Sadržaj

1. Uvod

U 21. stoljeću, upravljački sustavi imaju vrlo važnu ulogu u cjelokupnoj industriji. Središnji su dio proizvodnog procesa i distribucije u raznim industrijama [1]. Upravljački sustav je naziv za sustav koji upravlja radom drugih sustava ili uređaja pomoću upravljačkih petlji koje su njegove temeljne građevne jedinke [2]. Neki od ciljeva primjene upravljačkih sustava su automatizacija procesa, povećanje efikasnosti, optimizacija, olakšanje nadzora, upravljanja itd.

Njihovo područje primjene je vrlo široko. Može ih se pronaći u uređajima koje koristimo ili s kojima se susrećemo svakodnevno kao što su termostat, automatska vrata, semafor, automatska perilica rublja i audio sustavi. Također, njihova se primjena proteže na cjelokupne grane industrije poput farmaceutske industrije, proizvodnje i distribucije električne energije, naftne industrije, prehrambene industrije te metaloprerađivačke industrije. Od svih primjena, uz ovaj rad najviše je povezana primjena u elektroenergetskim sustavima i to iz razloga što je inspiracija za ovaj rad odabran znanstveni rad u kojem su predloženi mehanizmi namijenjeni za takve sustave.

Elektroenergetski sustav uključuje proizvodnju, distribuciju i potrošnju električne energije te ga sačinjavaju elektrane, trafostanice, dalekovodi te krajnji uređaji koji koriste električnu energiju za svoj rad [3]. U takvom sustavu sigurnost je izuzetno bitan faktor. Razlog je tomu što su posljedice proboja u takav sustav katastrofalne. Na primjer ogromna financijska šteta, nedostupnost električne energije ili čak i gubitak ljudskih života. To su samo neki od razloga nastanka normi za komunikaciju u elektroenergetskim sustavima. Dvije vrlo važne norme u ovom području su IEC 61850 i IEC 62351.

IEC 61850 standard olakšava komunikaciju između inteligentnih elektroničkih uređaja (IED-a) na način da definira strukture podataka te protokole za razmjenu informacija u trafostanicama (*substations*) [4]. Za razliku od njega, IEC 62351 standard nastao je kao dopuna standardu IEC 61850 na način da dodaje sigurnosne značajke koje su nedostajale jer je prepoznata važnost sigurnosti u tako osjetljivom sustavu. Podaci i naredbe koje se prenose u komunikaciji unutar trafostanice su vrlo osjetljivi stoga ih je bitno na pravi način zaštiti.

Način na koji se to može ostvariti je osiguravanje njihove tajnosti i cjelovitosti. Rješenje za takav problem je korištenje kriptografskih mehanizama. Centar cjelokupnog kriptografskog mehanizma su ključevi koji se koriste jer je njihovom kompromitacijom kompromitiran cijeli sustav. Zbog toga je vrlo bitno sigurno raspodijeliti odgovarajuće kriptografske ključeve svim sudionicima komunikacije unutar sustava. Upravo je raspodjela kriptografskih ključeva unutar sustava koji implementira IEC 61850 standard tema ovog rada. Kao polazna točka za implementaciju tog mehanizma u sklopu ovog rada odabrana je biblioteka otvorenog koda libIEC61850 koja implementira standard IEC 61850. Ona će biti proširena mehanizmom raspodjele kriptografskih ključeva čija je namjena pružiti kriptografsku zaštitu protokolima koji su dio navedene biblioteke. Kao protokol za raspodjelu kriptografskih ključeva koji će biti implementiran je odabran protokol predložen u [5].

Rad je strukturiran na sljedeći način. U drugom poglavlju biti će opisan standard IEC 61850 zajedno sa sustavima automatizacije trafostanice te modelima podataka i načinima komuniciranja unutar njih. Nakon toga, u trećem poglavlju je ukratko predstavljena biblioteka libIEC61850 koja je temelj za implementaciju protokola raspodjele ključeva u sklopu ovog rada. Zatim je u četvrtom poglavlju predstavljena sigurnost standarda IEC 61850 na način da su objašnjene prijetnje i napadi na njega kao te standard IEC 62351 koji je nadopuna prethodno spomenutom standardu. U petom poglavlju je opisan protokol koji se implementira u sklopu ovog rada zajedno sa detaljima implementacije. Konačno, u šestom poglavlju se nalazi zaključak donesen nakon proučavanja mnogo znanstvenih radova na ovu temu te nakon samog pisanja ovog rada.

2. IEC 61850 standard

U ovom poglavlju opisan je standard IEC 61850, njegova važnost i posebnosti. On je izuzetno bitan za ovaj rad iz razloga što je cilj raspodjele kriptografskih ključeva osigurati komunikaciju unutar sustava koji implementiraju navedeni standard.

IEC 61850 međunarodni je komunikacijski standard namijenjen primjeni unutar trafostanica. Autor mu je Međunarodno elektrotehničko povjerenstvo (*International Electrotechnical Commission* - IEC). IEC je vodeća svjetska organizacija koja se bavi izradom i objavljivanjem međunarodnih standarda za cjelokupnu električnu tehnologiju [6]. Kao što je već ranije spomenuto, IEC 61850 olakšava komunikaciju između inteligentnih elektroničkih uređaja (IED-a) na način da definira strukture podataka te protokole za razmjenu informacija unutar trafostanica [4]. IEC 61850 je s vremenom postao najpopularniji te najperspektivniji standard za komunikaciju u svom području iz razloga što koristi objektno orijentiran dizajn koji omogućuje interoperabilnost u sustavu koji koristi uređaje različitih proizvođača [7].

Osim toga, prednosti su mu i smanjenje troškova održavanja i postavljanja, pružanje platforme koju je lako nadograđivati u budućnosti sukladno s razvojem tehnologije, olakšano dodavanje novih uređaja i njihovo postavljanje u trenutni sustav itd. U narednim potpoglavljima opisana je struktura te specifičnosti IEC 61850 standarda, sustav automatizacije trafostanice te modeli podataka i protokoli korišteni za komunikaciju unutar trafostanica.

2.1. Struktura IEC 61850 standarda

Prva verzija standarda se sastoji od 10 glavnih dijelova [7]. U prvom dijelu nalazi se uvod te pregled samog standarda. U njemu se također definira svrha standarda. Drugi dio je rječnik pojmova koji se koriste u radu i koji čitatelju pomaže razumijevanje standarda. Treći dio definira opće zahtjeve. Četvrti dio propisuje upute za upravljanje sustavom i projektima. U petom dijelu

nalaze se komunikacijski zahtjevi definirani za jedan takav sustav. Šesti dio definira opisni jezik za konfiguraciju IED-a koji se naziva *System Configuration description Language* (SCL) [7]. Sedmi dio čini opis struktura komunikacije. U njemu su prikazani razni građevni blokovi kao što su *Abstract Communication Service Interface* (ACSI), *Logical Node* (LN), *Common Data Class* (CDC) i *Data Object* (DO) [7]. Osmi i deveti dio prikazuju mapiranje definiranih struktura podataka i usluga na 4 protokola, a oni su *Manufacturing Message Specification* (MMS), *Generic Object Oriented System Event* (GOOSE), *Sampled Values* (SV) i *Sampled Measure Values* (SMV). Posljednje deseto poglavlje daje upute na koji način provesti testiranje same implementacije onoga što je navedeno u standardu.

2.2. Sustav automatizacije trafostanice

Standard je prvotno bio zamišljen, a samim time i objavljen kao standard isključivo za automatizaciju trafostanica, ali je s vremenom izrastao u ono što je danas, a to je komunikacijski standard za trafostanice. Implementacija IEC 61850 standarda ima za cilj stvaranje jedinstvene platforme unutar trafostanice koja prije svega omogućuje automatizaciju unutar nje. Automatizacija sama po sebi jamči uštede na resursima što je vrlo bitan faktor prilikom donošenja odluke da li implementirati neki standard u postojeći sustav. Za ostvarivanje automatizacije je naravno nužno osigurati sigurnu i pouzdanu komunikaciju kao dio te platforme. Komunikacija je ključna jer se njenim kanalima prenose razne naredbe kojima se upravlja trafostanicom, ali i razna mjerenja koja su provedena te koja su ključna za analizu i nadzor rada trafostanice. Uz sve to, potrebni su i sigurnosni mehanizmi kako bi se trafostanice zaštitile od potencijalnih napadača koji mogu prouzročiti veliku štetu.

Upravo navedene karakteristike su sastavni dijelovi sustava automatizacije trafostanice (*Substation Automation Systems* - SAS). SAS je naziv za hardverske i softverske komponente koje se nalaze unutar trafostanice te koje služe automatizaciji radnog procesa te omogućuju njeno upravljanje i nadzor [8]. SAS je suvremena trafostanica koja koristi informacijsku tehnologiju u svom radu tj. na standardno fizičko ožičenje koje se koristi kod klasičnih trafostanica se nadodala upotreba industrijskih komunikacijskih protokola kao što su fieldbus i ethernet.

Problem s klasičnim trafostanicama je u tome što nema opcije udaljenog nadzora i analize te upozorenja u slučaju kvara prilikom kojeg bi se puno brže moglo reagirati. Ali nije samo stvar u vremenu potrebnom da se uoči prisutnost problema već i u mogućnosti da se taj problem brzo identificira i otkloni. To bi posljedično značilo da su prekidi u opskrbi električnom energijom krajnjim korisnicima bili češći i duži. Ovi problemi su znatno manje naglašeni u SAS-u jer u svakom trenutku postoji opcija provjere rada trafostanice što je omogućeno kontinuiranim mjerenjima i slanjem istih, ali i ostalih dijagnostičkih informacija kako bi se u što kraćem roku identificirao ili čak i predvidio kvar te kako bi se na njega pravovremeno reagiralo.

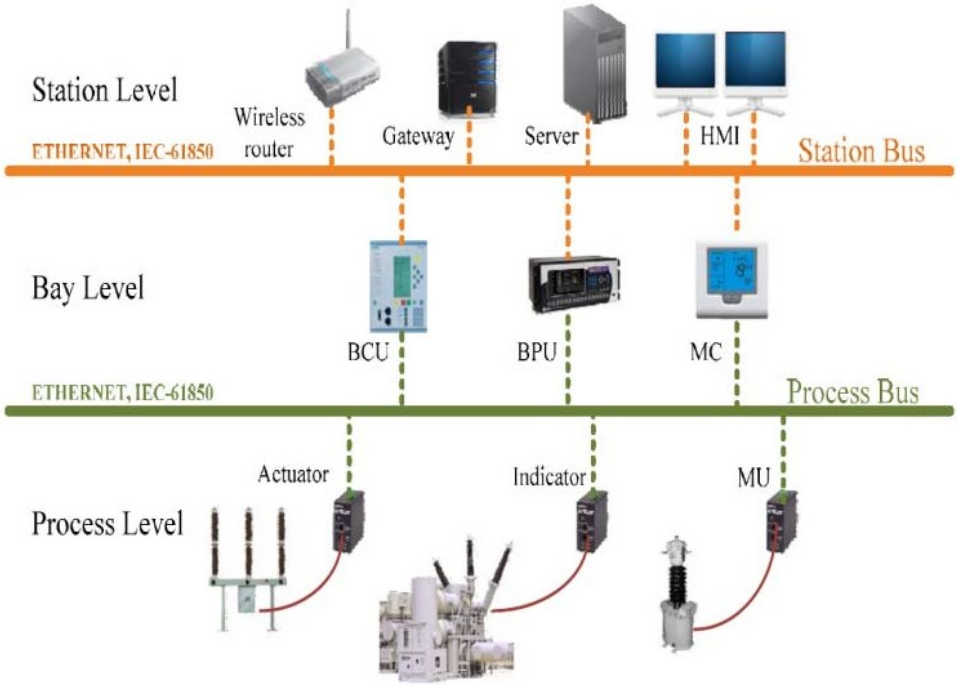
2.2.1. 3-razinski raspored SAS-a

Kako bi navedene funkcionalnosti u SAS-u bile izvedive, potrebni su i uređaji koji ih ostvaruju. U trafostanicama postoje dvije vrste opreme, a to su primarna oprema i sekundarna oprema [9]. Primarna oprema uključuje transformatore, koji su jezgra same trafostanice te služe tome da snižavaju ili povećavaju razine napona ovisno o potrebi, ali i opremu poput raznih prekidača i sabirnica. Sekundarnu opremu čine upravljačke ploče, zaštitna oprema poput zaštitnih releja, mjerni uređaji i klasična komunikacijska oprema kao što su preklopnici, koncentratori, usmjerivači i modemi. U standardu IEC 61850 spomenuta sekundarna oprema može se podijeliti na 3 razine kao što je to prikazano na slici (Slika 2.1), a one su razina procesa (*process level*), razina polja (*bay level*) i razina stanice (*station level*) [9].

Razina procesa obuhvaća fizičke procese i uređaje poput raznih senzora koji su direktno povezani s primarnom opremom preko bakrenih žica te nad njima provode mjerenja primjerice napona i struje. Povezanost uređaja procesne razine poput senzora na primarnu opremu i na procesnu sabirnicu vidi se na dnu slike (Slika 2.1).

Prikupljeni podaci na razini procesa šalju se komunikacijskim kanalom koristeći definirane protokole IED-ima na razini polja koji onda nad tim podacima provode analizu kako bi se utvrdilo da li trafostanica ispravno funkcionira i da li su potrebne neke preventivne radnje kako bi se potencijalni kvar spriječio ili možda i uklonio. Dakle razina polja se sastoji od IED-a koji obavljaju funkcije nadzora i upravljanja unutar trafostanice. Uređaji na razini polja spojeni su procesnu

sabirnicu, ali i na sabirnicu koja pripada razini stanice što je vidljivo na sredini slike (Slika 2.1). Zelena (donja) debela linija predstavlja procesnu sabirnicu, a narančasta (gornja) sabirnicu razine stanice.



Slika 2.1 3- slojni dizajn SAS-a (preuzeto iz [9])

Okosnicu razine stanice čine *Human-Machine Interface* (HMI) i *Supervisory control and data acquisition* (SCADA). HMI je korisničko sučelje, koje može biti fizičko poput gumba ili računalne kontrolne ploče, koje omogućuje operateru pristup kontroleru za industrijski sustav [10]. Za razliku od njega, SCADA je sustav za upravljanje industrijskim procesima što naravno podrazumijeva i kontinuirano prikupljanje podataka s udaljenih lokacija kako bi se isti mogli obrađivati te kako bi se posljedično moglo upravljati uređajima unutar sustava [11]. SCADA je vrsta upravljačkog sustava te centar cjelokupnog nadzora i upravljanja trafostanicom. SCADA upravlja svim komponentama trafostanice preko HMI-a. Kao što je vidljivo u gornjem desnom vrhu slike (Slika 2.1), HMI je spojen na sabirnicu razine stanice što omogućava slanje naredbi i upravljanje uređajima na razini polja.

2.3. Komunikacija i model podataka

Kako bi čitav sustav ispravno funkcionirao, neophodno je osigurati da komponente unutar njega imaju mogućnost razmjene podataka i naredbi. Da bi komunikacija između različitih razina SAS-a bila moguća potrebni su modeli podataka i protokoli za obavljanje cjelokupne komunikacije unutar trafostanice.

2.3.1. Model podataka IEC 61850 standarda

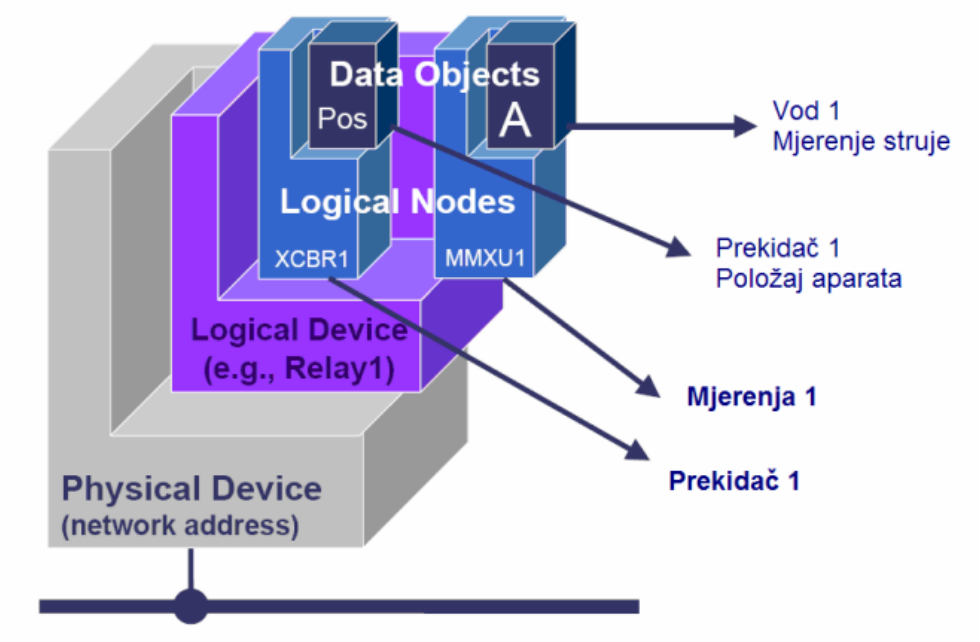
Kao što je već ranije spomenuto, IEC 61850 koristi objektno orijentiran dizajn koji pridonosi tome da se složene strukture podataka pretvore u manje i jednostavnije strukture. Objektni model biti će objašnjen na primjeru preuzetom iz [12].

Pri prvom pogledu na sliku (Slika 2.2), odmah je vidljiv objektni model podataka jer je jedna složena struktura raščlanjena na više jednostavnijih struktura koje su prikazane različitim bojama. Najveća i vanjska struktura je IED, odnosno na slici (Slika 2.2) *Physical Device*, tj. uređaj koji obavlja definirane radnje. U ovom primjeru IED je relej. Radnje koje relej izvršava se dijele u cjeline koje se nazivaju logički uređaji (*Logical Devices* - LD). Logički uređaj na slici (Slika 2.2) je prikazan ljubičastom bojom. Primjer logičkog uređaja za relej bi bila zaštita jer releji generalno služe za zaštitu električnih komponenti od oštećenja.

Sljedeća još manja struktura koju sadrže logički uređaji su logička čvorišta (*Logical nodes* - LN) koja su na slici (Slika 2.2) prikazani svjetlo plavom bojom. Logička čvorišta predstavljaju funkcionalnosti uređaja. Na slici (Slika 2.2), dvije donje strelice prikazuju dva logička čvorišta releja koja su *Mjerenja 1* i *Prekidač 1* što znači da su to funkcionalnosti koje taj relej pruža.

Identifikator logičkog čvorišta sastoji se od 4 slova te broja koji je prisutan i u samom nazivu logičkog čvorišta koji predstavlja koji je redni broj tog logičkog čvorišta koji ima istu funkciju. Cjelokupna nomenklatura za identifikatore logičkih čvorišta se nalazi u standardu IEC 61850. Spomenuti identifikatori se nalaze na dnu tamno plavih kvadara koji predstavljaju logička čvorišta.

Specifičnost logičkog čvorišta je u tome da ono može komunicirati s logičkim čvorištem koje se nalazi „unutar“ nekog drugog uređaja.



Slika 2.2 Shema pojednostavljenog logičkog modela (preuzeto iz [12])

Osim što predstavlja funkcionalnost uređaja, logičko čvorište može sadržavati više podatkovnih objekata (*Data Objects - DO*). Podatkovni objekti reprezentiraju podatke i svojstva koja su vezana za konkretno nadređeno logičko čvorište. U spomenutoj komunikaciji između logičkih čvorišta različitih uređaja glavnu ulogu imaju podatkovni objekti jer se njima prenose potrebne informacije koje se generiraju primjerice prilikom mjerenja struje kao što je i prikazano u gornjem desnom kutu slike (Slika 2.2) gdje se vidi tamno plavi kvadar koji označava podatkovni objekt točnije mjerenje struje što je naznačeno strelicom koja kreće iz njega.

2.3.2. Komunikacija u IEC 61850 sustavu

Osim što je potrebno omogućiti samu komunikaciju unutar IEC 61850 sustava, potrebno je i osigurati da ta komunikacija bude pouzdana, brza i sigurna. Za te potrebe unutar IEC 61850 standarda definirane su apstraktne usluge poput *abstract communication service interface* (ACSI),

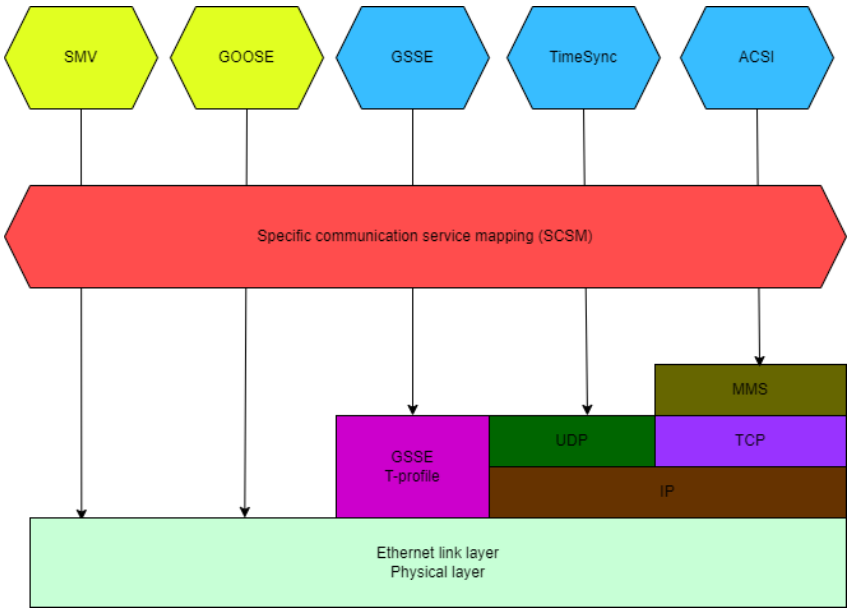
generic object oriented substation events (GOOSE), *TimeSync*, *generic substation state event* (GSSE) i *sampled measured value* (SMV) [13].

Za razliku od protokola za komunikaciju koji se koriste u IEC 61850 standardu, ACSI je sučelje za IED koje omogućuje i olakšava komunikaciju na način da pruža apstraktne komunikacijske usluge [14]. Njegova posebnost je u tome da je njegov rad neovisan o tome kako se odvija komunikacija između na primjer IED-a u sustavu. Upravo to svojstvo omogućuje interoperabilnost u sustavu tako da se mogu koristiti uređaji raznih proizvođača. Samim time to znači da će puno više sustava moći primijeniti ovaj standard za razliku od slučaja u kojem bi se koristio neki specifičan način komunikacije. To zapravo znači da ga univerzalnost ga čini znatno popularnijim.

ACSI koristi model klijent-poslužitelj u kojem klijent šalje zahtjeve za podatke ili usluge poslužitelju, a poslužitelj na njih odgovara i izvršava zatražene radnje ili naredbe. Navedeno sučelje podržava usluge kao što su upravljanje uređajima, pristup podacima te izvješćivanje. ACSI je mapiran na protokol *manufacturing message specification* (MMS) kao što je vidljivo na najdesnijoj strelici na slici (Slika 2.3). MMS je protokol čija je namjena razmjena podataka između sustava poput SCADA-e i IED-a i to preko ethernet-a [15]. Podržava operacije kao što su čitanje i pisanje podataka, kreiranje te brisanje objekata te izvješćivanje. MMS je mapiran na protokole TCP i IP što se vidi u donjem desnom kutu slike (Slika 2.3).

Mapiranje ACSI-a, ali i ostalih usluga i protokola, je provedeno preko servisa *specific communication service mapping* (SCSM) što je vidljivo kao crveni šesterokut koji ide preko svih strelica na sredini slike (Slika 2.3). SCSM je procedura koja definira mapiranje servisa kao što je primjerice ACSI na konkretan komunikacijski protokol ili tehnologiju [16].

U IEC 61850 standardu, definiran je i servis za sinkronizaciju vremena (*TimeSync*) čija namjena je osigurati vremensku sinkronizaciju uređaja za koje je neophodno da imaju vrlo precizno usklađeno vrijeme zbog vremenskih ograničenja koja su postavljenja na određenu komunikaciju unutar trafostanice kao što su primjerice slanje upravljačkih naredbi poput uključivanja i isključivanja uređaja. Servis za sinkronizaciju vremena je mapiran na UDP protokol kao što je vidljivo na drugoj po redu strelici s desne strane na slici (Slika 2.3).



Slika 2.3 Arhitektura protokola u IEC 61850 standardu (preslikano iz [13])

Za konkretnu primjenu u IEC 61850 standardu, definiran je servis koji se zove *Generic Substation Event* (GSE). On služi za razašiljanje vremenski kritičnih poruka o nekom događaju unutar trafostanice. Razašiljanje u ovom kontekstu znači da se jedna poruka o nekom događaju unutar trafostanice šalje na više različitih odredišta odnosno uređaja. Primjeri vremenski kritičnih poruka unutar trafostanice bi bile primjerice poruke o pokretanju ili gašenju nekog mehanizma ili uređaja te poruke o određenom kvaru. GSE kao servis se dijeli na *Generic Object Oriented Substation Events* (GOOSE) i *Generic Substation State Events* (GSSE). GSSE je prethodnik protokolu GOOSE koji ga zamjenjuje u primjeni u modernim SAS sustavima.

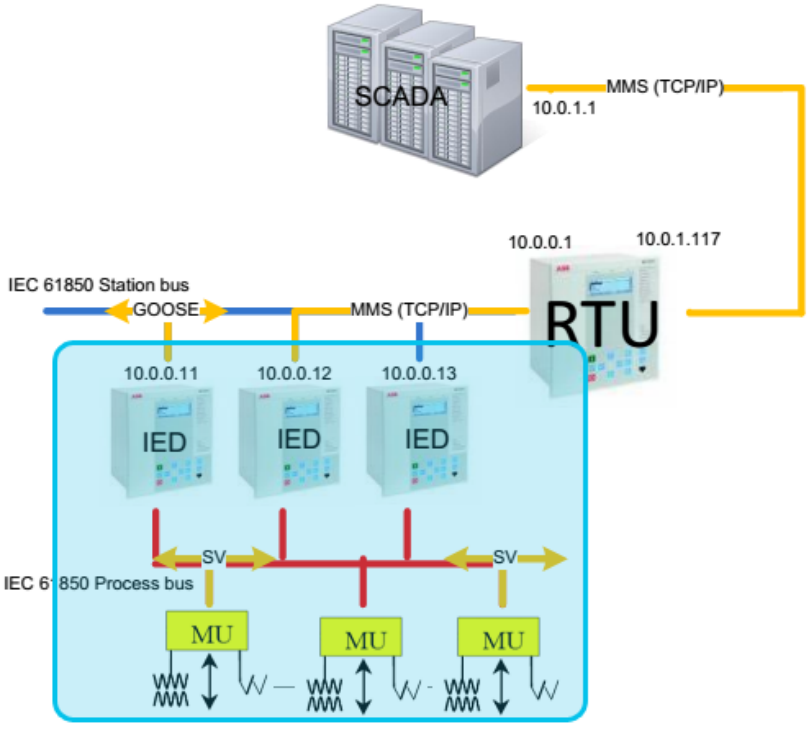
GOOSE je protokol koji je temeljen na izdavač-pretplatnik obliku komunikacije. Pretplatnik je uređaj ili više uređaja kojima je bitna ili potrebna informacija koju u nekom trenutku šalje neki drugi uređaj stoga se ti uređaji pretplaćuju da primaju određenu poruku od tog uređaja. U trenutku kada se dogodi neki specifičan događaj, uređaj koji generira poruku za koju postoje pretplatnici tu istu poruku razašilje svima koji su pretplaćeni na nju. GOOSE poruke ne sadrže adresu destinacije i ne postoji mehanizam koji radi provjeru da li je poruka uspješno primljena na strani primatelja. Budući da su poruke koje se šalju protokolom GOOSE vremenski kritične, za njih postoji definirano maksimalno kašnjenje od 3 ms [17]. GOOSE poruke su mapirane direktno na ethernet sloj što je vidljivo sa drugom po redu strelicom slijeva na slici (Slika 2.3). To omogućuje da se

izbjegne korištenje više različitih zaglavlja što posljedično znači da su i veličine GOOSE poruka manje, a samim time je i vrijeme potrebno za slanje kraće [4].

Protokol koji ima puno sličnosti s protokolom GOOSE je *sampled measured value* (SMV). Kao i GOOSE, SMV se temelji na izdavač-pretplatnik obliku komunikacije. SMV protokol se koristi za komunikaciju između IED-a i spojnih jedinica (*Merging Units* - MU). MU je uređaj koji pretvara provedeno mjerenje (na primjer struje) iz nekog drugog uređaja u digitalizirane podatke te ih zatim šalje dalje preko procesne sabirnice.

Upravo to je vidljivo na slici (Slika 2.4), uz razliku što na slici piše SV kao *sampled values* koji je zapravo isto što i SMV, ali malo širi pojam što za ovaj primjer i objašnjenje trenutno nije bitno. Na slici (Slika 2.4) unutar plavog okvira se vide žuti pravokutnici koji predstavljaju MU-e koji pretvaraju mjerenja u digitalizirane podatke koji se onda preko SV-a (SMV-a) šalju procesnom sabirnicom IED-ima što je naznačeno narančastim strelicama.

SMV je kao i GOOSE mapiran direktno na ethernet sloj što je vidljivo na prvoj strelici slijeva na slici (Slika 2.3). Također porukama oba protokola se dodjeljuje najviši prioritet tijekom prijenosa jer njihove poruke vremenski kritične i imaju definirano maksimalno kašnjenje.



Slika 2.4 SV/SMV komunikacija unutar trafostanice (preuzeto iz [26])

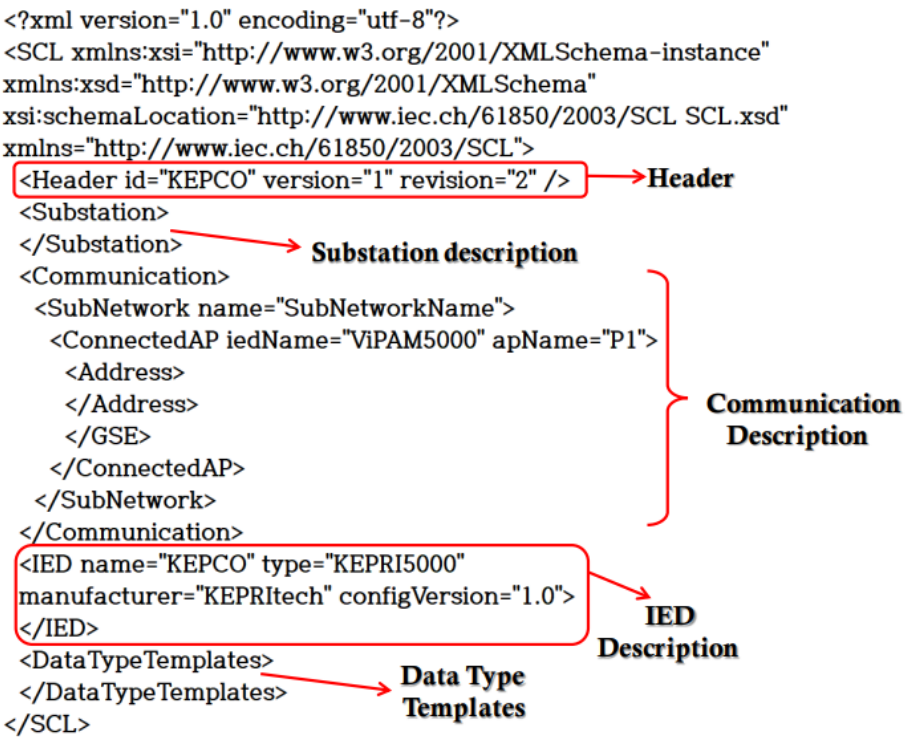
2.4. Substation Configuration Language (SCL)

Osim što je bitno da uređaji i cijeli sustav unutar trafostanice imaju mogućnost komuniciranja, potrebno je i omogućiti način na koji se definiraju sami dijelovi trafostanice kao primjerice IED-ovi te njihovi odnosi s drugim IED-ima kao što je primjerice određivanje koji IED-ovi međusobno komuniciraju.

Upravo to je omogućeno u standardu IEC 61850 koristeći jezik temeljen na XML-u koji se naziva Substation Configuration Language (SCL). Koristeći ga, mogu se definirati IED-i, podaci koji se prenose kao poruke u komunikaciji te ostale značajke komunikacije kao i cjelokupni SAS. Primjer SCL datoteke te njezinih dijelova nalazi se na slici (Slika 2.5). Prilikom prvog pogleda na SCL datoteku na slici (Slika 2.5), vidljiva je sličnost sa XML datotekama budući da je SCL jezik temeljen na XML-u.

U gornjem crvenom pravokutniku nalazi se zaglavlje koje je definirano identifikatorom te oznakom verzije i revizije. Nakon toga slijedi definicija odnosno opis trafostanice koji je u ovom slučaju prazan te naznačen crvenom strelicom koja pokazuje na tekst *Substation description*. Zatim se definira sama komunikacija koja je na slici (Slika 2.5) označena crvenom vitičastom zagradom. Unutar nje se nalaze elementi poput podmreže koja je definirana imenom, pristupne točke definirane imenom i IED-om te adrese.

Naravno neizostavni dio je svakako opis IED-a koji se nalazi u donjem crvenom pravokutniku na slici (Slika 2.5) te koji se sastoji od imena IED-a, tipa, proizvođača te konfiguracijske verzije. Konačno, na dnu slike (Slika 2.5) odnosno SCL datoteke nalaze se označeni predlošci za tipove podataka, a oni mogu biti ranije spomenuti LN i DO.



Slika 2.5 Primjer SCL datoteke te njenih dijelova (preuzeto iz [27])

3. Biblioteka libIEC61850

Budući da je tema ovog rada implementacija mehanizma upravljanja kriptografskim ključevima što je samo jedan dio mali dio u sklopu komunikacije unutar sustava koji implementira IEC 61850 standard, kao polazna točka za implementaciju mehanizma raspodjele ključeva u sklopu ovog rada odabrana je biblioteka otvorenog koda libIEC61850 koja implementira komunikacijske mehanizme iz standarda IEC 61850. Biblioteka libIEC61850 će biti proširena mehanizmom raspodjele kriptografskih ključeva čija je namjena pružiti kriptografsku zaštitu protokolima koji su dio navedene biblioteke.

3.1. Općenito o biblioteci libIEC61850

Kao što piše na službenoj stranici biblioteke [18] te na službenom GitHub repozitoriju [19] koji sadrži izvorni kod biblioteke, LibIEC61850 je biblioteka otvorenog koda implementirana u programskom jeziku C na principu klijent-poslužitelj koja pruža implementaciju sljedećih komunikacijskih protokola iz standard IEC 61850: GOOSE, MMS i SV. Autor biblioteke je njemačka tvrtka MZ Automation koja se bavi implementacijom i integracijom komunikacijskih protokola u području sustava automatizacije.

Jedan od ciljeva implementacije ove biblioteke bio je osiguravanje što veće moguće portabilnosti kako bi se biblioteka mogla koristiti na raznim uređajima što je uspješno i napravljeno tako da se biblioteka može koristiti na uređajima i računalima raznih procesnih mogućnosti koja koriste operativne sustave Windows, Linux i MacOS te na ugradbenim uređajima.

Velika prednost biblioteke je u tome što izgrađena kao platforma koju se vrlo lako može nadograditi i koristiti čak i u komercijalnim projektima. Osim za komercijalnu primjenu, biblioteka služi kao i vrlo dobar uvid u to što je zapravo komunikacija u IEC 61850 standardu i kako funkcionira te je vrlo dobar temelj za vlastitu implementaciju IEC 61850 komunikacije pogotovo zahvaljujući brojnim primjerima primjene uključenim u biblioteku. Navedeni primjeri su vrlo jednostavna prezentacija određenog protokola ili funkcionalnosti.

Zadnje verzija biblioteke je verzija 1.5.1 iz ožujka 2022. godine. Iako je biblioteka napisana u programskom jeziku C, biblioteka se može koristiti i u programskom jeziku C# budući da sadrži .NET omotač [18]. Kada se biblioteka preuzme sa službenog GitHub repozitorija [19] biblioteke, sljedeći korak je izgradnja biblioteke koja je napravljena da bude vrlo jednostavna te pristupačna korisnicima. Biblioteka libIEC61850 se može brzo i jednostavno izgraditi pomoću alata *cmake* i *make*. U slučaju korištenje alata *make* dovoljno je samo pozivanje istoimene naredbe u željenom direktoriju ili u glavnom direktoriju kako bi se izgradila cijela biblioteka. Detaljnije upute i mogućnosti oko izgradnje dane su u GitHub repozitoriju biblioteke.

Kao što je već ranije spomenuto, biblioteka libIEC61850 pruža implementacija komunikacijskih mehanizama u unutar sustava koji implementira IEC 61850 standard, a neke od konkretnih funkcionalnosti i značajki implementiranih u biblioteci su:

- MMS klijent-poslužitelj
- MMS datotečne usluge kao što su dohvaćanje, postavljanje, brisanje te pregledavanje
- SV izdavač-pretplatnik
- GOOSE izdavač-pretplatnik
- Podrška za TLS
- Usluga bilježenja implementirana koristeći SQLite bazu podataka
- Podrška za praćenje usluga
- C i C#/.NET API
- Izvješća
- Statička implementacija IED modela pomoću SCL datoteka
- Dinamičko kreiranje IED modela korištenjem konfiguracijskih datoteka
- Alat za pretvorbu SCL datoteka u IED modele
- Alat za pretvorbu SCL datoteka u konfiguracijske datoteke za poslužitelje
- Usluge za skupove podataka kao što su stvaranje i brisanje
- Podrška za upravljačke modele
- Podrška za operativne sustave Windows, Linux i MacOS
- Primjena u ugradbenim uređajima slabe računalne moći
- *Hardware abstraction layer* (HAL)
- Usluga pridruživanja uz autentifikaciju [20], [18]

3.2. Demonstracija pokretanja GOOSE primjera

Kao što je već ranije spomenuto, unutar biblioteke libIEC 61850 nalaze se brojni jednostavni primjeri koji prikazuju osnovne funkcionalnosti. Među tim primjerima su i GOOSE izdavač (*publisher*) te GOOSE pretplatnik (*subscriber*) ondnosno GOOSE osmatrač (*observer*) koji mu je identičan po svemu, ali ima i dodatan ispis za traženje grešaka u kodu. Ovi primjeri demonstriraju GOOSE protokol koji je temeljenu na izdavač-pretplatnik obliku komunikacije.

Da bi se primjeri mogli pokrenuti prvo je naravno potrebno u naredbenom retku naredbom `make examples` u glavnom direktoriju biblioteke izgraditi primjere. Sljedeći korak je u naredbenom retku napraviti virtualna ethernet sučelja te ih potom povezati i pokrenuti sljedećim naredbama:

```
sudo ip link add veth0 type veth peer name veth1
sudo ip link set veth0 up
sudo ip link set veth1 up
sudo ip link
```

Nakon što se izvrši posljednja navedena naredba, na ekranu se pojavi sljedeće dvije linije ispisa koji prikazuje listu mrežnih sučelja u sustavu s njihovim rednim brojem, stanjem, adresom i ostalim sličnim informacijama. Na njemu je vidljivo da su uspješno kreirana dva nova virtualna ethernet sučelja.

```
3: veth1@veth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
noqueue state UP mode DEFAULT group default qlen 1000
    link/ether 32:2d:fb:4e:16:fd brd ff:ff:ff:ff:ff:ff
4: veth0@veth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
noqueue state UP mode DEFAULT group default qlen 1000
    link/ether 0e:0a:7e:70:59:c0 brd ff:ff:ff:ff:ff:ff
```

Nakon toga, potrebno je podesiti da izdavač šalje poruke na adresu od *veth1* koja se dobije izvođenjem naredbe koja rezultira ispisom kao prethodni navedeni. Spomenuto podešavanje se radi direktno u kodu primjera *goose_publisher_example.c* i to na način da se odgovarajući elementi liste *dstAddress* redom postavaljaju na heksadecimalnu vrijednost iz MAC adrese i to na način da nulti element liste predstavlja prve dva znaka iz MAC adrese gledajući slijeva. Budući da su prva

dva znaka adrese sučelja *veth1* jednaka 32, potrebno je kao nulti element liste postaviti navedenu vrijednost. Sljedeća linija koda demonstrira navedeno:

```
gooseCommParameters.dstAddress[0] = 0x32;
```

Zatim preostaje pokrenuti primjer *goose_observer.c*, ali na način da sluša na sučelju *veth1*. To se postiže naredbom `sudo ./goose_observer veth1` što rezultira sljedećim ispisom `Set interface id: veth1.`

Naravno, potrebno je pokrenuti i primjer *goose_publisher_example.c*, ali na način da šalje na sučelje *veth0*. To se postiže naredbom `sudo ./goose_publisher_example veth0` što rezultira sljedećim ispisom `Using interface veth0.`

Nakon pokretanja obje naredbe u zasebnim naredbenim retcima, u naredbenom retku u kojem je pokrenut primjer *goose_observer.c* se pojavi prikazani ispis (Ispis 3.1). Na ispisu (Ispis 3.1) iz naredbenog retka se vidi primjer GOOSE pouke implantirane u biblioteci libIEC 61850 koju osmatrač primi od izdavača. Kao što je vidljivo na ispisu (Ispis 3.1), poruka je zapravo *GOOSE event* koja u sebi nosi informacije poput izvorišne i odredišne MAC adrese, TTL-a, vremenske oznake te izjave o valjanosti poruke.

```
GOOSE event:
  vlanTag: NOT found
  appId: 1000
  srcMac: 0E:0A:7E:70:59:C0
  dstMac: 32:2D:FB:4E:16:FD
  goId: simpleIOGenericIO/LLN0$GO$gcbAnalogValues
  goCbRef: simpleIOGenericIO/LLN0$GO$gcbAnalogValues
  dataSet: simpleIOGenericIO/LLN0$AnalogValues
  confRev: 1
  ndsCom: false
  simul: false
  stNum: 1 sqNum: 0
  timeToLive: 500
  timestamp: 1685990164.497
  message is valid
  AllData: {1234,19840101000000.000Z,5678}.
```

Ispis 3.1 Struktura GOOSE eventa

4. Sigurnost IEC 61850 standarda

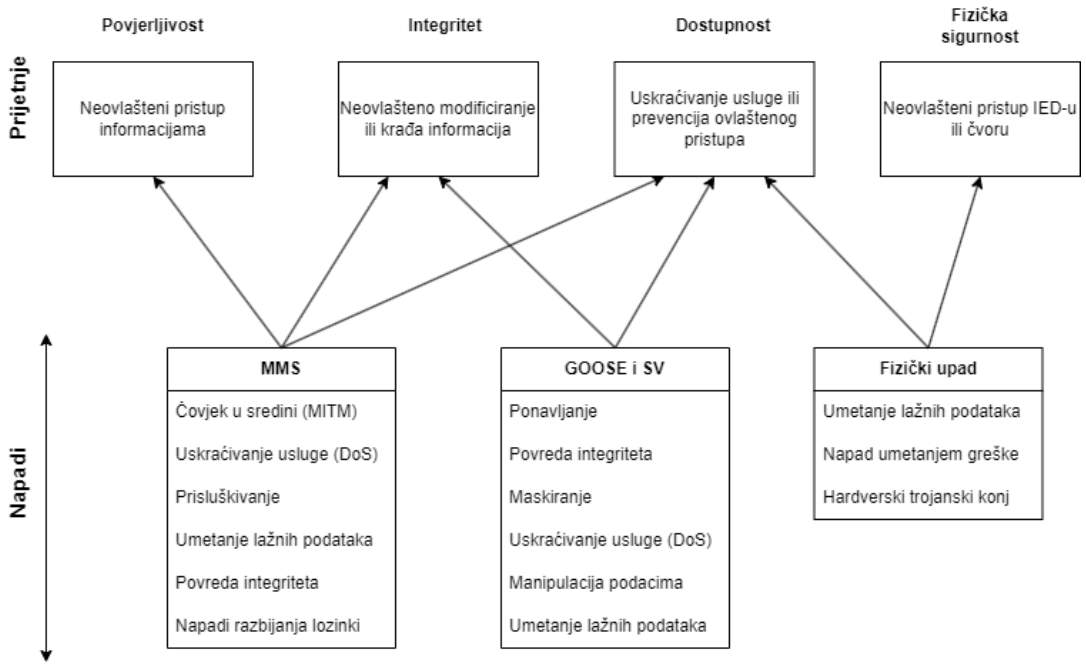
Prije nastanka standarda IEC 61850, sigurnosti unutar elektroenergetskog sustava se nije pridavalo previše pažnje jer se smatralo da je komunikacija unutar sustava jako specijalizirana i drugačija, odnosno glavno načelo sigurnosti je bilo sigurnost temeljena na prikrivanju (*security by obscurity*) [7]. Navedeno načelo se naravno nije moglo još dugo primjenjivati dolaskom standarda IEC 61850 iz razloga što su svi detalji komunikacije kao što su protokoli i strukture podataka javno poznati i svatko ima uvid u način funkcioniranja sustava koji implementira navedeni standard.

4.1. Prijetnje i napadi na IEC 61850 sustave

Budući da se komunikacija u elektroenergetskim sustavima standardizirala pomoću IEC 61850, samim time su se sustavi otvorili prema napadačima kojima su vrlo zanimljivi iz razloga što se probojem u elektroenergetski sustav mogu napraviti ogromne štete poput novčane, reputacijske, ali i ljudskih života.

Analizom IEC 61850 standarda uočeni su njegovi potencijalni sigurnosni problemi te mogućnosti napada. Ciljevi napada na trafostanice koje implementiraju IEC 61850 standard su pristup povjerljivim informacijama te onemogućavanje i ometanje rada trafostanice [7]. Na slici (Slika 4.1) se nalazi prikaz sigurnosnih prijetnji s njihovim pripadajućim napadima. Iznad svake prijetnje se nalazi odgovarajući sigurnosni zahtjev koji je neispunjen u slučaju da se navedena prijetnja uspješno iskoristi.

Napadi su sortirani prema vrsti napada koji mogu biti napadi na protokole kao što su MMS, GOOSE i SV te fizički upad u sustav. Strelice na slici (Slika 4.1) označavaju koji napad (na koji protokol) vodi do koje prijetnje. Primjerice napad uskraćivanjem usluge MMS protokola vodi do prevencije ovlaštenog pristupa što posljedično narušava dostupnost kao sigurnosni zahtjev. Na slici (Slika 4.1) je vidljivo da se korištenjem istih vrste napada mogu ostvariti različite prijetnje [7].



Slika 4.1 Prikaz sigurnosnih prijetnji s pripadajućim napadima (precrtano iz [7])

Napadi na protokole GOOSE, MMS i SV mogu biti različiti, a neki od njih su napad čovjeka u sredini koji bi značio preusmjeravanje prometa od izvora do odredišta preko napadača, prisluškivanje komunikacije unutar sustava, povreda integriteta koja bi značila izmjenu poslanih podataka ili naredbi preko određenog protokola te ponavljanje određene poruke primjerice GOOSE protokola kako bi se iskoristila određena slabost samog protokola.

Osim napada na same protokole, mogući su i napadi koji koriste fizički upad u sustav. Neovlaštenim fizičkim pristupom sustavu se može prouzročiti značajna šteta. Primjerice napadači mogu umetnuti greške i lažne podatke ili i isključiti te namjerno oštetiti dio opreme kao što su IED-ovi te time prouzročiti značajnu novčanu štetu te mogu dovesti do uskraćivanja usluge koje je više izraženo u elektroenergetskim sustavima u kojima su posljedice jednog takvog napada puno veće nego primjerice u nekom drugom sustavu koji nije toliko kritičan za svoju okolinu [21]. Razlog tome je što takvim napadom moguće potpuno obustaviti opskrbu električnom energijom te zaustaviti mnoge procese ovisne o njoj što znači da je narušena dostupnost kao sigurnosni zahtjev.

4.2. IEC 62351 standard

U prvom izdanju standarda IEC 61850 izostajale su sigurnosne mjere za zaštitu komunikacije jer je za to bio predviđen zaseban standard IEC 62351 koji bi pokrivao sigurnost protokola u standardu IEC 61850, ali i u drugim standardima kao što su IEC 60870, IEC 61970 i IEC 61968. Između ostaloga, IEC 62351 standard definira ranjivosti komunikacije u standardu IEC 61850 te mjere za ublažavanje i sprječavanje istih.

Sigurnosne prijetnje koje identificira standard IEC 62351 se dijele na namjerne i nenamjerne. Pod pojmom namjernih prijetnji se ubrajaju hakerski napadi, industrijska špijunaža koju provodi konkurencija s namjerom neovlaštenog prikupljanja informacija ili krađom tuđe ideje u cilju iskorištavanja za vlastitu novčanu dobit, krađa identiteta osobe koja je ovlaštena za pristup određenim resursima i informacijama, zaposlenici tvrtke koji zloupotrebljavaju svoju poziciju i ovlasti te napredne ustrajne prijetnje (*Advanced persistent threats* - APT). S druge strane, nenamjerne prijetnje mogu biti korištenje slabih lozinki, odgađanje ažuriranja softvera na računalu, prirodne katastrofe te krađa uređaja [22].

Da bi se zadovoljili sigurnosni zahtjevi odnosno osigurala sigurnost sustava, standard IEC 62351 propisuje svoje prijedloge za zaštitu komunikacije. Za GOOSE i SV protokole odnosno njihove poruke koje se prenose, IEC 62351 propisuje integritet i autentičnost kao dva bitna sigurnosna zahtjeva.

Kao očigledno rješenje za ove zahtjeve se javlja upotreba algoritama šifriranja, ali nažalost oni u ovoj specifičnoj primjeni nisu pravo i potpuno rješenje zbog ograničenja u sustavima koji implementiraju IEC 61850 standard. Problem primjene algoritama šifriranja je to da su IED-ovi uređaji slabe računalne moći te zbog toga ne mogu u zadanim vremenskim ograničenjima obaviti operacije šifriranja i dešifriranja koje su za njih vrlo zahtjevne. Spomenuto vremensko ograničenje je to da je maksimalno kašnjenje s kraja na kraj GOOSE poruke jednako 3 ms [7].

Budući da algoritmi šifriranja nisu odgovarajuće rješenje, IEC 62351 standard predlaže upotrebu digitalnih potpisa koristeći funkciju sažimanje SHA256 i algoritam javnog ključa RSA odnosno još preciznije algoritam RSASA-PSS.

Iako se navedeni prijedlog nalazi u standardu, eksperimentalno je dokazano da čak ni on ne zadovoljava vremensko ograničenje od 3 ms pa je iz tog razloga predloženo i rješenje koje umjesto digitalnih potpisa koristi MAC algoritam koji zaista zadovoljava spomenuto ograničenje [7]. Nedostatci upotrebe MAC algoritma su to što njime nije zadovoljena povjerljivost koja je u nekim slučajevima upotrebe bitna te se u tom slučaju moraju koristiti i algoritmi šifriranja uz opreznost oko vremenskog ograničenja te unaprijed raspodijeljeni ključevi [7].

Kao bitne sigurnosne zahtjeve za MMS poruke, standard IEC 62351 definira integritet, povjerljivost i autentičnost. MMS protokol odnosno poruke unutar njega su osigurane u IEC 62351 standardu koristeći TLS protokol koji osigurava navedene zahtjeve [21]. Proces razmjene poruka putem protokola MMS se sastoji od početnog rukovanja za koji je potrebno osigurati integritet i autentičnost te prijenosa podataka za koji je potrebno osigurati tajnost korištenjem šifriranja podataka [7].

5. Implementacija protokola raspodjele ključeva

Kao što je već spomenuto u prošlom poglavlju, određeni kriptografski mehanizmi zahtijevaju unaprijed raspodijeljene ključeve ili bar materijale za ključeve. To je vrlo kompleksan problem te se često svodi na to da su neki ključevi raspodijeljeni ne-kriptografskim metodama te se oni koriste dalje za implementaciju samog mehanizma raspodjele ključeva. U narednim potpoglavljima opisati će se protokol te njegova implementacija u sklopu ovog rada.

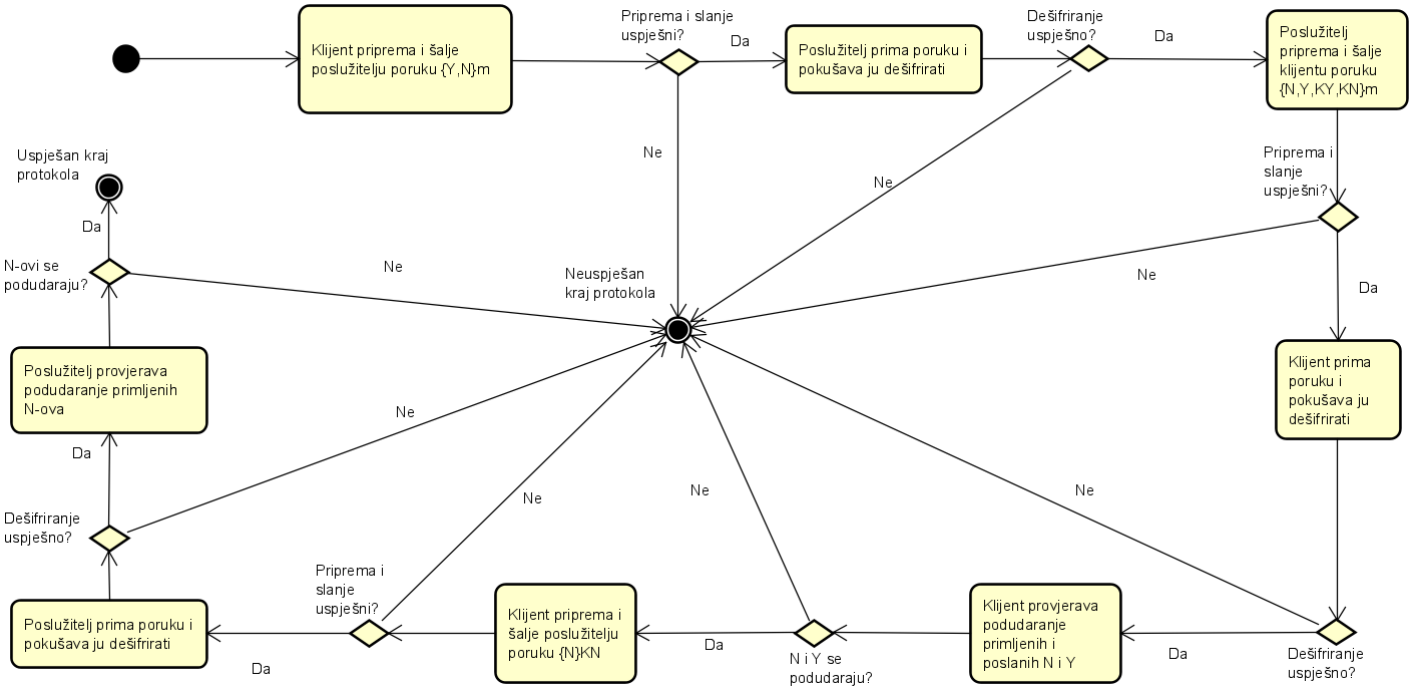
5.1. Protokol raspodjele ključeva

Kao protokol koji je implementiran u sklopu ovog rada odabran je protokol predložen u [5]. Njegovi autori zaključili su kako je od ispitanih mogućnosti koje uključuju raspodjelu simetričnih ključeva i raspodjelu ključeva za mehanizam javnog ključa, najbolje rješenje koje uključuje raspodjelu simetričnih ključeva jer je ono najjeftiniji mehanizam s obzirom na sve ostale faktore. Upravo je protokol raspodjele simetričnih ključeva odabran i u ovome radu kao protokol koji će biti implementiran.

Početna pretpostavka protokola je da novi IED koji se priključuje u mrežu trafostanice i kojem se trebaju raspodijeliti odgovarajući ključevi ima ključ za paljenje m (*ignition key*) koji je 128-bitni AES ključ koji je prilikom proizvodnje IED-a ispisan na njegovom pakiranju [5]. Ključ za paljenje pojedinog IED-a se unosi u kontroler trafostanice kako bi on mogao komunicirati s IED-om te kako bi se mogao izvršiti protokol raspodjele ključeva. Točnije, ključ za paljenje služi za šifriranje/dešifriranje prve dvije poruke u protokolu raspodjele ključeva. Budući da se koristi simetrični algoritam šifriranja, potrebno je ključ za paljenje unaprijed raspodijeliti kako je i opisano. U opisu implementacije protokola će se umjesto kontrolera trafostanice koristiti izraz poslužitelj, a umjesto IED-a klijent. Nakon što se IED spoji na mrežu trafostanice, tada je sve spremno za početak protokola raspodjele ključeva.

Na slici (Slika 5.1) se nalazi UML dijagram koji prikazuje tijek protokola koji je implementiran. Kao što se može vidjeti na slici (Slika 5.1), prvi korak je da novi IED koji upravo spojen na mrežu trafostanice šalje zahtjev za pridruživanjem poslužitelju. Zahtjev za produživanjem je poruka koja

se sastoji od jedinstvene oznake (identifikatora) IED-a Y te nasumično generiranog broja N koji su šifrirani ključem za paljenje koji je IED-u unaprijed dodijeljen.



Slika 5.1 UML dijagram aktivnosti protokola raspodjele kriptografskih ključeva

Ako se šifriranje i slanje navedene poruke ne izvrše uspješno, onda je to kraj protokola kao što je vidljivo na slici (Slika 5.1), a ako se izvrše uspješno onda se prelazi na idući korak. Idući korak protokola je da poslužitelj prima poslanu poruku te ju zatim pokušava dešifrirati ključem za paljenje koji je inženjer upisao u njega tijekom postavljanja IED-a. Ključ za paljenje određenog IED-a je naravno isti na obje strane komunikacije kako bi se uspješno provodili postupci šifriranja i dešifriranja budući da se koriste simetrični algoritmi. Ako poslužitelj ne uspije dešifrirati poruku koju primi, znači da je došlo do određene pogreške te se time staje s protokolom uz neuspješnu raspodjelu ključeva.

U slučaju da se poruka uspješno dešifrira, slijedi idući korak, a to je da poslužitelj priprema odgovor klijentu na način da primljenu i dešifriranu oznaku IED-a Y te primljeni nasumični broj N šifrira zajedno s ključem uređaja KY te mrežnim ključem KN. Ključ uređaja KY je različit za svaki IED te se svakom IED-u mora raspodijeliti ključ namijenjen samo njemu. Mrežni ključ KN je isti za sve uređaje u mreži te se prilikom njegove izmjene mora svim uređajima u mreži poslati

novi mrežni ključ. Šifriranje odgovora klijentu se obavlja korištenjem ključa za paljenje korištenog u prethodnim koracima protokola. Šifrirani sadržaj se šalju klijentu kao odgovor. Ako slanje bude neuspješno ili se dogodi greška prilikom šifriranja, onda je to kraj protokola.

U slučaju da navedeno slanje bude uspješno, klijent prima poruku od poslužitelja i pokušava ju dešifrirati ključem za paljenje kao što je vidljivo i na slici (Slika 5.1). Ako klijent ne uspije dešifrirati primljenu poruku, to je opet prijevremeni neuspješni kraj protokola. A ako uspije dešifrirati primljenu poruku, onda klijent uspoređuje primljenu oznaku IED-a Y te primljeni nasumično generirani broj N s onima koje je on na početku poslao poslužitelju u zahtjevu za pridruživanje. Ako se vrijednosti ne podudaraju, onda je došlo do greške te se protokol završava.

U slučaju da se vrijednosti podudaraju, onda klijent priprema posljednju poruku poslužitelju. U toj poruci ponovno šalje isti nasumični broj N koji se i do sada koristio u komunikaciji šifriran s primljenim mrežnim ključem KN. Ako se poruka uspješno pošalje, nastavlja se s protokolom. Ovo je ujedno kraj protokola s klijentske strane.

Sada poslužitelj prima poruku od klijenta i pokušava ju dešifrirati s mrežnim ključem KN. Ako dešifriranje ne uspije, to je ponovno neuspješan kraj protokola. U slučaju da poslužitelj uspije uspješno dešifrirati primljenu poruku, tada provjerava podudara li se primljeni nasumični broj N s onim kojeg je već ranije primio. Ako se nasumični brojevi podudaraju, to je ujedno i uspješan kraj protokola u kojem su klijentu raspodijeljeni mrežni ključ KN te jedinstveni ključ uređaja KY. U suprotnom slučaju to je neuspješan kraj protokola. U navedenom postupku se nasumični broj N i oznaka IED-a Y koriste kao kontrolne vrijednosti za provjeru da li je dosadašnji tok protokola bio uspješan te da se provjeri sudionik komunikacije s druge strane, pogotovo u slučaju da poslužitelj komunicira s više IED-a u isto vrijeme.

5.2. Implementacija protokola

Budući da je implementacija opisanog protokola zamišljena kao proširenje biblioteke libIEC 61850, sam programski kod je pisan u programskom jeziku C te uključen u biblioteku na način da su kreirani novi primjeri u mapi *examples*. Programski kod kao i sve ostale datoteke su podijeljeni

u dva odvojena direktorija unutar direktorija *examples* biblioteke libIEC 61850. Ta dva direktorija su *tcp_server* koji predstavlja poslužitelja i *tcp_client* koji predstavlja klijenta.

Direktorij *tcp_client* sastoji se od datoteke *tcp_client.c* u kojoj se nalazi sav programski kod vezan uz klijentsku stranu, tekstualne datoteke *ignition_key.txt* u kojoj se nalazi ključ za paljenje koji se čita iz datoteke tijekom izvođenja protokola te datoteke *Makefile* koja služi za izgradnju primjera.

Direktorij *tcp_server* sastoji se od datoteke *tcp_server.c* u kojoj se nalazi sav programski kod vezan uz poslužiteljsku stranu, tekstualnih datoteka *ignition_key.txt*, *network_key.txt*, *device_key.txt* u kojima se redom nalaze ključ za paljenje, mrežni ključ i ključ uređaja koji se čitaju iz datoteke tijekom izvođenja protokola te datoteke *Makefile* koja služi za izgradnju primjera.

5.2.1. Implementacija komunikacije koristeći TCP vezu

U implementaciji u sklopu ovog rada klijent i poslužitelj komuniciraju putem TCP veze, stoga je naravno bilo potrebno implementirati takvu komunikaciju koristeći TCP spojne točke. Na primjeru koda (Kod 5.1) će biti objašnjeno koje su ključne naredbe i stavke za otvaranje TCP veze u ovom primjeru sa poslužiteljske strane. Kod za cjelokupnu implementacija komunikacije preko TCP veze je preuzet iz [23], [24].

Prva stvar koju je potrebno napraviti je kreiranje spojne točke (*socket*) koja omogućuje komunikaciju između poslužitelja i klijenta preko TCP veze. Spojna točke se stvara putem naredbe `socket` s odgovarajućim argumentima što je prikazano u prvoj liniji koda (Kod 5.1). Osim kreiranja spojne točke, bitno je i podesiti neke njene karakteristike kao što je primjerice to da se omogući ponovna upotreba lokalnih adresa kako ne bi došlo do potencijalnih grešaka koje nastaju kada je neka adresa zauzeta nekim drugim procesom ili ostala zauzeta od prošlog pokretanja programa. Navedena mogućnost se ostvaruje naredbom `setsockopt` kako je i prikazano u drugoj liniji koda.

U središnjem odlomku koda (Kod 5.1) je prikazano postavljenje opcija obitelji adresa, same adrese te porta. Još je potrebno povezati spojnu točku sa zadanom IP adresom što se radi narebom `bind`,

zadati poslužitelju da sluša na zadanoj adresi i portu naredbom `listen` te zadati da prihvaća podatke od klijenta naredbom `accept`.

```
int socketfiledesc = socket(AF_INET, SOCK_STREAM, 0);
setsockopt(socketfiledesc, SOL_SOCKET, SO_REUSEADDR , &rse,
sizeof(rse))

socketAddressServer.sin_family = AF_INET;
socketAddressServer.sin_addr.s_addr = htonl(INADDR_ANY);
socketAddressServer.sin_port = htons(PORT);

int bnd = bind(socketfiledesc, (SA *)&socketAddressServer,
sizeof(socketAddressServer));
int lstn = listen(socketfiledesc, 7);
int socket_commu = accept(socketfiledesc, (SA
*)&socketAddressClient, &size);
```

Kod 5.1 Programsko postavljanje TCP veze

Slanje preko TCP veze je ostvareno naredbom `send` kao što je prikazano u idućoj liniji koda:

`send(socketfiledesc, ciphertext, sizeof(ciphertext), 0).` Prvi argument je opisnik datoteke spojne točke preko koje se šalju podaci. Preostali argumenti su redom sami podaci koji se šalju, njihova veličina te postavljene zastavice koje definiraju moguće opcije oko slanja.

Primanje poruke preko TCP veze je ostvareno naredbom `recv` kao što je prikazano u idućoj liniji koda: `recv(socketfiledesc, bfr, SIZE, 0) == -1).` Argumenti su slični kao i kod slanja te se sastoje od opisnika datoteke spojne toče, varijable u koju se spremaju primljeni podaci, maksimalna duljina primljenih podataka te postavljene zastavice.

Prilikom uzastopnog korištenja naredbi `send` i `recv` potrebno je bilo između njihovih poziva u kodu dodavati pozive funkcije `sleep` kako ne bi dolazilo do greške uslijed uzastopnog slanja/primanja poruka koristeći spojnu točku.

Također je prilikom svakog slanja ili primanja dodana i provjera vraćaju li odgovarajuće funkcije vrijednost koja označava uspješnu operaciju ili ne. Primjerice ako funkcija `recv` vrati vrijednost -

1, to znači da se dogodila greška prilikom primanja poruke te je potrebno grešku ispisati i prekinuti izvođenje programa. Na primjeru primanja poruke je demonstrirana navedena provjera u sljedećih nekoliko linija koda:

```
if (recv(socket_commu, ciphertextlbl_dec, SIZE, 0) == -1) {
      perror("[-]Error in receiving file.");
      exit(1);
}
```

5.2.2. Implementacija šifriranja/dešifriranja

Budući da se poruke prije slanja TCP vezom moraju šifrirati i da je to glavna funkcionalnost navedenog protokola, bilo je potrebno implementirati odgovarajući algoritam. U [5] je navedeno da je ključ za paljenje `m` zapravo 128-bitni AES ključ stoga je implementiran upravo AES algoritam koji koristi 128-bitni ključ. Način pripreme za šifriranje i samo pozivanje funkcije za šifriranje je prikazano u kodu (Kod 5.2).

```
unsigned char *key = (unsigned char *)prekey;
unsigned char *iv = (unsigned char *) init_vec;
unsigned char *plaintext = (unsigned char *)challenge;
unsigned char ciphertext[SIZE];
int ciphertext_len = encrypt(plaintext, strlen ((char
*)plaintext), key, iv, ciphertext);
```

Kod 5.2 Priprema za poziv i poziv funkcije encrypt

Dakle, potrebno je pripremiti ključ za šifriranje koji se čita iz datoteke, inicijalizacijski vektor koji se nasumično generira prije samog korištenja te sami sadržaj koji će se šifrirati. Sve navedene varijable je potrebno definirati kao `unsigned char` jer je to format koji se koristi u programskom jeziku C prilikom korištenja šifriranja. Nakon navedene pripreme potrebno je samo još pozvati funkciju `encrypt` s argumentima koji odgovaraju prethodno spomenutim varijablama. U funkciji `encrypt` se nalazi programski kod samog šifriranja koji je prikazan kodom (Kod 5.3) koji je preuzet iz [25].

```
int encrypt(unsigned char *plaintext, int plaintext_len, unsigned
char *key, unsigned char *iv, unsigned char *ciphertext){
```

```
    EVP_CIPHER_CTX *ctx;
    int len;
    int ciphertext_len;
    if(!(ctx = EVP_CIPHER_CTX_new()))
        handleErrors();
    if(1 != EVP_EncryptInit_ex(ctx, EVP_aes_128_cbc(), NULL, key,
iv))
        handleErrors();
    if(1 != EVP_EncryptUpdate(ctx, ciphertext, &len, plaintext,
plaintext_len))
        handleErrors();
    ciphertext_len = len;
    if(1 != EVP_EncryptFinal_ex(ctx, ciphertext + len, &len))
        handleErrors();
    ciphertext_len += len;
    EVP_CIPHER_CTX_free(ctx);
    return ciphertext_len;
}
```

Kod 5.3 Program za šifriranje koristeći AES algoritam

Kod (Kod 5.3) omogućava šifriranje algoritmom AES uz 128-bitni ključ te način rada CBC. Algoritam šifriranja je ostvaren korištenjem EVP funkcionalnosti biblioteke OpenSSL koji predstavlja sučelje za kriptografske funkcije. Funkcija `encrypt` se sastoji od nekoliko generalnih koraka. Prvi korak je stvaranje i inicijaliziranje konteksta koristeći `EVP_CIPHER_CTX_new`. Zatim se inicijalizira samo šifriranje koristeći `EVP_EncryptInit_ex` uz odgovarajuće argumente kao što su kontekst, točan algoritam, ključ i inicijalizacijski vektor. Nakon toga slijedi poziv funkcije `EVP_EncryptUpdate` koja pruža funkcionalnost samog šifriranja te u varijablu `ciphertext` sprema šifrirani tekst. Zadnji korak je finaliziranje šifriranja pozivom funkcije `EVP_EncryptFinal_ex`.

U kodu (Kod 5.3) funkcije `encrypt` je vidljivo pozivanje funkcije `handleErrors` čija namjena je obrada grešaka odnosno ispis o konkretnoj grešci koja se dogodila u postupku šifriranja ili dešifriranja. Funkcija je prikazana kodom (Kod 5.4) u kojem je vidljivo da se ispis o konkretnoj grešci ostvaruje uz pomoć funkcije `ERR_print_errors_fp` koja je također dio OpenSSL biblioteke.

```
void handleErrors(void){
      printf("Something went wrong!\n");
      ERR_print_errors_fp(stderr);
      abort();
}
```

Kod 5.4 Funkcija za obradu grešaka tijekom šifriranja/dešifriranja

Dešifriranje je implementirano na gotovo identičan način te je kod za dešifriranje i obradu grešaka također preuzet iz [25] . Za razliku od šifriranje, dešifriranje ima dodatak funkcije `remove_padd` čija namjena je da se nakon dešifriranja ukloni dopuna koja se dodaje prilikom šifriranja kako bi se tekst nadopunio do zadane veličine.

Navedena funkcija je prikazana kodom (Kod 5.5). Njena funkcionalnost je ostvarena tako da se *for* petljom prolazi po svim znakovima dešifriranog niza te se u trenutku o kojem *for* petlja dođe do znaka koji nije alfanumerički program prekida, a prije toga se na to mjesto postavi znak za kraj znakovnog niza. Time je sva dopuna koja se nalazi iza tog znaka zanemarena i uklonjena. Sada je taj isti znakovni niz spreman za daljnje korištenje u programu jer je ekvivalentan onome koji je poslan te šifriran.

```
unsigned char* remove_padd(unsigned char input[]){
      int sz = strlen((char *)input);
      for (int i = 0; i < sz; i++) {
        if ( isalnum(input[i]) ){
            continue;
        }
            else{
                  input[i] = '\0';
                  break;
            }
    }
      return input;
}
```

Kod 5.5 Funkcija za uklanjanje dopune nakon dešifriranja

Prilikom slanja poruka putem TCP veze, u porukama se osim sadržaja koji je već prethodno naveden prilikom objašnjavanja protokola šalju i inicijalizacijski vektor koji je korišten prilikom šifriranja kako bi druga strana mogla dešifrirati tu istu poruku te cjelobrojne vrijednosti koje predstavljaju duljinu šifriranog teksta koju je vratila funkcija `encrypt`. Razlog slanja duljina šifriranog teksta je u tome što tu vrijednost prima funkcija `decrpyt` prilikom dešifriranja te je to pouzdanije nego računanje duljine na primateljevoj strani budući da je riječ o varijablama tipa *unsigned char* koje predstavljaju bajtove, a ne čisti tekst.

Sve varijable korištene u postupku šifriranja/dešifriranja su tipa *unsigned char* jer je on jedini prikladan za ispravno prikazivanje bajtova u programskom jeziku C. U slučaju korištenja varijabli tipa *char* bi došlo do grešaka jer se neke bajtovne vrijednosti uopće ne bi mogle ispravno prikazati budući da tip *char* može prikazati raspon vrijednosti od -128 od 127 za razliku od tipa unsigned char koji prikazuje vrijednosti iz raspona od 0 do 255.

Podaci poput ključeva za paljenje te mrežnih ključeva i jedinstvenih ključeva uređaja se prilikom korištenja u šifriranju/dešifriranju, slanja ili primanja čitaju ili zapisuju u tekstualne datoteke unutar svog direktorija. Svi navedeni ključevi su svi 128-bitni ključevi. Inicijalizacijski vektori su također 128-bitne vrijednosti koje se generiraju prije samog korištenja na način prikazan kodom (Kod 5.6) koji će biti ukratko objašnjen.

Da bi se generirao inicijalizacijski vektor, prvo je potrebno osigurati generiranje pseudo-slučajnih brojeva. To je u kodu (Kod 5.6) napravljeno na način da se funkcijom `srand` postavlja sjeme za generator pseudo-slučajnih brojeva korištenjem varijable `t1` u kojoj se dohvaća trenutno vrijeme funkcijom `time`.

Budući da je inicijalizacijski vektor veličine 128 bitova odnosno 16 bajtova, potrebno je koristiti niz znakova veličine 17 jer je svaki znak u programskom jeziku C jednak jednom bajtu uz dodatak oznake za kraj niza koja također zauzima jedan bajt. Generiranje samih znamenki inicijalizacijskog vektora se u kodu (Kod 5.6) obavlja u `for` petlji pozivom funkcije `rand` za koju je postavljeno sjeme te dodavanjem generirane znamenke na kraj niza. Za kraj je još samo potrebno na kraj niza dodati oznaku kraja niza za koju je rezerviran posljednji bajt.

```
time_t t1;
srand ( (unsigned) time (&t1));
```

```
char init_vec[17];

for (int i = 0; i < 16; i++) {
      init_vec[i] = '0' + rand() % 10;
}
init_vec[16] = '\0';
```

Kod 5.6 Program za generiranje inicijalizacijskog vektora

5.3. Izgradnja primjera raspodjele ključeva

Kao što je već ranije spomenuto, kompletan opisani kod i sve popratne datoteke su uključene u biblioteku libIEC61850. Da bi se mogao pokrenuti primjer raspodjele ključeva potrebno je prvo izgraditi navedene primjere. To je moguće na dva načina. Prvi način je da se u glavnom direktoriju biblioteke pokrene naredbeni redak i iz njega pokrene naredba `make examples` koja će izgraditi sve već postojeće primjere koji su dio biblioteke, ali također i primjer raspodjele ključeva. Drugi način je da se uđe u direktorij konkretnog primjera te se zatim iz njega otvori naredbeni redak i pokrene naredba `make` koja će izgraditi konkretan primjer.

Da bi se to ostvarilo bilo je potrebno modificirati postojeće datoteke *Makefile* u kojima se definira izgrađivanje primjera kako bi se dalo uputstvo za izgradnju i novo dodanih primjera u biblioteci.

Također je bilo potrebno i napisati dvije datoteke *Makefile* za primjere *tcp_client.c* i *tcp_server.c*. Obje datoteke su gotovo pa identične, stoga će biti prikazan primjer *Makefile* datoteke za primjer *tcp_server.c*. U navedenoj datoteci se definira na koji način je potrebno izgraditi određeni kod. Kao što je vidljivo u kodu (Kod 5.7), koristi se *gcc* prevoditelj, zastavice za prevoditelj *-Wall -g* koje definiraju da su omogućena sva upozorenja te informacije te informacije za traženje grešaka u kodu, te zastavice za povezivač *-lssl -lcrypto* koje označuju korištenje biblioteke OpenSSL. Unutar *Makefile-a* potrebno je i definirati izvor odnosno datoteku koju je potrebno izgraditi te datoteke u kojoj će se spremiti izvršni kod. Osim toga, definirano je i da se prilikom izgradnje primjera provede instalacija biblioteke OpenSSL kako bi se primjeri mogli pokrenuti. U slučaju da je biblioteka već instalirana, samo će se provesti ažuriranje u slučaju da instalirana verzija nije najnovija. Navedeno je u kodu (Kod 5.7) ostvareno na samom dnu u kodu ispod oznake

install_openssl. Jedino ograničenje za pokretanje ovih primjera je da se mora koristiti operacijski sustav Linux zbog određenih korištenih naredbi.

```
CC = gcc
CFLAGS = -Wall -g
LDFLAGS = -lssl -lcrypto

TARGET = tcp_server
SOURCE = tcp_server.c

all: install_openssl $(TARGET)

$(TARGET): $(SOURCE)
	$(CC) $(CFLAGS) -o $(TARGET) $(SOURCE) $(LDFLAGS)

clean:
	rm -f $(TARGET)

install_openssl:
	@echo "Installing OpenSSL..."
	@sudo apt-get install -y libssl-dev
	@echo "OpenSSL installed successfully."
```

Kod 5.7 Prikaz sadržaja Makefile datoteke za primjer tcp_server

5.4. Demonstracija izgradnje i pokretanja primjera

Nakon što je u prošlom potpoglavlju opisana izgradnja implementiranih primjera, sada će ona biti demonstrirana zajedno s pokretanjem primjera. Kao što je već opisano, za izgradnju je samo potrebno u naredbenom retku unutar direktorija primjera pokrenuti naredbu make kojom se pokreću naredbe iz datoteke makefile te se time generira prikazani ispis (Ispis 5.1). Na ispisu (Ispis 5.1) je prikazana izgradnja primjera *tcp_server* u kojem se vidi da se prvo pokreće instalacija OpenSSL biblioteke čija najnovija verzija je već instalirana što je vidljivo kroz cijeli ispis osim u zadnjoj liniji koja označava kompiliranje programa.

```
Installing OpenSSL...
[sudo] password for student:
Reading package lists... Done
Building dependency tree
Reading state information... Done
libssl-dev is already the newest version (1.1.1f-1ubuntu2.19).
0 upgraded, 0 newly installed, 0 to remove and 247 not upgraded.
OpenSSL installed successfully.
gcc -Wall -g -o tcp_server tcp_server.c -lssl -lcrypto
```

Ispis 5.1 Prikaz ispisa generiranog nakon pokretanja naredbe make za primjer tcp_server

Nakon uspješne izgradnje, primjeri *tcp_server* i *tcp_client* spremni su za pokretanje. Primjeri se pokreću standardnim pokretanjem na operacijskom sustavu Linux, a to je u primjeru *tcp_server* naredba `./tcp_server`. Primjere tcp_server i tcp_server je potrebno na navedeni način pokrenuti u odvojenim naredbenim retcima.

Pokretanje primjera *tcp_client* generira prikazani ispis (Ispis 5.2). U ispisu (Ispis 5.2) su vidljivi koraci protokola sa strane klijenta koji su slanje prve poruke poslužitelju koja je zapravo zahtjev za pridruživanjem, primanjem odgovora od poslužitelja, dešifriranje primljenog odgovora, provjera da li se primljene i poslane verzije oznake IED-a te nasumičnog broja podudaraju, slanje odgovora poslužitelju kao potvrdu uspješnosti protokola s klijentske strane te konačna poruka o uspješnom kraju protokola.

```
Sending 1 successfully completed!
Reception completed successfully!!
Decryption of the message from the server successfully completed!
The received label and challenge match the sent versions!
Sending 2 successfully completed!
Protocol successfully completed!!
```

Ispis 5.2 Rezultat pokretanja primjera tcp_client

Pokretanje primjera *tcp_server* generira prikazani ispis (Ispis 5.3). U ispisu (Ispis 5.3) su vidljivi koraci protokola sa strane poslužitelja koji uključuju primanje prve poruke/zahtjeva za

pridruživanje od klijenta, dešifriranje primljene poruke, ispis oznake IED-a koji šalje zahtjev za pridruživanjem, slanje odgovora klijentu na njegov zahtjev, ponovno primanje odgovora od klijenta koji predstavlja potvrdu uspješnosti protokola s klijentske strane, dešifriranje druge primljene poruke, provjeru podudaranja primljenih nasumičnih brojeva iz prve i druge poruke te potvrda o uspješnosti i završetku cjelokupnog protokola.

```
Reception 1 completed successfully!
Decryption of the message from the client successfully completed!
IED label: IED21345!!!
Sending successfully completed!
Reception 2 completed successfully!!
Decryption of the message from the client successfully completed!
The received challenge matches the one received at the beginning!
Protocol successfully completed!!
```

Ispis 5.3 Rezultat pokretanja primjera tcp_server

Krajnji rezultat pokretanja protokola je da se u direktoriju *tcp_client* stvore dvije nove tekstualne datoteke koje su na slici (Slika 5.2) označene crvenom bojom i u kojima se nalaze mrežni ključ te jedinstveni ključ uređaja. Time je poslužitelj na siguran način raspodijelio klijentu potrebne ključeve koje klijent sada može koristiti za primjeri šifriranje GOOSE poruka.

Slika 5.2 Prikaz direktorija tcp_client nakon uspješnog završetka protokola raspodjele ključeva

6. Zaključak

Napadi na elektroenergetske sustave su izrazito opasni iz razloga što se probojem u elektroenergetski sustav mogu napraviti ogromne štete poput novčane, reputacijske, ali i ljudskih života. Primjerice, moguće je zaustaviti opskrbu električnom energijom za čitave gradove. Napadi su dodatno olakšani primjenom standarda IEC 61850 u praksi budući da način funkcioniranja sustava i komunikacije unutar njega više nije nepoznat nego je potpuno javan i svatko se s njime može upoznati, a samim time i potencijalni napadači. Primjeri napada na sustave koji implementiraju standard IEC 61850 su napad uskraćivanjem usluge, napad čovjeka u sredini, umetanje lažnih podataka te grešaka. Smjernice za njihovo ublažavanje i sprječavanje su dane u standardu IEC 62351. Neka od predloženih rješenja uključuju i korištenje algoritama šifriranja/dešifriranja za koje postoji problem na koji način osigurati sigurnu raspodjelu kriptografskih ključeva koji su temelj za šifriranje. U znanstvenim radovima postoje određeni prijedlozi za ostvarenje navedenog od kojih je odabran jedan za ovaj rad.

Protokol za raspodjelu ključeva implementiran u sklopu ovog rada kao nadogradnja biblioteke libIEC61850 dodaje mnogo novih mogućnosti za implementaciju sigurnosnih mehanizama unutar same biblioteke. Samim time je ostvarena ideja ovog rada, a to je da se na temelju prikazanog i implementiranog može dalje nadograđivati i istraživati. Primjerice njegovom upotrebom je moguće raspodijeliti kriptografske ključeve koji će se zatim koristiti za zaštitu odnosno šifriranje poruka protokola od kritične važnosti kao što je GOOSE kojim se šalju upravljačke naredbe. Upravo navedeno je jedan od mogućih smjerova za daljnje istraživanje te implementaciju zaštite komunikacije unutar elektroenergetskog sustava.

7. Literatura

[1] P. Kirvan, »Control system,« TechTarget, [Mrežno]. Dostupno: https://www.techtarget.com/whatis/definition/control-system. [Pokušaj pristupa 26 Svibanj 2023].

[2] Wikipedia, »Control system,« [Mrežno]. Dostupno: https://en.wikipedia.org/wiki/Control_system. [Pokušaj pristupa 26 Svibanj 2023].

[3] Leksikografski zavod Miroslav Krleža, »elektroenergetski sustav,« Hrvatska enciklopedija, mrežno izdanje, [Mrežno]. Dostupno: https://www.enciklopedija.hr/natuknica.aspx?id=17604. [Pokušaj pristupa 28 Svibanj 2023].

[4] S. M. S. Hussain, S. M. Farooq i T. S. Ustun, »Analysis and Implementation of Message Authentication Code (MAC) Algorithms for GOOSE Message Security,« *IEEE Access,* 2019.

[5] S. Fuloria, R. Anderson, F. Alvarez i K. McGrath, »Key management for substations: Symmetric keys, public keys or no keys?,« u *IEEE/PES Power Systems Conference and Exposition*, Phoenix, 2011.

[6] International Electrotechnical Commission, »Who we are,« [Mrežno]. Dostupno: https://www.iec.ch/who-we-are. [Pokušaj pristupa 30 Svibanj 2023].

[7] S. M. S. Hussain, T. S. Ustun i A. Kalam, »A Review of IEC 62351 Security Mechanisms for IEC 61850 Message Exchanges,« *IEEE Transactions on Industrial Informatics,* Rujan 2020.

[8] Eaton Corporation, »Substation automation: fundamentals of substation automation,« [Mrežno]. Dostupno: https://www.eaton.com/us/en-us/products/utility-grid-solutions/grid-automation-system-solutions/fundamentals-of-substation-automation.html. [Pokušaj pristupa 1 Lipanj 2023].

[9] H. Zeynal, M. Eidiani i D. Yazdanpanah, »Intelligent Substation Automation Systems for robust operation of smart grids,« u *IEEE Innovative Smart Grid Technologies - Asia (ISGT Asia)*, Kuala Lumpur, 2014.

[10] I. Wigmore, »human-machine interface (HMI),« TechTarget, [Mrežno]. Dostupno: https://www.techtarget.com/whatis/definition/human-machine-interface-HMI. [Pokušaj pristupa 2 Lipanj 2023].

[11] P. Loshin, »SCADA (supervisory control and data acquisition),« TechTarget, [Mrežno]. Dostupno: https://www.techtarget.com/whatis/definition/SCADA-supervisory-control-and-data-acquisition. [Pokušaj pristupa 2 Lipanj 2023].

[12] F. Vidović, I. Vukić, G. Leci i D. Ožanić, »PROTOKOL IEC 61850 – PRIJEDLOG TEHNIČKOG RJEŠENJA SEKUNDARNOG SUSTAVA DISTRIBUCIJSKE TRANSFORMATORSKE STANICE,« u *HRVATSKI OGRANAK MEĈUNARODNE ELEKTRODISTRIBUCIJSKE KONFERENCIJE*, Šibenik, 2008..

[13] X. Lu, W. Wang i J. Ma, »Authentication and Integrity in the Smart Grid: An Empirical Study in Substation Automation Systems,« *International Journal of Distributed Sensor Networks,* 2012.

[14] »Abstract Communication Service Interface (ACSI),« International Electrotechnical Commission, [Mrežno]. Dostupno: https://std.iec.ch/terms/terms.nsf/3385f156e728849bc1256e8c00278ad2/6cb7f02cfa2274c9c12574ab0036770a?OpenDocument. [Pokušaj pristupa 3 Lipanj 2023].

[15] »IEC 61850 MMS Protocol,« Typhoon HIL, [Mrežno]. Dostupno: https://www.typhoon-hil.com/documentation/typhoon-hil-software-manual/References/iec_61850_mms_protocol.html. [Pokušaj pristupa 3 Lipanj 2023].

[16] »Specific Communication Service Mapping, SCSM,« International Electrotechnical Commission, [Mrežno]. Dostupno: https://std.iec.ch/terms/terms.nsf/3385f156e728849bc1256e8c00278ad2/5b38de600b30bf1ec1256db10027c3a3?OpenDocument. [Pokušaj pristupa 3 Lipanj 2023].

[17] R. B. Scarselli, L. F. Soares i I. M. Moraes, »Evaluating Cryptographic Algorithms in IEC 61850 Networks,« u *10th International Conference on Networks of the Future (NoF)*, Rim, 2019.

[18] MZ Automation, »libIEC61850 / lib60870,« [Mrežno]. Dostupno: https://libiec61850.com/. [Pokušaj pristupa 4 Lipanj 2023].

[19] MZ Automation, »libiec61850,« [Mrežno]. Dostupno: https://github.com/mz-automation/libiec61850. [Pokušaj pristupa 4 Lipanj 2023].

[20] MZ Automation, »API Reference Manual,« [Mrežno]. Dostupno: https://support.mz-automation.de/doc/libiec61850/c/latest/. [Pokušaj pristupa 5 Lipanj 2023].

[21] J. O'Raw, D. M. Laverty i D. J. Morrow, »IEC 61850 substation configuration language as a basis for automated security and SDN configuration,« u *IEEE Power & Energy Society General Meeting*, Chicago, 2017.

[22] S. Sučić, »Sigurnosni komunikacijski protokoli u elektroenergetskom sustavu,« 2009..

[23] Y. Shukla, »TCP Server-Client implementation in C,« GeeksforGeeks, [Mrežno]. Dostupno: https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/. [Pokušaj pristupa 15 Ožujak 2023].

[24] N. K. Tomar, »File Transfer using TCP Socket in C,« Idiot Developer, [Mrežno]. Dostupno: https://idiotdeveloper.com/file-transfer-using-tcp-socket-in-c/. [Pokušaj pristupa 16 Ožujak 2023].

[25] »EVP Symmetric Encryption and Decryption,« OpenSSL wiki, [Mrežno]. Dostupno: https://wiki.openssl.org/index.php/EVP_Symmetric_Encryption_and_Decryption. [Pokušaj pristupa 25 Ožujak 2023].

[26] »IEC 61850 Sampled Values protocol,« Typhoon HIL, [Mrežno]. Dostupno: https://www.typhoon-hil.com/documentation/typhoon-hil-software-manual/References/iec_61850_sampled_values_protocol.html. [Pokušaj pristupa 3 Lipanj 2023].

[27] B. Jang, A. Abubakari i N. Kim, »IEC 61850 SCL Validation Using UML Model in Modern Digital Substation,« *Smart Grid and Renewable Energy,* 2018.

Sažetak

Implementacija mehanizma upravljanja kriptografskim ključevima u programskoj biblioteci libiec61850

Ovaj rad predstavlja nadogradnju biblioteke libIEC61850 na način da se implementira mehanizam raspodjele kriptografskih ključeva koji se unutar biblioteke mogu koristiti za zaštitu komunikacije putem protokola kao što su GOOSE i SV. Na samom početku rada opisan je standard IEC 61850 te njegova struktura. Prikazan je i objašnjen sustav automatizacije trafostanice kome je standard IEC 61850 i namijenjen te njegov 3-razinski raspored. Nastavno na SAS, dan je kratak pregled modela podataka koji se koriste unutar njega, opis komunikacijskih mehanizama i protokola kojima obavlja prijenos podataka kao i njihovo mapiranje na protokole poput TCP-a, IP-a, UDP-a i ostalih. U radu se nalazi i jednostavan primjer SCL datoteke na kojem je ukratko predstavljen SCL koji služi tome da se definiraju IED-i, podaci koji se prenose u komunikaciji između njih te cjelokupna komunikacija unutar SAS-a. Također je dan i uvid u sigurnost IEC 61850 standarda na način da su objašnjene prijetnje te napadi na sustave koji ga implementiraju, ali i kako ih ublažiti i spriječiti koristeći standard IEC 62351 čija namjena je da bude nadopuna spomenutom standardu. Budući da je praktični dio temeljen na njemu, ukratko je opisana biblioteka libIEC61850 te je prikazano pokretanje jednog od primjera iz nje. Implementirani protokol raspodjele ključeva je preuzet iz [5] te je opisan u sklopu ovog rada. Konačno, rad sadrži i način implementacije spomenutog protokola, načini izgradnje implementiranih primjera te demonstraciju implementiranog protokola.

Ključne riječi: IEC 61850, libIEC61850, IEC 62351, upravljanje kriptografskim ključevima, raspodjela kriptografskih ključeva, kibernetička sigurnost, elektroenergetski sustav, protokol

Abstract

Implementation of the cryptographic key management mechanism in the libiec61850 program library

This paper presents an upgrade of the libIEC61850 library in a way to implement a cryptographic key distribution mechanism that can be used within the library to protect communication through protocols such as GOOSE and SV. At the very beginning of the work, the IEC 61850 standard and its structure are described. The substation automation system for which the IEC 61850 standard is intended and its 3-level layout are presented and explained. Continuing on SAS, a brief overview of the data model used within it is given, a description of the communication mechanisms and protocols used to transfer data, as well as their mapping to protocols such as TCP, IP, UDP and others. The paper also contains a simple example of an SCL file that briefly presents the SCL used to define IEDs, the data transmitted in communication between them, and the entire communication within SAS. An insight into the security of the IEC 61850 standard is also given, in such a way that the threats and attacks on the systems that implement it are explained, as well as how to mitigate and prevent them using the IEC 62351 standard, which is intended to be a supplement to the mentioned standard. Since the practical part is based on it, the libIEC61850 library is briefly described and the running of one of the examples from it is shown. The implemented key distribution protocol was taken from [5] and is described as part of this paper. Finally, the paper contains the implementation method of the mentioned protocol, way to build implemented code examples and the demonstration of the implemented protocol.

Keywords: IEC 61850, libIEC61850, IEC 62351, management of cryptographic keys, distribution of cryptographic keys, cybersecurity, power system, protocol