. *Controller Area Network*) i FlexRay [3]. Odabir protokola ovisi o zahtjevima mreže, brzini i cijeni. CAN je osmišljen kao jednostavan i efikasan protokol za razmjenu informacija između ECU-a sa strogim vremenskim zahtjevima i bez ugrađenih sigurnosnih mehanizama. Kako broj ECU-a u modernim automobilima prilično naglo raste, CAN je postao jedna od kritičnih točaka sigurnosti sustava.

Naime, CAN protokol emitira poruke, ne provjerava identitet poruka, priključuje sve ECU-e na zajednički medij, te ne koristi kriptografiju. Navedena svojstva CAN komunikacije predstavljaju rizik jer neovlaštene osobe mogu utjecati na CAN komunikaciju u automobilu. Pri tome je napade moguće izvršiti izravnim fizičkim pristupom CAN mreži ili putem bežičnih sučelja [4]. Postoji opasnost od mijenjanja podataka, isključivanja ECU-a, lažiranja senzorskih podataka, isključivanja sigurnosnih sustava, kao što su primjerice ABS i zračni jastuci. Jasno je kako opisani scenariji mogu dovesti do situacija u kojima je ugrožena sigurnost automobila, vozača i ostalih sudionika u prometu.

Tijekom godina osmišljeno je nekoliko mjera zaštite CAN komunikacije. Radi se o kombinaciji nekih sigurnosnih mehanizama kao što su kriptiranje, digitalni potpisi, kontrola pristupa, utvrđivanje podatkovnog integriteta, autentikacija i drugo [4]. Izbor rješenja ovisi o složenosti mreže, zahtjevima mreže, složenosti implementacije zaštite, te cijeni. U svakom

rješenju nezaobilazan je mehanizam kriptiranja. Kriptiranje, enkripcija ili šifriranje je postupak u kojem se otvoreni tekst na temelju zadanog algoritma pretvara u nečitljiv oblik, šifrat, kako bi se njegov sadržaj zaštitio prilikom prijenosa. Pojam otvoreni tekst označava izvornu poruku koju pošiljalac želi poslati primaocu, a pojam šifrat označava naizgled nečitljivu poruku dobivenu kriptografskim postupkom. Porast količine ECU-a i količine podataka na CAN sabirnici zahtijevaju razvoj efikasnijih rješenja. S obzirom da vrlo mala latencija komunikacije predstavlja glavni uvjet sigurnog automobila, nije moguća upotreba zaštitnih rješenja koja uzrokuju velika kašnjenja u komunikaciji.

U okviru ovog diplomskog rada ostvarena je zaštita CAN komunikacije na ugradbenoj računalnoj platformi temeljena na *Data Encryption Standard* (DES) i *Advanced Encryption Standard* (AES) kriptografskim algoritmima. Algoritmi su najprije implementirani u programskom jeziku C u Eclipse razvojnom okruženju, nakon čega su prilagođeni za dostupnu ugradbenu računalnu platformu temeljenu na mikroupravljaču STM32F407. CAN komunikacija je ostvarena između dvije ugradbene računalne platforme uporabom HAL (engl. *Hardware Abstraction Layer*) CAN biblioteke u STM32CubeIDE razvojnom okruženju. Provedena je verifikacija i evaluacija predloženog rješenja zaštite CAN komunikacije.

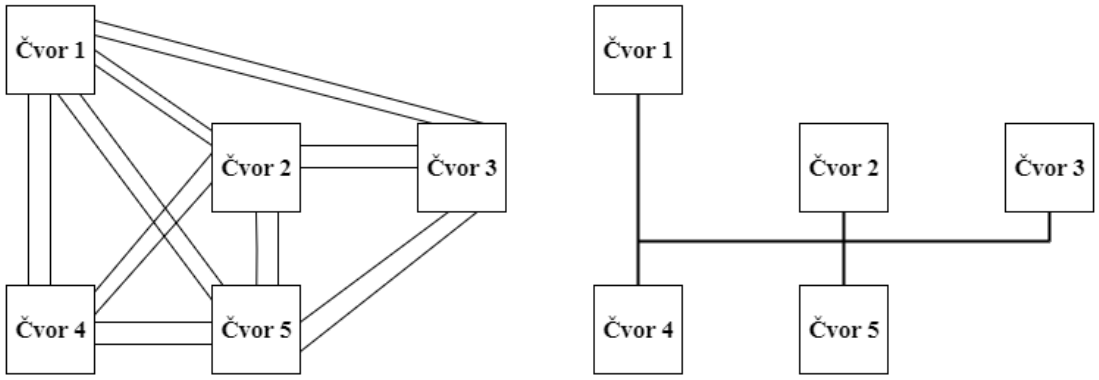
U nastavku rada, u drugom poglavlju je opisana specifikacija CAN protokola. Navedeni su sigurnosni nedostaci specifikacije, te neka od rješenja zaštite CAN komunikacije. Odabrani kriptografski algoritmi predstavljeni su u trećem poglavlju. U nastavku je dana implementacija algoritama kriptiranja zasnovana na odabranim kriptografskim standardima, te implementacija sigurne CAN komunikacije na odabranoj ugradbenoj računalnoj platformi. Četvrto poglavlje opisuje način verifikacije ispravnosti rada algoritama, te rezultate evaluacije na temelju vremena izvođenja programskog koda i razine prometa na sabirnici. U petom poglavlju dan je zaključak rada.

2. PROBLEM SIGURNOSTI CAN KOMUNIKACIJE

CAN je mrežna tehnologija osmišljena i razvijena za upotrebu u automobilskoj industriji od strane tvrtke Bosch. Osnovni motivi za razvoj potpuno novog komunikacijskog protokola uključivali su jednostavnost implementacije, te strogi stvarno-vremenski zahtjevi komunikacije. U ovom poglavlju opisana je specifikacija CAN protokola, princip pristupa zajedničkom mediju, primanje odgovarajućih poruka, detekcija i upravljanje pogreškama, te temeljna ograničenja komunikacije. Nadalje, navedeni su nedostaci specifikacije, sigurnosni problemi, te nekoliko postojećih rješenja navedenih problema.

2.1. Opis fizičkog i podatkovnog sloja CAN specifikacije

U veljači 1986. godine na SAE (engl. *Society of Automotive Engineers*) kongresu u Detroitu inženjeri Kiencke, Dais i Litschel predstavili su novi mrežni sustav kao *„Automotive Serial Controller Area Network“* [5]. Ovaj mrežni sustav podrazumijeva serijsku komunikaciju između više čvorova ili elektroničkih upravljačkih jedinica (ECU). Na slici 2.1. a) prikazan je primjer topologije sa nekoliko čvorova, gdje su svi čvorovi međusobno povezani. Glavni nedostaci tako dizajnirane mreže su veliki troškovi same izrade, te kompleksnost održavanja. Na slici 2.1. b) se nalazi primjer topologije CAN mrežnog sustava, gdje je svaki čvor povezan na jednu zajedničku sabirnicu, čime su smanjeni troškovi izrade i kompleksnost održavanja.



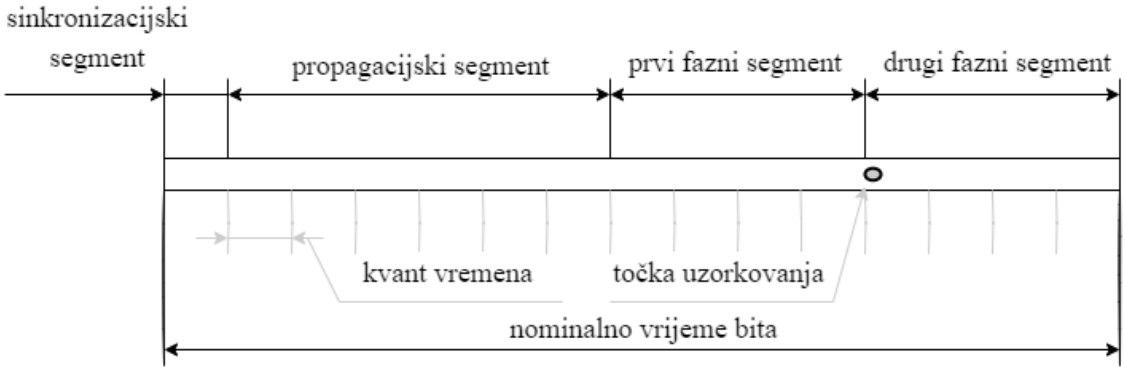
a) b)

Slika 2.1. Primjer topologije mreže, (a) bez CAN standarda, (b) sa CAN standardom [6]

Generalni princip rada CAN komunikacije se temelji na emitiranju poruka: određeni čvor emitira svoju poruku sabirnicom, a preostali čvorovi na temelju filtriranja identifikatora poruke primaju ili ignoriraju emitiranu poruku. U slučaju kada dva čvora pokušavaju istovremeno

emitirati poruku, prioritet slanja određen je samom vrijednosti identifikatora, čime su izbjegnute kolizije između poruka koje se šalju. Omogućena je detekcija pogrešaka, zahtjev ponovnog slanja oštećenih poruka, te isključivanje neispravnih čvorova.

Izvorna CAN specifikacija definira kodiranje bita, nominalno vrijeme bita i sinkronizaciju, a kasniji ISO (engl. *International Organization for Standardization*) 11898-2 standard [7] detaljnije definira fizički sloj. Korišteno je NRZ (engl. *Non-Return-to-Zero*) kodiranje bita koje osigurava visoku otpornost na vanjske smetnje. Kako bi svaki čvor točno očitavao vrijednost bita koji se trenutno prenosi sabirnicom, definiran je sinkronizacijski protokol koji koristi padajuće bridove za sinkronizaciju čvorova. S obzirom da protokol koristi bridove, odnosno prijelaze iz logičke jedinice u logičku nulu, duge sekvence bitova jednakih vrijednosti izbjegnute su umetanjem bitova (engl. *bit stuffing*). Umetanje bitova je definirano na način da se svaki šesti uzastopni bit zamijeni bitom suprotne vrijednosti. Osim sinkronizacije, na taj način je ostvarena i mogućnost detekcije pogreške u prijenosu. U svrhu sinkronizacije, definirano je i nominalno vrijeme bita (Slika 2.2.) koje se sastoji od četiri dijela: sinkronizacijski segment, propagacijski segment, prvi fazni segment i drugi fazni segment.

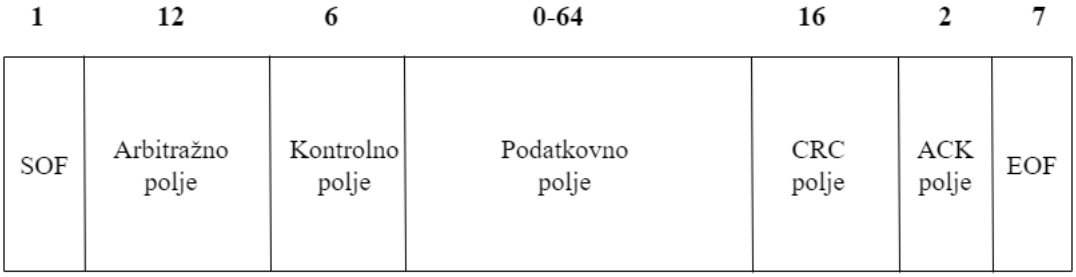


Slika 2.2. Nominalno vrijeme bita [8]

Sinkronizacijski segment predstavlja referentni interval u kojemu se očekuje početni brid bita. Propagacijski segment služi za kompenzaciju vremena potrebnog za propagaciju signala između dva najudaljenija čvora. Fazni segmenti se koriste za kompenzaciju faznih grešaka u položaju početnog brida bita. Oni se mogu produžiti ili skratiti kako bi se ponovno sinkronizirao položaj sinkronizacijskog segmenta s obzirom na brid sljedećeg bita. Točka uzorkovanja označava vremenski trenutak u kojem se određuje vrijednost bita koji se prenosi sabirnicom, a nalazi se između dva fazna segmenta. Veličine određenih segmenata definirane su preko kvanta

vremena, odnosno minimalne rezolucije u definiciji trajanja bita i najveće pretpostavljene pogreške za bitno orijentirane sinkronizacijske protokole. Tako je sinkronizacijski segment veličine jednog kvanta, a propagacijski segment i suma faznih segmenata veličine između jednog i osam kvanta uz uvjet da je drugi fazni segment jednaku maksimumu između prvog faznog segmenta i vremena procesiranja informacije, koje je uvijek manje ili jednako dva kvanta. Specifikacijom su definirane dvije vrste sinkronizacije, čvrsta sinkronizacija (engl. *hard synchronization*) i resinkronizacija (engl. *re-synchronization*). Čvrsta sinkronizacija se odvija na početku okvira detekcijom padajućeg brida SOF (engl. *Start Of Frame*) bita na način da kraj trenutnog kvantuma postaje kraj sinkronizacijskog segmenta, što osigurava da brid SOF bita leži unutar sinkronizacijskog segmenta. Resinkronizacija se odvija tijekom prijenosa detekcijom padajućih bridova, a uključuje skraćenje ili produljenje faznih segmenata, što osigurava da brid sljedećeg bita počinje unutar sljedećeg sinkronizirajućeg segmenta.

Podatkovni sloj definirana četiri vrste CAN okvira: podatkovni okvir (engl. *data frame*) koji sadrži same podatke, okvir zahtjeva *(engl. remote frame)* koji zahtijeva prijenos podatkovnog okvira s jednakim identifikatorom, okvir pogreške (engl. *error frame*) koji se prenosi u slučaju detekcije pogreške, te okvir preopterećenja (engl. *overload frame*) koji se koristi za kontrolu toka. Na slici 2.3. prikazana je struktura podatkovnog okvira, te veličina svakog dijela okvira izražena u bitovima.



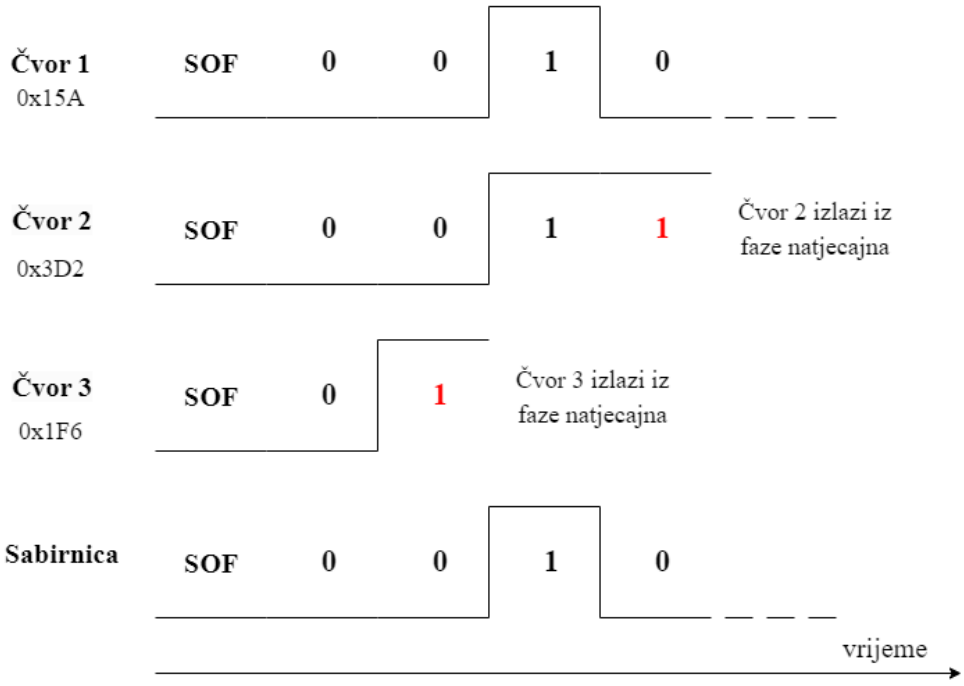
Slika 2.3. Struktura podatkovnog CAN okvira [8]

CAN okvir se sastoji od SOF bita, arbitražnog polja, kontrolnog polja, podatkovnog polja, CRC (engl. *Cyclic Redundancy Check*) polja, ACK (engl. *ACKnowledgement*) polja i EOF (engl. *End Of Frame*) bitova. U standardnom CAN-u, arbitražno polje se sastoji od jedanaest bitnog identifikatora i RTR (engl. *Remote Transmission Request*) bita koji je postavljen u vrijednost nula ako se radi o podatkovnom, a u vrijednost jedan ako se radi o okviru zahtjeva. Identifikator je jedinstvena oznaka okvira na temelju koje se vrši emitiranje i filtriranje okvira. Pri tome je važno napomenuti kako u istom vremenskom trenutku u CAN mreži ne mogu

postojati dva okvira sa jednakom vrijednosti identifikatora. Vrijednost koja je ranije pridijeljena nekom okviru se može dodijeliti novom okviru tek nakon što je izvršen prijenos prethodnog okvira. Kontrolno polje se sastoji od IDE (engl. *IDentifier Extension*) bita, jednog rezerviranog bita, te četiri DLC (engl. *Data Length Content*) bita. IDE bit je postavljen u vrijednost jedan ako se radi od standardnom identifikatoru, odnosno u vrijednost nula ako se radi o proširenom identifikatoru. DLC bitovi daju informaciju o tome kolika je veličina podatkovnog polja, a mogući raspon je od nula do osam bajtova. CRC polje služi za detekciju pogreške. Vrijednost CRC polja se definira prilikom slanja okvira, a jednaka je ostatku dijeljenja ulaznog polinoma i generatorskog polinoma $x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$. Pri tome je ulazni polinom određen bitovima od SOF bita do početka podatkovnog polja. Nakon primitka okvira ponovno se računa vrijednost CRC polja na jednak način. Ako postoji razlika između izračunate vrijednosti i vrijednosti pohranjene unutar CRC polja primljenog okvira, detektirana je pogreška u prijenosu. ACK bit je postavljen u vrijednost nula od strane čvora pošiljatelja, dok čvor primatelj signalizira uspješno primanje okvira postavljanjem ACK bita u vrijednost jedan. Oba polja, CRC i ACK, završavaju po jednim graničnim bitom postavljenim u vrijednost nula.

Dodjela prava emitiranja čvorovima temelji se na prioritetu identifikatora okvira kojeg čvor šalje. U trenutku kada se sabirnica nalazi u stanju mirovanja, čvorovi koji imaju okvire za slanje ulaze u fazu natjecanja za pristup zajedničkom mediju: sinkroniziraju se na bridu SOF bita, počinju emitirati identifikatore svojih okvira i osluškuju sabirnicu. Identifikatori se emitiraju bit po bit, pri čemu se nad bitovima na istim težinskim položajima u identifikatoru izvodi logička operacija I. Na taj način na sabirnici opstaje samo jedna rezultantna vrijednosti bita na pojedinom težinskom položaju. Svaki čvor koji emitira identifikator ujedno i osluškuje sabirnicu. Ako pojedini čvor očita rezultantnu vrijednost bita identifikatora različitu od vrijednosti svog emitiranog bita identifikatora, povlači se iz natjecanja za pristup zajedničkom mediju. Proces traje sve dok se ne emitira posljednji bit identifikatora. U tom trenutku na sabirnici je opstao jedan rezultantni identifikator. Čvor koji utvrdi da je pročitao identifikator jednak identifikatoru okvira koji želi poslati shvaća da je dobio pristup zajedničkom mediju, te počinje prijenos ostatka podatkovnog okvira. Na taj način je izbjegnuta kolizija okvira prilikom prijenosa. Proces dodijele prava emitiranja je neprekidan, ako se pojave novi čvorovi spremni za emitiranje okvira, oni ne sudjeluju u trenutnom procesu, nego osluškuju sljedeće stanje mirovanja sabirnice. Također, ranije je napomenuto kako u CAN mreži istovremeno ne mogu postojati dva okvira s jednakim identifikatorima. U slučajevima kada se u istom trenutku započne slanje okvira zahtjeva i podatkovnog okvira, čije su vrijednosti identifikatora jednake,

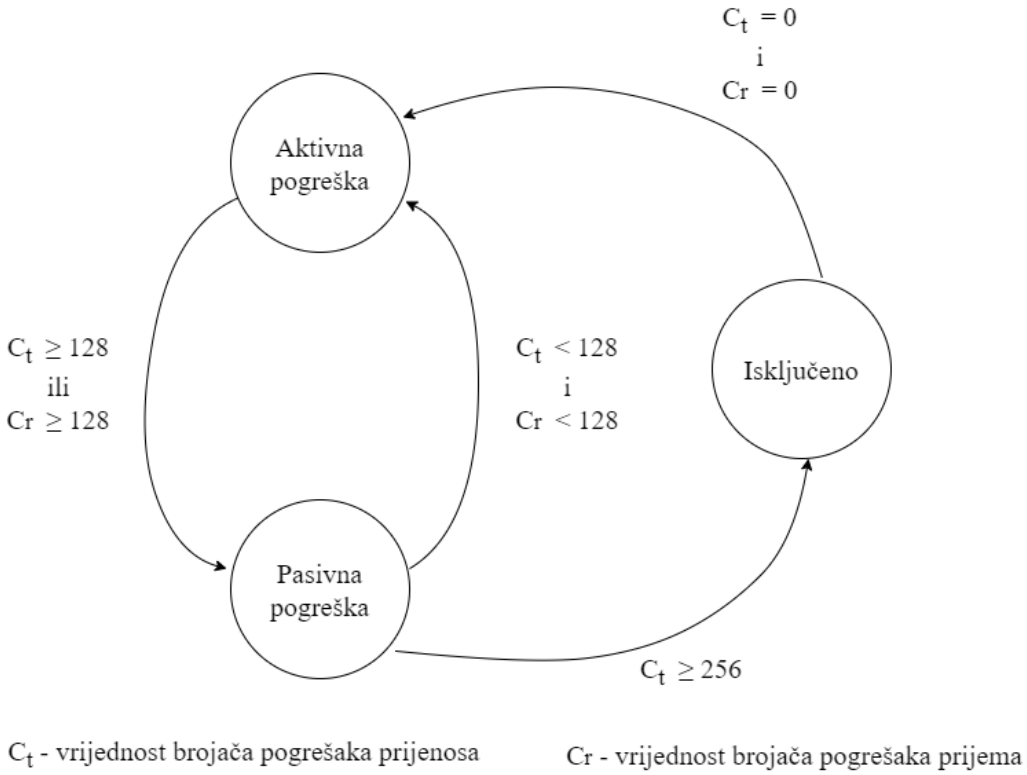
jedinstvenost arbitražnog polja je ostvarena RTR bitom. Vrijednost RTR bita podatkovnog okvira daje veći prioritet podatkovnom okviru u odnosu na okvir zahtjeva. Odnosno, okvir zahtjeva ispada u fazi natjecanja za pristup zajedničkom mediju i čeka sljedeći proces dodijele prava emitiranja. Na slici 2.4. prikazan je primjer dodjele prava emitiranja između tri čvora s heksadecimalnim vrijednostima identifikatora okvira 0x15A, 0x3D2, 0x1F6. Sva tri čvora emitiraju prvi bit nula, rezultat logičke operacije I je nula, pa svi čvorovi ostaju u fazi natjecanja. Sljedeći emitirani bit iznosi nula za čvorove jedan i dva, a jedan za čvor tri. Rezultat logičke operacije I iznosi nula, pa na sabirnici opstaje rezultantni bit vrijednosti nula. Čvor tri kojemu je emitirani bit iznosio jedan očitava rezultatnu vrijednost nula, detektira promjenu vrijednosti i izlazi iz faze natjecanja. Čvorovi jedan i dva očitavaju rezultantnu vrijednost bita jednaku vrijednostima njihovih emitiranih bitova, te ostaju u fazi natjecanja. Preostali čvorovi emitiraju sljedeći bit vrijednosti jedan, a kako rezultantna vrijednost iznosi jedan, oba čvora ostaju u fazi natjecanja. U sljedećem koraku čvor jedan emitira bit nula, dok čvor dva emitira bit jedan. Rezultat logičke operacije I iznosi nula pa osluškujući čvor dva izlazi iz faze natjecanja. Preostalom čvoru jedan dodijeljena su prava za emitiranje zajedničkim medijem. Iz ovog primjera vidljivo je kako uvijek najveći prioritet ima okvir s najmanjom vrijednosti identifikatora. Također, potvrđen je veći prioritet podatkovnog okvira u odnosu na okvir zahtjeva, što proizlazi iz RTR bita postavljenog u vrijednost nula jer u kontekstu CAN komunikacije logička nula dominanta vrijednost bita.



Slika 2.4. Primjer dodjele prava emitiranja čvorovima na CAN sabirnici [8]

S obzirom da se u CAN komunikaciji ne koristi adresiranje čvorova, pojedini čvorovi moraju primljene okvire filtrirati na temelju njihovih identifikatora. Svaki čvor sadrži registar u koji pohranjuje primljeni okvir, maske i filtere. Maska definira nad kojim bitovima identifikatora filtri trebaju djelovati kako bi detektirali moguće podudaranje. Primjerice, ako maska sadrži bitove jedan na prvom, petom i osmom položaju, samo ti bitovi identifikatora se uspoređuju sa definiranim filtrima. Ako maskirani identifikator odgovara nekom od filtera na čvoru, okvir se dalje obrađuje, u protivnom čvor odbacuje okvir jer nije njemu namijenjen.

Protokol može detektirati pet vrsta pogrešaka: pogreška bita (engl. *bit error*) se detektira u trenutku osluškivanja prilikom prijenosa okvira ako čvor detektira neispravan bit, pogreška umetanja bitova (engl. *stuff error*) se detektira pojavljivanjem šest uzastopnih bitova iste vrijednosti u poljima okvira nad kojima se vrši umetanje bitova, CRC pogreška se detektira ako se proračunata CRC vrijednost razlikuje od primljene, pogreška oblika (engl. *form error*) se detektira ako fiksno oblikovano polje bitova sadrži jedan ili više nedozvoljenih bitova, ACK pogreška se detektira ako predajnik detektira vrijednost ACK bita postavljenu u nula. Svaki čvor sadrži dva brojača pogrešaka, brojač pogrešaka prijenosa (engl. *transmit error counter*) i brojač pogrešaka prijema (engl. *receive error counter*). Ovisno o iznosu tih brojača, čvor može biti u stanju aktivne pogreške, stanju pasivne pogreške ili isključenom (engl. *bus-off)* stanju. Kada se čvor nalazi u stanju aktivne pogreške, te otkrije neku od pet navedenih pogreška, povećava svoj brojač pogrešaka prijenosa za osam, a brojače pogrešaka prijema preostalih čvorova za jedan. Detekcija pogreške na čvoru u stanju aktivne pogreške uzrokuje odbacivanje primljene poruke na svim čvorovima u mreži. Ako vrijednost brojača pogrešaka prijenosa ili vrijednost brojača pogrešaka prijema postane veća od stotinu dvadeset i osam, čvor iz stanja aktivne pogreške prelazi u stanje pasivne pogreške. Detekcija pogreške na takvom čvoru neće uzrokovati odbacivanje poruke na svim čvorovima u mreži i takav čvor čeka određeno vrijeme prije nego što ponovno može sudjelovati u natjecanju za pristup zajedničkom mediju. Ako vrijednost brojača pogrešaka prijenosa poprimi vrijednost veću od dvije stotine pedeset i šest, čvor prelazi u isključeno stanje i više ne sudjeluje u razmjeni podataka. Svaki uspješan prijem poruke smanjuje iznose brojača. Čvor iz stanja pasivne pogreške može prijeći u stanje aktivne pogreške ako vrijednost oba brojača padne ispod vrijednosti stotinu dvadeset i osam, te iz isključenog stanja može prijeći u stanje aktivne pogreške ako vrijednost oba brojača iznosi nula. Opisana stanja, prijelazi i uvjeti prelaska iz jedno u drugo stanje čvora ilustrirani su na slici 2.5.



Slika 2.5. Dijagram stanja čvorova prema CAN protokolu [8]

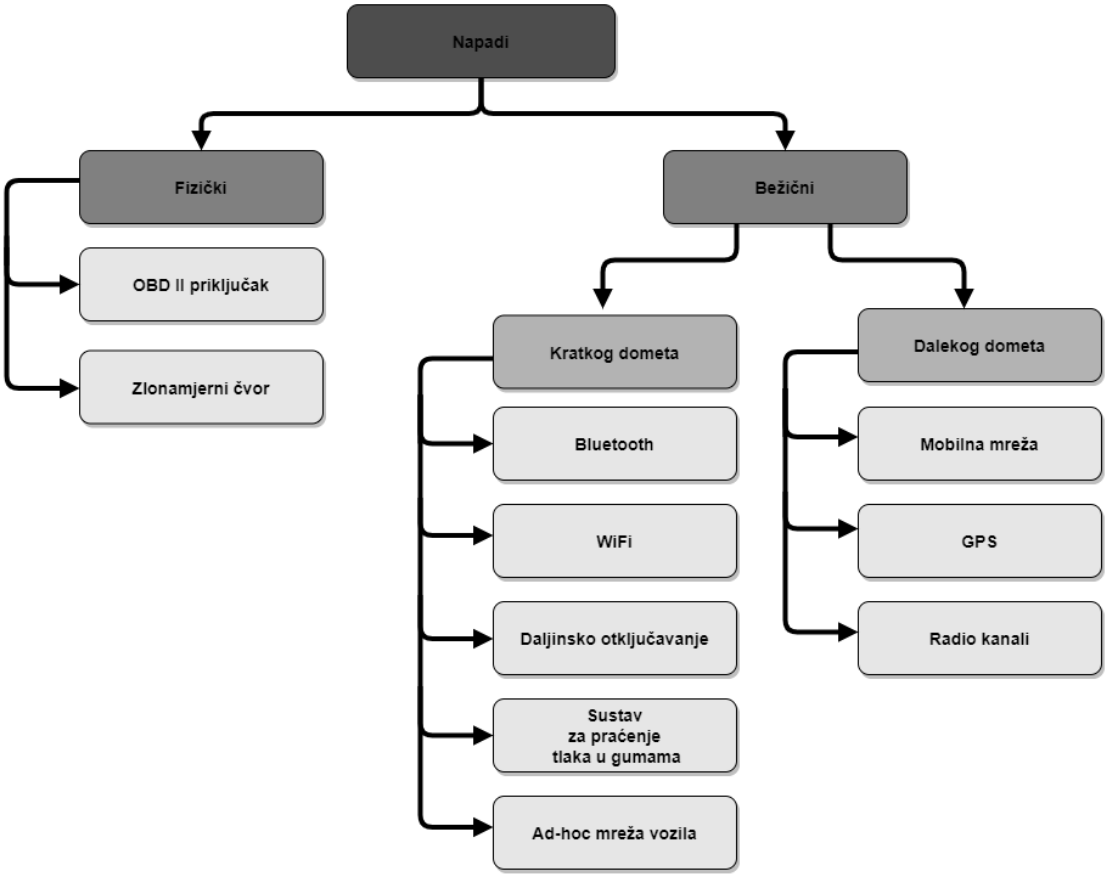
Detekcija pogreške na čvoru u aktivnom stanju se propagira na druge čvorove slanjem šest uzastopnih bitova vrijednosti jedan. Uzastopno slanje šest jednakih bitova krši pravilo umetanja bitova, a kao posljedica toga, svi ostali čvorovi otkrivaju pogrešku umetanja bitova i počinju redom slati bitove za pogrešku. Dakle, okvir pogreške je rezultat procesa signalizacije pogreške, duljine između šest i dvanaest bitova, a završava graničnikom pogreške (engl. *error delimiter*) duljine osam bita.

2.2. Nedostatci CAN komunikacije i mogući napadi

Sigurna komunikacija zadovoljava tri osnovna kriterija: povjerljivost, integritet i dostupnost. Pojam povjerljivosti označava mogućnost pristupa podacima samo ovlaštenim osobama. Kako CAN ne koristi nikakve kriptografske algoritme, napadač može pristupiti podacima i time narušiti privatnosti. Dakle, povjerljivosti nije osigurana. Integritet se odnosi na točnost, cjelovitost i valjanost podataka. CAN ima CRC provjeru kako bi se osigurao integritet uslijed grešaka pri prijenosu, no ni na koji način nije spriječen unos podataka od strane napadača, što narušava integritet. Pojam dostupnosti označava stanje mreže u kojemu ovlašteni korisnici

mogu koristiti sustav u bilo kojem trenutku. S obzirom na ranije spomenuti princip pristupa zajedničkom mediju koji se temelji na prioritetu poruka, ako se umetne poruka s najvišim prioritetom, mreža će biti nedostupna čvorovima koji šalju poruke nižeg prioriteta. Dakle, CAN protokol ne zadovoljava nijedan od kriterija, pa je komunikacija nesigurna.

Kibernetički napad je svaki namjerni pokušaj krađe, razotkrivanja, izmjene, onemogućavanja ili uništavanja podataka, aplikacija ili druge imovine putem neovlaštenog pristupa mreži, računalnom sustavu ili digitalnom uređaju. Razlikujemo fizički napad ostvaren izravnim pristupom sabirnici, te bežični napad putem bežičnih sučelja. Izravan pristup može se ostvariti putem OBD-II (engl. *On-Board Diagnostic*) priključka ili zlonamjernog čvora. OBD je podsustav koji prikuplja i pohranjuje informacije iz mreže senzora unutar vozila. Svrha takvog podsustava je pohraniti određene veličine pomoću kojih je moguće dijagnosticirati probleme i kvarove na automobilu. Priključak na OBD podsustav je standardiziran 1994. godine pod nazivom OBD-II, a danas je postao norma u svim modernim automobilima zbog prikupljanja informacija o emisiji štetnih plinova [9]. Pojam zlonamjerni čvor opisuje čvor unutar komunikacijske mreže koji je pod kontrolom napadača, a svojim radom narušava normalan i ispravan rad sustava. Pri tome pojam čvor označava bilo kakav uređaj izravno povezan na sabirnicu. Bežični napad može se ostvariti preko mnogobrojnih bežičnih sučelja koja se nalaze u modernim automobilima, a koja su potrebna kako bi automobil ostvario komunikaciju sa vanjskim uređajima. Neki od primjera bežičnih sučelja su sustav pasivne protuprovalne zaštite, sustav za praćenje tlaka u gumama, Bluetooth i radio. Slika 2.6. detaljnije ilustrira podjelu kibernetičkih napada prema načinu pristupa CAN mreži.



Slika 2.6. Podjela napada prema načinu pristupa CAN mreži [10]

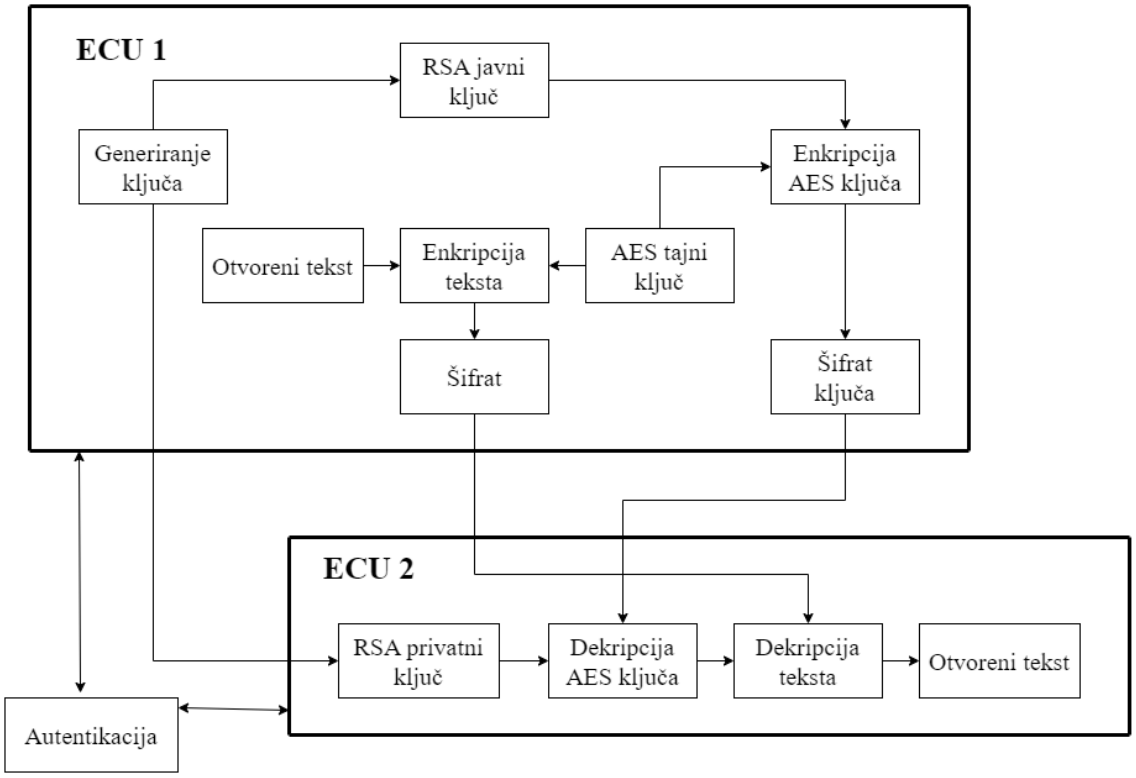
Ovisno o učinku napada na komunikacijsku mrežu, napad može biti gotovo bezazlen ili vrlo ozbiljan. U radu [11] je navedeno kako se izravnim fizičkim pristupom ostvaruje pristup svim čvorovima mreže, čime je moguće upravljati kočnicama i motorom vozila. Također, napadač može izvesti napad uskraćivanjem resursa (engl. *Distributed Denial-Of-Service attack*). Primjerice, generiranjem podatkovnih okvira s identifikatorom najvišeg prioriteta zlonamjerni čvor sprječava drugim čvorovima u mreži pristup dijeljenom mediju. Moguće je ometati normalan rad mreže konstantnim generiranjem pogrešaka, koje rezultiraju ponovnim emitiranjem okvira. Osim nad osnovnim funkcijama automobila, moguće je izvesti napad primjerice nad radiom ili uređajima za osvjetljenje unutar automobila. Iako uglavnom bezazleni, iznenadno bljeskanje unutarnjeg osvjetljenja ili preglasna reprodukcija zvučnog zapisa može uplaši vozača, što može imati kobne posljedice [12]. Kako bi mreža bila zaštićena, bežična sučelja komuniciraju s CAN-om putem pristupnog čvora GECU (engl. *Gateway Electronic Control Unit*). Međutim, istraživanja [13] su dokazala mogućnost kompromitacije GECU-a i pristup izoliranom CAN-u. Mogući napadi uključuju krađu automobila, te

upravljanje motorom, kočnicama i brisačima. Poseban sigurnosni problem predstavlja udaljeno isporučivanje softverskih nadogradnji (engl. *over-the-air update*) preko kojeg napadač može instalirati zlonamjerni softver [14]. U bliskoj budućnosti, automobili će biti opremljeni komunikacijom između vozila (engl. *vehicle-to-vehicle*) i komunikacijom između vozila i infrastrukture (engl. *vehicle-to-infrastructure*). Tako stvorena mreža za cilj ima optimizaciju prometa i izbjegavanje sudara, a kako bi to bilo ostvarivo, mreža mora imati pristup senzorima u automobilima bežičnom vezom. Navedeno otvara mogućnost za slanje krivotvorenih poruka, što može rezultirati poremećajem komunikacijske mreže unutar automobila.

Prilikom razvoja sigurnosnih rješenja koriste se četiri mehanizma: segmentacija mreže, autentikacija, otkrivanje upada i kriptografija. Segmentacijom CAN mreže omogućava se kontrola pristupa pojedinim segmentima. Na taj način je moguće odvojiti sigurnosno kritične funkcije automobila od na primjer informacijsko-zabavnog sustava (engl. *infotainment*). Nedostatak ovakvog pristupa je povećanje troškova mreže i teže održavanje mreže. Mehanizam autentikacije ovjerava autentičnosti poruke, te povezuje poruku sa njenim izvorom. Tako je moguće detektirati zlonamjerne poruke, čvor koji je njihov izvor, te isključenje kompromitiranog čvora iz komunikacije. Nedostatak ovog mehanizma je nekompatibilnost sa tradicionalnim CAN-om, te povećanje prometa i računalnog opterećenja čvorova. Metode za otkrivanje upada mogu se podijeliti na detekciju upada temeljenu na potpisima (engl. *signature*) i detekciju upada temeljenu na anomalijama. Detekcija temeljena na potpisima provjerava poznate napade u bazi podataka, što zahtijeva redovito ažuriranje za nove napade. Metoda temeljena na anomalijama analizira ponašanje mreže i prepoznaje odstupanje od normalnog ponašanja. Točnost je obično niža nego kod detekcije temeljene na potpisima, no za razliku od detekcije temeljene na potpisima, moguće je otkriti tip napada koji se ne nalazi u bazi poznatih napada. Kriptografija podrazumijeva uporabu različitih kriptografskih algoritama u svrhu zaštite sadržaja poruka. Dakle, prednost kriptografije leži u tome da se poruke razmjenjuju samo između sudionika komunikacije. Treća strana koja nema informaciju o kriptografskim algoritmima i ključevima neće biti u mogućnosti pristupiti informacijama koje nisu njoj namijenjene. Glavni problemi primjene kriptografije leže u podatkovnom polju ograničene veličine, ograničenoj računalnoj snazi čvorova, mogućnosti proboja statičkog tajnog ključ, te kašnjenje na čvorovima s ograničenim resursima. U sljedećem potpoglavlju dan je pregled nekoliko postojećih rješenja koja kombinacijom nekih od sigurnosnih mehanizama ostvaruju sigurnu CAN komunikaciju.

2.3. Pregled postojećih rješenja za ostvarenje sigurne CAN komunikacije

Većina postojećih rješenja u CAN komunikaciju ugrađuju kriptografiju s-kraja-na-kraj (engl. *end-to-end*) određenim algoritmom i mogućnost autentikacije ECU-a, dok neka rješenja predstavljaju nove pakete protokola koji se nadograđuju na CAN protokol. Jedno od rješenja kriptografije s-kraja-na-kraj predstavljeno je u radu [15] gdje je kombinacijom simetričnog AES algoritma i nesimetričnog RSA (engl. *Rivest-Shamir-Adleman*) algoritma (Slika 2.7.) zaštićen sadržaj CAN okvira. AES koristi jedan tajni ključ za postupak enkripcije i dekripcije, pri čemu je očuvanje tajnosti ključa nužno za sigurnost podataka. RSA koristi dva ključa, obično jedan javni ključ za postupak enkripcije koji je dostupan svima, te drugi tajni privatni ključ za postupak dekripcije. U radu je opisana sigurna razmjena CAN poruke između ECU pošiljatelja i ECU primatelja u sedam koraka. U prvom koraku RSA algoritam, koji se temelji na problemu faktorizacije velikih brojeva, generira javni i privatni ključ koje nije moguće odrediti od strane napadača u razumnom vremenu. Nakon toga se provodi enkripcija poruke AES algoritmom uporabom AES tajnog ključa, te enkripcija AES tajnog ključa uporabom javnog ključa RSA algoritma. Obje enkriptirane poruke se šalju prema ECU primatelju koji odgovara autentikacijskim signalom. U šestom i sedmom koraku algoritma se provodi dekripcija AES ključa pomoću privatnog ključa RSA algoritma, te dekripcija same CAN poruke pomoću tajnog AES ključa. Prednost predstavljenog rješenja je svakako jednostavnost implementacije. Kriptografski algoritmi ne zahtijevaju dodatne uređaje, nego su ostvareni na već postojećim ECU-ima. Nedostatak predloženog rješenja leži u procesu autentikacije koji nije detaljno specificiran. Navedeno je samo slanje nedefiniranog autentikacijskog signala, pri čemu nije navedeno što se događa u scenariju kada odgovor sa potvrdnim signalom ne dolazi do ECU-a.



Slika 2.7. Blok dijagram rješenja zaštite CAN komunikacije uporabom AES i RSA kriptografskih algoritama [15]

U radu [16] je predstavljen novi protokol nazvan AuthentiCAN, koji se temelji na CAN-FD (engl. *Controller Area Network Flexible Data-rate*), a implementira kriptografiju s-kraja-na-kraj i mogućnost autentikacije. AuthentiCAN razlikuje četiri stanja prijenosa poruka. Stanja se razlikuju pomoću prva dva najvažnija bita (engl. *most significant bit)* u šezdeset i četiri bajtnom podatkovnom dijelu CAN-FD okvira, a navedena stanja su 00 - emitiranje javnog ključa, 01 - slanje popisa za jednokratnu upotrebu, 10 - slanje poruke i 11 - sinkronizacija popisa za jednokratnu upotrebu. Prilikom faze emitiranja javnog ključa, ECU šalje CAN okvir čiji podatkovni dio sadrži javni ključ. Preostali ECU-i spojeni na sabirnicu prilikom primitka poruke pohranjuju javni ključ ako već nije spremljen u njihovoj lokalnoj memoriji, te emitiraju svoje javne ključeve. Ovaj princip izbjegava unaprijed kodirane (engl. *hardcoded*) parove ključeva koji se koriste u drugim rješenjima, što se pokazalo kao izvor nesigurnosti. Naime, ako napadač pristupi ECU-u koji ima pohranjen ključ u svojoj lokalnoj memoriji, tada on može dekriptirati cijeli sadržaj te komunikacijske mreže. Nakon razmjene javnih ključeva, prvi ECU sastavlja popis za jednokratnu uporabu i kodira ga javnim ključem drugog ECU-a. Tada drugi ECU može kodirati onoliko poruka koliko je jednokratnih ključeva dobio u primljenom popisu.

Autori navode veličinu jednokratnog ključa od osam bita kako bi se podatkovni dio CAN-FD popunio sa što više elemenata liste, čime se smanjuje količina CAN-FD okvira poslanih tijekom komunikacije za razmjenu jednokratnih listi. Takav dizajn algoritma je posebno važan kod CAN mreža s velikim brojem ECU-a. U fazi razmjene poruka ECU pošiljatelj pomoću svog javnog ključa enkriptira sadržaj CAN okvira koji je ulančan s jednokratnim ključem koji je primio od ECU-a kojem želi poslati poruku, te briše iskorišteni jednokratni ključ sa popisa ključeva. Na taj način poruku može dekriptirati samo određeni ECU primatelj koji ima kopiju tog istog jednokratnog ključa lokalno pohranjenu. ECU-i ulaze u fazu sinkronizacije popisa za jednokratnu upotrebu samo ako su iscrpili prethodno pohranjenu listu za određeni ECU kojem žele poslati podatke. Sama faza je jednaka inicijalnom slanju popisa s jednokratnim ključevima. U predstavljenom radu autori opisuju scenarij kod kojeg, uslijed gubitka poruke ili pogreške na poruci, dva ECU-a ne mogu komunicirati uslijed nesinkroniziranih popisa jednokratnih ključeva. Ako ECU odbije unaprijed određeni broj poruka od istog izvorišnog ECU-a, algoritam detektira da je došlo do desinkronizacije u popisima i taj problem rješava ulaskom u fazu sinkronizacije. Autori navode postojanje ranijih rješenja koja su izbjegla unaprijed kodirane ključeve uporabom GECU-a. Uloga GECU-a u takvim rješenjima je dodjela ključa ostalim ECU-ima u mreži. Prednost predstavljenog rješenja je uklanjanje tzv. *single point of vulnerability* koji postoji u ranije predstavljenim radovima. Nedostatak rješenja leži u činjenici da svaka detekcija pogreške uslijed koje dolazi do desinkronizacije popisa inicira fazu sinkronizacije. Faza sinkronizacije traje određeno vrijeme, a poruke koje emitiraju ključ predstavljaju redundanciju u komunikaciji.

U radu [17] predstavljen je paket protokola za autentikaciju entiteta, upravljanje ključevima, siguran protok okvira zahtjeva i ažuriranje ključa sesije prilikom povezivanja vozila sa vanjskim uređajima. Uvode dva nova protokola, RTRP (*engl. RTR frame transmission Protocol*) i NSKUP (engl. *New Session Key Update Protocol*), te modificiraju ISDP (*engl. Initial Session key Distribution Protocol*), SKUP(engl. *Session Key Update Protocol*) i VCP (engl. *Vehicle Connectivity with an external device Protocol*) protokole kako bi ispravili nedostatke postojećeg seta protokola. ISDP protokol uspostavlja inicijalni ključ sesije između ECU-a i GECU-a, varijanta je AKEP2 (engl. *Authenticated Key Exchange Protocol 2*) protokola i osigurava uzajamnu autentikaciju entiteta i implicitnu autentikaciju ključa. S obzirom da se ključevi prethodne sesije koriste za generiranje ključeva sljedeće sesije, glavni nedostatak standardnog ISDP protokola je činjenica da poznavanje ključa k-te sesije ugrožava svaku k+1 sesiju. Stoga, u radu je predložena tzv. tajnost ključa sesije (engl. *session key*

secrecy) gdje ECU-i koji sudjeluju u komunikaciji dogovaraju ključ sesije koji nikako nije vezan za ključeve prethodnih sesija kojima je napadač mogao pristupiti. Nadalje, ECU koji zahtjeva podatke šalje okvir zahtjeva s postavljenim RTR bitom i identifikatorom traženog data okvira. RTRP protokol na temelju inkrementa brojača ECU-a koji sudjeluju u razmjeni podataka, osigurava autentikaciju i integritet okvira zahtjeva. Za samu razmjenu podataka korišten je DFP (engl. *Data Frame transmission Protocol*) protokol koji koristi AES-128 enkripciju i 32-bitni MAC (engl. *Media Access Control*) za autentikaciju poruke. SKUP protokol periodično ažurira ključ sesije. Glavni nedostatak ovog protokola, kao i kod ISDP protokola je ovisnost ključeva budućih sesija o ključevima prethodnih sesija. Autori navode ranije pokušaje rješenja ovog problema gdje se ključ sesije ažurira prema unaprijed postavljenim vremenskim okidačima, uslijed prelijevanja vrijednosti brojača ECU-a ili prilikom odspajanja vozila s vanjskog uređaja. Nedostatak ovog rješenja je u nesinkroniziranosti brojača ostalih ECU-ova i ECU-a kod kojeg se dogodilo prelijevanje, te činjenica da odspajanje s vanjskog uređaja može detektirati samo GECU koji ne može pokrenuti ažuriranje brojača kod drugih ECU-a jer za to ne postoji protokol. Predlažu svoje rješenje gdje je veličina brojača dovoljno velika, pa ne dolazi do prelijevanja, a ažuriranje ključa provodi GECU sa svakim ECU-om. NSKUP protokol ažurira ključ sesije VCP protokola kako bi spriječio napade uslijed ponovne upotrebe starog ključa. Protokol prati kriptografsku razmjenu modificiranog VCP protokola. VCP protokol uspostavlja ključ između GECU-a i vanjskog uređaja. Koristi ECDH (engl. *Elliptic Curve Diffie-Hellman*) algoritam za razmjenu ključa, no autori navode kako VCP protokol nije dobro definiran, te postoje opasnosti od napada ponavljanjem (engl. *replay attack*), te ugroženost sigurnosti budućih ključeva koji ovise o prethodnim ključevima. U svom radu autori predstavljaju modificirani VCP protokol koji na temelju digitalnog potpisa povezuje poruke sa sesijom, pa je mogućnost ponavljanja poruka eliminirana. Autori navode kako prednost predstavljenog rješenja u odnosu na ranije postojeća rješenja leži u efektivnoj zaštiti od napada pretvaranjem (engl. *masquerade attack*), napada ponavljanjem i napada uskraćivanja resursa. Ipak, kao nedostatak ističu potrebu za dodatnim sklopovljem kako bi se implementirala sigurna komunikacija, te dodatne troškove same implementacije.

U radu [18] je navedeno kako ranije osmišljena rješenja koja štite CAN sabirnicu od napada pretvaranjem zauzimaju 100% kapaciteta mreže ili zahtijevaju dodatan hardver. Kao rješenje navedenog problema, predstavljen je novi protokol pod nazivom MAuth-CAN, koji ujedno štiti i od napada isključivanja. Rješenje se temelji na postojanju jednog ECU-a kojeg autori nazivaju

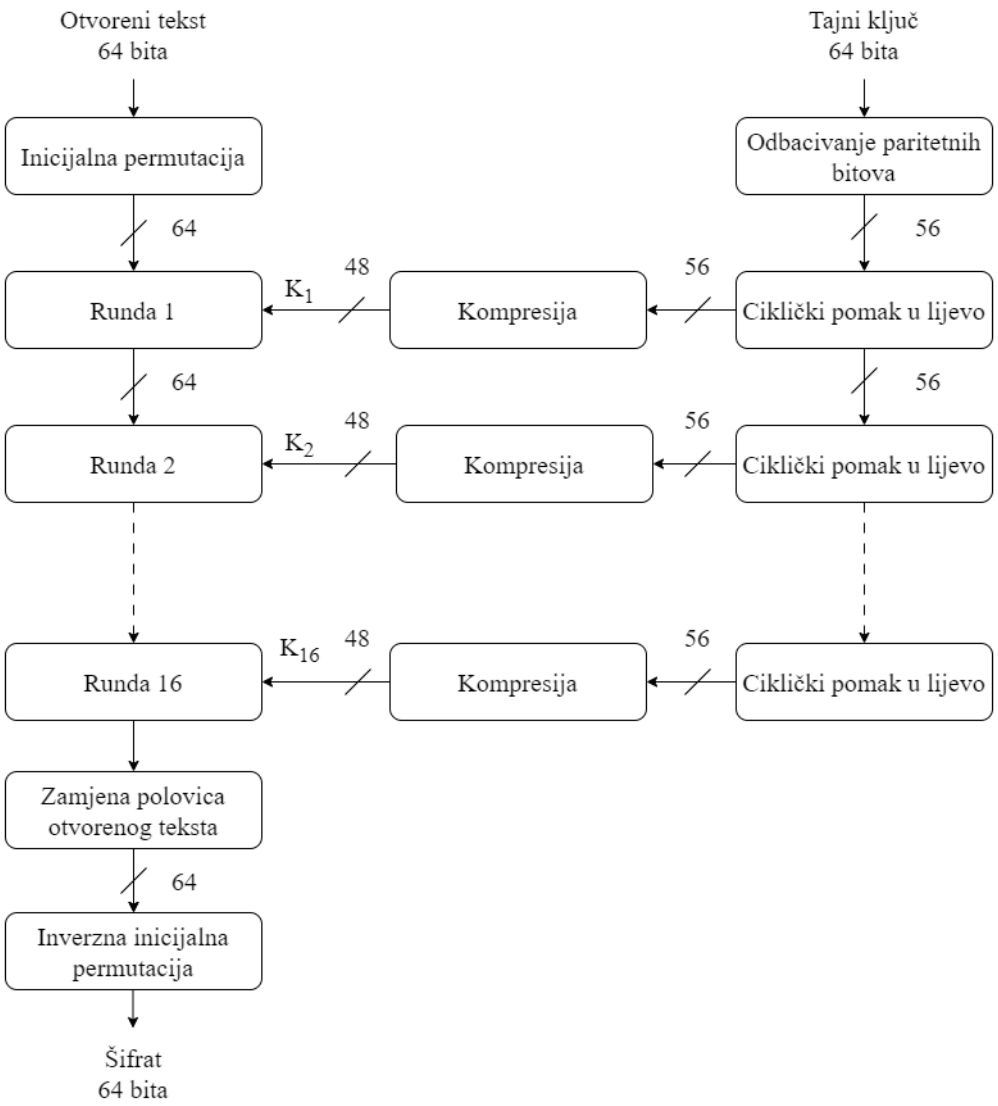
provjeritelj (engl. *authenticator*), koji u svrhu autentikacije komunicira sa svim ostalim ECU-ima. Osnovna pretpostavka je da je ECU provjeritelju prilikom izrade vozila dodijeljen tajni ključ za simetričnu enkripciju, te da napadač ne može kompromitirati sigurnost ECU provjeritelja. MAuth-CAN se sastoji od tri faze. U prvoj fazi, fazi inicijalizacije, ECU provjeritelj prema tablici trajnih ključeva ostalih ECU-a svakom od ECU-a generira i dodjeljuje njegov ključ sesije. Ova faza se temelji na AKEP2 protokolu koji omogućava uzajamnu autentikaciju entiteta i razmjenu ključeva. Nakon inicijalizacije, ECU-i periodično šalju poruke na sabirnicu. ECU pošiljatelj računa MAC vrijednost na temelju ključa koji mu je dodijeljen prilikom inicijalizacije, te šalje poruku i MAC vrijednost na sabirnicu. U ovoj fazi, preostali ECU-i čekaju unaprijed određeno vrijeme tijekom kojeg ECU provjeritelj ovjeri poruku ECU pošiljatelja. Autentikacija se provodi pomoću primljene MAC vrijednosti i ključa koji je provjeritelj dodijelio tom ECU prilikom inicijalizacije. Ako je autentikacija uspješna, ECU primatelji primaju poruku i komunikacija se nastavlja. U slučaju neuspješne autentikacije ECU povjeritelj generira poruku prijave (engl. *report message*) koja sprječava ECU primatelje da procesiraju lažnu zlonamjernu poruku. Autori predlažu adaptivno definiranje vremena tijekom kojeg ECU-i čekaju autentikaciju poruke, čime bi se vrijeme ažuriralo u skladu sa stanjem na CAN sabirnici. Posljednja faza predstavlja ažuriranje ključeva sesije, gdje ECU provjeritelj na temelju trajnih ključeva, nasumične vrijednosti generirane u fazi inicijalizacije, vrijednosti brojača i identifikatora ECU-a, generira nove ključeve sesije. Time se izbjegava prelijevanje brojača ECU-a i štiti ključ od napada uslijed analize uzastopnih MAC vrijednosti koje se računaju prilikom autentikacije poruka. Kako bi algoritam štitio od napada uskraćivanjem resursa na centralni ECU provjeritelj postavljena su dva vremenska zahtjeva. Ako napadač koristi poruke s točnim MAC vrijednostima vrijeme potrebno da ECU provjeritelj ovjeri poruku mora biti kraće nego vrijeme potrebno za transmisiju komprimirane poruke. Ako napadač koristi poruke s netočnim MAC vrijednostima moguć je scenarij da ECU provjeritelj generira toliko poruka prijave da dođe do napada uskraćivanjem resursa. Kako bi se spriječio ovaj scenarij potrebno je osigurati da je suma vremena potrebnog da ECU provjeritelj potvrdi poruku, te vremena potrebnog da ECU provjeritelj generira poruku prijave bude manja od samog vremena prijenosa poruke prijave. Osnovna prednost predstavljenog rješenja u odnosu na druga postojeća rješenja je manje zauzimanje kapaciteta CAN sabirnice, te implementaciju bez dodatnog hardware-a. Glavni nedostatak rješenja je scenarij u kojem napadač na neki način dođe do tajnog ključa za simetričnu enkripciju koji je dodijeljen vozilu prilikom izrade ili ako uspije kompromitirati ECU provjeritelj.

3. PRIJEDLOG ZAŠTITE CAN KOMUNIKACIJE NA UGRADBENOJ RAČUNALNOJ PLATFORMI

U ovom poglavlju opisan je prijedlog zaštite CAN komunikacije na ugradbenoj računalnoj platformi koji se temelji na implementaciji kriptografskih algoritama. Prva dva potpoglavlja sadrže informacije o standardima korištenih kriptografskih algoritama. Nakon toga je detaljno predstavljena implementacija vlastite enkripcije i dekripcije, a prikazan je i dijagram toka algoritama. Naposljetku poglavlja je opisana korištena ugradbena računalna platforma za implementaciju kriptirane CAN komunikacije. Priložena je shema spoja i detaljan opis podešavanja i programiranja mikroupravljača STM32F407 na kojemu je temeljena ugradbena računalna platforma.

3.1. *Data Encryption Standard*

National Bureau of Standards, danas poznat pod nazivom NIST (engl. *National Institute of Standards and Technology*) predstavlja *Data Encryption Standard* (DES) 1977. godine. Standard definira simetrični blok algoritam enkripcije i dekripcije s tajnim ključem. Temelji se na načelu koje je definirao Horst Feistel, a koje glasi kako možemo aproksimirati idealnu blok šifru koristeći koncept produktne šifre (engl. *product cipher*), što uključuje izvršavanje dvije ili više jednostavnih blok šifri, odnosno operacija, jednu za drugom na način da je konačni rezultat ili produkt kriptografski jači od bilo koje od komponentnih blok šifri. Operacije uključuju supstituciju - zamjenu svakog elementa otvorenog teksta s nekim drugim elementom u šifratu, te transpoziciju - permutaciju ili premještanje elemenata otvorenog teksta. DES koristi blokove otvorenog teksta veličine šezdeset i četiri bita, te tajni ključ veličine pedeset i šest bitova, a kao rezultat daje blokove šifrata veličine šezdeset i četiri bita. Slika 3.1. prikazuje DES postupak enkripcije i dekripcije.



Slika 3.1. DES postupak enkripcije i dekripcije [19]

Blok otvorenog teksta prolazi redom inicijalnu permutaciju, šesnaest rundi DES algoritma, zamjenu lijeve i desne polovice bloka, te inverznu inicijalnu permutaciju. Usporedno, tajni ključ inicijalne veličine šezdeset i četiri bita ispušta paritetne bitove na svakom osmom položaju, nakon čega ulazi u šesnaest rundi proračuna potključeva veličine četrdeset i osam bitova. Svaka runda kao ulaz uzima lijevu i desnu polovicu bloka iz prethodne runde, te potključ iteracije prema izrazima (3-1) i (3-2).

$$L_i = R_{i-1} \tag{3-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i) \tag{3-2}$$

gdje je:

- L_{i-1} – lijeva polovica otvorenog teksta prethodne runde,

- R_{i-1} – desna polovica otvorenog teksta prethodne runde,
- L_i – lijeva polovica otvorenog teksta trenutne runde,
- R_i – desna polovica otvorenog teksta trenutne runde,
- K_i – potključ trenutne runde.

Inicijalna permutacija je transpozicija bitova bloka otvorenog teksta definirana tablicom 3.1. Tablica prikazuje nove položaje pojedinih bitova, tako da će se pedeset i osmi bit ulaznog bloka sada nalaziti na prvom mjestu, pedeseti bit ulaznog bloka na drugom mjestu, četrdeset i drugi bit ulaznog bloka na trećem mjestu, i tako dalje sve do sedmog bita ulaznog bloka na posljednjem mjestu. Inverzna inicijalna permutacija se vrši ekvivalentno prema tablici 3.2.

Tablica 3.1. Transpozicijska tablica inicijalne permutacije u postupku DES enkripcije i dekripcije [19]

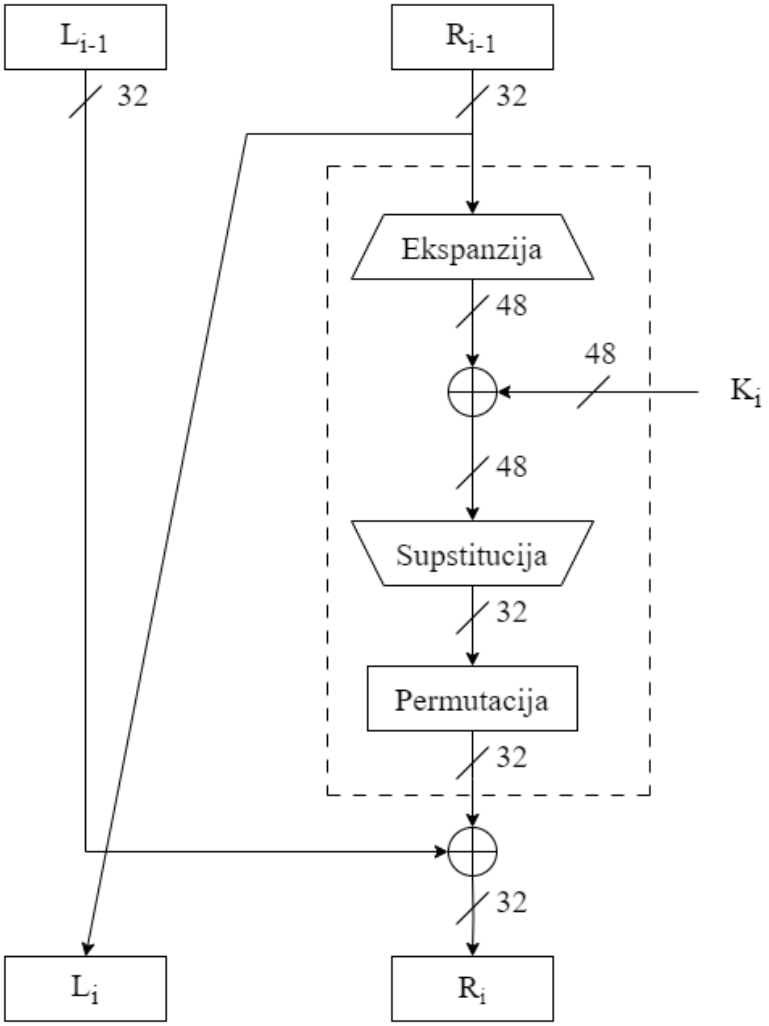
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Tablica 3.2. Transpozicijska tablica inverzne inicijalne permutacije u postupku DES enkripcije i dekripcije [19]

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Jedna runda DES algoritma prikazana na slici 3.2. uključuje operacije ekspanzije, zbrajanja, supstitucije i permutacije. Ekspanzija se vrši nad trideset i dva bitnom desnom polovicom bloka otvorenog teksta prema tablici 3.3. tako da se određeni bitovi ponavljaju čime se ostvaruje

proširenje do četrdeset i osam bita. Nakon ekspanzije, dobivena polovica bloka se zbraja logičkom operacijom isključivo ILI sa potključem runde. Supstitucija se izvodi pomoću osam tablica tzv. S-kutija u prilogu P.2.1., čiji su elementi cijeli brojevi između nula i petnaest. U svaku tablicu ulazi šest bitova na način da prvi i šesti bit označavaju red tablice, a drugi, treći, četvrti i peti bit označavaju stupac tablice. Kao rezultat se dobiva element na zadanom retku i zadanom stupcu, zapisan pomoću četiri bita, što za osam tablica iznosi ukupno trideset i dva bita. Posljednji korak runde uključuje permutaciju prema tablici 3.4.



Slika 3.2. Prikaz operacija jedne runde algoritma DES enkripcije [19]

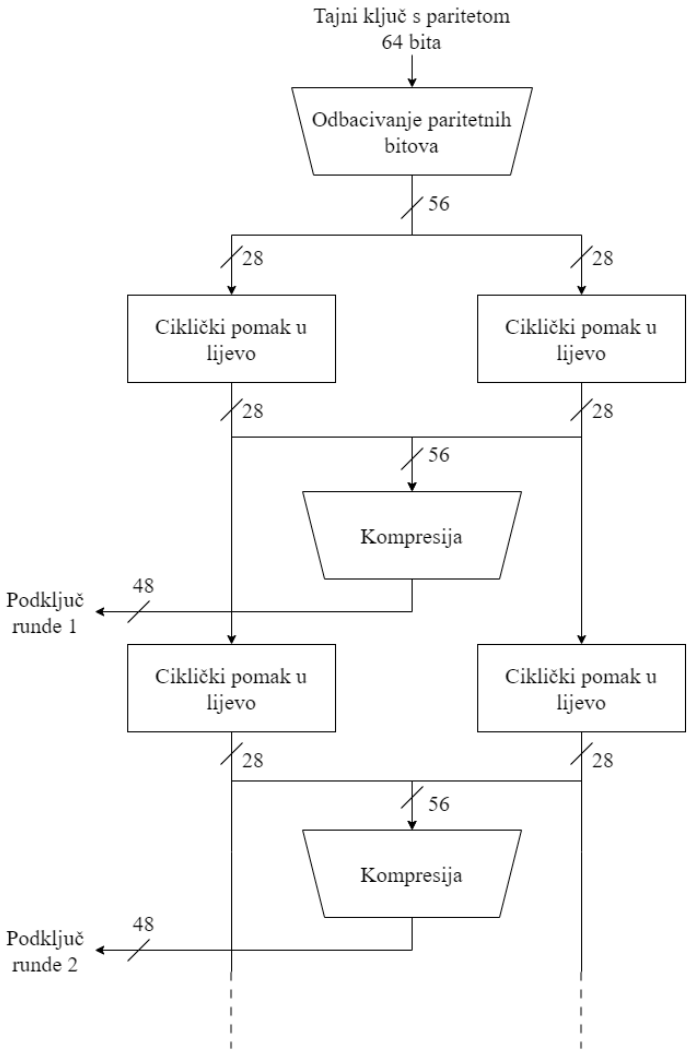
Tablica 3.3. Ekspanzijska tablica u postupku DES enkripcije i dekripcije [19]

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Tablica 3.4. Permutacijska tablica u postupku DES enkripcije i dekripcije [19]

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Jedna runda proračuna potključa (slika 3.3.) uključuje ciklički pomak polovica ključa u lijevo i kompresiju. Ovisno o broju runde, polovice ključa se pomiču u lijevo za jedan ili dva bita. Nakon spajanja polovica ključa vrši se kompresija čime se dobije četrdeset i osam bitni potključ runde. Svaka sljedeća runda kao ulaz uzima lijevu i desnu polovicu bloka iz prethodne runde. Prilikom enkripcije, zamjena lijeve i desne polovice bloka prije inverzne permutacije omogućava postupak dekripcije identičan postupku enkripcije, pri čemu se potključevi rundi koriste obrnutim redoslijedom.



Slika 3.3. DES algoritam proračuna potključeva rundi [20]

Sigurnost DES standarda leži u efektu lavine (engl. *avalanche effect*), nelinearnosti supstitucije pomoću S-kutija i velikom broju rundi. Efekt lavine je svojstvo enkripcijskog algoritma da za malu promjenu u otvorenom tekstu ili ključu rezultira velikim promjenama u šifratu. Za DES algoritam je pokazano [19] da nakon osme runde šifrat postaje slučajna funkcija bitova otvorenog teksta i ključa. Napad poznatog otvorenog teksta (engl. *known plaintext attack*) je tehnika kriptoanalize kod koje napadač s pristupom otvorenom tekstu može otkriti ključ nakon čega može dekriptirati sve šifrate dobivene s tim ključem. Jedina nelinearna operacija DES algoritma, supstitucija pomoću S-kutija, pruža zaštitu od takvog napada. Veliki broj rundi DES algoritma osigurava neučinkovitost napada diferencijalnom kriptoanalizom (engl. *differential cryptanalysis attack*). Napad diferencijalnom kriptoanalizom vrsta je napada odabranog otvorenog teksta (engl. *chosen plaintext attack*) koji analizira parove otvorenih

tekstova umjesto pojedinačnih otvorenih tekstova, tako da napadač može odrediti kako ciljani algoritam radi kada naiđe na različite vrste podataka s ciljem dolaska do informacije o korištenom ključu. Glavni nedostatak DES standarda je mala veličina ključa od pedeset i šest bitova što daje 7.2×10^{16} mogućih ključeva. S razvojem tehnologije algoritam je postao podložan napadu grubom silom (engl. *brute-force attack*) gdje napadač isprobava svaki od mogućih ključeva dok ne pronađe onaj koji je korišten. Računalo koje je grubom silom dekriptiralo DES algoritam za pedeset i šest sati konstruirala je *Electronic Frontier Foundation* 1998. godine [21]. Tablica 3.5. prikazuje prosječno vrijeme potrebno za pronalazak ključa DES i AES algoritama, ovisno o njegovoj veličini, napadom grube sile uporabom današnjih računala koja koriste više središnjih procesorskih jedinica.

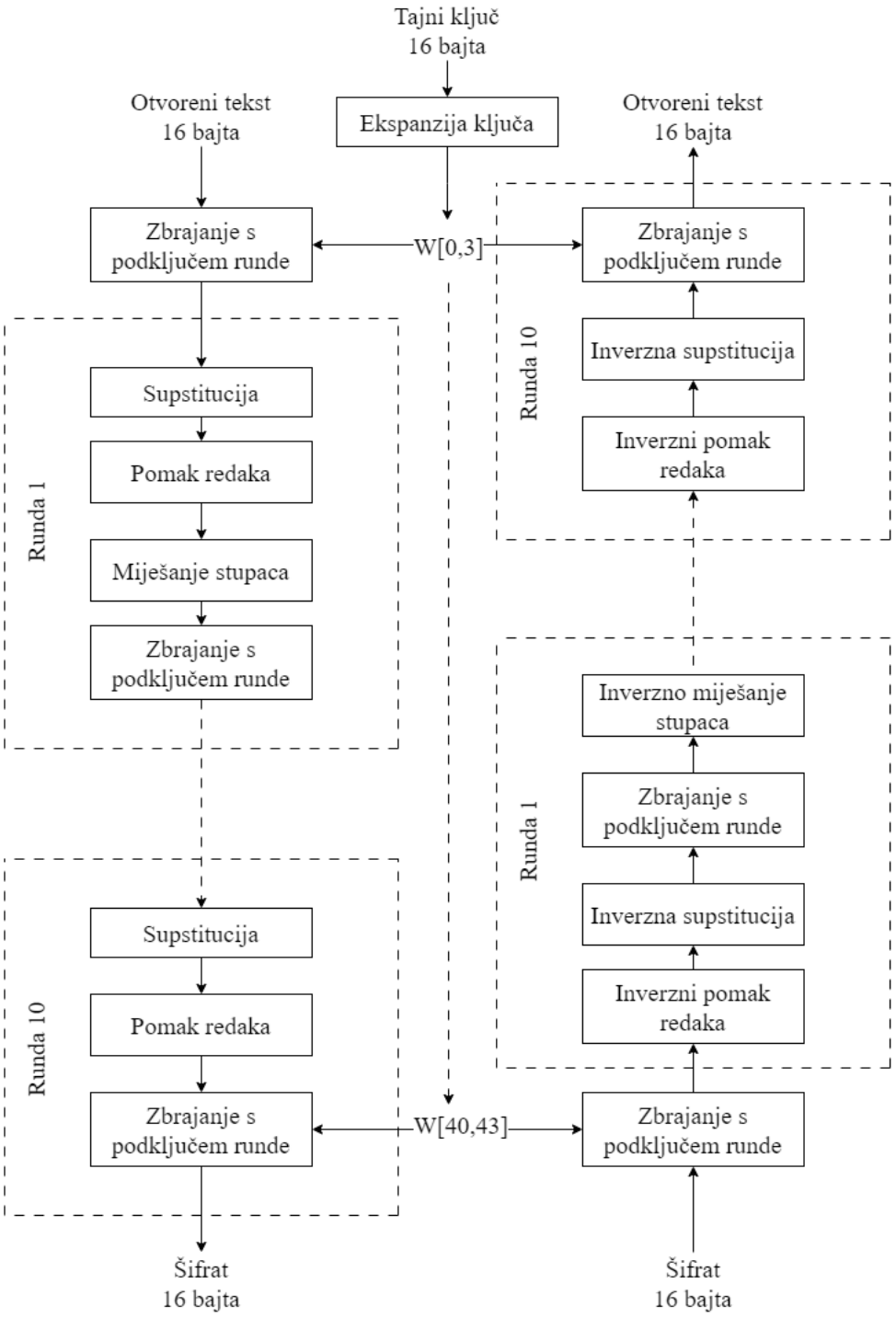
Tablica 3.5. Prosječno vrijeme potrebno za pronalazak ključa DES i AES algoritama pomoću napada grubom silom [21]

Veličina ključa [bit]	**Kriptografski algoritam**	**Broj mogućih ključeva**	**Vrijeme potrebno za razbijanje enkripcije prilikom 10^{13} proračuna u sekundi**
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	1 sat
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	5.3×10^{17} godina
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	5.8×10^{29} godina
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	9.8×10^{36} godina
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	1.8×10^{56} godina

3.2. *Advanced Encryption Standard*

Kriptografi Joan Daemen i Vincent Rijmen razvili su simetrični blok algoritam enkripcije i dekripcije s tajnim ključem pod nazivom RIJNDAEL, koji 2001. godine NIST objavljuje kao *Advanced Encryption Standard* (AES). Algoritam koristi blokove otvorenog teksta i daje blokove šifrata veličine stotinu dvadeset i osam bita. Standard definira tajni ključ triju mogućih veličina i broj rundi u ovisnosti o veličini ključa: deset rundi za ključ od stotinu dvadeset i osam bitova, dvanaest rundi za ključ od stotinu devedeset i dva bita, te četrnaest rundi za ključ od dvjesto pedeset i šest bitova. Algoritam obrađuje blok otvorenog u obliku matrice bajtova

veličine 4x4. Na slici 3.4. a) prikazan je AES postupak enkripcije, dok je na slici 3.4. b) prikazan AES postupak dekripcije.



Slika 3.4. AES postupak enkripcije (a) i dekripcije (b) [19]

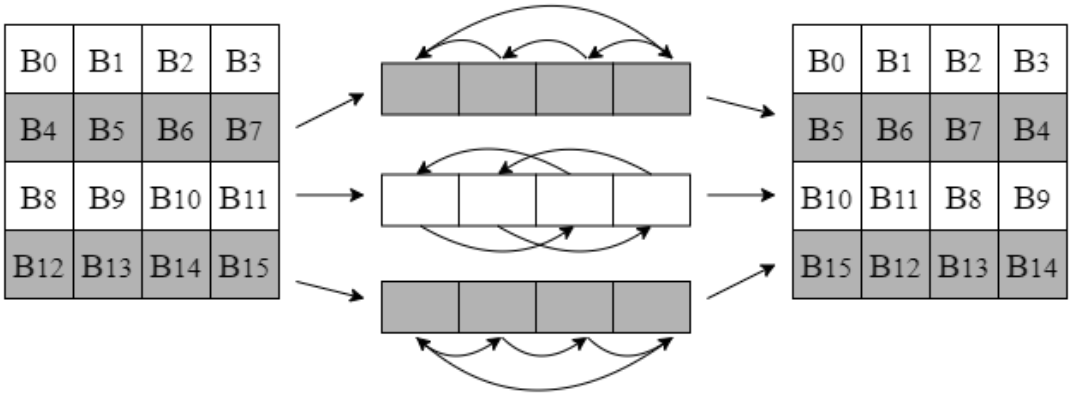
Nulta runda uključuje inicijalnu operaciju zbrajanja s ključem, nakon čega se provodi zadani broj rundi AES algoritma, sa izuzetkom operacije miješanja stupaca u posljednjoj rundi. Prilikom dekriptiranja koriste se operacije inverzne operacijama korištenima u postupku enkripcije, te obrnuti redoslijed potključeva. Jedna runda AES algoritma (Slika 3.4.) uključuje operacije supstitucije, pomaka redova, miješanja stupaca, te zbrajanja. Operacija supstitucije i njen inverz, izvode se pomoću S-kutije (Tablica 3.6.) i inverzne S-kutije (Tablica 3.7.). Za svaki pojedinačni bajt, prva četiri bita određuju redak tablice, a druga četiri bita stupac tablice. Kao rezultat se dobiva novi bajt zapisan u tablici u zadanom retku i stupcu.

Tablica 3.6. Supstitucijska S-kutija u postupku AES enkripcije[19]

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Tablica 3.7. Inverzna supstitucijska S-kutija u postupku AES dekripcije [19]

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D



Slika 3.5. Operacija pomak redova u postupku AES enkripcije [19]

Operacija pomak redova ciklički pomiče i-ti redak matrice za i mjesta ulijevo kako je to prikazano na slici 3.5. Njoj inverzna operacija vrši odgovarajući ciklički pomak u desno. Kod operacije miješanja stupaca i inverzne operacije miješanja stupaca svaki bajt iz stupca se mijenja novom vrijednosti koja predstavlja funkciju od svih bajtova u tom stupcu, prema formulama (3-3) i (3-4).

$$MA = A' \tag{3-3}$$

$$NB' = B \tag{3-4}$$

$$M = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \quad (3\text{-}5)$$

$$N = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \quad (3\text{-}6)$$

gdje je:

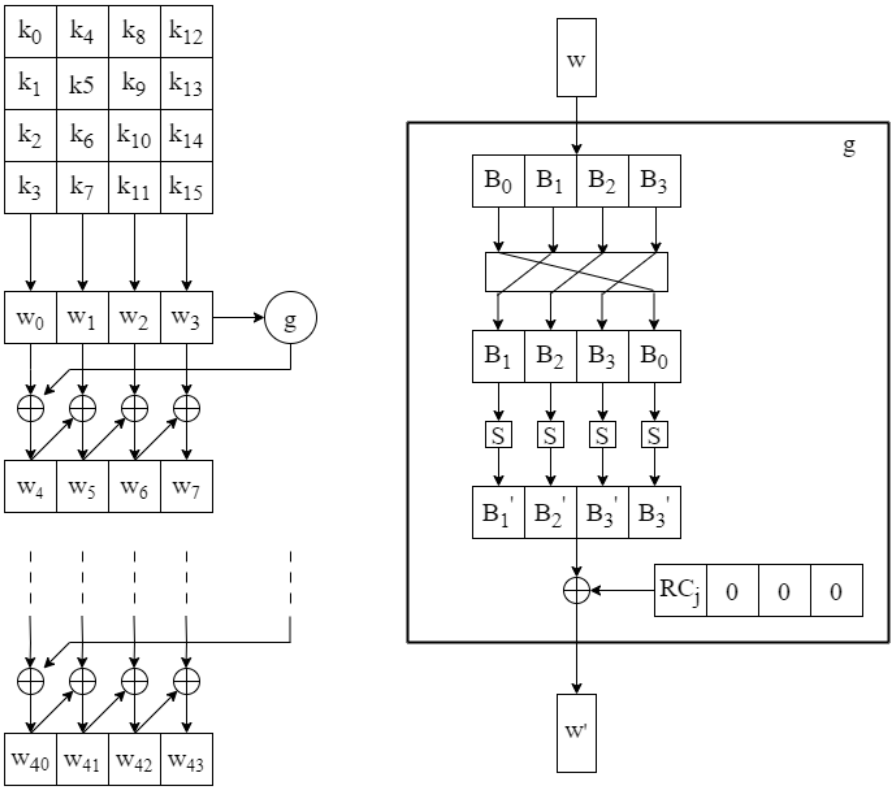
- A – ulazna matrica operacije miješanja stupaca,
- A' – rezultantna matrica operacije miješanja stupaca,
- M – matrica koeficijenata operacije miješanja stupaca,
- B – ulazna matrica inverzne operacije miješanja stupaca,
- B' – rezultantna matrica inverzne operacije miješanja stupaca,
- N – matrica koeficijenata inverzne operacije miješanja stupaca.

Zbrajanja i množenja kod operacija miješanja stupaca, te inverznog miješanja stupaca, provode se unutar Galois polja $GF(2^8)$. Koeficijenti matrica M i N osiguravaju maksimalnu distancu između kodnih riječi. Nakon miješanja stupaca, dobivena matrica se zbraja logičkom operacijom isključivo ILI sa potključem runde. Inverzna operacija zbrajanja je identična jer je logička operacija isključivo ILI sama sebi inverz.

Algoritam za ekspanziju ključa prikazan je na slici 3.6. Tajni ključ sastoji se od četiri početne riječi, svaka veličine četiri bajta. Svaka sljedeća riječ $w[i]$ dobije se iz prethodne riječi $w[i-1]$ tako da se primjeni logička operacija isključivo ILI sa riječi $w[i-4]$. Ako je i višekratnik broja četiri, prije operacije isključivo ILI izvršava se ciklički pomak prethodne riječi za jedan bajt u lijevo i operacija supstitucije pomoću S-kutije (Tablica 3.7.). Nakon toga se prethodna riječ zbraja operacijom isključivo ILI sa retkom $(RC[j], 0, 0, 0)$ gdje su vrijednosti konstante runde $RC[j]$ dane u tablici 3.8.

Tablica 3.8. Konstanta runde RC[j] gdje j označava broj runde [20]

j	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

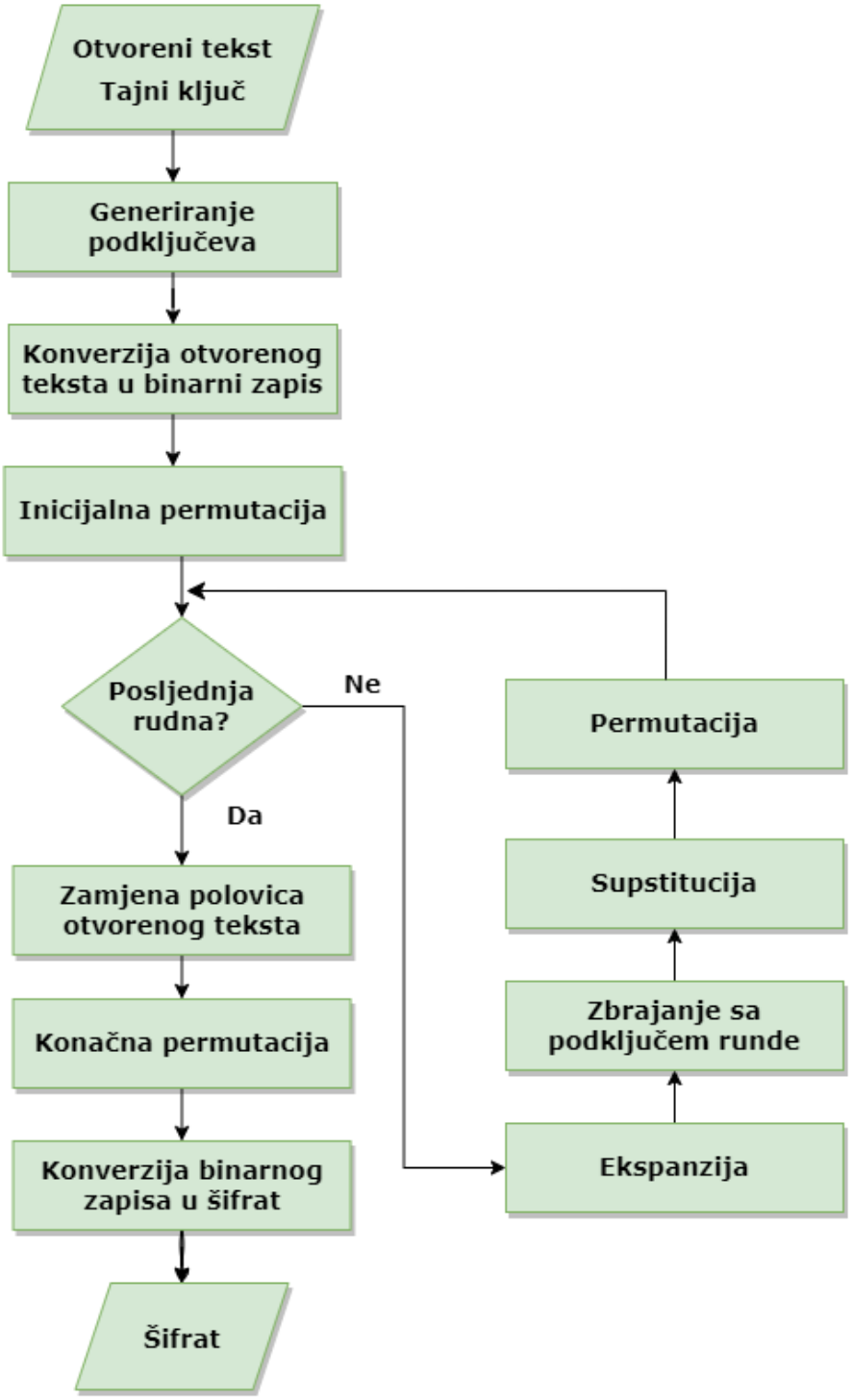


Slika 3.6. AES algoritam za ekspanziju ključa [20]

3.3. Implementacija algoritma kriptiranja zasnovanog na *Data Encryption Standardu*

U ovom potpoglavlju opisana je implementacija algoritma kriptiranja zasnovana na DES-u. Algoritam je izrađen u C programskom jeziku u Eclipse razvojnom okruženju. Eclipse nudi mogućnost lakog *debugiranja* koda, pa je DES algoritam enkripcije i dekripcije najprije razvijen na osobnom računalu, nakon čega je prebačen na ugradbenu platformu. Dijagram toka izrađenog algoritma enkripcije zasnovanog na DES-u prikazan je na slici 3.7. Ulazne varijable algoritma su otvoreni tekst i tajni ključ. Prije samog početka enkripcije, izvodi se generiranje potključeva na temelju ulaznog tajnog ključa, te se izvodi konverzija otvorenog teksta u prikladan binarni zapis. Enkripcija započinje inicijalnom permutacijom, nakon čega se izvodi deset rundi DES operacija. Sa završetkom posljednje rudne izvodi se zamjena polovica otvorenog teksta, te konačna permutacija. Konačni šifrat se pretvara iz binarnog zapisa u prikladan oblik, te on predstavlja izlaznu varijablu algoritma. Kako je ranije opisano, postupak enkripcije i dekripcije su kod DES-a jednaki uz napomenu da se potključevi runde koriste

obrnutim redoslijedom. U skladu s time, izrađeni algoritam dekripcije ima dijagram toka ekvivalentan dijagramu prikazanom na slici 3.7.



Slika 3.7. Dijagram toka enkripcije i dekripcije izrađenog algoritma zasnovanog na DES-u

Struktura projekta sastoji se od tri dijela: glavne izvorne datoteke `main.c`, izvorne DES datoteke `des.c` i datoteke DES zaglavlja `des.h`. Unutar datoteke DES zaglavlja `des.h` definirane su sljedeće varijable pohranjene izravno na ugradbenoj računalnoj platformi:

- `uint8_t Key_With_Parity[64]` – tajni ključ veličine šezdeset i četiri bita,
- `uint8_t Round_Keys[16][48]` – šesnaest potključeva runde veličine četrdeset i osam bita,
- `uint8_t Initial_Key_Permutation_Table[56]` – tablica za ispuštanje paritetnih bitova ključa,
- `uint8_t Key_Compression_Permutation_Table[48]` – tablica kompresije potključa runde,
- `uint8_t Initial_Permutation_Table[64]` – tablica inicijalne permutacije bloka otvorenog teksta,
- `uint8_t Final_Permutation_Table[64]` – tablica inverzne inicijalne permutacije bloka otvorenog teksta,
- `uint8_t D_Box[48]` – tablica ekspanzije polovice bloka otvorenog teksta,
- `uint8_t S_Box[8][4][16]` – tablica supstitucijskih S-kutija,
- `uint8_t Round_Permutation_Table[32]` – tablica permutacije runde.

Unutar izvorne DES datoteke `des.c` definirana su dva seta funkcija. Prvi set funkcija se izvodi nad blokom otvorenog teksta, a izvršava operacije inicijalne permutacije, ekspanzije, zbrajanja, supstitucije, permutacije, te inverzne inicijalne permutacije. Drugi set funkcija se izvodi nad tajnim ključem u svrhu generiranja potključeva rundi. Tablice 3.9. i 3.10. daju pregled i opis najvažnijih funkcija iz oba seta.

Tablica 3.9. Funkcije korištene u postupku DES enkripcije i dekripcije

FUNKCIJE NAD BLOKOM OTVORENOG TEKSTA	**OPIS**
`void Convert_Bytes_To_Binary(uint8_t Can_Input[], uint8_t Input[]);`	Funkcija pretvara predano polje bajtova `Can_Input[]` u njihov binarni zapis i pohranjuje ga u predanu varijablu `Input[].`

`void Convert_Binary_To_Bytes(uint8_t Input[], uint8_t Can_Input[]);`	Funkcija pretvara predani binarni zapis Input[] u polje bajtova i pohranjuje ga u predanu varijablu Can_Input[].
`void Initial_Permutation(uint8_t Input[]);`	Funkcija izvodi inicijalnu permutaciju nad predanim blokom otvorenog teksta `Input[]`. Veličina polja je pohranjena unutar funkcije i iznosi šezdeset i četiri.
`void Final_Permutation(uint8_t Input[]);`	Funkcija izvodi inverznu inicijalnu permutaciju nad predanim blokom otvorenog teksta `Input[]`. Veličina polja je pohranjena unutar funkcije i iznosi šezdeset i četiri.
`void Expansion_Permutation(uint8_t Right_Half[], uint8_t Expanded_Right_Half[]);`	Funkcija proširuje predanu polovicu bloka otvorenog teksta `Right_Half[]` sa trideset i dva bita na četrdeset i osam bita. Rezultat operacije je pohranjen u predanu varijablu `Expanded_Right_Half[].`
`void Add_Round_Key(uint8_t Expanded_Right_Half, int Round);`	Funkcija zbraja proširenu polovicu bloka otvorenog teksta `Expanded_Right_Half[]` sa potključem runde. Odgovarajući potključ runde se dohvaća unutar funkcije na temelju predane varijable `Round` koja označava trenutnu rundu.
`void S_Box_Supstitution(uint8_t Expanded_Right_Half[],uint8_t Right_Half[]);`	Funkcija izvodi supstituciju elemenata predanog polja Expanded_Right_Half[] pomoću S-kutija. Rezultat operacije se pohranjuje u predano polje Right_Half[] jer operacija supstitucije kod DES-a ujedno vrši i kompresiju sa četrdeset i osam bita na trideset i dva bita.
`void Round_Permutation(uint8_t Right_Half[]);`	Funkcija izvodi permutaciju runde nad predanim poljem Right_Half[].

Tablica 3.10. Funkcije korištene u DES postupku generiranja potključeva runde

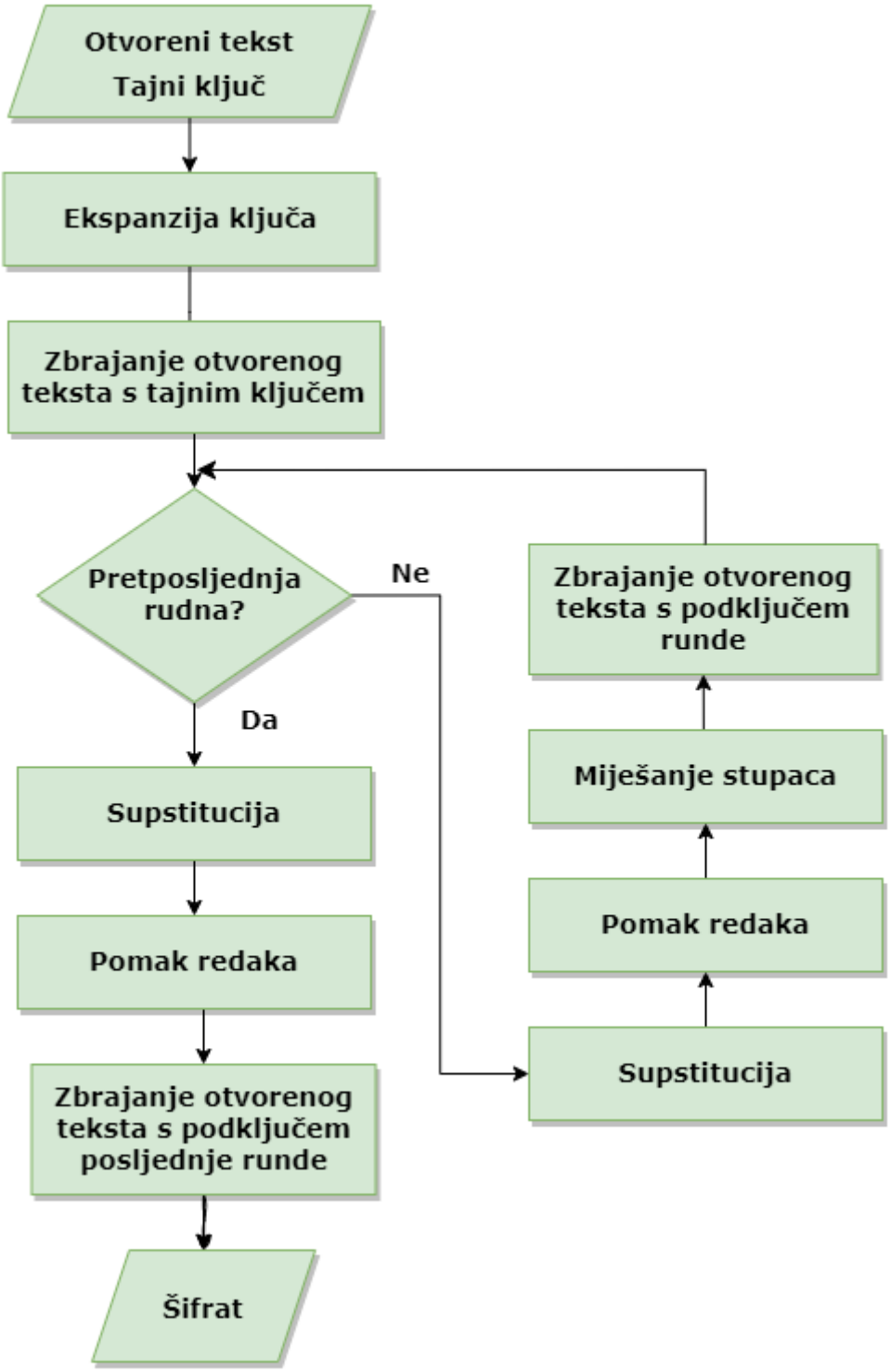
FUNKCIJE NAD TAJNIM KLJUČEM	OPIS
`void Initial_Key_Permutation(uint8_t Key_With_Parity[], uint8_t Key[]);`	Funkcija izvodi inicijalnu permutaciju ključa nad predanim tajnim ključem `Key_With_Parity[]`. Rezultat operacije se pohranjuje u predano polje `Key[]` jer operacija inicijalne permutacije ključa kod DES-a ujedno vrši ispuštanje paritetnih bitova, čime se ključ sa veličine šezdeset i četiri bita sažima na veličinu pedeset i šest bita.
`void Key_Comperssion_Permutation(int Round);`	Funkcija izvodi kompresiju potključa runde sa veličine pedeset i šest bita na veličinu četrdeset i osam bita. Odgovarajući potključ runde se dohvaća unutar funkcije na temelju predane varijable `Round` koja označava trenutnu rundu.
`void Left_Shift(uint8_t Input[], int Size);`	Funkcija izvodi ciklički pomak u lijevo predanog polja `Input[]` zadane veličine `Size`.
`void Expand_Key(uint8_t Key_With_Parity[]);`	Funkcija na temelju predanog tajnog ključa `Key_With_Parity[]` generira potključeve rundi. Unutar funkcije se nalaze pozivi pomoćnih funkcija koje izvršavaju operacije nad tajnim ključem, a potključevi rundi se pohranjuju u globalnu varijablu `Round_Keys[16][48].`

Unutar glavne izvorne datoteke `main.c` nalazi se poziv funkcija `void DES_Encrypt(uint8_t Can_Input[])`, odnosno `void DES_Decrypt(uint8_t Can_Input[])`. Funkcija `void DES_Encrypt(uint8_t Can_Input[])` izvodi enkripciju

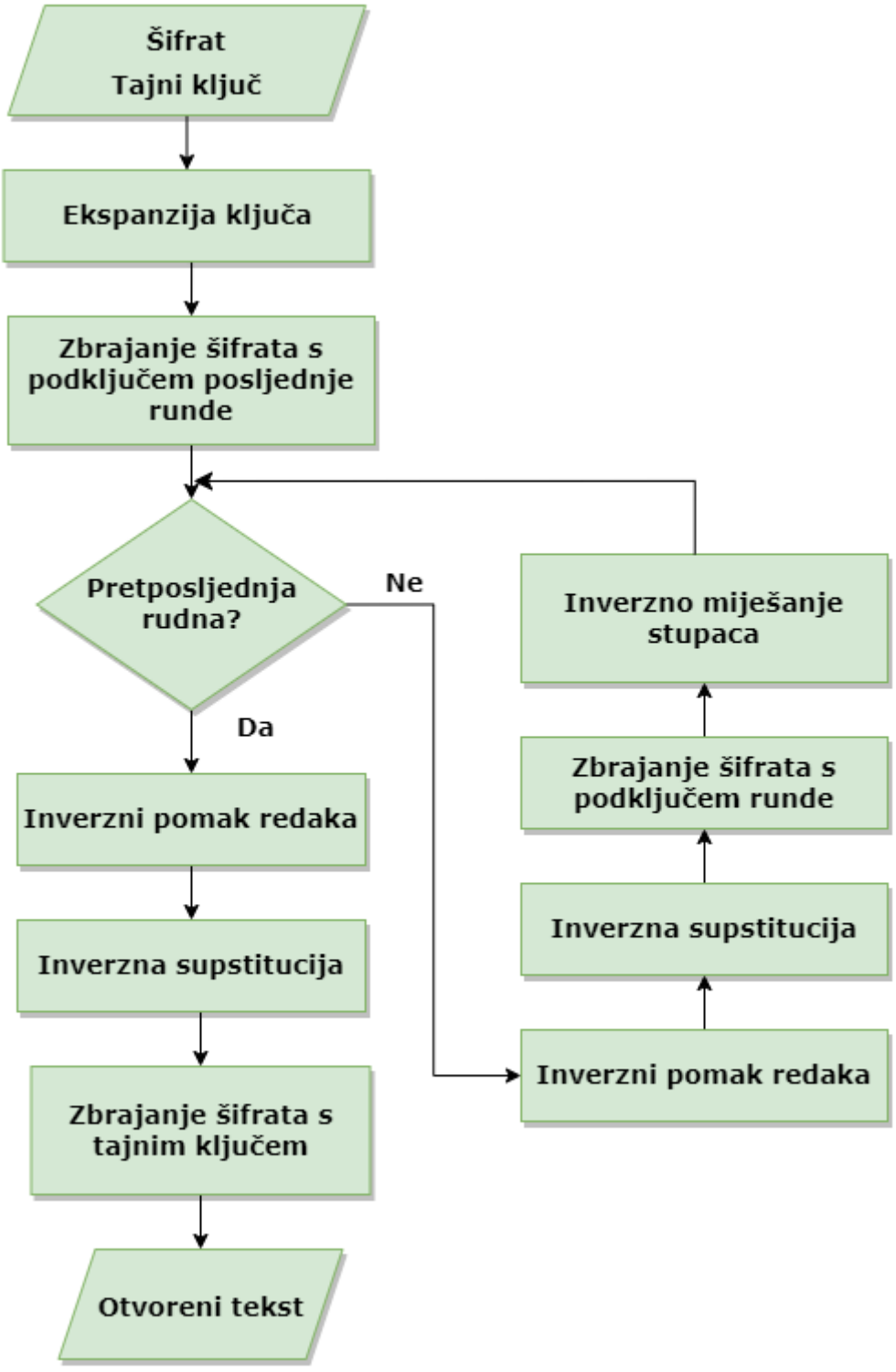
predanog polja Can_`Input[]`, pri čemu koristi ranije opisane pomoćne funkcije za izvođenje operacija inicijalne permutacije, ekspanzije, zbrajanja, supstitucije, permutacije, te inverzne inicijalne permutacije. Funkcija `void DES_Decrypt(uint8_t Can_Input[])` izvodi dekripciju predanog polja Can_`Input[]`, pri čemu je postupak ekvivalentan postupku enkripcije, ali uz obrnuti redoslijed potključeva. Povratna vrijednost funkcija je `void` jer je rezultat enkripcije, odnosno dekripcije, pohranjen u predano polje Can_`Input[]`.

3.4. Implementacija algoritma kriptiranja zasnovanog na *Advanced Encryption Standardu*

U ovom potpoglavlju opisan je vlastiti algoritma zasnovan na AES-u. Algoritam je izrađen u C programskom jeziku u Eclipse razvojnom okruženju. Eclipse nudi mogućnost lakog *debugiranja* koda, pa je AES algoritam enkripcije i dekripcije najprije razvijen na osobnom računalu, nakon čega je prebačen na ugradbenu platformu. Dijagram toka izrađenog algoritma enkripcije zasnovanog na AES-u prikazan je na slici 3.8. Ulazne varijable algoritma su otvoreni tekst i tajni ključ. Prije same enkripcije provodi se ekspanzija ključa, a dijelovi proširenog ključa predstavljaju potključeve rundi. Izvodi se nulta runda u kojoj se otvoreni tekst zbraja sa tajnim ključem. Nakon toga algoritam izvodi određeni broj rundi AES operacija. Posljednja rudna isključuje operaciju miješanja stupaca. Dobiveni šifrat predstavlja izlaz algoritma enkripcije. Dijagram toka izrađenog algoritma dekripcije zasnovanog na AES-u prikazan je na slici 3.9. Ulazne varijable algoritma su šifrat i tajni ključ, nakon čega se provodi ekspanzija ključa. Generirani potključevi rundi sada se koriste obrnutim redosljedom, pa nulta runda uključuje zbrajanje šifrata sa potključem posljednje runde. Izvodi se određeni broj rundi inverznih AES operacija. Posljednja runda isključuje inverznu operaciju miješanja stupaca, a šifrat se zbraja sa tajnim ključem. Dobiveni otvoreni tekst predstavlja izlaz algoritma dekripcije.



Slika 3.8. Dijagram toka enkripcije izrađenog algoritma zasnovanog na AES-u



Slika 3.9. Dijagram toka dekripcije izrađenog algoritma zasnovanog na AES-u

Struktura projekta sastoji se od tri dijela: glavne izvorne datoteke `main.c`, izvorne AES datoteke `aes.c` i datoteke AES zaglavlja `aes.h`. Unutar datoteke AES zaglavlja `aes.h` definirane su sljedeće varijable pohranjene izravno na ugradbenoj računalnoj platformi:

- `uint8_t Key_Size` – veličina tajnog ključa,
- `uint8_t Key[Key_Size]` – tajni ključ veličine 16, 24 ili 32 bajta,
- `uint8_t Expanded_Key_Size` – veličina proširenog tajnog ključa,
- `uint8_t Round_Keys[Number_Of_Rounds+1][16]` – potključevi rundi veličine šesnaest bajta,
- `uint8_t Number_Of_Rounds` – broj rundi,
- `uint8_t S_Box[16][16]` – supstitucijska tablica,
- `uint8_t Inverse_S_Box[16][16]` – inverzna supstitucijska tablica,
- `uint8_t Round_Constant[29]` – tablica konstanti runde.

Unutar izvorne AES datoteke `aes.c` definirana su dva seta funkcija. Prvi set funkcija se izvodi nad blokom otvorenog teksta u procesu enkripcije, te nad tajnim ključem u procesu generiranja proširenog tajnog ključa. Navedene funkcije izvršavaju operacije supstitucije pomoću S-kutije, zbrajanja bloka otvorenog teksta s potključem runde, miješanja stupaca, te pomaka redaka. Drugi set funkcija se izvodi nad blokom otvorenog teksta u procesu dekripcije, a izvršava operacije inverzne onima u procesu enkripcije. Tablice 3.11. i 3.12. daju pregled i opis najvažnijih funkcija iz oba seta.

Tablica 3.11. Funkcije korištene u postupku AES enkripcije i AES ekspanzije ključa

FUNKCIJE ENKRIPCIJE I EKSPANZIJE KLJUČA	**OPIS**
`void Supstitute_Bytes(uint8_t Input[], int Size);`	Funkcija izvodi supstituciju elemenata predanog polja `Input[]` pomoću S-kutije. Varijabla `Size` označava veličinu predanog polja.
`uint8_t Galois_Multiplication(uint8_t Operand_A, uint8_t Operand_B);`	Funkcija množi dva predana broja, `Operand_A` i `Operand_B`, unutar Galois polja $GF(2^8)$. Povratna vrijednost je umnožak brojeva tipa `uint8_t`.

`void Add_Round_Key(uint8_t Input[], uint8_t Round_Key[], int Size);`	Funkcija zbraja predano polje `Input[]` sa predanim potključem runde `Round_Key[]`. Varijabla `Size` označava veličinu polja i potključa runde, te iznosi 16.
`int Calculate_Column(uint8_t Input[], uint8_t Size);`	Funkcija računa vrijednost predanog stupca `Input[]` za operaciju miješanja stupaca. Varijabla `Size` označava veličinu stupca i onda mora biti jednaka 4. Povratna vrijednost je tipa `int`, te služi za detekciju greške ukoliko vrijednost varijable `Size` nije jednaka 4.
`int Mix_Columns(uint8_t Input[], int Size);`	Funkcija izvodi operaciju miješanja stupaca nad blokom otvorenog teksta `Input[].` Varijabla `Size` označava veličinu bloka otvorenog teksta i ona mora biti jednaka 16. Povratna vrijednost je tipa `int`, te služi za detekciju greške ukoliko vrijednost varijable Size nije jednaka 16.
`int Shift_Rows(uint8_t Input[], int Size);`	Funkcija izvodi operaciju pomicanja redaka nad blokom otvorenog teksta `Input[]`. Varijabla `Size` označava veličinu bloka otvorenog teksta i ona mora biti jednaka 16. Povratna vrijednost je tipa `int`, te služi za detekciju greške ukoliko vrijednost varijable `Size` nije jednaka 16.

Tablica 3.12. Funkcije korištene u postupku AES dekripcije

FUNKCIJE DEKRIPCIJE	OPIS
`void Inverse_Supstitute_Bytes(uint8_t Input[], int Size);`	Funkcija izvodi inverznu supstituciju elemenata predanog polja `Input[]` pomoću inverzne S-kutije. Varijabla `Size` označava veličinu predanog polja.

`int Inverse_Calculate_Column(uint8_t Input[], uint8_t Size);`	Funkcija računa vrijednost predanog stupca `Input[]` za inverznu operaciju miješanja stupaca. Varijabla `Size` označava veličinu stupca i onda mora biti jednaka 4. Povratna vrijednost je tipa `int`, te služi za detekciju greške ukoliko vrijednost varijable `Size` nije jednaka 4.
`int Inverse_Mix_Columns(uint8_t Input[], int Size);`	Funkcija izvodi inverznu operaciju miješanja stupaca nad blokom otvorenog teksta `Input[]`. Varijabla `Size` označava veličinu bloka otvorenog teksta i ona mora biti jednaka 16. Povratna vrijednost je tipa `int`, te služi za detekciju greške ukoliko vrijednost varijable Size nije jednaka 16.
`int Inverse_Shift_Rows(uint8_t Input[], int Size);`	Funkcija izvodi inverznu operaciju pomicanja redaka nad blokom otvorenog teksta `Input[]`. Varijabla `Size` označava veličinu bloka otvorenog teksta i ona mora biti jednaka 16. Povratna vrijednost je tipa `int`, te služi za detekciju greške ukoliko vrijednost varijable `Size` nije jednaka 16.

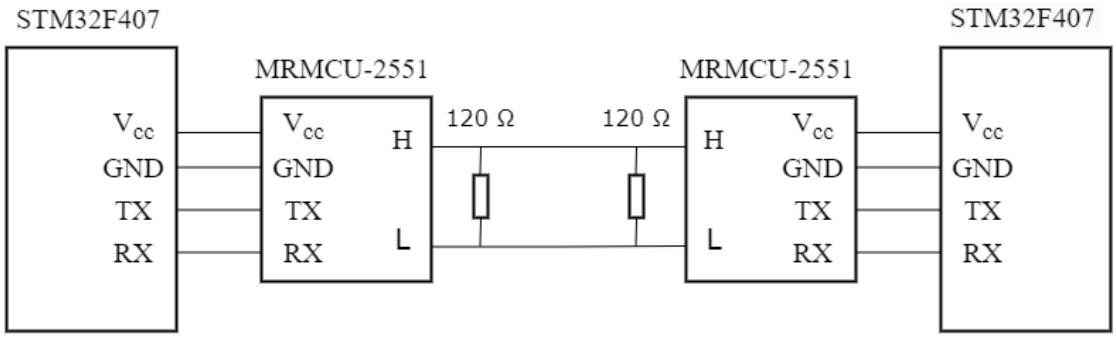
Unutar glavne izvorne datoteke `main.c` nalazi se poziv funkcija `void AES_Encrypt(uint8_t Input[])`, odnosno `void AES_Decrypt(uint8_t Input[])`. Funkcija `void AES_Encrypt(uint8_t Input[])` izvodi enkripciju predanog polja `Input[]`, pri čemu koristi ranije opisane pomoćne funkcije za izvođenje operacija supstitucije, zbrajanja, miješanja i pomaka. Funkcija `void AES_Decrypt(uint8_t Input[])` izvodi dekripciju predanog polja `Input[]`, pri čemu koristi ranije opisane pomoćne funkcije za izvođenje inverznih operacija supstitucije, miješanja i pomaka. Povratna vrijednost funkcija je `void` jer je rezultat enkripcije, odnosno dekripcije, pohranjen u predano polje `Input[]`.

3.5. Implementacija kriptirane CAN komunikacije na ugradbenoj računalnoj platformi

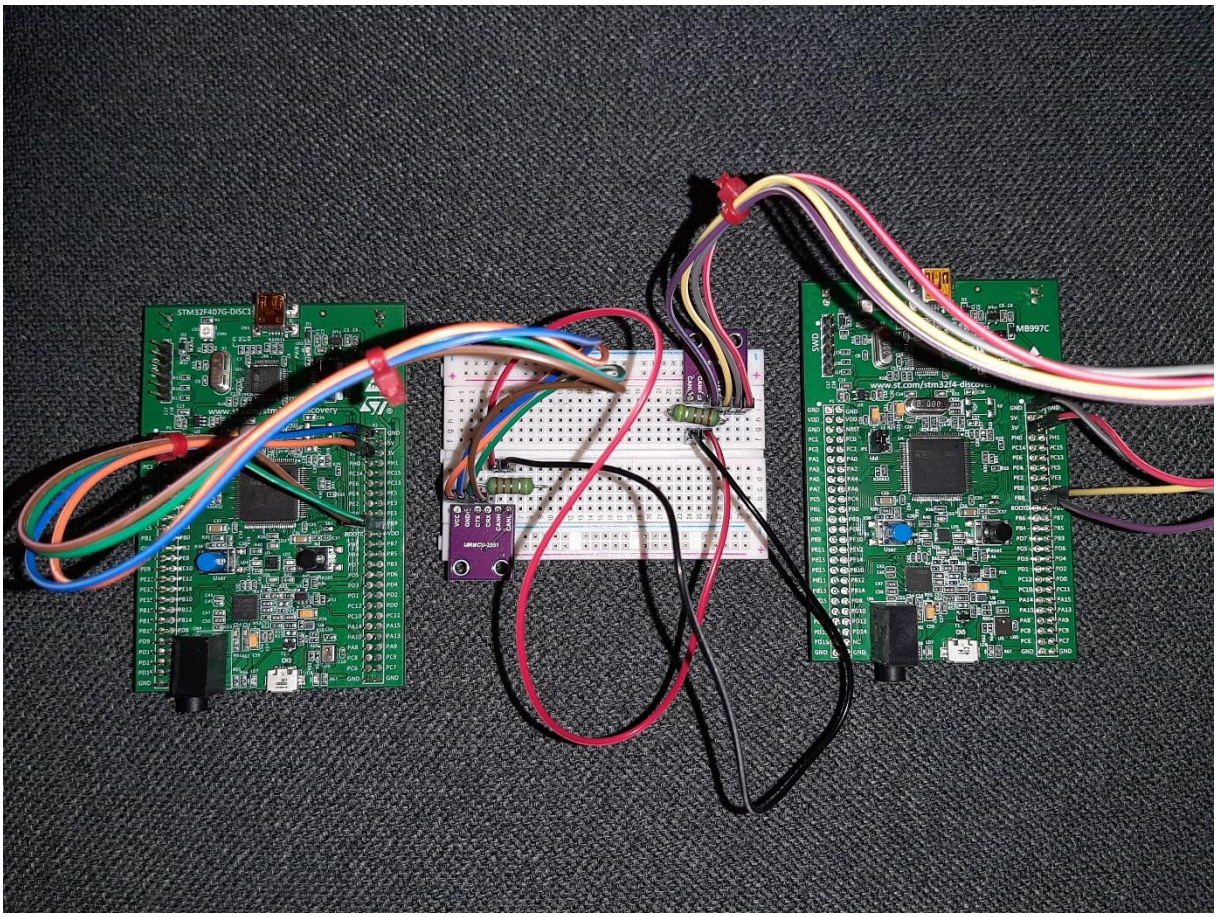
U ovom potpoglavlju opisana je implementacija kriptirane CAN komunikacije na ugradbenoj računalnoj platformi. Komunikacija je ostvarena između dva STM32F407 mikroupravljača tvrtke STMicroelectronics sljedećih značajki:

- 32 bitni Arm Cortex -M4 sa FPU (engl. *Floating-Point Unit*) jezgrom,
- 1 MB *flash* memorije,
- 192 kB radne memorije,
- mini USB priključak i mikro USB priključak,
- izlazni *stereo* priključak za slušalice,
- 2 tipkala,
- 4 korisničke LED,
- 3 mogućnosti napajanja: ST-LINK, USB VBUS, vanjski izvori,
- vanjsko napajanje od 3V i 5V,
- ST-LINK/V2-A *debugger*/programator s mogućnošću USB reenumeracije: masovno skladište, virtualni COM port i *debug* port.

Detaljnije informacije mogu se pronaći na službenoj stranici proizvoda [22] tvrtke STMicroelectronics, a *datasheet* [23] sadrži sheme mikroupravljača, raspored pinova, priključaka, popis funkcionalnosti i slično. Također, korišteni su i CAN primopredajnici MRMCU-2551 [24], te terminirajući otpornici od 120 Ω, čija je uloga spriječiti refleksiju signala na sabirnici koja se pojavljuje na bakrenom vodiču zbog nepoklapanja impedancije. Korištena je razina napajanja od 3.3 V. Električna shema spoja prikazana je na slici 3.10., dok je na slici 3.11. prikazana ostvarena CAN mreža.



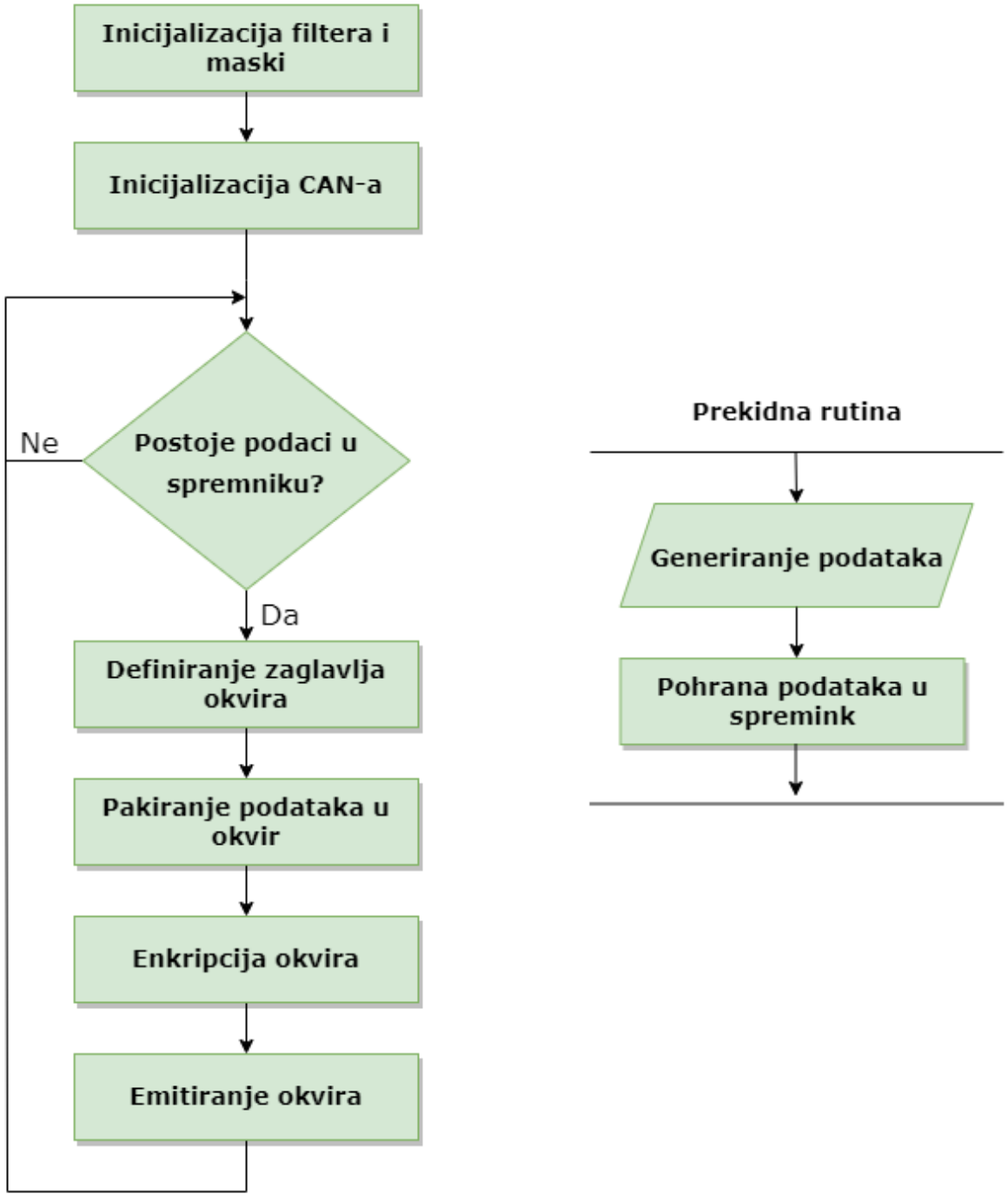
Slika 3.10. Električna shema spoja CAN komunikacije između dva mikroupravljača



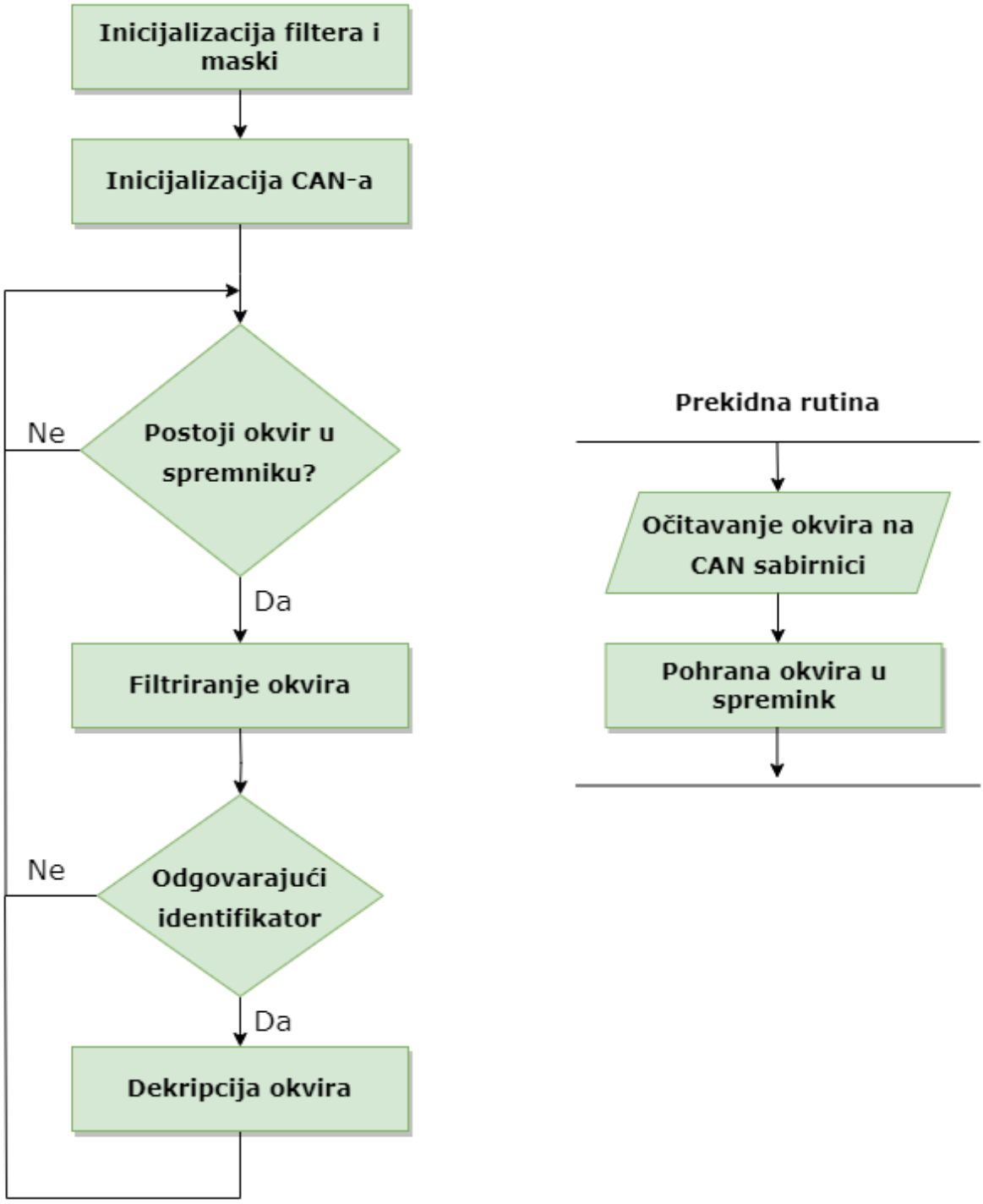
Slika 3.11. Spoj dva mikroupravljača preko CAN primopredajnika i sabirnice

Mikroupravljači su programirani u STM32CubeIDE [25] razvojnom okruženju. Na slici 3.12. prikazan je dijagram toka programskog koda mikroupravljača pošiljatelja, dok je na slici 3.13. prikazan dijagram toka programskog koda mikroupravljača primatelja. Oba mikroupravljača provode inicijalizaciju filtara, maski i potrebnih elemenata za ostvarenje CAN komunikacije, nakon čega se program odvija u beskonačnoj petlji u glavnoj funkciji. U trenutku pritiska na tipkalo, generira se zahtjev za prekidom te se poziva odgovarajuća prekidna rutina (engl. *Interrupt Service Routine*) koja generira podatke za emitiranje na sabirnici. Generirani podaci se pakiraju u okvir, definira se zaglavlje okvira, te se izvodi enkripcija. Trenutni okvir se emitira, nakon čega se provjerava jesu li poslani svi podaci. Ako nisu, izvršavanje programa se vraća na enkripciju i emitiranje sljedećeg okvira. Nakon što su poslani svi podaci, pošiljatelj se vraća na izvođenje beskonačne petlje u glavnoj funkciji i čeka generiranje novih podataka za emitiranje. Usporedno, u trenutku pojave okvira na CAN sabirnici, u mikroupravljaču primatelju se generira odgovarajući zahtjev za prekidom. Poziva se odgovarajuća prekidna rutina koja pohranjuje detektirani okvir u spremnik, nakon čega se pohranjeni okvir filtrira. Ako

okvir nema odgovarajući identifikator, algoritam se vraća u beskonačnu petlju i čeka pojavu novog okvira na CAN sabirnici. Ako okvir sadrži odgovarajući identifikator, izvodi se dekripcija okvira, te se program vraća u beskonačnu petlju u glavnom programu.



Slika 3.12. Dijagram toka ostvarene zaštićene CAN komunikacije na mikroupravljaču pošiljatelju



Slika 3.13. Dijagram toka ostvarene zaštićene CAN komunikacije na mikroupravljaču primatelju

Za ostvarenje CAN komunikacije između dva mikroupravljača korištena je `stm32f4xx_hal_can.h` HAL biblioteka s pripadajućom dokumentacijom [26]. Unutar glavne izvorne datoteke `main.c` deklarirane su sljedeće varijable:

- CAN_FilterTypeDef CANFilter,
- CAN_TxHeaderTypeDef TxHeader,
- CAN_RxHeaderTypeDef RxHeader,
- uint32_t TxMailbox,
- uint8_t TxData[8],
- uint8_t RxData[8].

Konfiguracijom mikroupravljača definirana je brzina ostvarene komunikacije od 500 kbit/s, normalan operacijski način rada, CAN1 RX0 prekid, te CAN1_RX pin PB9 i CAN1_TX pin PB8. U zaglavlju paketa definirana je veličina podatkovnog polja od maksimalnih osam bajta, naznačeno je da se radi o standardnom CAN identifikatoru, a paketu je dodijeljena vrijednost identifikatora 0x100. Za prijem poruka definiran je CAN_RX_FIFO0 *first-in first-out* registar. S obzirom da u komunikaciji sudjeluju samo dva mikroupravljača, te oba moraju primiti sve poruke od onog drugoga, maske za filtriranje su postavljene na vrijednost nula. Nakon podešavanja filtera i zaglavlja paketa, funkcijom HAL_CAN_ActivateNotification(&hcan1, CAN_IT_RX_FIFO0_MSG_PENDING) omogućen je prekid koji okida HAL_CAN_RxFifo0MsgPendingCallback(CAN_HandleTypeDef *hcan) funkciju povratnog poziva. Unutar te funkcije ostvaruje se primanje poruka funkcijom HAL_CAN_GetRxMessage(hcan, CAN_RX_FIFO0, &RxHeader, RxData) koja poruku s CAN_RX_FIFO0 registra pohranjuje u RxData varijablu, a zaglavlje poruke u RxHeader varijablu. Emitiranje poruka je ostvareno uporabom prekida koji pritiskom korisničkog tipkala na mikroupravljaču okida funkciju povratnog poziva HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin). Unutar nje se nalazi funkcija HAL_CAN_AddTxMessage(&hcan1, &TxHeader, TxData, &TxMailbox) koja zaglavlje poruke TxHeader i podatkovno polje poruke TxData predaje prvom slobodnom sandučiću TxMailbox i pokreće zahtjev za prijenosom poruke putem sabirnice. Prilikom slanja poruke uključuje se crvena LED kao indikator „poruka je poslana“, dok se nakon primitka poruke uključuje zelena LED kao indikator „poruka je primljena“. Enkripcija je ostvarena pozivom DES_Encrypt(Can_Input) funkcije ili AES_Encrypt(Input) funkcije prije slanja poruke, dok je dekripcija ostvarena pozivom DES_Decrypt(Can_Input) funkcije ili AES_Decrypt(Can_Input) funkcije nakon primitka poruke.

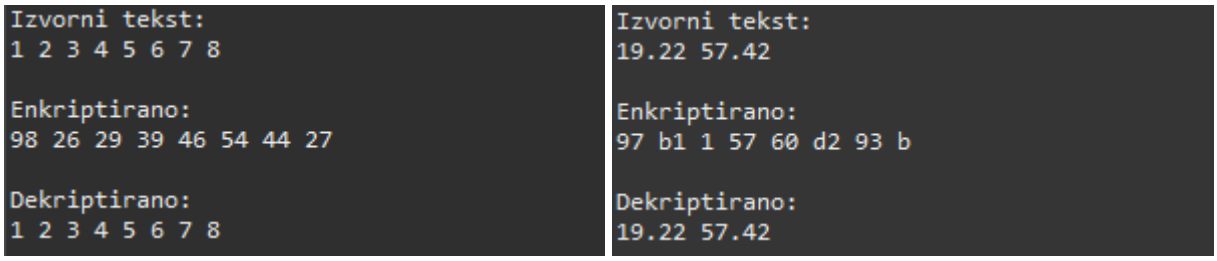
4. VERIFIKACIJA I EVALUACIJA PREDLOŽENOG RJEŠENJA ZAŠTITE CAN KOMUNIKACIJE NA UGRADBENOJ RAČUNALNOJ PLATFORMI

Pojam verifikacije označava postupak utvrđivanja ispravnosti nekog rješenja, dok pojam evaluacije označava postupak utvrđivanja efikasnosti nekog rješenja. U prvom potpoglavlju najprije je opisana verifikacija implementiranih DES i AES kriptografskih algoritama na osobnom računalu, nakon čega je opisana verifikacija na ugradbenoj računalnoj platformi temeljenoj na mikroupravljaču STM32F470. U drugom potpoglavlju opisana je evaluacija implementiranih kriptografskih algoritama, gdje su kao parametri kvalitete rješenja promatrani vrijeme izvođenja računalnog koda enkripcije i dekripcije, te razina prometa na CAN sabirnici.

4.1. Verifikacije predloženog rješenja zaštite CAN komunikacije na osobnom računalu i na ugradbenoj računalnoj platformi

Provedena su tri eksperimenta kako bi se verificirali implementirani DES i AES kriptografski algoritmi. Svi eksperimenti su provedeni nad dva tipa podatka, `uint8_t` i `float`, kako bi se potvrdila ispravnost rada implementiranih algoritma i sa cijelim brojevima i sa decimalnim brojevima. Pri tome je važno napomenuti kako se radi o `float` tipu veličine trideset i dva bita.

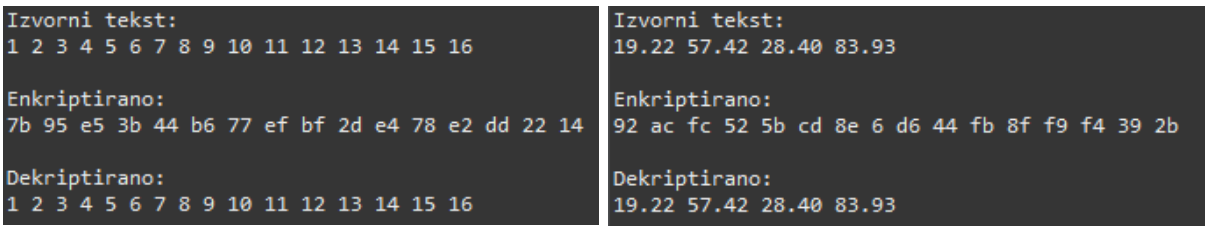
Prvi eksperiment je proveden na osobnom računalu pomoću Eclipse razvojnog okruženja, a uključivao je enkripciju i dekripciju jednog bloka otvorenog teksta poznate vrijednosti. Veličina bloka otvorenog teksta za DES kriptografski algoritam iznosi osam bajta, a za AES kriptografski algoritam šesnaest bajta. Poznati blok otvorenog teksta je prvo enkriptiran, a nakon toga dekriptiran, te je dekriptirana vrijednost bloka ispisana na zaslonu računala. Ispravnost implementacije DES i AES kriptografskih algoritama je provjerena usporedbom poznate i dekriptirane vrijednosti bloka otvorenog teksta. Na slici 4.1. a) prikazan je ispis prvog eksperimenta za DES kriptografski algoritam i blok popunjen podacima `uint8_t` tipa, dok je na slici 4.2. b) prikazan ispis prvog eksperimenta za DES kriptografski algoritam i blok popunjen podacima `float` tipa. S obzirom da je dobivena dekriptirana vrijednost bloka jednaka vrijednosti izvornog bloka, potvrđen je ispravan rad DES kriptografskog algoritma na osobnom računalu za oba tipa podataka.



a) b)

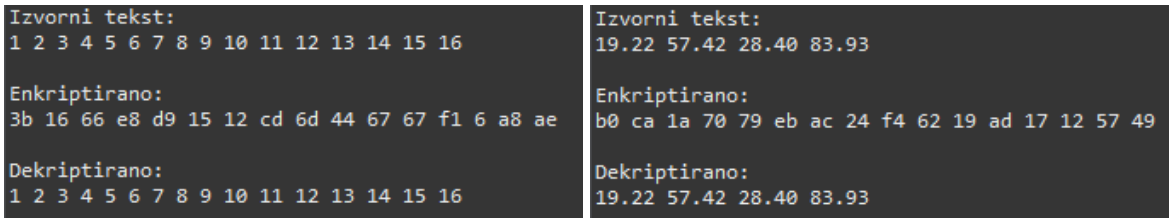
Slika 4.1. Verifikacija DES algoritama na osobnom računalu pomoću Eclipse razvojnog okruženja za blok popunjen vrijednostima, (a) uint8_t tipa i (b) float tipa

Na slici 4.2. a) prikazan je ispis prvog eksperimenta za AES kriptografski algoritam sa ključem veličine šesnaest bajta i blok popunjen podacima `uint8_t` tipa, dok je na slici 4.2. b) prikazan ispis prvog eksperimenta za AES kriptografski algoritam sa ključem veličine šesnaest bajta i blok popunjen podacima `float` tipa. Na slici 4.3. a) prikazan je ispis prvog eksperimenta za AES kriptografski algoritam sa ključem veličine dvadeset i četiri bajta, te blok popunjen podacima `uint8_t` tipa, dok je na slici 4.3. b) prikazan ispis prvog eksperimenta za AES kriptografski algoritam sa ključem veličine dvadeset i četiri bajta, te blok popunjen podacima `float` tipa. Na slici 4.4. a) prikazan je ispis prvog eksperimenta za AES kriptografski algoritam sa ključem veličine trideset i dva bajta, te blok popunjen podacima `uint8_t` tipa, dok je na slici 4.4. b) prikazan ispis prvog eksperimenta za AES kriptografski algoritam sa ključem veličine trideset i dva bajta, te blok popunjen podacima `float` tipa. Za sve slučajeve je dobivena dekriptirana vrijednost bloka jednaka vrijednosti izvornog bloka, pa je potvrđen ispravan rad AES kriptografskog algoritma na osobnom računalu za sve veličine ključa i za oba tipa podataka.



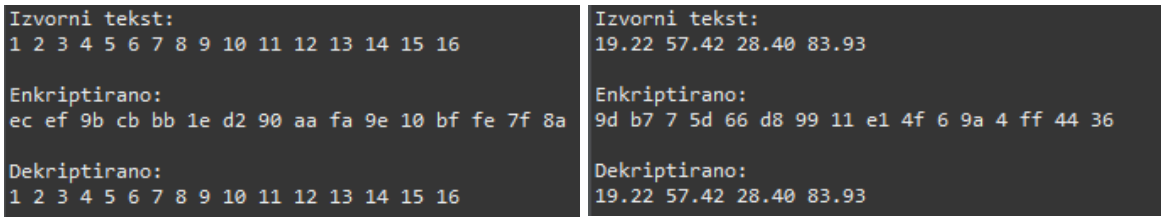
a) b)

Slika 4.2. Verifikacija AES algoritama na osobnom računalu pomoću Eclipse razvojnog okruženja za ključ veličine šesnaest bajta i blok popunjen vrijednostima, (a) uint8_t tipa i (b) float tipa



a) b)

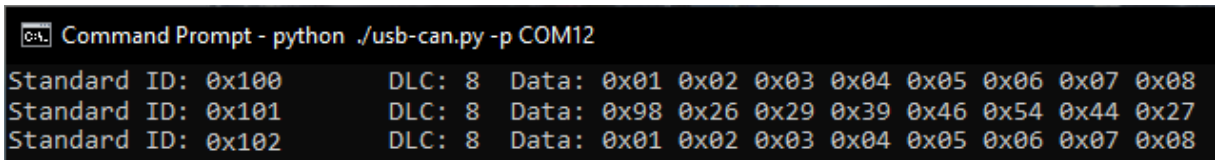
Slika 4.3. Verifikacija AES algoritama na osobnom računalu pomoću Eclipse razvojnog okruženja za ključ veličine dvadeset i četiri bajta, te blok popunjen vrijednostima, (a) uint8_t tipa i (b) float tipa



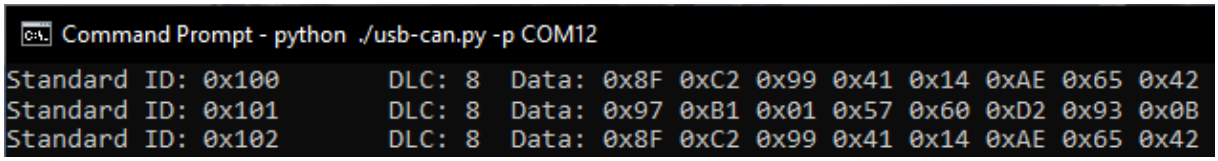
a) b)

Slika 4.4. Verifikacija AES algoritama na osobnom računalu pomoću Eclipse razvojnog okruženja za ključ veličine trideset i dva bajta, te blok popunjen vrijednostima, (a) uint8_t tipa i (b) float tipa

Drugi eksperiment je također uključivao enkripciju i dekripciju jednog bloka otvorenog teksta poznate vrijednosti, no proveden je na ugradbenoj računalnoj platformi temeljenoj na mikroupravljaču STM32F407. Na CAN sabirnicu su priključeni mikroupravljač i USB-CAN adapter pomoću kojeg je u *command promptu* na zaslonu osobnog računala prikazan snimljeni promet na sabirnici. Mikroupravljač je emitirao tri bloka, prvi poznate vrijednosti, drugi enkriptirane vrijednosti i treći dekriptirane vrijednosti. Veličina bloka otvorenog teksta za DES kriptografski algoritam iznosi osam bajta, a za AES kriptografski algoritam šesnaest bajta. Ispravnost implementacije DES i AES kriptografskih algoritama je provjerena usporedbom poznate i dekriptirane vrijednosti bloka otvorenog teksta. Na slici 4.5. prikazan je ispis drugog eksperimenta za DES kriptografski algoritam i blok popunjen podacima `uint8_t` tipa, dok je na slici 4.6. prikazan ispis drugog eksperimenta za DES kriptografski algoritam i blok popunjen podacima `float` tipa. S obzirom da je dobivena dekriptirana vrijednost bloka jednaka vrijednosti izvornog bloka, potvrđen je ispravan rad DES kriptografskog algoritma ugradbenoj računalnoj platformi za oba tipa podataka.

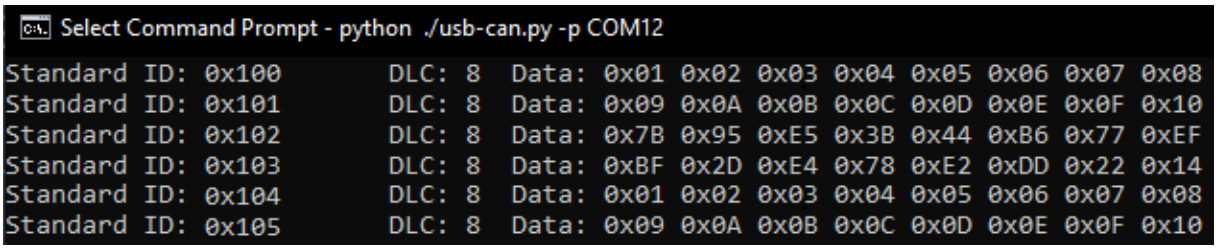


Slika 4.5. Verifikacija DES algoritama na ugradbenoj računalnoj platformi pomoću USB-CAN adaptera za blok popunjen vrijednostima uint8_t tipa

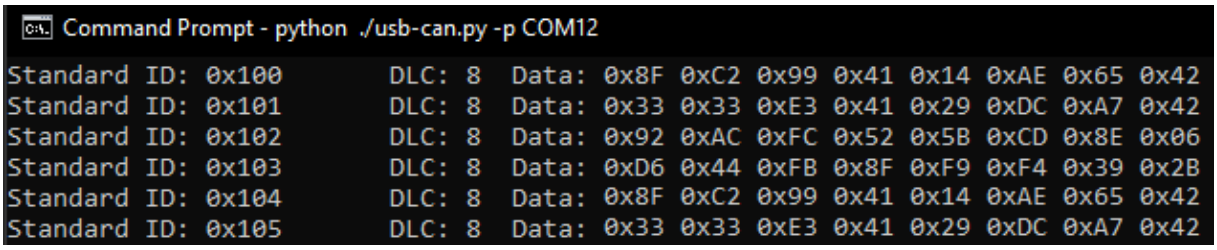


Slika 4.6. Verifikacija DES algoritama na ugradbenoj računalnoj platformi pomoću USB-CAN adaptera za blok popunjen vrijednostima float tipa

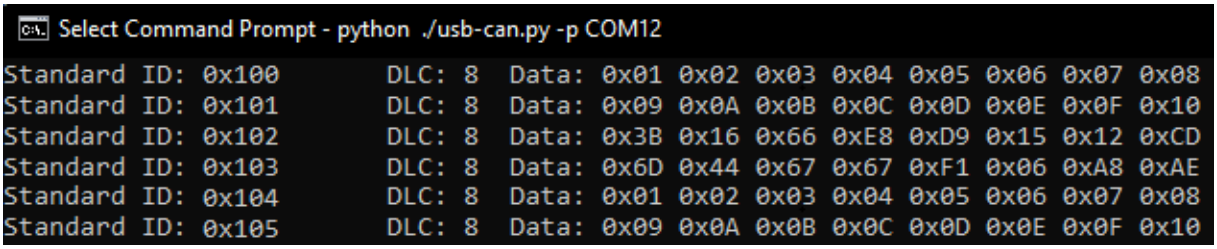
Na slici 4.7. prikazan je ispis drugog eksperimenta za AES kriptografski algoritam sa ključem veličine šesnaest bajta, te blok popunjen podacima `uint8_t` tipa, dok je na slici 4.8. prikazan ispis drugog eksperimenta za AES kriptografski algoritam sa ključem veličine šesnaest bajta, te blok popunjen podacima `float` tipa. Na slici 4.9. prikazan je ispis drugog eksperimenta za AES kriptografski algoritam sa ključem veličine dvadeset i četiri bajta, te blok popunjen podacima `uint8_t` tipa, dok je na slici 4.10. prikazan ispis drugog eksperimenta za AES kriptografski algoritam sa ključem veličine dvadeset i četiri bajta, te blok popunjen podacima `float` tipa. Na slici 4.11. prikazan je ispis drugog eksperimenta za AES kriptografski algoritam sa ključem veličine trideset i dva bajta, te blok popunjen podacima `uint8_t` tipa, dok je na slici 4.12. prikazan ispis drugog eksperimenta za AES kriptografski algoritam sa ključem veličine trideset i dva bajta, te blok popunjen podacima `float` tipa. Za sve slučajeve je dobivena dekriptirana vrijednost bloka jednaka vrijednosti izvornog bloka, pa je potvrđen ispravan rad AES kriptografskog algoritma na ugradbenoj računalnoj platformi za sve veličine ključa i za oba tipa podataka.



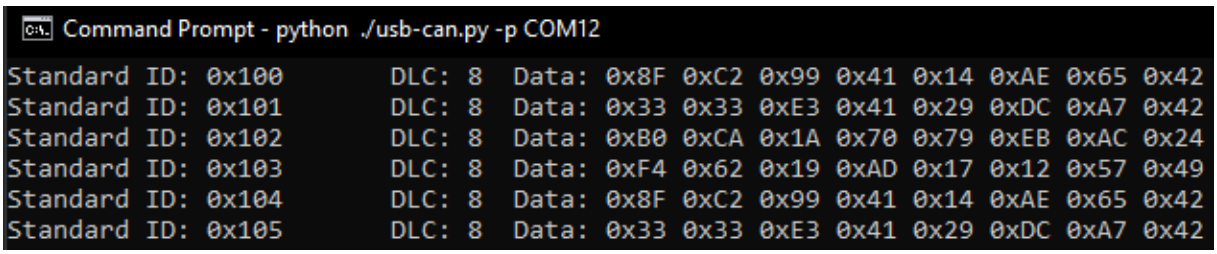
Slika 4.7. Verifikacija AES algoritama na ugradbenoj računalnoj platformi pomoću USB-CAN adaptera za ključ veličine šesnaest bajta, te blok popunjen vrijednostima int tipa



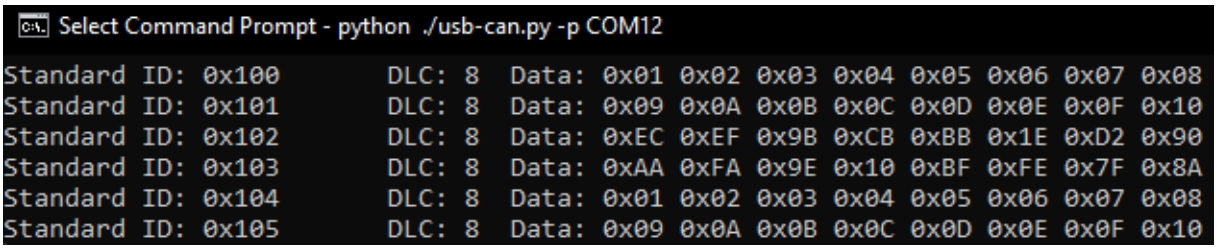
Slika 4.8. Verifikacija AES algoritama na ugradbenoj računalnoj platformi pomoću USB-CAN adaptera za ključ veličine šesnaest bajta, te blok popunjen vrijednostima float tipa



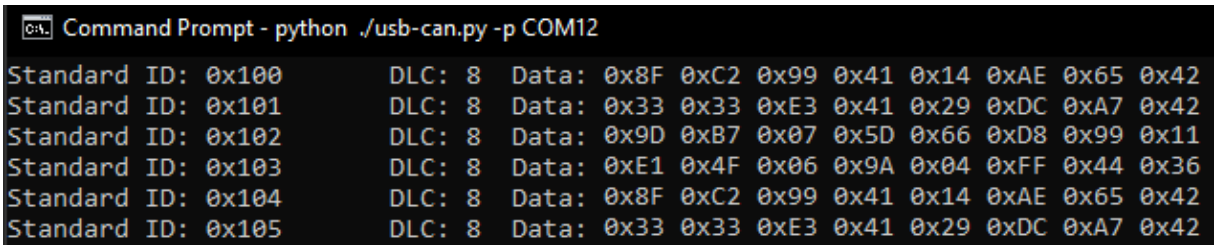
Slika 4.9. Verifikacija AES algoritama na ugradbenoj računalnoj platformi pomoću USB-CAN adaptera za ključ veličine dvadeset i četiri bajta, te blok popunjen vrijednostima int tipa



Slika 4.10. Verifikacija AES algoritama na ugradbenoj računalnoj platformi pomoću USB-CAN adaptera za ključ veličine dvadeset i četiri bajta, te blok popunjen vrijednostima float tipa



Slika 4.11. Verifikacija AES algoritama na ugradbenoj računalnoj platformi pomoću USB-CAN adaptera za ključ veličine trideset i dva bajta, te blok popunjen vrijednostima int tipa



Slika 4.12. Verifikacija AES algoritama na ugradbenoj računalnoj platformi pomoću USB-CAN adaptera za ključ veličine trideset i dva bajta, te blok popunjen vrijednostima float tipa

Treći eksperiment je također proveden na ugradbenoj računalnoj platformi temeljenoj na mikroupravljaču STM32F407, a uključivao je enkripciju i dekripciju velikog broja blokova otvorenog teksta. Na CAN sabirnicu su priključeni mikroupravljač predajnik i mikroupravljač prijemnik kao što je prikazano na slici 3.10. Na mikroupravljaču pošiljatelju je generirano i enkriptirano deset tisuća blokova nasumičnih vrijednosti tipa `uint_8` u rasponu od $uint8\_t\_\min = 0$ do $uint8\_t\_\max = 255$, te deset tisuća blokova nasumičnih vrijednosti tipa `float` u rasponu od $float\_\min = 1.175494\ x\ 10^{-38}$ do $float\_\max = 3.4028237\ x\ 10^{38}$. Mikroupravljač primatelj je dekriptirao primljene blokove. Pri tome su na osobnom računalu putem serijske veze UART, unutar terminala emuliranih Tera Term *open source* programom, prikazane poslane i primljene vrijednosti blokova. Eksperiment je najprije proveden za DES kriptografski algoritam, nakon čega je proveden za AES kriptografski algoritam za sve veličine ključa. Usporedbom ispisa potvrđeno je kako su svi poslani enkriptirani blokovi uspješno primljeni i uspješno dekriptirani, čime je potvrđena ispravnost implementacije DES i AES kriptografskih algoritama na ugradbenoj računalnoj platformi.

4.2. Evaluacija predloženog rješenja zaštite CAN komunikacije na ugradbenoj računalnoj platformi

Evaluacija rješenja je provedena s obzirom na vrijeme izvođenja računalnog koda na ugradbenoj računalnoj platformi temeljenoj na mikroupravljaču STM32F407, te s obzirom na razinu prometa na CAN sabirnici. U eksperimentima gdje je mjereno vrijeme izvođenja računalnog koda, oba implementirana kriptografska algoritma i sva mjerenja su se izvodila na jednom mikroupravljaču koji je radio u *loopback* modu rada. U *loopback* modu rada, mikroupravljač izvršava ulogu pošiljatelja i primatelja, odnosno sve podatke koje enkriptira i pošalje može primiti i dekriptirati. Pri tome su posebno mjerena vremena izvođenja enkripcije, a posebno vremena dekripcije. Vrijeme izvođenja računalnog koda je izmjereno uporabom mjerača vremena (engl. *timer*) na mikroupravljaču. U konfiguraciji je definirana frekvencija mjerača vremena od 84 MHz i *prescaler* vrijednosti 8400, tako da jedan takt brojača mjeri 100 μs. Resetiranje brojača je postavljeno na 65535-tom taktu, pa je maksimalno vrijeme koje se na ovaj način može izmjeriti jednako oko 6.5 sekundi. Sva mjerenja su izvedena jednom jer rezolucija mjerača vremena od 100 μs ne detektira promjene u vremenu izvođenja prilikom ponavljanja mjerenja. Pojam razine prometa odnosi se na iznos ukupne količine prenesenih podataka u odnosu na iznos prenesenih korisnih, informacijskih podataka. U eksperimentima gdje je mjerena razina prometa na CAN sabirnici, mikroupravljač predajnik i mikroupravljač prijemnik priključeni su na CAN sabirnicu kao što je prikazano na slici 3.10.

U daljnjem tekstu, pojam podatak označava jednu brojčanu vrijednost, dok pojam blok označava grupaciju podataka, odnosno više brojčanih vrijednosti grupiranih zajedno. Pri tome je jedan DES blok veličine osam bajta, a AES blok veličine šesnaest bajta. Implementirani DES i AES kriptografski algoritmi se izvode nad jednim blokom odgovarajuće veličine. Pri prijenosu CAN sabirnicom, za emitiranje jednog DES bloka potreban je jedan CAN okvir, dok su za emitiranje jednog AES bloka potrebna dva uzastopna CAN okvira.

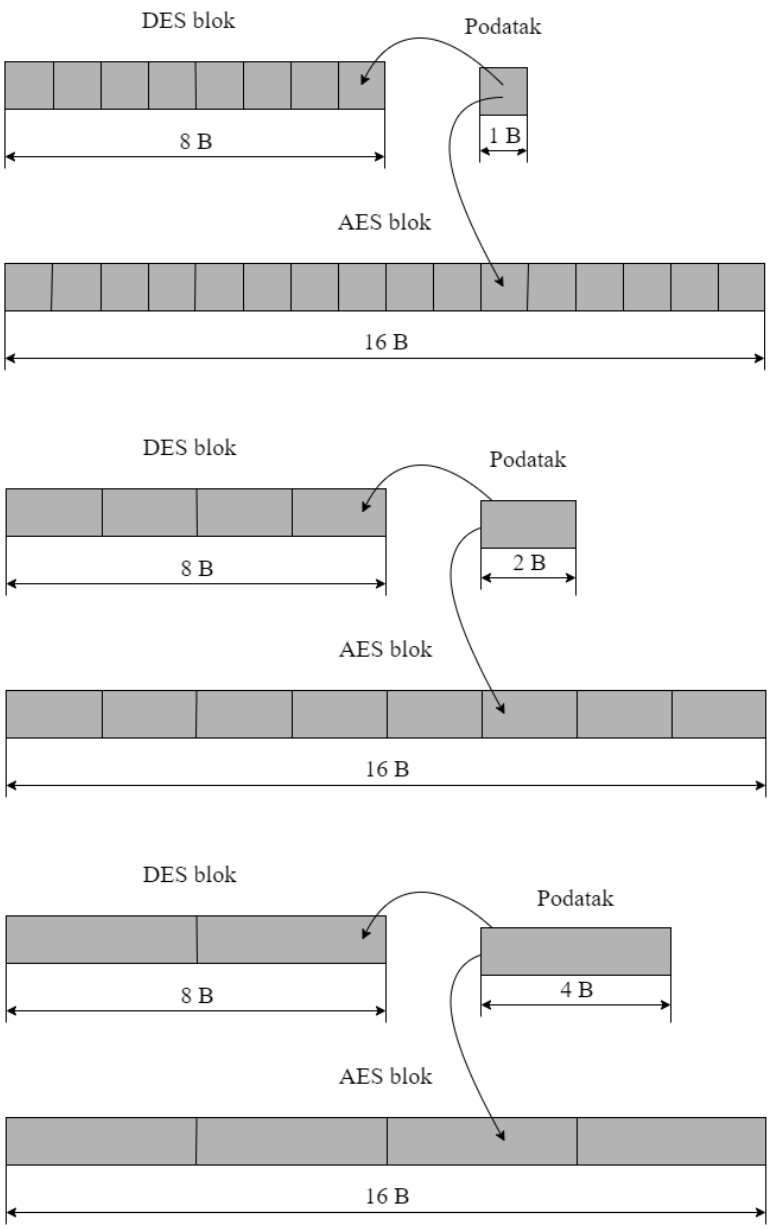
U prvom eksperimentu izmjereno je vrijeme izvođenja enkripcije i dekripcije jednog bloka otvorenog teksta. Veličina bloka otvorenog teksta za DES kriptografski algoritam iznosi osam bajta, a za AES kriptografski algoritam šesnaest bajta. Eksperiment je proveden nad dva tipa podatka, `uint8_t` i `float`, kako bi se utvrdilo ovisi li vrijeme izvođenja implementiranih kriptografskih algoritama o sadržaju bloka. U tablici 4.1. prikazana su dobivena vremena izvođenja enkripcije i dekripcije jednog bloka otvorenog teksta za implementirane DES i AES kriptografske algoritme.

Tablica 4.1. Vrijeme enkripcije i vrijeme dekripcije jednog bloka otvorenog teksta implementiranih DES i AES algoritama za različite veličine ključa

	DES	**AES**		
Veličina ključa [B]	**8**	**16**	**24**	**32**
Vrijeme enkripcije [ms]	2.4	1.9	2.4	2.8
Vrijeme dekripcije [ms]	2.5	2.1	2.4	2.9

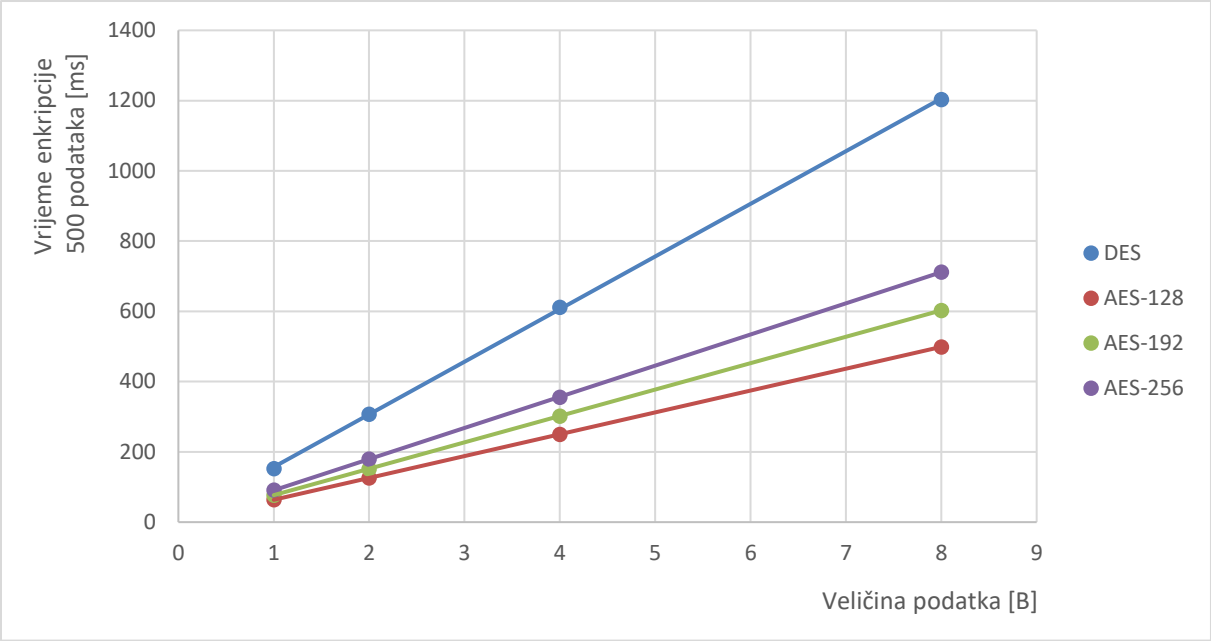
Mjerenja su pokazala kako vrijeme izvođenja implementiranih kriptografskih algoritama ne ovisi o sadržaju bloka. Dobiveni rezultat je očekivan jer algoritmi u oba slučaja izvode operacije nad blokovima jednake veličine. Drugim riječima, DES kriptografski algoritam uvijek radi sa blokom veličine osam bajta, neovisno o tome je li blok popunjen sa osam `uint8_t` podataka ili dva `float` podatka. Također, vidljivo je da se kod AES enkripcije i dekripcije vrijeme izvođenja povećava s povećanjem veličine tajnog ključa. Takvi rezultati su očekivani jer se za veličinu ključa od šesnaest bajta provodi deset AES rundi, za veličinu ključa od dvadeset i četiri bajta provodi dvanaest AES rundi, a za veličinu ključa od trideset i dva bajta provodi četrnaest AES rundi. Važno je napomenuti da, ako se DES algoritmom želi poslati šesnaest bajta podataka, potrebno je provesti dva zasebna procesa enkripcije i dekripcije. Tada je ukupno vrijeme izvođenja udvostručeno, odnosno ono iznosi 4.8 ms za enkripciju i 5 ms za dekripciju, što je značajno više nego kod AES implementacije.

U drugom eksperimentu analiziran je utjecaj promjene veličine podatka na vrijeme izvođenja računalnog koda i razinu prometa na CAN sabirnici. Odabrane veličine podatka su jedan bajt, dva bajta, četiri bajta i osam bajta. Provedena su mjerenja za prijenos pet stotina podataka, a podaci su pakirani u blokove veličine osam bajta za DES algoritam, odnosno blokove veličine šesnaest bajta za AES algoritam, kako je prikazano na slici 4.13.

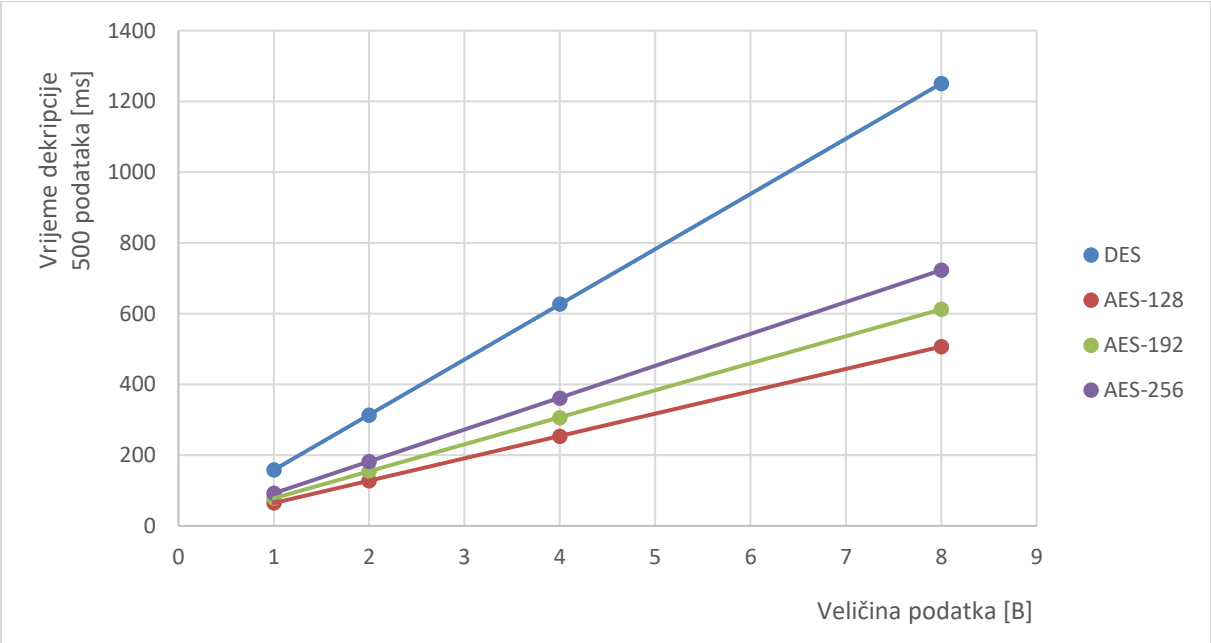


Slika 4.13. Pakiranje podataka različitih veličina u DES i AES blokove stalnih veličina

S obzirom na veličinu pojedinog podatka, na slici 4.14. prikazana su vremena izvođenja DES i AES algoritama enkripcije prilikom slanja pet stotina podataka, dok su na slici 4.15. prikazana vremena izvođenja DES i AES algoritama dekripcije prilikom slanja pet stotina podataka.



Slika 4.14. Vremena izvođenja implementiranih DES i AES algoritama enkripcije prilikom slanja pet stotina podataka prikazana u ovisnosti o veličini pojedinog podatka



Slika 4.15. Vremena izvođenja implementiranih DES i AES algoritama dekripcije prilikom slanja pet stotina podataka prikazana u ovisnosti o veličini pojedinog podatka

Vidljivo je kako s povećanjem veličine podatka linearno raste ukupno vrijeme izvođenja računalnog koda. Naime, ako se kriptiraju podaci veličina manjih od veličine bloka kriptografskog algoritma, moguće je združiti nekoliko podatka u jedan kriptografski blok. Time

je smanjen ukupni broj pojedinih operacija enkripcije i dekripcije, odnosno njihovo ukupno vrijeme izvođenja. Primjerice kod AES algoritma, pet stotina podatka veličine jednog bajta stane u svega trideset i dva bloka za enkripciju, dok je za pet stotina podatka veličine osam bajta potrebno dvije stotine i pedeset blokova za enkripciju. S obzirom na veličinu pojedinog podatka, tablica 4.2. prikazuje razinu prometa na CAN sabirnici prilikom slanja pet stotina podataka kriptiranih implementiranim DES algoritmom, dok tablica 4.3. prikazuje razinu prometa na CAN sabirnici prilikom slanja pet stotina podataka kriptiranih implementiranim AES algoritmom.

Tablica 4.2. Razina prometa prilikom slanja pet stotina podataka kriptiranih implementiranim DES algoritmom prikazana u ovisnosti o veličini pojedinog podatka

Veličina podatka [B]	**Količina informacijskih podataka [B]**	**Količina umetnutih podataka [B]**	**Ukupno poslanih podataka [B]**
1	500	4	504
2	1000	0	1000
4	2000	0	2000
8	4000	0	4000

Tablica 4.3. Razina prometa prilikom slanja pet stotina podataka kriptiranih implementiranim AES algoritmom prikazana u ovisnosti o veličini pojedinog podatka

Veličina podatka [B]	**Količina informacijskih podataka [B]**	**Količina umetnutih podataka [B]**	**Ukupno poslanih podataka [B]**
1	500	12	512
2	1000	8	1008
4	2000	0	2000
8	4000	0	4000

U slučajevima kada količina informacijskih podataka nije cjelobrojno djeljiva s veličinom bloka, potrebno je umetnuti određenu količinu podataka kako bi se popunio prazan prostor u kriptografskom bloku stalne veličine. Tako razina prometa na sabirnici iznosi:

- za DES algoritam potrebno je obraditi i poslati pet stotina i četiri bajta podataka kako bi se prenijelo pet stotina bajta informacijskih podataka,
- za AES algoritam potrebno je obraditi i poslati pet stotina i dvanaest bajta podataka kako bi se prenijelo pet stotina bajta informacijskih podataka,
- za AES algoritam potrebno je obraditi i poslati tisuću i osam bajta podataka kako bi se prenijelo tisuću bajta informacijskih podataka,
- za AES algoritam potrebno je obraditi i poslati tri tisuće i osam bajta podataka kako bi se prenijelo tri tisuće bajta informacijskih podataka.

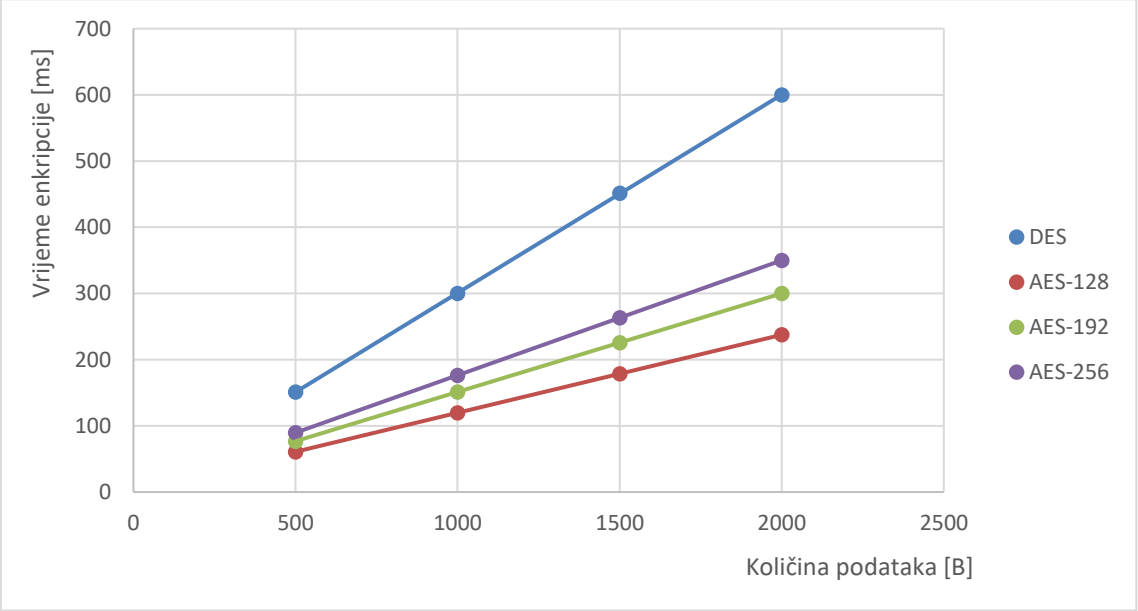
Vidljivo je kako implementirani kriptografski algoritmi ne povećavaju značajno količinu prometa na sabirnici u odnosu na nezaštićenu CAN komunikaciju. Naime, u slučaju najveće količine umetnutih podataka, umetnuti podaci iznose svega 2.34% od ukupne količine poslanih podataka.

U trećem eksperimentu analiziran je utjecaj povećanja ukupnog broja podataka na vrijeme izvođenja računalnog koda. Veličina jednog podatka je iznosila jedan bajt, a odabrane su ukupne količine podataka od pet stotina bajta, tisuću bajta, tisuću i pet stotina bajta, te dvije tisuće bajta. Kao i u prethodnom primjeru, u slučajevima kada količina informacijskih podataka nije cjelobrojno djeljiva s veličinom bloka, izvršeno je umetanje podataka do potrebne veličine bloka. Vremena izvođenja implementacija DES i AES kriptografskih algoritama za različite količine podataka prikazani su u tablici 4.4.

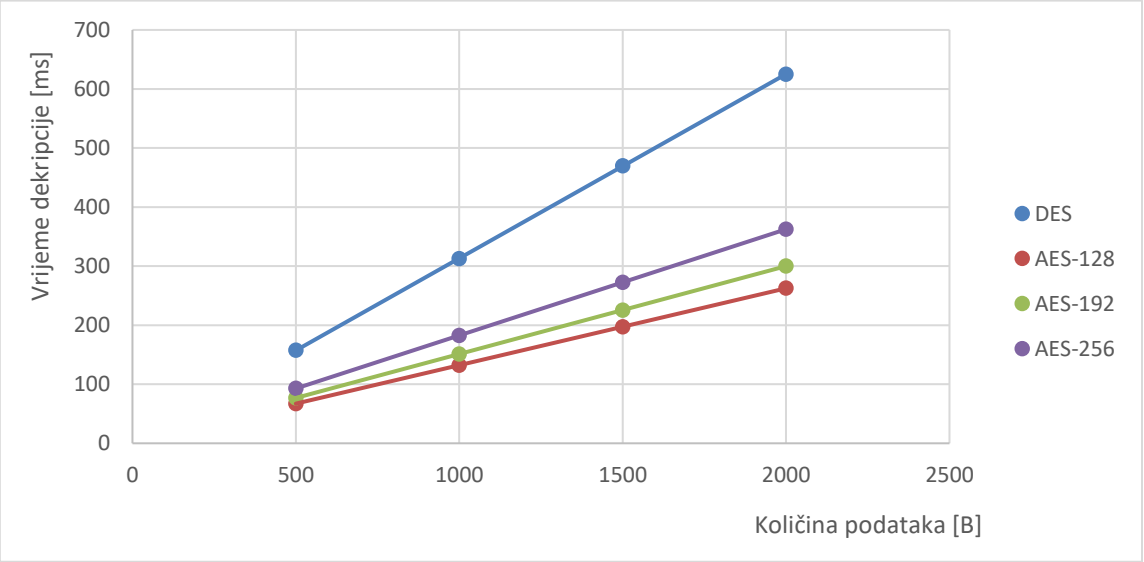
Tablica 4.4. Vrijeme izvođenja implementacija DES i AES kriptografskih algoritama za različite količine podataka

Količina podataka [B]	**Ukupno vrijeme enkripcije [ms]**				**Ukupno vrijeme dekripcije [ms]**			
	DES	**AES-128**	**AES-192**	**AES-256**	**DES**	**AES-128**	**AES-192**	**AES-256**
500	151.2	60.8	76.8	89.6	157.5	67.2	76.8	92.8
1000	300.0	119.7	151.2	176.4	312.5	132.3	151.2	182.7
1500	451.2	178.6	225.6	263.2	470	197.4	225.6	272.6
2000	600.0	237.5	300.0	350.0	625.0	262.5	300.0	362.5

Vidljivo je kako ukupno vrijeme izvođenja linearno raste s porastom broja podataka, što je očekivano s obzirom na povećanje broja pojedinih kriptografskih operacija. Slika 4.16. prikazuje graf linearne ovisnosti količine podataka i vremena izvođenja za DES i AES kriptografske algoritme enkripcije, dok slika 4.17. prikazuje graf linearne ovisnosti količine podataka i vremena izvođenja za DES i AES algoritme dekripcije.



Slika 4.16. Linearna ovisnost količine podataka i vremena izvođenja implementiranih DES i AES algoritama enkripcije



Slika 4.17. Linearna ovisnost količine podataka i vremena izvođenja implementiranih DES i AES algoritama dekripcije

Usporedbom dobivenih rezultata eksperimenata, može se zaključiti kako je najefikasnije kriptografsko rješenje AES algoritam sa veličinom ključa od šesnaest bajta. Naime, izuzev činjenice kako je DES algoritam danas nesiguran od napada grubom silom, vidljivo je kako je njegovo vrijeme izvođenja u usporedbi s preostalim algoritmima najduže. Kako AES algoritam pruža potrebnu zaštitu za sve veličine ključa, prirodno je odabrati rješenje s najkraćim vremenom izvođenja. Neznatno povećanje razine prometa je u većini slučajeva neizbježno, no odgovarajućim grupiranjem podataka u blokove otvorenog teksta, moguće je smanjiti broj umetnutih podataka.

5. ZAKLJUČAK

U okviru ovog rada su izrađeni algoritmi za enkripciju i dekripciju CAN komunikacije na ugradbenoj računalnoj platformi temeljenoj na mikroupravljaču STM32F407. Algoritmi se zasnivaju na *Data Encryption* i *Advanced Encryption* standardima. Sam proces zaštićene komunikacije se odvija u tri koraka koji uključuju enkripciju podataka na strani pošiljatelja, slanje podataka CAN sabirnicom, te primanje i dekripciju podataka na strani primatelja. Implementacija kriptografskih algoritama najprije je izvedena u C programskom jeziku u Eclipse razvojnom okruženju na osobnom računalu, a zatim je implementacija prilagođena za izvođenje na mikroupravljaču STM32F407. CAN komunikacija ostvarena je uporabom dostupne ugradbene računalne platforme temeljene na mikroupravljačima STM32F407 i MRMCU-2551 primopredajnicima. Mikroupravljači su programirani uporabom STM32CubeIDE razvojnog okruženja uz pomoć HAL_CAN biblioteke. Provedeni su eksperimenti u svrhu verifikacije i evaluacije implementiranog rješenja zaštite CAN komunikacije. Rezultati verifikacije su pokazali kako oba rješenja pružaju određeni stupanj zaštite CAN komunikacije od neovlaštenog čitanja podataka koji se razmjenjuju putem CAN sabirnice. Prema rezultatima evaluacije zaključeno je kako oba rješenja u određenoj mjeri povećavaju razinu prometa na CAN sabirnici, no da je rješenje temeljeno na *Advanced Encryption Standard* algoritmu efikasnije u pogledu vremena izvođenja.

Ipak, zbog korištenja samo jednog sigurnosnog mehanizma, predloženi način zaštite CAN komunikacije u svojoj sadašnjoj izvedbi nije prikladan za upotrebu u stvarnim sustavima. Kombiniranjem razvijenog rješenja sa mehanizmima segmentacije mreže, autentikacije i otkrivanja upada, te prikladnom optimizacijom i testiranjem konačnog algoritma moglo bi se ostvariti rješenje primjenjivo u stvarnim sustavima.

LITERATURA

[1] Statista, „*Automotive electronics cost as a percentage of total car cost worldwide from 1970 to 2030*“, https://www.statista.com/statistics/277931/automotive-electronics-cost-as-a-share-of-total-car-cost-worldwide/, pristup ostvaren 27.8.2023.

[2] CSS Electronics, „*LIN Bus Explained - A Simple Intro [2023]*“, https://www.csselectronics.com/pages/lin-bus-protocol-intro-basics, pristup ostvaren 27.8.2023.

[3] NI, „*FlexRay Automotive Communication Bus Overview*“, https://www.ni.com/en/shop/seamlessly-connect-to-third-party-devices-and-supervisory-system/flexray-automotive-communication-bus-overview.html, pristup ostvaren 27.8.2023.

[4] M. Bozdal, M. Samie, I. Jennions, „*A Survey on CAN Bus Protocol: Attacks, Challenges, and Potential Solutions*“, 2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE), Ujedinjeno Kraljevstvo, 2018.

[5] CAN in Automation, „*History of CAN technology“*, https://www.can-cia.org/can-knowledge/can/can-history , pristup ostvaren 13.06.2023.

[6] W. Voss, „*Controller Area Network Prototyping with Arduino*“, Copperhill Technologies Corporation, Sjedinjene Američke Države, 2014.

[7] ISO, „*Road vehicles - Controller area network (CAN) - Part 1: Data link layer and physical signalling*“, ISO, Švicarska, 2003.

[8] M. Natale, H. Zeng, P. Giusto, A. Ghosal, „*Understanding and Using the Controller Area Network Communication Protocol - Theory and Practice*“, Springer, Sjedinjene Američke Države, 2012.

[9] Noregon, „*What Does OBD Stand For?“*, https://www.noregon.com/what-is-obd/, pristup ostvaren 25.8.2023.

[10] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T.Kohno, „*Experimental security analysis of a modern automobile*“, Proceedings of the IEEE Symposium on Security and Privacy, Sjedinjene Američke Države, 2010.

[11] S. Checkoway, D. McCoy, B. Kantor, „*Comprehensive experimental analyses of automotive attack surfaces*“, Proceedings of the 20th USENIX conference on Security, Sjedinjene Američke Države, 2014.

[12] T. Hoppe, S. Kiltz, J. Dittmann, „*Security threats to automotive CAN networks - Practical examples and selected short-term countermeasures*“, 2011.

[13] McAfee, C. Beek, R. Samani, „*DEFCON—Connected Sar Security*“, https://securingtomorrow.mcafee.com/other-blogs/mcafee-labs/defcon-connected-car-security/ , pristup ostvaren 25.07.2023.

[14] M. Bozdal, M.Samie, S. Aslam, I. Jennions „*Evaluation of CAN Bus Security Challenges*“, Proceedings of the 2018 International Conference on Computing, Electronics & Communications Engineering (iCCECE), Ujedinjeno Kraljevstvo, 2018.

[15] R. Deeksha, K. Paramasivam, „*Design and validation of CAN protocol with enhanced Security in electric vehicle*“, Proceedings of the Fifth International Conference on Computing Methodologies and Communication, 2021.

[16] E. Marasco, F Quaglia, „*AuthentiCAN: a Protocol for Improved Security over CAN*“, Fourth World Conference on Smart Trends in Systems, Security and Sustainability, 2020.

[17] B. Palaniswamy, S. Camtepe, E. Foo, J. Pieprzyk, „*An Efficient Authentication Scheme for Intra-Vehicular Controller Area Network*“, IEEE Transactions On Information Forensics And Security, vol. 15, 2020.

[18] H. J. Jo, J. H. Kim, H. Choi, W. Choi, D. H. Lee, I. Lee, „*MAuth-CAN: Masquerade-Attack-Proof Authentication for In-Vehicle Networks*“, IEEE Transactions on Vehicular Technology, vol. 69, no. 2, 2020.

[19] W. Stallings, „*Cryptography and Network Security - Principles and Practice*“, Pearson Education Limited, Ujedinjeno Kraljevstvo, 2022.

[20] K. Grgić, predavanje s kolegija Sigurnost računalnih sustava, „*Simetrični kriptosustavi*“, FERIT Osijek, 2021.

[21] W. Stallings, „*Network Security Essentials - Applications and Standard*“, Pearson Education Limited, Ujedinjeno Kraljevstvo, 2017.

[22] STM32F407/417 značajke, https://www.st.com/en/microcontrollers-microprocessors/stm32f407-417.html, pristup ostvaren 02.02.2023.

[23] STM32F407 *datasheet*, https://www.alldatasheet.com/view.jsp?Searchword=STM32F407&sField=2, pristup ostvaren 02.02.2023.

[24] MRMCU-2551 *datasheet*, https://ww1.microchip.com/downloads/en/devicedoc/20001667g.pdf, pristup ostvaren 19.02.2023.

[25] STM32CubeIDE razvojno okruženje i dokumentacija, https://www.st.com/en/development-tools/stm32cubeide.html, pristup ostvaren 05.02.2023.

[26] Dokumentacija HAL_CAN biblioteke, https://www.st.com/resource/en/user_manual/um1725-description-of-stm32f4-hal-and-lowlayer-drivers-stmicroelectronics.pdf, pristup ostvaren 22.02.2023.

SAŽETAK

U radu je predstavljen CAN standard, nakon čega su opisani njegovi nedostatci u pogledu sigurnosti. Navedena su određena postojeća rješenja zaštite CAN komunikacije, te pregled prednosti i nedostataka navedenih rješenja. U okviru ovog rada implementirana su dva rješenja zaštite CAN komunikacije temeljena na *Data Encryption Standard* i *Advanced Encryption Standard* kriptografskim algoritmima. Predstavljeni su standardi odabranih kriptografskih algoritama, a njihova implementacija je izrađena u programskom jeziku C u Eclipse razvojnom okruženju. Za ostvarenje zaštićene CAN komunikacije korištene su ugradbene računalne platforme temeljene na mikroupravljačima STM32F407. Naposlijetku je provedena verifikacija i evaluacija implementiranih kriptografskih algoritama. U svrhu verifikacije su izvedena tri eksperimenta, prvo pomoću Eclipse razvojnog okruženja na osobnom računalu, a nakon toga pomoću USB-CAN adaptera na ugradbenoj računalnoj platformi temeljenoj na mikroupravljačima STM32F407. Evaluacija implementiranih kriptografskih algoritama je provedena s obzirom na vrijeme izvođenja programskog koda, te s obzirom na razinu prometa na CAN sabirnici. Izvedena su tri eksperimenta na ugradbenoj računalnoj platformi temeljenoj na mikroupravljačima STM32F407. Promatrani su slučajevi različitog sadržaja bloka kriptografskih algoritama, različite veličine pojedinog podatka koji se pakira u blok konstantne veličine, te različite ukupne količine kriptiranih podataka.

Ključne riječi: CAN, kriptografija, DES, AES, mikroupravljač.

ENCRYPTING CAN COMMUNICATION ON AN EMBEDDED COMPUTER PLATFORM

ABSTRACT

This master's thesis presents the CAN standard, followed by a description of its security flaws. Certain existing solutions for protecting CAN communication are mentioned, along with an overview of the advantages and disadvantages of these solutions. Within this work, two solutions for securing CAN communication based on Data Encryption Standard and Advanced Encryption Standard cryptographic algorithms are implemented. The standards of the selected cryptographic algorithms are introduced, and their implementation is done in the C programming language in the Eclipse development environment. Embedded computer platforms based on STM32F407 microcontrollers are used for achieving secure CAN communication. Finally, the verification and evaluation of the implemented cryptographic algorithms are conducted. Three experiments were carried out for verification, first using the Eclipse development environment on a personal computer, and then using a USB-CAN adapter on an embedded computer platform based on STM32F407 microcontrollers. The evaluation of the implemented cryptographic algorithms is done concerning the execution time of the code and the level of traffic on the CAN bus. Three experiments were performed on the embedded computer platform based on STM32F407 microcontrollers, considering different content of cryptographic algorithm blocks, different sizes of individual data packed into blocks of constant size, and different total amounts of encrypted data.

Keywords: CAN, cryptography, DES, AES, microcontroller.

ŽIVOTOPIS

Iva Horvat rođena je 29. kolovoza 1997. godine u Osijeku. Nakon završene Elektrotehničke i prometne škole u Osijeku, 2016. godine ostvaruje upis na preddiplomski sveučilišni studij elektrotehnike i informacijske tehnologije na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija, tadašnji Elektrotehnički fakultet u Osijeku. 2020. godine stječe akademski naziv sveučilišne prvostupnice (lat. *baccalaureus*) inženjerka elektrotehnike i informacijske tehnologije. Iste godine upisuje diplomski sveučilišni studij elektrotehnike, smjer komunikacije i informatika.

PRILOZI

Prilog P.2.1. Supstitucijske S-kutije u postupku DES enkripcije i dekripcije

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	01	08	14	06	11	03	04	09	07	02	13	12	00	05	10
1	03	13	04	07	15	02	08	14	12	00	01	10	06	09	11	05
2	00	14	07	11	10	04	13	01	05	08	12	06	09	03	02	15
3	13	08	10	01	03	15	04	02	11	06	07	12	00	05	14	09

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	10	00	09	14	06	03	15	05	01	13	12	07	11	04	02	08
1	13	07	00	09	03	04	06	10	02	08	05	14	12	11	15	01
2	13	06	04	09	08	15	03	00	11	01	02	12	05	10	14	07
3	01	10	13	00	06	09	08	07	04	15	14	03	11	05	02	12

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	07	13	14	03	00	6	09	10	1	02	08	05	11	12	04	15
1	13	08	11	05	06	15	00	03	04	07	02	12	01	10	14	09
2	10	06	09	00	12	11	07	13	15	01	03	14	05	02	08	04
3	03	15	00	06	10	01	13	08	09	04	05	11	12	07	02	14

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	02	12	04	01	07	10	11	06	08	05	03	15	13	00	14	09
1	14	11	02	12	04	07	13	01	05	00	15	10	03	09	08	06
2	04	02	01	11	10	13	07	08	15	09	12	05	06	03	00	14
3	11	08	12	07	01	14	02	13	06	15	00	09	10	04	05	03

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	01	10	15	09	02	06	08	00	13	03	04	14	07	05	11
1	10	15	04	02	07	12	09	05	06	01	13	14	00	11	03	08
2	09	14	15	05	02	08	12	03	07	00	04	10	01	13	11	06
3	04	03	02	12	09	05	15	10	11	14	01	07	10	00	08	13

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	11	2	14	15	00	08	13	03	12	09	07	05	10	06	01
1	13	00	11	07	04	09	01	10	14	03	05	12	02	15	08	06
2	01	04	11	13	12	03	07	14	10	15	06	08	00	05	09	02
3	06	11	13	08	01	04	10	07	09	05	00	15	14	02	03	12

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	02	08	04	06	15	11	01	10	09	03	14	05	00	12	07
1	01	15	13	08	10	03	07	04	12	05	06	11	10	14	09	02
2	07	11	04	01	09	12	14	02	00	06	10	10	15	03	05	08
3	02	01	14	07	04	10	8	13	15	12	09	09	03	05	06	11