Univerzitet u Beogradu
Matematički fakultet

Kriptoanaliza A5/2

Master rad

Student:
Martin Hofer 1012/2011

Mentor:
prof. dr Miodrag Živković

Septembar, 2016.

Mentor: prof. dr Miodrag Živković
Matematički fakultet,
Univerzitet u Beogradu

Članovi komisije: prof. dr Filip Marić
Matematički fakultet,
Univerzitet u Beogradu

doc. dr Mladen Nikolić
Matematički fakultet,
Univerzitet u Beogradu

Datum odbrane: _____

Sadržaj

Predgovor

U ovom radu predstavljena je kriptoanaliza šifre A5/2 sa poznavanjem otvorenog teksta i njemu odgovarajućeg šifrata. Šifra A5/2 koristi se za zaštitu podataka u komunikaciji u GSM sistemima.

Rad je podeljen u devet poglavlja. Najpre, u poglavlju 1, opisan je GSM sistem u okviru kojeg se vrše napadi kriptoanalizom, kao i osnovni pojmovi kriptologije: kriptografija i kriptoanaliza. U 2. poglavlju opisana je arhitektura GSM mreže, gde su predstavljeni osnovni pojmovi i elementi koji čine mrežu. U poglavlju 3, opisana je sigurnost, kao i privatnost korisnika. Pored toga, ukratko su opisani algoritmi za autentifikaciju korisnika, generisanje ključa, algoritam za šifrovanje podataka, kao i procedura za uspostavljanje poziva između mreže i korisnika. Zatim, sledi pogravlje 4 u okviru kojeg je opisan rad algoritma A5/2, njegova inicijalizacija i generisanje niza ključa. Pre napada na A5/2, u poglavlju 5 prikazani su neki tehnički aspekti GSM mreže, koji napad čine mogućim. U okviru 6. poglavlja opisani su pasivni napadi na A5/2 i to sa poznavanjem otvorenog teksta i njemu odgovarajućeg šifrata i napad sa poznavanjem samo šifrata. U 7. poglavlju predstavljeni su aktivni napadi na mrežu, u okviru kojih napadač aktivno učestvuje u razmeni informacija (presreće informacije, šalje mreži, kao i telefonu). Ovi napadi se zasnivaju na manama protokola komunikacije telefona i mreže. U poglavlju 8, predstavljeni su mogući scenariji napada, tj. na koje sve načine napadač može da dođe do njemu bitnih informacija za napad i eventualno samog ključa za šifrovanje. Nakon svih opisanih tehničkih detalja, u 9. poglavlju predstavljena je programska realizacija napada iz tačke 6.1.3.

1 Uvod

Jedno od ključnih sredstava poslovanja i komunikacije u današnje vreme jesu informacije koje se svakodnevno prenose putem mobilnih telefona. Pojavom druge generacije GSM mreže javlja se potreba za razvojem kriptografskih algoritama koji omogućavaju šifrovanje podataka koji se šalju putem mreže.
Za razliku od kriptografije, cilj kriptoanalize je pronalaženje slabosti algoritama, kako bi se otkrili sadržaji šifrovanih informacija bez poznavanja tajnog ključa. Tokom razvoja kriptografskih algoritama, razvijeni su i razni napadi na kriptografske sisteme. Neki jednostavniji napadi se odnose na analiziranje šifrovanih tekstova kako bi se eventualno uočila neka sličnost koju je moguće iskoristiti za dešifrovanje, dok malo složeniji napadi uključuju obradu nekih složenijih matematičkih operacija. Napad na svaki kriptografski algoritam je moguće sprovesti pretraživanjem svih mogućnosti (engl. brute force), ali to obično nije praksa, jer zahteva previše vremena.

1.1 GSM

CEPT (European Conference of Postal and Telecommunications Administrations, originalno *Conférence européenne des administrations des postes et des télécommunications*) 1988. godine osniva ETSI (European Telecommunications Standards Institute), nezavisnu organizaciju za standardizaciju u oblasti telekomunikacija. ETSI je bio zadužen za razvoj i proizvodnju globalno primenljivih standarda za informacione i komunikacione tehnologije, uključujući fiksne, mobilne, radio i internet tehnologije. GSM predstavlja jedan od standarda razvijen od strane ETSI koji opisuje protokole za digitalnu mobilnu mrežu druge generacije (2G) koju koriste mobilni telefoni. Osnovi ovog standarda usvojeni su i predstavljeni 1991. godine u Finskoj.
S obzirom da prethodni mobilni sistemi nisu imali nikakav vid zaštite, te su predstavljali predmet kriminalnih radnji, kao što su prisluškivanje poziva, kloniranje telefona, krađa poziva, GSM je bio prvi mobilni sistem koji je razmatrao pretnje po pitanju bezbednosti sistema. Primer je uvođenje hardverske komponente u telefonu, SIM (Subscriber Identity Module) kartica, koja je predstavljala dodatnu zaštitu, jer je korisnik pre uspostavljanja veze morao da odradi proces prijavljivanja na mrežu. Cilj je bio i obezbediti privatnost korisnika, što je u osnovi obezbeđeno uglavnom šifrovanjem. GSM koristi A5 kao jedan od algoritama za šifrovanje podataka. A5/2 je protočna šifra, koja obezbeđuje privatnost razgovora u GSM-u. Razvijena je od strane ETSI, za zemlje Azije koje su imale zabranu korišćenja tehnologija zapadnih zemalja. Šifra A5/2 predstavlja slabiju verziju A5/1. Ispostavilo se da je previše slaba za šifrovanje, toliko slaba da je moguće bilo "provaliti" i sa slabom opremom u realnom vremenu. To se i desilo 1999. godine kada su Ian Goldberg i David Wagner [2] uspeli da dekriptiraju A5/2. Prema udruženju GSMA (GSM Association), jula 2006. godine, mobilni telefoni prestaju da podržavaju A5/2, te se prelazi na korišćenje A5/1.

1.2 Osnovni pojmovi kriptologije

Kriptologija (engl. cryptology) je oblast matematike koja obuhvata kriptografiju i kriptoanalizu.
Kriptografija (engl. cryptography) je nauka koja se bavi metodima očuvanja tajnosti informacija. Ona omogućava da osoba A sigurno pošalje svoju poruku osobi B, a da neka treća strana osoba C, ne može da dođe do sadržaja poruke.
Poruka koja se šalje naziva se **otvoreni tekst** (engl. plaintext). Šifrovana poruka se naziva **šifrat** (engl. ciphertext). Proces transformacije otvorenog teksta u šifrat, koji za cilj ima prikrivanje sadržaja njegovog sadržaja se naziva **šifrovanje** (engl. encryption).

Dešifrovanje (engl. decryption) je inverzna tranformacija šifrovanju, tj. proces vraćanja šifrata u otvoreni tekst.
Neka je sa P označen otvoreni tekst, šifrat sa C, funkcija šifrovanja sa E i funkcija dešifrovanja sa D. Proces šifrovanja poruke matematički se zapisuje $E(P) = C$, a proces dešifrovanja šifrata $D(C) = P$. Kroz operacije šifrovanja i dešifrovanja prenosi se originalna poruka i treba da važi $D(E(P)) = P$.
Kriptografija mora da obezbedi:

- Integritet informacija; Podaci koji se šifruju ne smeju biti promenjeni, brisani ili zamenjeni drugim informacijama.
- Tajnost informacija; Sadržaj informacija je dostupan samo ovlašćenim osobama, odnosno osobama koje poseduju ključ.
- Provera identiteta; Osobe koje komuniciraju prvo moraju da se predstave jedna drugoj, nakon čega počinju razmenu informacija.

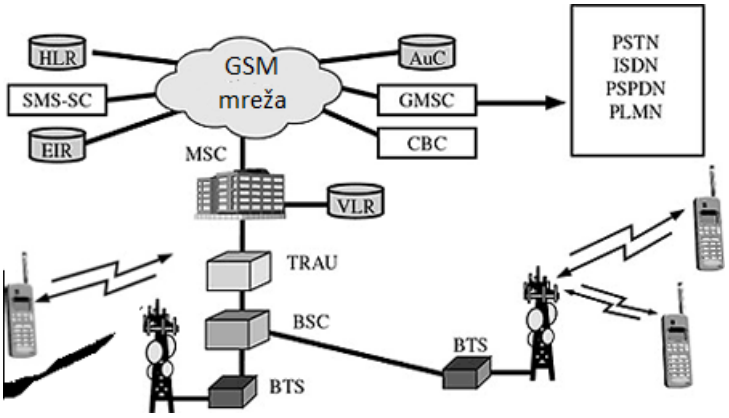
Kriptografski algoritam ili **šifra** (engl. cipher) je matematička funkcija koja se koristi za šifrovanje i dešifrovanje. Postoji nekoliko podela šifri: prema alfabetu šifrata, prema ključu, tj. algoritmima šifrovanja i dešifrovanja, prema tipu operacija pri šifrovanju, prema načinu na koji se obrađuje otvoreni tekst. Nama je od značaja poznavanje sifri prema načinu na koji se obrađuje otvoreni tekst i to protočna šifra. Protočna šifra transformiše otvoreni tekst simbol po simbol, odnosno kombinovanjem simbol po simbol sa nizom ključa, koji se dobija iz kriptografskog generatora pseudoslučajnih brojeva.
Kriptoanaliza (engl. cryptoanalysis) je naučna disciplina koja proučava metode otkrivanja značenja šifrovanih informacija bez korišćenja tajnih informacija za dešifrovanje. Pokušaj kriptoanalize naziva se napad. Uspešna, ili makar delimična, kriptoanaliza naziva se dekriptiranje. Iako je cilj oduvek isti, metode i tehnike kriptoanalize su se tokom istorije kriptografije menjale, prilagođavajući se povećanju kompleknosti kriptografije. Najpre su te metode podrazumevale papir i olovku, zatim mašine, kao što je Enigma tokom Drugog svetskog rata, pa sve do napada nad algoritme šifrovanja koji se primenjuju u računarskim sistemima.

2 Arhitektura GSM mreže

Arhitektura GSM mreže je struktuirana u nekoliko izolovanih sekcija:

- Mobilna stanica
- Podsistem baznih stanica
- Mrežni komutatorski podsistem

Mobila stanica (engl. Mobile Station - MS) može biti bilo koji uređaj, ali u većini slučajeva to je mobilni telefon koji bežičnim putem komunicira sa primopredajnikom bazne stanice (engl. Base Transceiver Station - BTS). Ovi primopredajnici su rasporedjeni po ćelijama. Engleski naziv za ćeliju je *cell*, otuda i naziv *cellular network*, tj. mobilna mreža. Kroz usmerene antene, područje oko jednog baznog primopredajnika, podeljeno u sektore, može da opsluži sve susedne ćelije. Svaki bazni primopredajnik komunicira sa kontrolerom bazne stanice (engl. Base Station Controller - BSC) i zajedno čine podsistem bazne stanice (engl. Base Station Subsystem - BSS). Mrežni komutatorski podsistem (engl. Network SubSystem - NSS) je deo GSM sistema koji obavlja funkcije komutiranja i upravljanja komunikacije između mobilnih telefona i komutirane javne telefonske mreže. On je razvijen i u vlasništvu je operatora mobilne telefonije i dozvoljava



Slika 1: Struktura GSM mreže

mobilnim telefonima da komuniciraju jedni sa drugima i telefonima u široj javnoj telefonskoj mreži. Mrežni komutatorski podsistem takođe se naziva kao osnovna GSM mreža.
Kontroler bazne stanice komunicira sa mrežnim komutatorskim podsistemom gde je priključen na mobilni komutacioni centar (engl. Mobile service Switching Centre - MSC) koji na kraju upravlja svim komunikacijama i unutar i van sistema. U MSC se koriste tri baze podataka:

- Registar vlastitih pretplatnika (engl. The Home Location Register - HLR) predstavlja centralnu bazu podataka. HLR sadrži detalje svakog pretplatnika mobilnog telefona kojem je dozvoljeno korišćenje osnovne GSM mreže
- Autentifikacioni centar (The Authentication Centre - AUC) predstavlja zaštićenu bazu podataka koja se pridružuje svakom HLR-u. AUC sadrži kopije tajnih ključeva za autentifikaciju svih postojećih SIM kartica koji se koriste za proveru autentičnosti i kodovanje radio signala prilikom povezivanja na mrežu i tokom komuniciranja kroz istu.
- Registar gostujućih pretplatnika (engl. Visitors Location Register - VLR) predstavlja bazu podataka pretplatnika koji se trenutno nalaze u geografskoj oblasti koju kontroliše MSC. Svaka primopredajna bazna stanica u mreži služi za tačno jedan VLR, tako da pretplatnik ne može biti u više od jednog VLR-a u datom trenutku. Pored ovih registra postoji i registar identiteta uređaja (engl. Equipment Identity Register - EIR) koji je često integrisan sa HLR-om. EIR čuva spisak mobilnih uređaja (prepoznaju se po svom IMEI broju) koji su zabranjeni na mreži ili se nadziru. Ovaj registar je osmišljen kako bi se omogućilo praćenje ukradenih ili neispravnih mobilnih telefona.

MSC je osnovni čvor koji pruža usluge za GSM i odgovoran je za upravljanje pozivima i kratkim porukama, kao i za druge usluge kao što su konferencijski pozivi, faks itd. MSC uspostavlja, održava i raskida pozive, upravlja mobilnost i potrebe za prosleđivanjem tokom poziva. Postoje različiti nazivi za razne MSC-ove u različitim kontekstima, koji odražava njihove složene uloge u mreži.

3 Sigurnost i privatnost u okviru GSM mreže

Sigurnost i privatnost podataka u današnje vreme su od velikog značaja, pogotovu u komunikacijskim sistemima gde se podaci prenose putem radio talasa. Ovakva vrsta prenosa

podataka otvara vrata neželjenim upadima i prisluškivanjima na mreži. Nekoliko bezbedonosnih funkcija su ugrađeni u GSM mrežu kako bi zaštitili privatnost pretplatnika. To su:

- Autentifikacija registrovanog pretplatnika
- Sigurnost prenosa podataka šifrovanjem
- Zaštita identiteta pretplatnika
- Nefukncionalnost mobilnog telefna bez SIM kartice
- Duplirane SIM kartice nisu dozvoljene na mreži
- Bezbedno skladišten autentifikacioni ključ pretplatnika K_I

Ona najvažnija koja je predstavljena u ovom radu sigurnost podataka šifrovanjem.
Tokom razvoja u periodu od 1982. do 1991. godine, poboljšava se sigurnost GSM mreže. Sa razvitkom digitalne komunikacije, pojavljuje se potreba za korišćenjem kriptografskih algoritama koji direktno mogu da šifruju i dešifruju digitalni tok podataka. Takvi algoritmi su implementirani u odvojene hardverske komponente. GSM mreža koristi kriptografske algoritme u tri svrhe:

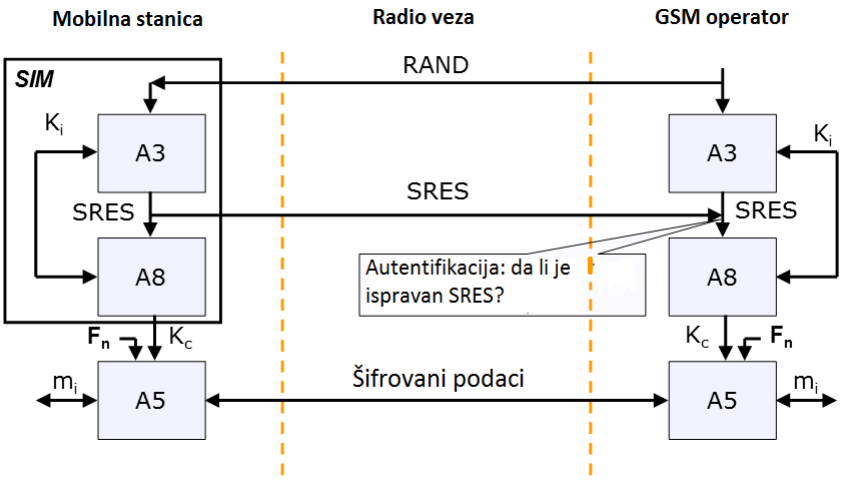
- Algoritam A3 - Autentifikacija
- Algoritam A8 - Generisanje ključa
- Algoritam A5 - Šifrovanje podataka

GSM mreža danas predstavlja najsigurniji sistem za mobilnu telekomunikaciju.

3.1 Autentifikacija - Algoritam A3

Postupak autentifikacije proverava ispravnost SIM kartice pretplatnika, a zatim se odlučuje da li je mobilnoj stanici dozvoljen pristup mreži. Operator mreže potvrđuje verodostojnost pretplatnika korišćenjem metode izazov-odgovor (engl. challenge-response method). Mobilna stanica izračunava 32-bitni odgovor (engl. signed response - *SRES*) izdavanjem komande SIM kartici. Ova komanda podrazumeva generisanje 128-bitnog slučajnog broja (engl. random number - *RAND*) od strane HLR. SIM koristi autentifikacioni ključ pretplatnika K_I, *RAND* i algoritam A3 za izračunavanje 128-bitnog odgovora koji se vraća mobilnoj stanici. Zatim mobilna stanica šalje *SRES* preko BTS, MSC, ponavlja se izračuvanje, da bi se na kraju potvrdio identitet pretplatnika u AUC. Ako se primljeni *SRES* slaže sa izračunatom vrednošću, autentikacija je potvrđena i mobilna stanica može nastaviti sa radom na mreži. Ako se vrednosti ne slažu, veza sa GSM mrežom se prekida. Za *SRES* se koriste samo 32 bita izračunate vrednosti. Kompletno izračunavanje *SRES*-a se vrši u okviru SIM. To obezbeđuje veću sigurnost poverljivih informacija, kao što su Internacionalni Identifikacioni Broj Pretplatnika (engl. International Mobile Subscriber Identity - IMSI[1]) ili ključ za autentikaciju K_I. A3 je implementiran u pametnoj karici (engl. SmartCard), tako da K_I pojedinačnog pretplatnika nikad ne napušta SIM, tj. nikada se ne prenosi putem radio kanala.
Najčešća implementacija A3 u okviru GSM mreže je verzija algoritma COMP128. 1997. godine objavljena je prva verzija COMP128 zbog procurelih dokumenata, a ubrzo nakon toga je i uspešno napadnut. Danas je poznat algoritam kojim se K_I može izvući iz ukradene SIM kartice za

[1] IMSI je obično 15-cifreni broj, ali može biti i kraći. Prve 3 cifre su mobilni kôd zemlje, a prate ih kôd mobilne mreže sa 2 cifre (Evropski standard) ili 3 cifre (Severno Američki standard). Preostale cifre su identifikacioni broj mobilnog telefona unutar mreže pretplatnika



Slika 2: A3 - Autentifikacija na GSM mrežu

manje od jednog minuta, čime se dobija pristup SIM i informacija o PIN kodu. Sa izvučenim K_I moguće je probiti sigurnosni sistem autentikacije. Zbog ovoga GSM mrežni operatori prelaze na korišćenje algoritma COMP128-2, COMP128-3 i COMP128-4 (koristi se u 3G mrežama) koji se još uvek drže u tajnosti.

3.1.1 Anonimnost

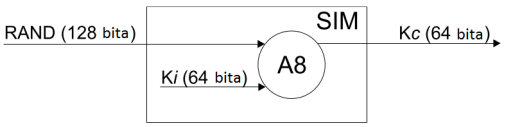
Kada GSM pretplatnik uključi svoj telefon prvi put, njegov IMSI se šalje AuC na mreži. Nakon toga, pretplatniku se dodeljuje privremeni identifikacioni broj pretplatnika (engl. Temporary Mobile Subscriber Identity - TMSI). IMSI se nakon ovog trenutka veoma retko prenosi, osim ako je to neophodno, jer se time sprečava da potencijalni priskluškivač otkrije identitet GSM korisnika na osnovu njegovog IMSI. Korisnik nastavlja da koristi TMSI, u zavisnosti koliko često promeni lokaciju. Svakom promenom lokacije, mreža dodeljuje novi TMSI mobilnom telefonu. TMSI se zajedno sa IMSI čuva na mreži. Mobilna stanica koristi TMSI da se prijavi na mrežu ili tokom iniciranja poziva. Na sličan način mreža komunicira sa mobilnom stanicom. VLR obavlja posao dodeljivanja, administriranja i ažuriranja TMSI. Kada se telefon isključi, TMSI se čuva na SIM kartici, da bi znali da je telefon dostupan kada se telefon ponovo uključi.

3.2 Generisanje ključa - Algoritam A8

Algoritam A8 služi za generisanje ključa i veoma je sličan algoritmu A3. Tačnije koristi se isti algoritam (COMP128) za proizvođenje 64-bitnog ključa za šifrovanje K_c koji se zatim koristi u algoritmu A5 za šifrovanje i dešifrovanje podataka. Na osnovu 128-bitnog $RAND$ primljenog od MSC i 128-bitnog K_I koji se nalazi na SIM kao dva ulazna parametra, A8 izračunava 64-bitni broj kao izlaz. Samo 54 bita se koriste kao ključ za šifrovanje K_c, dok preostalih 10 bitova su popunjeni nulama.

3.3 Šifrovanje podataka - Algoritam A5

Kako bi se zaštitili podaci koji se bežičnim putem šalju preko mreže, GSM šifruje podatke koristeći protočnu šifru poznatiju kao A5, tj. algoritam za šifrovanje podataka. Kada je korisnik



Slika 3: A8 - Generisanje ključa

autentifikovan na mrežu, $RAND$ se zajedno sa K_I šalje algoritmu A8 kako bi proizveo ključ za šifrovanje K_c. Ovaj ključ se koristi u algoritmu A5 kako bi se podaci šifrovali i dešifrovali. A5 je implementiran u mobilnom telefonu, kako bi mogao da šifruje i dešifruje podatke neposredne pred prenos .
Postoje četiri različite varijante A5 algoritma:

- A5/0 algoritam koji ne šifruje podatke. Uglavnom se koristio u zemljama pod sankcijom Ujedinjenih Nacija i pojedinim zemljama trećeg sveta.
- A5/1 je najjača verzija algoritma. Specificiran je sredinom 80-tih nakon rasprave o jačini algoritma koja se vodila između nekoliko članica NATO-a. Algoritam se koristio pretežno u Zapadnoj Evropi i Americi.
- A5/2 predstavlja oslabljenu verziju algoritma A5/1. Uglavnom se koristila u zemljama Azije.
- A5/3 je kasnije napravljen za korišćenje u 3G mreži i predstavlja novi algoritam zasnovan na blokovskoj šifri KASUMI.

3.4 Uspostavljanje veze u GSM mreži

Pozivi se uspostavljaju na sledeći način:

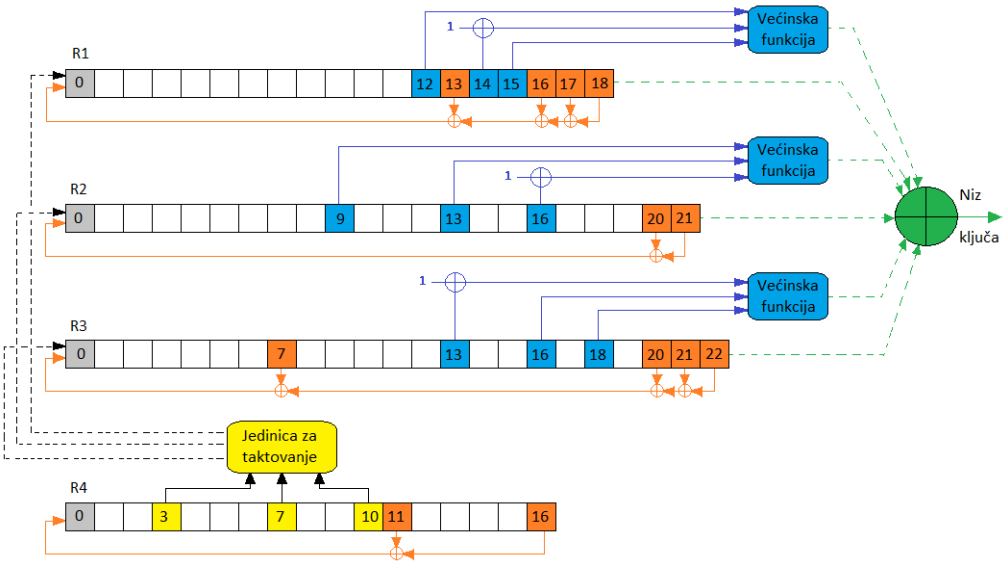
1. U slučaju da je poziv iniciran od strane mreže, mreža poziva telefon slanjem PAGING REQUEST, korišćenjem njegovog IMSI ili TMSI, na određenom kanalu (PAGCH).
2. Ukoliko je poziv iniciran od strane telefona, procedura počinje od ovog koraka.
   - Telefon šalje CHANNEL REQUEST poruku na RACH (engl. Random Access Channel) kanal. Ova poruka sadrži veoma malo informacija, svega 8 bita, od čega su 3 bita za povod za uspostavljanje veze.
   - Mreža šalje poruku IMMEDIATE ASSIGNMENT na PAGCH. Ova poruka između ostalog sadrži redni broj TDMA okvira (pogledati 5. poglavlje) u okviru kojeg je primljena CHANNEL REQUEST poruka, kao i detalji kanala koji je dodeljen mobilnom telefonu. Mobilni telefon se nakon ove poruke podešava na dodeljeni kanal za odvijanje razgovora (engl. traffic channel).
3. Servis za zahteve i razmatranje odluke o TMSI:
   - Mobilni uređaj šalje zahtev za uslugu. Ova poruka sadrži i TMSI. Pored toga sadrži i spisak verzija A5 algoritama koje telefon podržava (class-mark), kao i redni broj ključa za šifrovanje (0,...,6).
   - Mreža potvrđuje zahtev i ponavlja proveru TMSI. Razlog za to je zato što dva mobilna telefona na neki način mogu da registruju da im je dodeljen isti kanal. Stoga, samo TMSI telefona koji je potvrđen od mreže ostaje na kanalu, dok se drugi prekida.

4. Autentifikacija:

   - Mreža šalje zahtev za autentifikaciju (AUTHREQ). Taj zahtev sadrži slučajnu 128-bitnu vrednost $RAND$, kao i redni broj ključa za šifrovanje. Rezultat ove poruke je K_c.
   - Mobilni telefon odgovora na zahtev sa potpisanim odgovorom ($SRES$) u okviru odgovora za autentifikaciju (AUTHRES).
   - Mreža zahteva od telefona da započne šifrovanje (CIPHMODCMD). Mreža može da odredi koji će se algoritam za šifrovanje koristiti, a može da predloži i ključ za šifrovanje rednim brojem (0,..., 6). Nakon toga mreža započinje dešifrovanje dolazećih informacija.
   - Mobilni telefon započinje šifrovanje i dešifrovanje i odgovora šifrovanom CIPHMODCOM.

5. Mreža i telefon komuniciraju na kanalu. Može doći do promene kanala u okviru kojeg komuniciraju. U tom slučaju mreža šalje informaciju o novom kanalu. Ukoliko se razgovor šifruje, onda se šifruje i informacija o novom kanalu.

4 Algoritam A5/2



Slika 4: Interna struktura A5/2

A5/2 predstavlja slabiju verziju algoritma A5/1. Razvijen je 1999. godine u skladu sa ograničenjima politike izvoza, izvan Evropske Unije jer u to vreme nije bio dozvoljen izvoz jakih šifara. A5/2 je protočna šifra sačinjena od četiri pomeračka registra sa linearnom povratnom spregom $R1, R2, R3, R4$ dužine redom, 19, 22, 23 i 17 bita.
Na slici 4 prikazana je struktura A5/2, gde su ćelije pomeračkih registara numerisane sleva udesno sa $0, 1, \ldots, n-1$, gde je n dužina pomeračkog registra. Pre nego što se registar taktuje izračunava

se povratna vrednost, tj. XOR-uju se vrednosti na povratnim granama. Nakon toga sadržaj registra se pomera jedan bit u desno, pritom odbacujući bit sa indeksom $n-1$, a povratna vrednost se upisuje u ćeliju sa indeksom 0.
Registri se inicijalizuju 64-bitnim ključem K_c i javnom 22-bitnom vrednošću COUNT koju označavamo sa f (engl. GSM Frame Identifier). Procedura inicijalizovanja registara prikazana je na slici 5. Pri tome je i-ti bit K_c, odnosno f označen sa $K_c[i]$, odnosno $f[i]$, a indeks $i = 0$ je najmanje značajan bit (engl. LSB - Least Significant Bit).

1. $R1 = R2 = R3 = R4 = 0;$

2. for $i = 0$ to 63
   - taktovatiSvaCetiriRegistra();
   - $R1[0] = R1[0] \oplus K_c[i];$
   - $R2[0] = R2[0] \oplus K_c[i];$
   - $R3[0] = R3[0] \oplus K_c[i];$
   - $R4[0] = R4[0] \oplus K_c[i];$

3. for $i = 0$ to 21
   - taktovatiSvaCetiriRegistra();
   - $R1[0] = R1[0] \oplus f[i];$
   - $R2[0] = R2[0] \oplus f[i];$
   - $R3[0] = R3[0] \oplus f[i];$
   - $R4[0] = R4[0] \oplus f[i];$

4. $R1[15] = R2[16] = R3[18] = R4[10] = 1;$

Slika 5: Procedura postavljanja ključa A5/2

Stanje registra nakon postavljanja ključa, označeno sa $R1, R2, R3, R4$, je linearno u odnosu na bite K_c i f, bez bita $R1[15], R2[16], R3[18], R4[10]$, koji su postavljeni na 1.
A5/2 radi u ciklusima, gde je na kraju ciklusa proizveden jedan izlazni bit. Jedinica za taktovanje (Clocking Unit registra $R4$ - Slika 4) izračunava većinsku funkciju nad bitovima, gde se na osnovu izlaza te funkcije određuje koji registar će biti taktovan. Ulaz u većinsku funkciju su tri bita, a izlaz je jedan bit. Većinska funkcija je kvadratna, koja se zapisuje $majority(a, b, c) = a \cdot b \oplus b \cdot c \oplus c \cdot a$. U slučaju kada je potrebno odrediti taktovanje registra $R1, R2$ i $R3$ ulaz u funkciju su bitovi $R4[3], R4[7]$ i $R4[10]$. Nakon što se izračuna izlaz većinske funkcije, najmanje dva od tri registra $R1, R2$ i $R3$ se taktuju na sledeći način: $R1$ je taktovan akko bit $R4[10]$ je jednak izlazu većinske funkcije, $R2$ je taktovan akko bit $R4[3]$ jednak izlazu većinske funkcije i $R3$ je taktovan akko bit $R4[7]$ jednak izlazu većinske funkcije. Registar $R4$ se uvek taktuje. Taktovanje jednog registra podrazumeva pomeranje registra udesno, pri čemu se oslobađa pozicija u registru sa indeksom nula. Na tu poziciju se upisuje bit, koji se izračunava tako što se XOR-uju biti sa povratnih grana. Na slici 6 prikazano je izračunavanje izlaznog bita većinske funkcije za taktovanje registra $R1, R2$ i/ili $R3$.
Nakon taktovanja registra izračunava se izlazni bit, generisan XOR funkcijom sa argumentima: bit u ćeliji sa indeksom $n-1$ registra $R1, R2$ i $R3$, gde je n dužina odgovarajućeg registra i izlaza većinskih funkcija. Izlaz većinske funkcije se računa funkcijom *majority*, gde su argumenti: tri bita registra određeni svojim pozicijama, od kojih se vrednost jednog bita komplementira (Slika 4).
Prvih 99 bitova izlaza se odbacuju, što znači da se registri samo taktuju po prethodno definisanom pravilu, a sledećih 228 bitova izlaza se uzimaju kao izlazni niz ključa (engl. keystream). Procedura generisanje niza ključa prikazana je na slici 7.

$R_4[3]$	$R_4[7]$	$R_4[10]$	Većinska funkcija	R_1	R_2	R_3
0	0	0	0	takt	takt	takt
1	0	0	0	takt		takt
0	1	0	0	takt	takt	
1	1	0	1		takt	takt
0	0	1	0		takt	takt
1	0	1	1	takt	takt	
0	1	1	1	takt		takt
1	1	1	1	takt	takt	takt

Slika 6: Izračunavanje vrednosti za taktovanje registra $R1, R2$ i $R3$

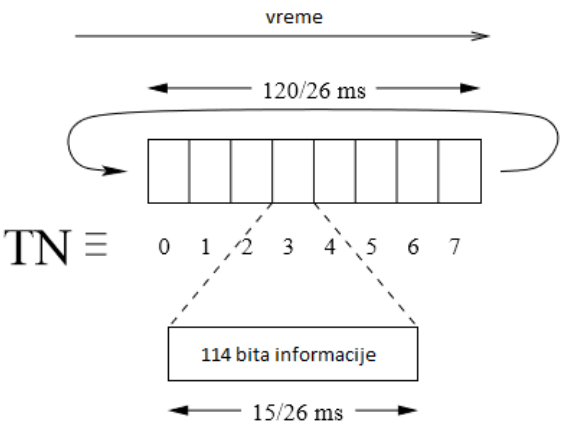
1. Pokrenuti postavljanje ključa sa ulaznim parametrima f (Slika 4).
2. Izvršiti 99 ciklusa i odbaciti izlaz.
3. Izvršiti 228 ciklusa i iskoristiti izlaz kao niz ključa.

Slika 7: Procedura generisanja niza ključa

Izlaz od 228 bita je podeljen u dva dela. Prva polovina od 114 bita se koristi kao niz ključa za šifrovanje podataka na vezi od mreže ka telefonu, a druga polovina od 114 bita za šifrovanje podataka na vezi od telefona ka mreži. Šifrovanje se vrši bitskom operacijom XOR poruke podeljene u okvire dužine 114 bita i niza ključa.

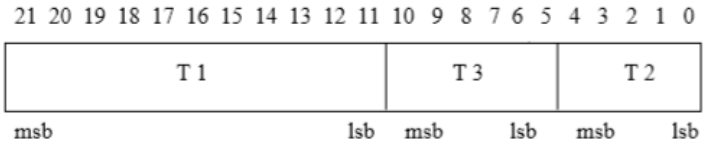
5 Tehnički aspekti GSM sistema

Pre prikazivanja napada, potrebno je objasniti neke tehničke aspekte GSM sistema koji su od značaja za napade. U GSM sistemima jedan fizički kanal može da opsluži i do osam različitih



Slika 8: TDMA okvir

telefona, dodeljujući fizički kanal po principu *round-robin*[1], gde svaki telefon prenosi podatke u jednom vremenskom slotu (engl. *time slot*) od 15/26 ms. Ovaj metod je poznat kao TDMA (engl. Time Division Multiple Access) (Slika 8). Svaki okvir sačinjen je od osam vremenskih slotova, koji odgovaraju njihovim brojevima (engl. TN - Time slot Number). Svaki TDMA okvir ima broj koji mu je pridružen. U svakom vremenskom slotu može biti preneto 114 bita podataka. Fizički kanal između mreže i telefona ima maksimalnu propusnost od 114 po TDMA okviru, tj. 24.7 Kbit/s.
Generisanje niza ključa koristeći A5/2 za određeni okvir zavisi od rednog broja TDMA okvira. Tačnije, A5/2 se ponovo pokreće za svaki TDMA okvir, kako bi se generisao novi niz ključa. U opisu šifre A5/2 pomenuto je da na postavljanje ključa utiče vrednost COUNT. Ta vrednost određena je iz rednog broja TDMA okvira (Slika 9).



Slika 9: Izvođenje vrednosti COUNT, odnosno f

Ovde T1 predstavlja količnik rednog broja okvira pri deljenju sa $51 \cdot 26 = 1362$, T2 je ostatak pri deljenju broja okvira sa 26, a T3 je ostatak pri deljenju broja okvira sa 51.

6 Napadi na A5/2

Kao što je već pomenuto kriptoanaliza predstavlja proces pomoću kojeg napadač pokušava da transformiše šifrat u odgovarajući tekst, a da pritom ne zna ključ. Dekriptiranje predstavlja uspešnu kriptoanalizu.
Cilj napada nije nužno dekriptiranje samo jedne šifrovane poruke, već sticanje informacija o ključu koji se koristi u datom sistemu, pri čemu se na taj način kompromituje prošla, a verovatno i buduća komunikacija. U odnosu na tip informacije koju napadač poseduje, napadi mogu biti podeljeni na:

- Napad sa poznavanjem samo šifrata
- Napad sa poznatim otvorenim tekstom
- Napad sa poznavanjem izabranog otvorenog teskta.

U ovom radu se obrađuje samo napad sa poznatim otvorenim tekstom i odgovarajućim šifratom.

6.1 Napadi na A5/2 sa poznatim otvorenim tekstom

Kod napada sa poznatim otvorenim tekstom napadač ima pristup i otvorenom tekstu i njemu odgovarajućem šifrovanom tekstu, i sprovodi analizu datih podataka sa ciljem pronalaženja ključa koji se koristi za šifrovanje.

[1]Round-robin je algoritam koji svakom procesu (u ovom slučaju telefonu) dodeljuje jednaki deo vremena u kružnom redosledu, obrađujući svaki proces bez prioriteta.

6.1.1 Napad Goldberg, Wagner i Green

Napad GWG (Goldberg, Wagner, Green) se zasniva na činjenici da, kako je $R4[10]$ postavljen na jedinicu nakon generisanja niza ključa, nezavisno od toga da li je bit $f[10]$ nula ili jedan, $R4$ će imati istu vrednosti. Kako $R4$ kontroliše taktovanje registara $R1, R2$ i $R3$, taktovanje ovih registara je nezavisno od vrednosti $f[10]$.
Kako bi postavio napad, napadač mora da pronađe dva različita okvira sa potpuno istim stanjem registra $R4$, a time i istim rasporedom pokretanja registara $R1, R2$ i $R3$, tj. dva okvira f koji imaju istu vrednost $f[10]$. Uzimajući u obzir način na koji se od rednog broja TDMA dobija COUNT, dovoljna su dva okvira koji su na razmaku tačno $26 \cdot 51 = 1326$ TDMA (oko 6 sekundi) razdvojena, gde je $f[10]$ prvog okvira jednako 0. Ukoliko je $f[10]$ prvog okvira jednako 1, onda je napadač primoran da sačeka dodatnih 6 sekundi, tj. naredni okvir čiji $f[10] = 0$. Napadač ne može da upotrebi okvir sa $f[10] = 1$ kao prvi okvir, jer zbog prenosa kod dodavanja jedinice, ostali bitovi COUNT-a su promenjeni, pa će tako i vrednost $R4$ biti različita u dva okvira. Iz tog razloga napadač je primoran da sačeka od 6 do 12 sekundi kako bi prikupio odgovorajuće podatke za napad.
Pretpostavke za napad su sledeće. Neka su f_1 i f_2 odgovarajuće COUNT vrednosti dva okvira na prethodno opisan način, sa odgovorajućim nizom ključa, k_1 i k_2. Stanja registara $R1, R2, R3, R4$ u prvom okviru, nakon postavljanja ključa, ali pre prvog koraka generisanja niza ključa, tj. nakon odbacivanja 99 bitova, su označena sa $R1_1, R2_1, R3_1, R4_1$. Na isti način su označena i stanja registara u drugom okviru sa $R1_2, R2_2, R3_2, R4_2$. S obzirom na način izbora okvira f_1 i f_2, vrednosti $R4_1$ i $R4_2$ su jednake, te su njihove vrednosti jednostavno obeležene sa $R4$. Stanja ostalih registra nisu ista, ali kako je proces inicijalizacije linearan (pogledati 6.1.4) u odnosu na bitove f_1 i f_2, razlika (XOR) između $R1_1, R2_1, R3_1$ i $R1_2, R2_2, R3_2$ je takođe linearna u odnosu na $f_1 \oplus f_2$. Ove razlike su fiksne, kako je $f_1 \oplus f_2 = 00000000000010000000000$. Iz ovog može se zaključiti sledeće: $R1_1 = R1_2 \oplus \delta_1, R2_1 = R2_2 \oplus \delta_2, R3_1 = R3_2 \oplus \delta_3$, gde su δ_1, δ_2 i δ_3 konstante koje zavise od ključa.
Sa zadatom vrednošću $R4$, može se pokazati da je (XOR) razlika između nizova ključa $k_1 \oplus k_2$ linearna u odnosu na $R1_1, R2_1, R3_1$. Sa poznatim sadržajem $R4$, poznato je i celokupno taktovanje ostalih registara (i ono je jednako u oba okvira f_1 i f_2, s obzirom da je $R4_1 = R4_2$). Neka su l_1, l_2 i l_3 ukupan broj takovanja registara $R1, R2$ i $R3$ na kraju svakog ciklusa i. Stoga, vrednosti registara $R1, R2$ i $R3$ na kraju ciklusa i za okvir f_i su $L1^{l_1} \cdot R1_1$, $L2^{l_2} \cdot R2_1$ i $L3^{l_3} \cdot R3_1$, gde su $L1$, $L2$ i $L3$ matrice koje izražavaju jedno taktovanje prva tri registra. Na sličan način, vrednosti registara za okvir f_2 na kraju ciklusa i su $L1^{l_1} \cdot (R1_1 \oplus \delta_1)$, $L2^{l_2} \cdot (R2_1 \oplus \delta_2)$ i $L3^{l_3} \cdot (R3_1 \oplus \delta_3)$. Neka $g_1(R1) \oplus g_2(R2) \oplus g_3(R3)$ predstavlja izlazni bit za trenutno zadata stanja registara $R1, R2$ i $R3$. S obzirom da se na po tri ćelije svakog od registra $R1, R2$ i $R3$ primenjuje većinska funkcija koja je kvadratna, i funkcije g_1, g_2 i g_3 su kvadratne.
Tačnije, neka su $x_0, \ldots, x_{18}, y_0, \ldots, y_{21}, z_0, \ldots, z_{22}$ promenljive koje predstavljaju bite registara $R1, R2$ i $R3$, odmah nakon što je generisan prvi bit niza ključa. Izlazni bitovi registara $R1, R2$ i $R3$ su redom:

$$izlazniBit(R1) = x_{12}x_{14} \oplus x_{12} \oplus x_{12}x_{15} \oplus x_{14}x_{15} \oplus x_{15} \oplus x_{18}$$
$$izlazniBit(R2) = y_9y_{13} \oplus y_9 \oplus y_9y_{16} \oplus y_{13} \oplus y_{13}y_{16} \oplus y_{21}$$
$$izlazniBit(R3) = z_{16} \oplus z_{13}z_{16} \oplus z_{18} \oplus z_{13}z_{18} \oplus z_{16}z_{18} \oplus z_{22}$$

Stoga, prvi bit niza ključa je zbir prethodna tri izlazna bita:

$$k_1[0] = izlazniBit(R1) \oplus izlazniBit(R2) \oplus izlazniBit(R3)$$

GWG su primetili da razlika izlaznih bitova može da se izrazi kao linearna funkcija internog stanja u odnosu na f_1. Razlika izlaznog bita u ciklusu i može da se definiše sa:

$$g_{\delta_1}(L1^{l_1} \cdot R1_1) \oplus g_{\delta_2}(L2^{l_2} \cdot R2_1) \oplus g_{\delta_3}(L1^{l_3} \cdot R3_1) \quad .$$

Funkcije g_{δ_1}, g_{δ_2} i g_{δ_3} su linearne funkcije što će biti dokazano.
Razlike u $R1_1, R2_1$ i $R3_1$ su linearne. Ostaje da se pokaže kako je sa datom kvadratnom funkcijom $g(x_1, \ldots, x_n)$ i $\Delta = \Delta_1, \ldots, \Delta_n$ gde su $x_i, \Delta_i \in \{0, 1\}$. Funkcija $g_\Delta = g(x_1, \ldots, x_n) \oplus g(x_1 \oplus \Delta_1, x_2 \oplus \Delta_2, \ldots, x_n \oplus \Delta_n)$ je linearna u odnosu na $x_1, \ldots, x_n$, a jednačine $\Delta \in \{\delta_1, \delta_2, \delta_3\}$. S obzirom da je g kvadratna funkcija, ona može da se zapiše sa:

$$g(x_1, \ldots, x_n) = \sum_{1 \le i,j \le n} a_{i,j} x_i x_j \oplus a_{0,0}$$

gde su $a_{i,j} \in \{0, 1\}$ fiksne za dato g. Pored ovoga treba uzeti u obzir da je $x_i x_i = x_i$. Stoga,

$$g_\Delta = \sum_{1 \le i,j \le n} a_{i,j}(x_i x_j \oplus (x_i \oplus \Delta_i)(x_j \oplus \Delta_j)) =$$

$$= \sum_{1 \le i,j \le n} a_{i,j}(x_i x_j \oplus x_i x_j \oplus x_i \Delta_j \oplus \Delta_i x_j \oplus \Delta_i \Delta_j) =$$

$$= \sum_{1 \le i,j \le n} a_{i,j}(x_i \Delta_j \oplus \Delta_i x_j \oplus \Delta_i \Delta_j)$$

Poslednji izraz je linearan u odnosu na $x_1, \ldots, x_n$ za dato $\Delta_1, \ldots, \Delta_n$.
Dakle, sa datim $R4$ i razlikom niza ključa, tj. $k_1 \oplus k_2$, interno stanje $R1_1, R2_1, R3_1$ može da se izračuna rešavanjem sistema jednačina. Na osnovu internih stanja registara $R1_1, R2_1, R3_1, R4_1$ i okvira f_1, pokretanjem algoritma za nalaženje ključa, obrtanjem registara unazad (6.1.5), može da se otkrije K_c. S obzirom da $R4$ nije poznato, napadač mora da pokuša reši sistem jednačina za svih 2^{16} vrednosti registra $R4$, dok ne pronađe konzistentno rešenje.

6.1.2 Napad Barkan, Biham i Keller

Za ovaj napad je potrebni su nizovi ključa bilo koja četiri okvira f. Cilj napada BBK (Barkan, Biham, Keller) je pronaći stanja registara $R1, R2, R3, R4$, nakon postavljanja ključa, ali pre 99 takta odbacivanja izlaznog bita. Na osnovu tih stanja traži se ključ K_c za šifrovanje algoritmom za nalaženje ključa, obrtanjem registara unazad (6.1.5).
Ovaj napad je unapređenje napada GWG (nije potrebno uzimati okvire koju su razdvojeni tačno 6 sekundi). Pogađa se početno stanje registra $R4$, a svaki izlazni bit se zapisuje kao kvadratna funkcija nekih ćelija registara $R1, R2$ i $R3$. Izlazni bit je moguće zapisati i na različitim okvirima kao kvadratne funkcije nekih ćelija registara $R1, R2$ i $R3$ u odnosu na okvir f_1. Na osnovu izlaznih bitova četiri okvira, pravi se sistem jednačina koji se rešava linerizacijom (proizvodi parova promenljivih smatraju se novim nepoznatima, posle čega sistem postaje linearan). Iz toga dobijaju se početna stanja registara $R1, R2$ i $R3$.
Neka su k_1, k_2, k_3 i k_4 nizovi ključa za okvire f_1, f_2, f_3 i f_4. Svaki k_j je dužine 114 bita. Sa $k_j[i]$ označen je i-ti bit okvira f_j.
U napadu GWG je objašnjeno da se svaki izlazni bit može zapisati kao kvadratna funkcija početnih stanja registara $R1, R2$ i $R3$. Potrebno je napraviti sistem kvadratnih jednačina koje

izražavaju jednakost kvadratnih članova za svaki izlazni bit i stvarne vrednosti poznatog niza ključa. Rešavanje sistema kvadratnih jednačina predstavlja NP kompletan problem. Međutim, u ovom slučaju sistem jednačina je predefinisan, tj. postoji 61 promenljiva i 114 kvadratnih jednačina. Dodavanjem jednačina ostalih okvira f složenost značajno opada, jer sistem postaje sve više i više predefinisan. Pri dodavanju jednačina potrebno je voditi računa da su jednačine nad istim promenljivima, npr. nad promenljivima početnih stanja registara $R1, R2$ i $R3$ za okvir f_1. Kada se spoje jednačina svih četiri okvira, traži se rešenje sistema metodom Gausove eliminacije.
Sistem jednačina se pravi i rešava za svaku od 2^{16} mogućih vrednosti za registar $R4$ za okvir f_1, sve dok se ne pronađe konzistentno rešenje. To rešenje predstavlja početna stanja registara za okvir f_1.
Svaka od tri većinskih funkcija vrši bitske operacije nad jednim od registra. Na taj način, svaki kvadratni (nelinearni) član se sastoji od para promenljivih istog registra. Linearizacijom se nelinarni članovi svode na lineralne uvođenjem nove promenljive, tj. svaki proizvod predstavlja novu promenljivu. U prvom registru postoje 19, u drugom 22, a trećem registru 23 linearne promenljive. Uzimajući u obzir da je na kraju algoritma za postavljanje ključa jedan od bita postavljen na jedinicu, onda u svakom od registara ima jedna promenljiva manje. Stoga, u prvom registru postoje 18 linearnih i $\binom{18}{2} = \frac{18 \cdot 17}{2} = 153$ nelinarnih članova, u drugom 21 linearni i $\binom{21}{2} = \frac{21 \cdot 20}{2} = 210$ i u trećem registru $\binom{22}{2} = \frac{22 \cdot 21}{2} = 231$, tj. ukupno $18 + 153 + 21 + 210 + 22 + 231 = 655$ promenljivih. Dodatno, uključuje se i konstanta 1 kao promenljiva koja označava deo afine transformacije. Na kraju, skup promenljivih se sastoji od 656 promenljivih. Skup ovih promenljivi za okvir f_i je označen sa S_i.
Potrebno je pokazati da se izlazni biti okvira f_2, f_3 i f_4 mogu predstaviti kao linearne kombinacije promenljivih iz skupa S_i u odnosu na okvir f_1. Polazi se od pretpostavke da je poznat sadržaj registra $R4_1$ i da je postavljanje ključa linearno u odnosu na vrednost COUNT (pogledati 6.1.4). Stoga, sa datom (XOR) razlikom vrednosti COUNT, poznata je (XOR) razlika vrednosti svakog registra nakon postavljanja ključa. S obzirom da se zna $R4_1$, kao i (XOR) razlika registara, poznat je i sadržaj registra $R4_2$. Takođe poznata je (XOR) razlika između $R1_1, R2_1, R3_1$ i $R1_2, R2_2, R3_2$.
Svaku promenljivu iz skupa S_2 je potrebno predstaviti kao promenljivu iz skupa S_1. Neka α_1 predstavlja nadovezane vrednosti linearnih promenljivih iz S_1, a g kvadratnu funkciju takvu da je $S_1 = g(\alpha_1)$. Na isti način se nadovežu vrednosti linearnih promenljivih iz skupa S_2 koja je obeležena sa α_2. Vrednosti iz skupa S_2 se takođe dobijaju korišćenjem funkcije g, tj. $S_2 = g(\alpha_2)$. Vrednost α_2 je moguće zapisati kao $\alpha_2 = \alpha_1 \oplus \delta_{1,2}$, gde $\delta_{1,2}$ predstavlja (XOR) razliku vrednosti odgovarajućih registara. S obzirom na osobinu linearnosti i razlike između S_1 i S_2 u odnosu na bite α_1, promenljive iz skupa S_2 mogu da se izraze preko linearnih promenljivih iz skupa S_1. Isti postupak je potrebno ponoviti i za preostala dva okvira, tj. f_3 i f_4.
Na kraju te procedure dobija se sistem kvadratnih jednačina predstavljene samo preko promenljivih iz skupa S_1, koristeći poznate nizove ključa četiri okvira. Sistem jednačina je zapisan kao $S_{R4_1} \cdot S_1 = k$, gde je S_{R4_1} matrica dimenzije 456×656, a k niz dužine 456, tj. nadovezane vredosti nizova ključa za sva četiri okvira. Skup S_{R4_1} zavisi od vrednosti registra $R4_1$, kao i XOR razlike između COUNT vrednosti okvira.
Dolazi se do zaključka da kada se prikupe 656 linearno nezavisnih jednačina, sistem je vrlo lako moguće rešiti koristeći metod Gausove eliminacije. Međutim, ono što predstavlja problem je prikupljanje jednačina, s obzirom da se vrlo često pokreće algoritam za postavljanje ključa, kako bi se dobio novi niz ključa koji se koristi za šifrovanje. BBK su eksperimentisali prilikom izvođenja napada i zaključili da se napad može izvršiti i ako se prikupi svega oko 450 jednačina, bez obzira što prošireni skup ima 656 promenljvih. Za rešavanje sistema su koristili Gausovu metodu eliminacije.

U slučaju da su podaci koje napadač koristi retki, postoje dodatne metode koje mogu da se koriste kako bi se smanjio broj potrebnih jednačina. Npr. gde god da se pojavi neka promenljiva x_i, svaka kvadratna jednačina oblika $x_i \cdot x_j$ može da se uprosti sa 0 ili x_j zavisno od toga da li je $x_i = 0$ ili $x_i = 1$.

Optimizovani napad BBK

Pored ovog napada BBK su napravili i optimizovanu verziju napada sa poznavanjem otvorenog teksta. Ta verzija napada nalazi ključ K_c za nekoliko milisekundi, s tim da koristi ranije izračunate tabele koje se čuvaju u memoriji. U poređenju sa prethodnim napadom, ovaj zahteva malo više podataka.
Osnovna ideja napada je sledeća. Napad se deli u dve faze. Prva faza napada je unapred računanje i sastavljanje sistema linearnih nezavisnih jednačina za sve vrednosti $R4_1$. Za svaki sistem traži se rešenje, tj. za sistem $S_{R4_1} \cdot S_1 = k$ traži se matrica T_{R4_1} koja sadrži rešenje, takva da je $T_{R4_1} \cdot S_{R4_1}$ rešenje nakon Gausove eliminacije za S_{R4_1}. Nakon ove faze, dolazi faza koja se izvodi u realnom vremenu napada, u kojoj se traži $t = T_{R4_1} \cdot k$ za svaku vrednost $R4_1$. Nakon što se pronađe odgovarajuće t, tj. ono za koje je niz ključa konzistentan sa testiranom vrednošću $R4_1$, vrednost $R4_1$ može da se proveri tako što se izračuna ključ i izvrši probno šifrovanje.
Za prvu fazu, računanja unapred je potrebno oko 40 minuta, isto koliko i za prethodno opisani napad. To je vreme računanja i rešavanja svih sistema jednačina za svako $R4_1$. Nakon što pronađe odgovarajuće $R4_1$, provera se njegova ispravnost i računa se K_c. Napad se izvršavao za manje od jedne sekundu na kućnom računaru.

6.1.3 Realizovana varijanta napada

Napad čija je realizacija opisana u poglavlju 9, zasniva se na prethodno opisanom (neoptimizovanom) napadu koji su implementirali BBK. BBK su koristili četiri (bilo koja) okvira i odgovarajuće nizove ključa. Na taj način moguće je napraviti 456 jednačina za promenljive iz skupa S_1 (oznake su iste kao u napadu BBK). Vršenjem velikog broja eksperimenata zaključili su da je dovoljno oko 450 jednačina, gde nakon rešavanja sistema dobijaju rešenja za 656 promenljivih. Za rešavanje sistema koristi se metoda Gausova eliminacije.
U realizovanom napadu se, za razliku od napada BBK, koriste šest proizvoljnih okvira $f = (f_1, \ldots, f_6)$ i njima odgovarajući nizovi ključa $k = (k_1, \ldots, k_6)$. Na ovaj način dobija se sistem od 684 jednačina, sa 656 promenljivih. Ovakav sistem je lakše rešiti jer ima više jednačina od broja promenljvih.
Kada se na prazan registar ubaci ključ K_c, dobija se stanje P. Registar sa stanjem P potrebno je obrnuti 22 puta sa ulaznim bitom 0. To stanje je obeleženo sa P'. Kada se na stanje P ubaci okvir f_1, dobija se novo stanje Q_1. Kada se na stanje P ubaci okvir f_2, dobija se stanje Q_2. Stanja Q_1 i Q_2 predstavljaju linearne kombinacije nepoznatih. Ove linearne kombinacije je moguće dobiti i na sledeći način. Kada se na prazan registar ubaci okvir f_1, dobije se neko stanje registra E_1, a kada se ubaci f_2, dobije se stanje registra E_2. Zbog osobine linearnosti, stanje $Q_1 = P' \oplus E_1$, a $Q_2 = P' \oplus E_2$, (XOR) razlika Q_1 i Q_2 je jednaka $E_1 \oplus E_2$. Iz ovoga se zaključuje da je $Q_2 = Q_1 \oplus E_1 \oplus E_2$. Stanja E_1 i E_2 predstavljaju nizove konstantnih bitova, čija dužina je jednaka dužini registra. Pre napada potrebno je sačuvati stanja registra $(E_1, \ldots, E_6)$ koja su dobijaju nakon ubacivanja okvira $f_1, \ldots, f_6$ na prazne registre. Kao i u napadu BBK, potrebno je napraviti i rešiti sisteme jednačina za svih 2^{16} mogućih vrednosti registra $R4$, za izuzetkom 10. bita koji je uvek 1. Prilikom pravljenja sistema potrebno je izšiti korekciju linearnih kombinacija u odnosu na stanje nakon ubacivanja f_1. To znači da na linearne kombinacije treba dodati vektore konstantnih bitova $(E_2, \ldots, E_6)$ na odgovarajućim pozicijama. Na ovaj način linearne kombinacije postaju afine kombinacije.

Nakon korekcije sledi odbacivanje 99 taktova, a odmah zatim i pravljenje jednačina, po 114 za svaki od 6 okvira. Nakon ovoga, dobija se sistem od 684 jednačina sa 719 promenljvih, koji se rešava metodom Gausove eliminacije. Nakon nalaženje sistema sa konzistentnim rešenjem, traži se ključ K_c pokretanjem algoritma za postavljanje ključa unazad.
Ono što je potrebno uočiti je da će ovaj dobijeni skup promenljvih biti manji, tj. činiće ga skup od 656 promenljvih, zbog postavljanja jedinica u registrima $R1, R2, R3, R4$ na kraju algoritma za postavljanje ključa. Za njih se već zna rešenje i nije potrebno tražiti ga kroz sistem jednačina.

6.1.4 Osobina linearnosti

Neka su promenljive $x = (x_1, \ldots, x_n)^T = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, gde je n dužina linearnog pomeračkog registra. Vektor $y = (y_1, y_2, \ldots, y_n)^T$ je linearna funkcija u odnosu x, ako je $y = Ax$, tj. $\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = A \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, gde je matrica A veličine $n \times n$, čiji elementi su $a_{i,j}$ konstante, $i, j = 1, \ldots, n; a_{i,j} \in \{0, 1\}$.

Neka je registar dužine n, sa povratnim granama, čije su pozicije unapred određene. Nakon svakog takta računa se povratna vrednost registra koja se upisuje u ćeliju sa indeksom 0, koja se sabira najpre sa 64 bita ključa K_c i 22 bita vrednosti f. Novo stanje registra, nakon svakog takta, može da se predstavi kao linearna funkcija prethodnog, što operaciju punjenja ključa K_c i vrednosti f čini linearnom. Neka je početno stanje registra X i na njega se deluje ključem K_c. Novo stanje registra je:

$$Y = BX + CK_c \tag{1}$$

gde su B i C konstante matrice veličine $n \times n$, odnosno $n \times 64$. Vektor X je dužine n, tj. $n \times 1$, a ključ K_c predstavljen kao vektor dužine 64×1, dobija novo stanje registara koje može da se predstavi kao linearna kombinacija na način prikazano u (1).
Ako se na stanje Y deluje vrednošću prvog okvira f_1, dobija se novo stanje Z_1 koje je jednako:

$$Z_1 = DY \oplus Ef_1 \tag{2}$$

Veličina matrice D je ista kao i B, samo što ona predstavlja linearne kombinacije u odnosu na stanje Y, dok je matrica E veličine $n \times 22$ (22-bitna vrednost f). Ako se na stanje Y deluje nekom drugom vrednošću, tj. drugom vrednošću za okvir f_2, dobija se stanje:

$$Z_2 = DY \oplus Ef_2 \tag{3}$$

Ukoliko se saberu (XOR) jednačine (2) i (3), dobija se:

$$Z_1 \oplus Z_2 = Ef_1 \oplus Ef_2 = E(f_1 \oplus f_2) \tag{4}$$

Razlika f_1 i f_2 je fiksna i može se predstaviti kao vektor veličine 22×1. S obzirom da je E matrica veličine $n \times 22$ dobija se da je razlika stanja dva registra nakon primene dva okvira linearna, tj. sabiranjem jednačina (2) i (3) dobija se (XOR) razlika koja je takođe linearna. Na kraju, stanje Z_2 je linearno u odnosu na stanje Z_1, odnosno stanje registra predstavlja linearnu kombinaciju prethodnog stanja kada se na njega deluje sa vrednošću f_2,

$$Z_2 = Z_1 \oplus E(f_1 \oplus f_2) \tag{5}$$

Ovo se dokazuje matematičkom indukcijom, ako se na stanje registra nakon ubacivanja ključa K_c, ubaci vrednost f_1, (XOR) razlika tih stanja je linearna. Za svaku sledeću vrednost f_k, $k = 2, ...,$ razlika stanja registra je linearna (4).
Ova osobina je korisna prilikom postavljanja napada na A5/2, sa i bez poznavanja odgovarajućeg otvorenog teksta.

6.1.5 Algoritam za nalaženje K_c

Nakon što se pronađe konzistentno rešenje za sistem jednačina, koji je opisan u prethodnim napadima, dobijaju se stanja registra pre postavljanja konstantnih bitova. Stanja registra u tom trenutku su linearna u odnosu na f_1 i K_c, što znači da od trenutnih stanja registra može doći do ključa K_c. To se postiže postavljanjem novog sistema jednačina, na osnovu bitova prva tri registra. Pre postavljanja sistema, potrebno je vratiti registre na stanje pre ubacivanja okvira f_1. S obzirom da je poznat rad algoritma za ubacivanje ključa i okvira i pravila za taktovanje registara, na sličan način funkcioniše i ovaj algoritam.
Potrebno je okrenuti sve registre unazad za 22 bita okvira. Takt jednog registra predstavljen je u nastavku ovog pasusa. Najpre se izračuna poslednji upisan bit na poziciji u svakom od registra sa indeksom 0. Neka je taj bit označen sa Z. Bit Z predstavlja XOR funkcija sa argumentima: trenutni bit registra na poziciji 0 i bit okvira f_1 na poziciji $f[21]$. Zatim, svaki registar se pomeri za jedno mesto ulevo. Nakon toga izračunava se vrednost XOR funkcija sa argumentima bitova povratnih registra. Rezultujući bit se sabira sa bitom Z i upisuje u ćeliju registra na najvišoj poziciji, tj. na $n - 1$, gde je n dužina registra.
Nakon okretanja registara $R1, R2, R3, R4$ 22 unazad, dobija se stanje registra koje je dobijeno ubacivanjem ključa K_c na prazne registre. Na osnovu tih stanja gradi se sistema linearnih jednačina.
Promenljive ključa se izražavaju kao linerna kombinacija promenljivih svakog od registra. Iz svakog registra formira se sistem jednačina koji je predstavljen matricom veličine $n \times 64$, gde je n dužina registra. U prvom sistemu forira se siste sa 19 jednačina, u drugom sa 22 jednačine i u trećem 23 jednačine, što na kraju daje sistem veličine 64×64, koji se rešava metodom Gausove eliminacije. Rešenje sistema predstavlja ključ K_c.

6.2 Napad na A5/2 sa poznavanjem samo šifrata

Za razliku od napada sa poznavanjem otvorenog teksta, gde su poznati odgovarajući parovi šifrat-otvoreni tekst, ovde napadač na raspolaganju ima samo šifrat. U ovom delu rada predstavljen je napad sa poznavanjem samo šifrata koji su predložili i implementirali Barkan, Biham i Keller[1].

GSM standard definiše nekoliko tipova komunikacijskih kanala [6]. Svaki kanal ima svoje šeme za kodiranje i premeštanje (engl. interleaving). Fokus u ovom napadu je na kanalu SACCH (engl. Slow Associated Control CHannel)[6].
U sistemima poput GSM, gde se podaci prenose radio talasima, pojava greške je uobičajena, što

ukazuje na potrebu korekcije greške. U GSM komunikaciji postoje više vrsta šema za korekciju greške. Obično se primenjuje kombinacija tri šema. Prva je Fire Code, zatim konvolucini kod i na kraju Interleaving. Za detaljnije opise ovih šema pogledati [7].
Tokom komunikacije, poruka se prvo podvrgne kodiranju za ispravljanje greške, što dovodi do znatnog povećanja dužine poruke. U kanalu SACCH, poruka koju je potrebno kodirati je fiksne dužine od 184 bita. Nakon što se izvrši korekcija greške, poruka je dužine 456 bita, čiji su biti ispremeštani i podeljeni u četiri dela. Ova poruka se šifruje i šalje dalje. Obično, u nekim sigurnijim sistemima, prvo se radi šifrovanje poruke, što znači da se svaki bit od 184 bita poruke najpre šifruje, pa se onda radi dodavanje redudantnih bitova informacija, sa kojim originalna poruka predstavlja poruku dužine 456 bita.
Na osnovu redudantnih bitova moguće je postaviti napad. Potrebno je napraviti sistem linearno nezavisnih jednačina koji se rešava na isti način kao u napadu BBK sa poznavanjem otvorenog teksta. Potrebne su barem dve poruke dužine 456 bita, kako bi se dobilo 450 jednačina za rešavanje. Nakon što se reši sistem za odgovorajuće $R4_1$, traži se K_c pokretanjem algoritma za određivanje ključa.
Ovakva vrsta napada u ovom radu nije detaljno obrađena. Za detaljniji opis napada pogledati[1]. Pored detaljnijeg opisa napada, postoji i alat koji simulira napad sa poznavanjem samo šifrata na A5/2, koji su implementirali Nicolas Paglieri i Olivier Benjamin [8].

7 Aktivni napadi na GSM mrežu

U ovom delu rada predstavljen je napad koji se zasniva na manama prilikom uspostavljanja veze sa GSM mrežom. Napadač na ovaj način može da kompromituje komunikaciju, tako što ima mogućnost da razbije slabiju šifru koja se često nalazi u telefonu žrtve. Pretpostavka je da napadač želi da kompromituje razgovor u mreži koja koristi A5/1 kroz kriptoanalizu A5/2.
Za razliku od napada objašnjenih u 6. poglavlju, gde napadač samo prisluškuje razgovor kako bi dobio informaciju, u aktivnim napadima napadač ima mogućnost kako da šalje, tako i da zahteva od mreže ili žrtve da šalju podatke. Na ovaj način napadač izlaže sebe riziku da bude razotkriven.
Prednosti aktivnih napada su što napadač može da prisluškuje mrežu koja koristi A5/1, dok žrtva koristi A5/2. Vremensko izvršavanje ovog napada ista je kao i za napad na A5/2. Pored ove, postoje još prednosti koje napadaču mogu da olakšaju napad. U većini napada, napadač može da predstavi sebe kroz žrtvin mobilni uređaj, koristeći lažne bazne stanice (pogledati 2. poglavlje za elemente GSM mreže). Na ovaj način, žrtvin mobilni telefon vidi napadača kao mrežu, te napadač može da kontroliše i komanduje žrtvi šta da pošalje, koji kanal da koristi, da koristi samo TDMA okvire koje napadač unapred odredi (ovako može unapred da izračuna tabele i pripremi se za napad sa poznavanjem samo šifrata). Neke od mana koje napadač može da iskoristi prilikom napada su:

- Protokol autentifikacije i usaglašavanja oko ključa između telefona i mreže može da bude izvršen na početku razgovora, na zahtev mreže. Telefon ne može da zatraži autentifikaciju. U tom slučaju ključ K_c ostaje isti kao iz prethodnog razgovora. Na ovaj način mreža može da autentifikuje telefon činjenicom da koristi ključ K_c, a telefon je dokazao da koristi K_c.

- Mreža bira algoritam za šifrovanje. Telefon samo pruža spisak šifri koje podržava u okviru poruke koja se naziva *class-mark*. Moguće je da mreža ne koristi nijedan algoritam za šifrovanje.

- Class-mark poruka nije zaštićena, te napadač može da je promeni.

- Lažne bazne stanice - Tokom autentifikacije samo telefon je autentifikovan od strane mreže, dok obrnuti mehanizam ne postoji.
- Ključ je uvek isti i on zavisi samo od $RAND$, koji se bira od strane mreže, bez obzira koji algoritam (A5/1, A5/2, A5/3) se koristi za šifrovanje.
- Isti $RAND$ je moguće koristiti više puta.

Neki od napada objašnjeni su u daljem tekstu koji se zasnivaju na manama protokola u komunikaciji između telefona i mreže.

7.1 Napad class-mark porukom

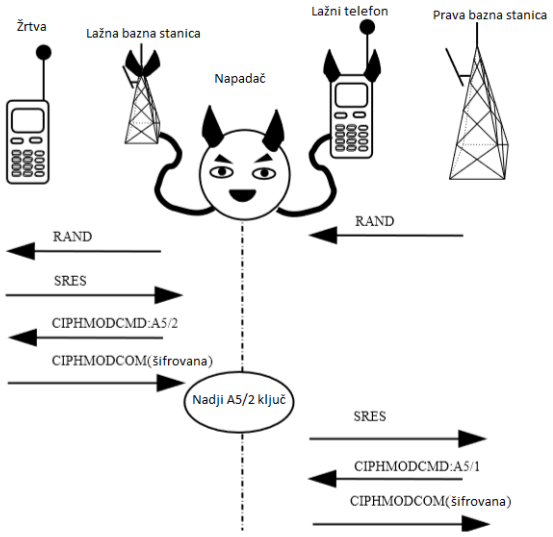
Class-mark predstavlja poruku koja sadrži spisak šifri koje mobilni telefon podržava.
Ovaj napad je jedan od najjednostavnijih napada, u okviru kojeg napadač menja sadržaj class-mark poruke koju telefon šalje mreži na početku razgovora. Napadač menja sadržaj tako da mreža misli da telefon podržava samo A5/2 šifru. Iako mreža češće koristi A5/1, u ovom slučaju će morati da koristi A5/2, s obzirom na poruku koju je napadač poslao. Na taj način napadač može da prisluškje razgovor i kriptoanalizom, slabije, A5/2 šifre dođe do ključa.
Neke mreže, zavisno od implementacije, mogu da odluče da ipak ne koriste A5/2 šifru i na taj način prekinu komunikaciju.

7.2 K_c na osnovu prošlog ili budućeg razgovora

Ključ K_c za šifrovanje je uvek isti, bez obzira koji algoritam za šifrovanje se koristi. Ideja ovog napada se zasniva na korišćenju lažnih baznih stanica, koje navode telefon žrtve da koristi A5/2. Daljim napadom sa poznavanjem samo šifrata, moguće je dobiti K_c.
Ovim napadom moguće je dobiti ključ K_c šifrovanog razgovora koji je zabeležen u prošlosti. Kako se ključ za šifrovanje ne menja kroz nekoliko narednih razgovora, postoji mogućnost da ključ koji se dobije će se koristiti i u budućim razgovorima. Pre napada, napadač snima šifrovan razgovor. U toku napada, napadač započinje vezu sa telefonom žrtve kroz svoju lažnu baznu stanicu. Zatim, napadač pokreće proceduru autentifikacije, koristeći istu vrednost $RAND$ koja se koristila kroz šifrovan razgovor. Telefon vraća $SRES$, koji je jednak $SRES$ snimljenog razgovora. Napadač zahteva od telefona da započne šifrovanje koristeći A5/2. Telefon šalje potvrdu, koja je šifrovana A5/2 i istim ključem K_c koji se koristio kada se snimao razgovor. Na kraju, napadač započinje napad sa poznavanjem samo šifrata da dobije K_c iz šifrovanog odgovora telefona. Napad može biti ponovljen više puta, za sve $RAND$ koji se pojavlju za vreme snimanja razgovora.

7.3 Napad Čovek-u-sredini

Napadač može da prisluškuje razgovor u realnom vremenu koristeći napad *Čovek-u-sredini* (engl. Man-in-the-middle). Napadač koristi lažnu baznu stanicu u komunikaciji sa žrtvom. Na taj način napadač lažno predstavlja mobilni telefon mreži. Kada započne proces autentifikacije, mreža šalje zahtev napadaču, koji napadač prosleđuje do žrtve. Žrtva izračunava $SRES$ i vraća ga napadaču, koji čuva, ne vraćajući ga mreži. Zatim napadač zahteva od žrtve da započne šifrovanje koristeći algoritam A5/2. Žrtva započinje šifrovanje i šalje šifrovanu potvrdu. Napadač u tom trenutku koristi neki od napada na A5/2, npr. napad zasnovan na poznavanju samo šifrata i nalazi ključ za manje od jedne sekunde. Tek nakon toga, napadač vraća $SRES$ mreži. Sada kada je napadač lažno autentifikovan na mrežu, mreža traži od napadača da započne šifrovanje koristeći A5/1. Napadač tada zna K_c i šalje odgovor šifrovan algoritmom A5/1. Od ovog trenutka pa na dalje, mreža vidi napadača kao mobilni telefon i napadač može da nastavi komunikaciju.



Slika 10: Napad Čovek u sredini

Ovde se postavlja pitanje, kako mreža ne može da primeti napad, s obzirom da postoji blago kašnjenje prilikom procedure autentifikacije. Ipak, ovo i nije tako lako otkriti, jer po standardu GSM-a dozvoljeno je 12 sekundi da telefon izvrši autentfikaciju, tj. izračunavanje $SRES$-a i vraćanje odgovora mreži, a napad traje čitavu jednu sekundu za dobijanje K_c.
Nakon što napadač zatraži od žrtve da započne šifrovanje koristeći A5/2, žrtva mora da ogovori sa šifrovanom CIPHMODCOM (engl. Cipher model complete) porukom, koja predstavlja potvrdu da je šifrovanje započeto. Ta poruka je dužine 456 bita. To je dovoljno za napad za poznatim otvorenim tekstom, ali nedovoljno za napad za poznavanjem samo šifrata (za ovaj napad potrebne su dve takve poruke). Da bi napadač potvrdio CIPHMODCOM poruku, potreban mu je K_c. Zato, on može da sačeka da mehanizam na telefonu ponovo pošalje šifovanu CIPHMODCOM poruku, što je dovoljno da izvrši napad za poznavanjem samo šifrata. Sa prethodnom pripremom, napad se može izvršiti nad CIPHMODCOM sa napadom sa poznatim otvorenim tekstom.

8 Scenariji mogućih napada

Napadi koji se primenjuju na algoritam A5/2 mogu biti korišćeni u više scenarija. U nastavku teksta opisani su neki od njih: prisluškivanje, preuzimanje, izmena (SMS) poruke sa podacima i krađa.

8.1 Scenario prisluškivanja

Komunikacija šifrovana u okviru GSM mreže, može biti dešifrovana i prisluškivana od strane napadača, ukoliko napadač ima ključ za šifrovanje. Ne samo da napadač može prisluškivati razgovor, već može prisluškivati razmenu podataka, kao i SMS poruka. Prisluškivanje u realnom vremenu, može biti izvedeno pomoću pasivnog napada. Pasivnim napadom, napadač ne može da razmenjuje podatke sa mrežom ili žrtvom ili da na bilo koji način utiče na promenu komunikacije, već pokušava da probije sistem samo na osnovu posmatranih podataka, tj. šifrata. Ovde napadač

može da primeni napade sa poznatim otvorenim tekstom i njegovim odgovorajućim šifratom, zavisno od toga šta je prikupio od podataka. Na mrežama koje koriste šifrovanje koje nije A5/2, neophodan je napad čovek u sredini (pogledati 7.3). U drugom mogućem napadu, prisluškivanjem sistema koji koriste šifru A5/1, napadač snima razgovor. Zatim, koristi lažne bazne stanice kako bi napadao telefon žrtve i preuzima ključ K_c. Ovo zahteva aktivno učešće u napadu. Jednom kada napadač ima ključ, može da dešifruje razgovor.

8.2 Scenario preuzimanja

Dok GSM mreža može da autentifikuje korisnika neposredno pre pokretanja razgovora, šifrovanje je sredstvo za sprečavanje lažnog predstavljanja u kasnijim fazama razgovora. Osnovna pretpostavka je da napadač nema K_c i da na taj način ne može obavljati šifrovane razgovore. Koristeći pasivne napade, napadač može otkriti ključ. Kada napadač ima ključ za šifrovanje, on može da izbaci žrtvu iz razgovora (npr. prenosom jačeg signala) i da predstavi sebe kao žrtvu na drugoj strani koristeći preuzeti ključ. Dakle, preuzimanje razgovora nakon autentifikacije je moguće.

8.3 Scenario izmene poruka sa podacima

Jednom kada je poziv preuzet, napadač odlučuje o tome šta želi da uradi sa sadržajem, uključujući sadržaj SMS poruka, s obzirom da su i one šifrovane istim ključem kao za razgovor. Napadač može prisluškivati sadržaj poruke koje su poslate od strane žrtve ili da ih prima. Napadač može zaustaviti poruku koja se šalje ili čak da pošalje svoju SMS poruku, čime narušava integritet GSM saobraćaja.

8.4 Scenario krađe - dinamičko kloniranje

Za GSM se verovalo da je siguran sistem protiv krađe zbog procedure autentifikacije algoritmima A3 i A8 (pogledati 3. poglavlje). Međutim, kada mreža zatraži autentifikaciju, napadač izvrši napad kojim koristi žrtvin telefon kao sredstvo za dobijanje $SRES$-a i K_c za dato $RAND$. Napadač inicira odlazni poziv ka mreži i paralelnu radio vezu ka žrtvi napada. Kada mreža zatraži od napadača autentifikaciju, napadač traži od žrtve autentifikaciju i prenosi dobijenu proveru identiteta nazad na mrežu. Nakon što uspe da dobije K_c, napadač može da zatvori vezu sa žrtvom i da nastavi vezu sa mrežom. Mreža ovo veoma teško uočava kao napad, jer ceo postupak autentifikacije izgleda kao normalna procedura. Žrtvin telefon neće zvoniti pa samim tim žrtva i ne zna da je žrtva. Ovim postupkom napadač može da napravi odlazne pozive na račun žrtve.

9 Programska realizacija napada sa poznatim otvorenim tekstom

Napad sa poznatm otvorenim tekstom, opisan u 6.1.3, realizovan je u programu A52Attack. Za razvoj aplikacije korišćen je programski jezik C++ i aplikacioni interfejs Qt Creator, verzija 4.0.0.

9.1 Organizacija programa

Program je podeljen u dve celine: A52KeystreamGeneration i A52Attack. Prvi program u osnovi se koristi za generisanje niza ključa kojim se šifruju informacije, kao i za pripremu

odgovarajućih podataka za sledeći program (A52Attack). Dakle, ovi programi se izvršavaju nezavisno.
U podsekciji 9.3 A52KeystremaGeneration objašnjena je struktura programa za generisanje niza ključa, a potom u podsekciji 9.4 A52Attack i struktura programa za napad na A5/2 sa poznatim otvorenim tekstom. U nastavku teksta, objašnjen je jos jedan program koji služi kao zajednički alat za ostale programe.

9.2 A52Utils

A52Utils je program koji ima jednu klasu A52Utils.cpp i njoj odgovarajuće zaglavlje A52Utils.h i koja sadrži sve one metode koje koriste programi A52KeystreamGeneration i A52Attack. Ovaj program napravljen je i preveden kao dinamička biblioteka (.dll).

```
typedef unsigned char byte;
typedef uint32_t word;
typedef uint64_t qword;
typedef vector<vector<byte>> matrix;

enum ClockDirection { LEFT, RIGHT };

class A52UTILSSHARED_EXPORT A52Utils
{
public:
    A52Utils();

    static void printBits( int sizeOfRegister, word R );

    static void printContentOfVector( const vector<byte> &V );

    static void printContentOfVector( const vector<word> &V );

    static void printContentOfVector( const vector<qword> & V );

    static void printRegisters(word R1, word R2, word R3, word R4);

    static void printMatrix(int n, int m, matrix &M);

    static void printMatrix(int n, int m, matrix &M, vector<byte> & rightSide);

    static word parity(word x);

    static word clockOneRegister(word reg, word mask, word taps, int size,
word framebit, ClockDirection direction);

    static word clockingUnit( word w1, word w2, word w3 );

    static void xorTwoVectors(vector<byte> &V1, vector<byte> &V2, int len);

    static word convertVectorToWord(const vector<byte> &v);
```

```
    static vector<byte> concatVectors( const vector<byte> &R1, const vector<byte> &R2,
const vector<byte> &R3, int withLastElement );

    static void concatMatrixs( const matrix &LFSR1, const matrix &LFSR2,
const matrix &LFSR3, matrix &LSE );

    static int checkLSE(const vector<byte> &testVector, const matrix &LSE,
const vector<byte> &rightSide);
};
```

Tipovi podaka koji se koriste u programima za generisanje niza ključa i napada su takođe definisane u ovoj klasi. Tip **byte** je definisana kao unsigned char i koristi se za predstavljanje jednog bita niza ključa.
Registri $R1, R2, R3, R4$ predstavljeni su tipom podataka **word**. Operacije nad registrima su bitske. S obzirom da je najveća dužina registra 23, onda je dovoljno koristiti prvi tip podataka koji je veći ili jednak od 23, a to je unsigned, koji je dužine 32 bita. Najniža pozicija registra je skroz desno, tj. 2^0, dok je najveća skroz levo, tj. 2^{n-1}. Sadržaj registra određen je bitskom operacijom & registra i njegove odgovarajuće heksadekadne maske. Ove maske su takođe definisane u ovom programu (R1MASK, R2MASK, R3MASK, R4MASK).
Ključ koji se koristi za šifrovanje je dužine 64 bita. Stoga, koristi se tip podataka unsigned long long. Definisan je tipom podataka **qword**.
Matrica je predstavljena vektorom vektora nad tipom byte. Radi lakšeg zapisa definisana je kao tip podataka **matrix**. Matrica se koristi kako bi se predstavio sistem jednačina.
Nabrojani tip (enum) **ClockDirection** ima vrednosti LEFT i RIGHT. Vrednosti predstavljaju informaciju u koju stranu se pomeraju registri. U algoritmu za generisanje niza ključa registri se pomeraju ulevo (LEFT), dok je u algoritmu za nalaženje ključa pomeranje registra u suprotnom smeru, samim tim i vrednost nabrojanog tipa je RIGHT.
Neke bitne metode su objašnjene u nastavku. Ostale metode mogu se videti u programu u zaglavlju klase, gde je detaljno opisan rad svake metode.

Metoda parity

Metoda parity se koristi za određivanje parnosti broja jedinica u registru, tj. suma bitova po modulu 2 (operacija XOR). Na primer, ako je sadržaj registra 1011, krajnji rezultat biće 1, jer na pozicijama $2^0, 2^1$ i 2^3 se nalaze jedinice, a njihova suma (XOR) je 1. Kako je i poslednji bit 1, operacija sa bitskim & biće 1, što predstavlja povratnu vrednost metode.
Ova metoda se koristi kako bi se u trenutku pre taktovanja odredila suma povratnih grana registra. Operacija & registara i odgovarajuće heksadekadne maske (R1TAPS, R2TAPS, R3TAPS i R4TAPS), daje vrednost bita na pozicijama za povratne grane.

Metoda clockOneRegister

Metoda se koristi taktovanje jednog registra. Takt podrazumeva pomeranje jednog registra u jednu od strana koja je određena poslednjim parametrom metode direction.
Postupak za taktovanje registra je sledeći:

- Kada direction ima vrednost LEFT, najpre se izračunava vrednost operacijom bitskog & nad registrom i odgovarajuće maske za povratne grane. Zatim se registar pomera jedno mesto ulevo, kako bi se oslobodila pozicija sa indeksom 0, tj. 2^0. Na toj poziciji se upisuje prethodno izračunata vrednost

- Kada direction ima vrednost RIGHT, najpre se uzima vrednost koja je u registru na poziciji 2^0 i vrednost koja je pre takta registra bila izračunata tako što su biti XOR-ovani sa povratnih grana. Te dve vrednosti se saberu (XOR). Nakon toga, registar se pomeri jedno mesto udesno, oslobađajući mesto na najvišoj poziciji u registru. Na toj poziciji upisuje se prethodna izračunata vrednost. Na vrednost bita na poziciji 2^0 utiče i vrednost bita okvira, ako je taj bit jednak 1.

Prvi deo metode se koristi se u algoritmu za postavljanje ključa, a drugi u algoritmu za nalaženje ključa na osnovu stanja registra.

Metoda clockingUnit

Metoda clockingUnit određuje izlaz jedinice za taktovanje. Ulazni parametri su biti registra R4 na pozicijama 3, 7 i 10. Kada su barem dva bita jednaka jedinici, onda je povratna vrednost metode 1, u suprotnom je 0. Na ovaj način uvek se taktuju dva ili tri registara. Ova metoda se koristi i za većinsku funkciju koja predstavlja jedan od izlaznih bitova koji ulaze u generisanje niza ključa. Ulazni parametri u tom slučaju su vrednosti bita nekih ćelija registra, koje su unapred određene.

9.3 A52KeystreamGeneration

U ovom programu predstavljen je rad šifre A5/2 i generisanje niza ključa. Pored toga, vrši se priprema potrebnih podataka za napad. Sve metode smeštene su u klasi Keygen.cpp. Ovaj program koristi biblioteku a52utils.h (9.2).

9.3.1 Klasa Keygen

Niz ključa se generiše algoritmom opisanim u poglavlju 4 (Slika 5 i 7).

```
class Keygen
{
public:
    Keygen();

    void keysetup (qword key, word frame );

    vector<byte> runForKeystream( qword key, word frame );

    void frameSetup( vector<word> frame );

    void produceKnownKeystreams( qword key, vector<word> frames );

    void makeFinalVectorOfRegisters( qword, word frame );

    void produceCorrectR4( qword key, word frame );

    void testKeysetup();

    void testLinearity( qword key, word frame );

private:
```

```
void initRegister();

void insertKey( qword key );

void insertFrame( word frame );

void setConstants();

void throwAway99Bits();

vector<byte> produceKeystream();

void clock( int clockAll );

byte getbit();

vector<byte> makeOneFinalVector(word R, unsigned size);
```

Kao i u prethodnom programu, i ovde su objašnjene samo bitne metode za rad algoritma za generisanje niza ključa.

Metoda clock

S obzirom da je definisana metoda koja taktuje jedan registar, taktovanje svih ili neki od tri registra se odvija u metodi clock. Ako je ulazni parametar clockAll = 1, onda se taktuju svi registri, a ako je jednako 0, taktuju se oni registri na osnovu pravila implementiranog u metodi clockingUnit u programu A52Utils. Na primer, ako je izlaz metode clockingUnit = 0, a biti na pozicijama 3, 7 i 10 registra $R4$ jednaki, redom 0, 0 i 1, taktovaće e registri $R2$ i $R3$, dok $R1$ neće. Nezavisno od prva tri registra, registar $R4$ se uvek taktuje.

Metoda keysetup

Metoda keysetup je podeljna u četiri koraka. Prvi je definisan u metodi initRegister(), gde se registri $R1, R2, R3, R4$ postavljaju na nule. Drugi korak je metoda insertKey(), gde se na prazan registar ubacuju biti 64-bitnog privatnog ključa K_c. Ubacivanje ključa podrazumeva taktovanje svakog registra, sabiranje bita ključa sa povratnom vrednošću registra i upisivanje rezultata na poziciji sa indeksom 0. Naredni korak je metoda insertFrame(), koja na isti način upisuje bitove okvira f (vrednost COUNT - poglavlje 5), kao i bitove ključa, s tom razlikom da je dužina okvira 22 bita. Poslednji korak je postavljanje konstantnih bitova na pozicijama određene heksadekadnim maskama R1CONSTBIT, R2CONSTBIT, R3CONSTBIT i R4CONSTBIT. Informacije o ključu i okvirima se čitaju iz datoteke *key_and_frames.txt* koja se nalazi u direktorijumu *myInputFiles*.

Metoda runForKeystream

Metoda runForKeystream koristi se za generisanje niza ključa. Nakon ubacivanja ključa, okvira i postavljanje konstanti, pokreće generisanje niza ključa. Pre generisanja izlaza, 99 bitova se odbacuju (metoda throwAway99Bits). To znači da se registri samo taktuju, a izlaz se odbacuje i ne koristi u generisanju niza ključa. Nakon ovoga generiše izlaz od 114 bitova (metoda pro-

duceKeystream).[1] Jedan izlazni bit se dobija metodom getbit() koja iz registara uzima bitove sa najviših pozicija i iz većinske funkcije svakog od registara. Sve vrednosti se XOR-uju, a rezultujući bit predstavlja jedan od 114(228) bitova izlaza.

Metoda frameSetup

Metoda frameSetup se koristi za pripremanje podataka, na osnovu kojih se gradi sistem jednačina. Sistem se gradi na osnovu osobine linearnosti (6.1.4). S obzirom da su poznati okviri i nizovi ključa generisani u tim okvirima, potrebno je da napraviti stanja registra tako što se na prazne registre ubace šest okvira (6.1.3). Stanja se zapisuju u datoteku *stages_after_frames.txt*, za svaki okvir po četiri stanja registra. Nakon kreiranja, datoteka se kopira u direktorijum *myOutputFiles*.

Metoda produceKnownKeystreams

Metoda produceKnownKeystreams generiše nizove ključa za svih šest okvira. S obzirom da ovaj program predstavlja simulaciju napada na A5/2, ovi nizovi ključa se generišu u okviru ovog programa. U realnim napadima najčešće se do ovih informacija dolazi prisluškivanjem.
Nakon generisanja nizova ključa, informacija u vidu nula i jedinica se zapisuje u datoteku *known_keystreams.txt*. Ovi nizovi predstavljaju desnu strane jednakosti u sistemu jednačina. Kao i u prethodnoj metodi, datoteka *known_keystreams.txt* se kopira u direktorijum *myOutputFiles*.

Pored ovih metoda, implementirane su i metode za proveru ispravnosti generisanja niza ključa: testKeysetyp i testiLinearity(6.1.4).

9.4 A52Attack

Program A52Attack čine klase Attack.cpp, KeygenReverse.cpp, LFSR.cpp i LSESolver.cpp i njihova odgovarajuća zaglavlja. Kao i prethodni program i ovaj koristi metode iz biblioteke A52Utils (a52utils.h).
Nakon pripremljenih ulaznih podataka može se započeti napad na A5/2. Pre klase u kojoj je implementiran napad, u nastavku teksta je objašnjena klasa LFSR.

9.4.1 Klasa LFSR

Klasa LFSR sadrži vektor koji čuva linearne (afine) kombinacije bita početog stanja. Ovim vektorom izražava se stanje svakog registra preko linearnih kombinacija. Svaki vektor je dužine odgovarajućeg registra koji on predstavlja. Na najnižoj poziciji u vektoru (pozicija 0) zapisana je linearna kombinacija 2^0, koja predstavlja promenljivu x_0, u sledećoj 2^1, tj. promenljiva x_1, sve do 2^{n-1}, tj. promenljiva x_{n-1}, gde je n dužina registra.

```
class LFSR
{
public:
    LFSR();
    LFSR(const unsigned numberOfRegister, const int size, const word mask,
const vector<int> positionsOfFeedbackTaps, const vector<int> positionsOfOutTaps);
```

[1] Generiše se 228 bitova, s tim da prva polovina od 114 bita se koristi kao niz ključa za šifrovanje podataka na vezi od mreže ka telefonu, a druga polovina od 114 bita za šifrovanje podataka na vezi od telefona ka mreži. Za napad je dovoljna jedna od strana.

```
    vector<byte> & getFinalVector();

    void init();

    void initFinalVector();

    void clock();

    void correct(word S1_R, word Sk_R);

    void putConstant();

    void produceEquations();

private:
    /* Fields */
    unsigned numberOfRegister;
    int size;
    word mask;
    vector<word> reg;
    vector<int> positionsOfFeedbackTaps;
    vector<int> positionsOfOutTaps;
    vector<byte> finalVector;

    /* Methods */
    word calculateNewAffineCombination();

    void clockOneAffineCombination(word LC);

    void multiplyTwoAffineCombination(word L1, word L2);

    int calculatePositionForNewValue( int p, int q );

    unsigned getSizeOfEquation();
};
```

Pored registra, informacije bitne za LFSR prosleđene su kroz konstruktor. To je broj registra **numberOfRegister** koji služi samo da označi koji registar je u pitanju. Zatim pozicije povratnih grana koje su definisane kao vektor koji čuva te pozicije. Na isti način se čuvaju i pozicije izlaznih grana. Takođe ova klasa sadrži i vektor **finalVector**, koji predstavlja jednu jednačinu registra sa linearnim i kvadratnim članovima.

Metoda init

Metoda init, inicijalizuje promenljivu **reg**, tipa vektor<word>, tako što na najnižoj poziciji upisuje 2^0, a na najvišoj 2^{n-1}, gde je n dužina registra.

Metoda initFinalVector

Metoda initFinalVector služi da inicijalizuje vektor **finalVector** koji predstavlja jednačinu sa linearnim i kvadratnim članovima. Dužina ovog registra određena je metodom getSizeOfEquation() koja se računa kao $n + \binom{n}{2} + 1$, gde je n dužina registra. Prvi deo vektora dužine n je linearni deo jednačine, a ostatak predstavlja kvadradne članove jednačine. Na kraju je dodata promenljiva koja označava konstantu zbog afinih tranformacija.

Metoda clock

Metoda clock taktuje registar, tj. pomera sve udesno, a prethodnu izračunatu povratnu vrednost upisuje na nultu poziciju u vektoru. Računanje povratne vrednosti vrši se sabiranjem onih afinih kombinacija, koji su određeni pozicijama u vektoru **positionsOfFeedbackTaps**. Ovo izračunanje se izršava u metodi calculateNewAffineCombination().

Metoda correct

Korekcija prilikom postavljanja sistema jednačina se vrši u odnosu na stanje registra na koje je ubačen okvir f_1. To stanje je jedna reč (tip word) dužine $n-1$, gde je n dužina bitskog registra. Neka je to stanje označeno sa X. Sabiranje X sa vektorom afinim kombinacijama se vrši na sledeći način: bit na poziciji 2^0 stanja X se XOR-uju sa afinom kombinacijom u vektoru na poziciji 0, bit na poziciji 2^1 sa afinom kombinacijom na poziciji 1, itd. Dodavanje se vrši tako što se bit stanja X pomeri ulevo za dužinu registra i XOR-uje sa bitom koji predstavlja konstantu u afinoj kombinaciji.

Metoda putConstants

Metoda putConstants na pozicijama predviđenim za konstante odgovarajućih registara upisuje jedinice na najvišim pozicijama afine kombinacije. To znači:

```
LFRS1.reg[15] = 1 << 19;
LFRS2.reg[16] = 1 << 22;
LFRS3.reg[18] = 1 << 23;
```

Metoda produceEquations

Metoda produceEquations je jedna od najznačajnih metoda u ovoj klasi koja simulira rad bitskog pomeračkog registra. Jedna jednačina napravljena je od registara $R1, R2$ i $R3$. Iz prvog registra proizvede se jednačina sa 191 promenljvom, iz drugog jednačina sa 254 promenljive i iz trećeg, jednačina sa 277 promenljivih. Ukupan broj promenljivih je 722, od kojih su tri konstane koje se kasnije prebacuju sa desne strane sistema. Dosta promenljivih iz ovih jednačina biće nula u celom sistemu jednačina, zbog postavljanja konstanti u prethodnoj metodi (putConstants).
Izlaz iz vektora proizvodi jednu jednačinu koja je sačinjena na osnovu promenljivih tog registra. Na početku u vektoru **finalVector**, koji predstavlja jednu jednačinu registra, su upisane sve nule. Na izlaz iz vektora utiče afina kombinacija na najvišoj poziciji u vektoru i izlaz većinske funkcije. To znači da je potrebno pokupiti sve jedinice iz afine kombinacije sa poslednje pozicije u vektoru i dodati (XOR) ih na istim pozicijama u **finalVector**. Ovo je implementirano u metodi clockOneAffineCombination(). Izlaz većinske funkcije podrazumeva množenje afinih kombinacija što dovodi do pojavljivanja kvadratnih članova. Množenje je implementirano u metodi multiplyTwoAffineCombination(). Dve afine kombinacije se množe tako što se pokupe pozicije na kojim se nalaze jedinice i jedne i druge afine kombinacije. Na osnovu tih pozicija računa se pozicija u

finalVector metodom calculatePositionForNewValue, čiji su ulazni parametri pozicije jedinica dveju kombinacija. Na izračunatoj poziciji u **finalVector** dodaje (XOR) se jedinica.

9.4.2 Klasa LSESolver

Klasa LSESolver sadrži dve metode za rešavanje sistema jednačina: solveForKey i gauss. Metoda **solveForKey** funkcioniše samo ako je kvadratna matrica ili ako je broj kolona manji od broja redova, dok je druga metoda **gauss** prilagođena nalaženju konzistentog rešenja za stanja registra na osnovu kojih se dalje traži ključ. I jedna i druga koriste Gausov metod eliminacije. Pored ovih metoda implementirana je i provera koja u fazi nakon eliminacije proverava da li sistem zadovoljava rešenje. Sistem nema rešenje ukoliko su sa leve strane jednakosti sve promenljive jednake nuli, a desna strana jednaka jedinici.

```
// Struktura koja čuva poziciju jedinice na dijagonali
typedef struct {
int row;
int col;
} position_of_ones;

// Struktura koja čuva rešenja pronađena rešavanjem sistema jednačina
typedef struct {
word R1;
word R2;
word R3;
} solution;

class LSESolver
{
public:
    LSESolver();

    static vector<byte> solveForKey(int n, int m, matrix &M,
vector<byte> & rightSideOfSystem);
    static solution gauss(int n, int m, matrix &M, vector<byte> &rightSide);
    static int isThereSolution(int n, int m, const matrix & M,
vector<byte> &rightSide);
};
```

9.4.3 Klasa Attack

U ovoj klasi započinje se napad na algoritam A5/2 na osnovu prikupljenih i analiziranih podataka.

```
class Attack
{
public:
    Attack();

    void init();
```

```
    void clock(int produceEquations, int frame, int row , vector<byte> &rightSide);

    void run(word correctR4);
private:
    /* Fields */
    LFSR LFSR1, LFSR2, LFSR3;
    word R4;
    matrix LSE;
    vector<byte> rightSide;
    vector<word> S1_to_S6_for_R1;
    vector<word> S1_to_S6_for_R2;
    vector<word> S1_to_S6_for_R3;
    vector<word> S1_to_S6_for_R4;
    vector<word> frames;
    vector<byte> testVector;

    /* Methods */
    word clockone(word reg, word mask, word taps);
    void initStages();
    void initKeystreams();
    void initFrames();
    void initTestVector();
};
```

Promenljive **LFSR1**, **LFSR2** i **LFSR3** su klasnog tipa LFSR. One čuvaju registre afinih kombinacija. Registar **R4** je bitski registar (word), s obzirom da on utiče samo na taktovanje ostalih registra. Matrica **LSE** definisana je tipom matrix i ona predstavlja sistem jednačina kojim se traži rešenje za stanja registra $R1, R2$ i $R3$ pre 99 taktova za odbacivanje izlaza. Stanja registra koja se dobijaju kada se okviri ubace na prazne registre čuvaju se u vektorima, za svaki registar po šest stanja. Desna strana čuva se u posebnom vektoru **rightSide**. U taj vektor su upisani šifrat koji su dobijeni prisluškivanjem, ili u ovom slučaju generisanjem niza ključa korišćenjem programa A52KeystreamGeneration. Poznati okviri se čuvaju u vektoru **frames**.

Metoda init

Metoda init, inicijalizuje potrebne podatke za napad. To su šifrati, odnosno nizovi ključa i stanja registra nakon ubacivanja okvira na prazne registre. Podaci se čitaju iz spoljnih datoteka *known_keystreams.txt* i *stages_after_frames.txt* smeštene u direktorijum *myOutputFiles*.

Metoda run

Metoda run započinje napad i traženje ključa K_c. Napad podrazumeva rešavanje sistema jednačina za sva moguća stanja za registar R4, tj. za 2^{16} mogućnosti, jer je 10. bit R4 uvek postavljen na 1.
Sistem se gradi na osnovu šest stanja svakog od registra koji su pročitani iz datoteke *stages_after_frames.txt*. Za svako stanje proizvodi se po 114 jednačina, što na kraju čini sistem jednačina od 684 jednačina i 719 promenljivih (matrica 684×719). Pre formiranja jednačina inicijalizuju se klase koje čuvaju vektore afinih kombinacije, zatim se vrši korekcija svakog registra u odnosu na okvir f_1. Nakon toga, postavljaju se konstante u vektorima afinih kombinacija i u registru R4. Nakon ovih koraka prelazi se na formiranje jednačina. Kao i u radu bitskog registra,

potrebno je prvo odbaciti 99 izlaza, pa tek onda napraviti 114 jednačina. Obe ove operacije podrazumevaju taktovanje registra. Taktovanje zavisi od stanja registra R4 i jedinice za taktovanje. Pri formiranju jednačina, konstantni član iz svake jednačine se prebacuje na desnu stranu i sabira sa odgovarajućim bitom šifrata. Nakon toga, formirane jednačine svakog od registra se spajaju u jednu jednačinu koja ima 719 promenljivih: 190 promenljivi od prvog registra, 253 od drugog i 276 promenljivi od trećeg registra.
Nakon formiranja sistema potrebno je isti i rešiti. Rešavanje sistema se postiže metodom **gauss** iz klase LSESolver. Za sistem koji ima rešenje pokušava se nalaženje ključa pokretanjem algoritma za nalaženje ključa unazad. Nalaženje ključa implementirano je u sledećoj klasi KeygenReverse.

9.4.4 Klasa KeygenReverse

Nakon izšrene analize nizova ključa, postavljanja sistema i rešavanja istog, potrebno je da se od trenutnih stanja registara $R1, R2, R3, R4$ dođe do ključa K_c. U klasi KeygenReverse implementirane su metode koje nalaze ključ K_c.

```
class KeygenReverse
{
public:
    KeygenReverse();

    qword findKey(word R1, word R2, word R3, word R4, word frame);

private:
    void clock( int clockAll, word framebit );

    vector<byte> calculateNewEquation(const vector<unsigned> positionsOfFeedbackTaps,
matrix &LFSR);

    void shiftRowsInMatrix(matrix &LFSR);

    void clockForKey(matrix &LFSR1,matrix &LFSR2,matrix &LFSR3);

    vector<byte> solveKey();
};
```

Metoda findKey

U okviru metode findKey odvijaju se operacije za nalaženje ključa. Ulazni parametri su četiri stanja registra i okvir f_1 za koji je generisan niz ključa.
Prvo je potrebno vratiti stanje registra pre ubacivanja okvira f_1, tj. nakon ubacivanja ključa K_c. Vraćanje na prethodno stanje se postiže metodom clock, koja taktuje registar unazad, tj. udesno, metodom clockOneRegister koja je definisana i implementirana u programu A52Utils. Nakon toga potrebno je napraviti sistem jednačina za trenutno stanje registara, čije rešenje predstavlja ključ K_c.
S obzirom da je ukupan broj bitova registara R1, R2 i R3 jednak 64, dovoljno je izraziti svaki bit iz odgovarajućeg registra jednačinom, koja ima 64 promenljive.
Svaki registar je predstavljen matricom veličine $n \times 64$, gde je n dužina registra. Elementi matrice su iz skupa $\{0, 1\}$. U iteraciji i, gde je $i = 0, \ldots, 63$, ubacuje se bit ključa na i-toj poziciji u vektoru koji predstavlja ključ. Bit koji se ubacuje predstavlja promenljivu ključa $K[i]$. Nakon

toga, određuje se nova jednačina koja se računa kada se odgovarajući redovi, određeni povratnim granama, saberu po pozicijama. Novonastala jednačina se upisuje u nulti red matrice.
Nakon 64 takta, matrice se spajaju u jednu matricu (64×64) za koju se traži rešenje. Za rešavanje sistema koristi se metoda solveForKey iz klase LSESolver.
Na kraju može izvrši provera ispravnosti ključa. To se postiže tako što se algoritam za nalaženje ključa vrati unazad za vrednost koja se dobila rešavanjem sistema, na isti način kao i za okvir f_1, Rezultat te operacije dovodi do toga da je sadržaj svih registra popunjen nulama. Ukoliko se takvo stanje ne dobije, znači da je negde došlo do greške u formiranju ili rešavanju sistema.

9.5 Upotreba i testiranje programa

U ovom delu rada opisano je korišćenje programa za pripremu napada i samog napada na A5/2. Oba programa se izvršavaju u okviru interfejsa komandne linije.

9.5.1 Organizacija datoteka na sistemu

Glavni direktorijum je A52. Programi su smešteni u svoje direktorijume, koji su smešteni u direktorijum A52. Pored toga, u direktorijumu A52 se nalazi i direktorijum *myInputFiles* sa datotekom *key_and_frame.txt* koja predstavlja ulazne testne podatke. U okviru direktorijuma A52 nalazi se i direktorijum *myOutputFiles* u koji program A52KeystreamGeneration upisuje datoteke u toku izvršavanja. Takođe, u direktorijumu A52 nalazi se i direktorijum *myFilesForAnalyze*, u kom su smeštene datoteke za eventualnu kasniju analizu programa. Za sada tu je samo datoteka koja sadrži rešenu matricu sistema iz koje se nalazi stanje registara nakon ubacivanja okvira f_1.
U istom direktorijumu se nalazi i direktorijum *myLibs*, koji sadrži dinamičku biblioteku A52Utils programa.

9.5.2 Prevođenje programa

Aplikacioni interfejs Qt Creator sa ugrađenim kompajlerom može se besplatno preuzeti sa zvanične Web stranice https://www.qt.io/download/#qt-creator. U izradi ovog programa korišćena je verzija Qt Creator 4.0.0, bazirana na verziji Qt 5.6.0, sa ugrađenim kompajlerom MinGW 4.9.2 32bit.

Nakon instalacije Qt Creator-a, program se može otvoriti na dva načina:

- Dvostrukim levim klikom na datoteku sa ekstenzijom .pro
- Izborom opcije Open File or Project iz padajuće kartice File, nakon čega se otvara prozor u kome je potrebno pronaći datoteku sa ekstenzijom .pro.

Qt za komercijalnu upotrebu podrazumevano koristi Debug mod za pokretanje aplikacija. Usled toga, i ova verzija zbog podešavanja u .pro datoteci radi samo u Debug modu.
Pre pokretanja programa, potrebno je uvezati dinamičku biblioteku A52Utils u oba programa. Nakon ubacivanja programa u Qt, on se podrazumevano postavlja za aktivan projekat, čime se postiže izmena podešavanja za taj program. Sa leve strane treba izabrati opciju Projects, u okviru kartice Build & Run i izabrati opciju Run pri vrhu ove stranice, čime se otvaraju podešavanja za pokretanje programa. U okviru dela Run Environment, sa desne strane potrebno je izabrati opciju Details, pri čemu se otvara potprozor za izmenu varijabli koje utiču na pokretanje programa u ovom okruženju. Potrebno je pronaći varijablu pod imenom PATH, zatim izabrati njenu vrednost i sa desne strane kliknuti na Edit. Na kraju te vrednosti potrebno je dodati *;../myLibs*.

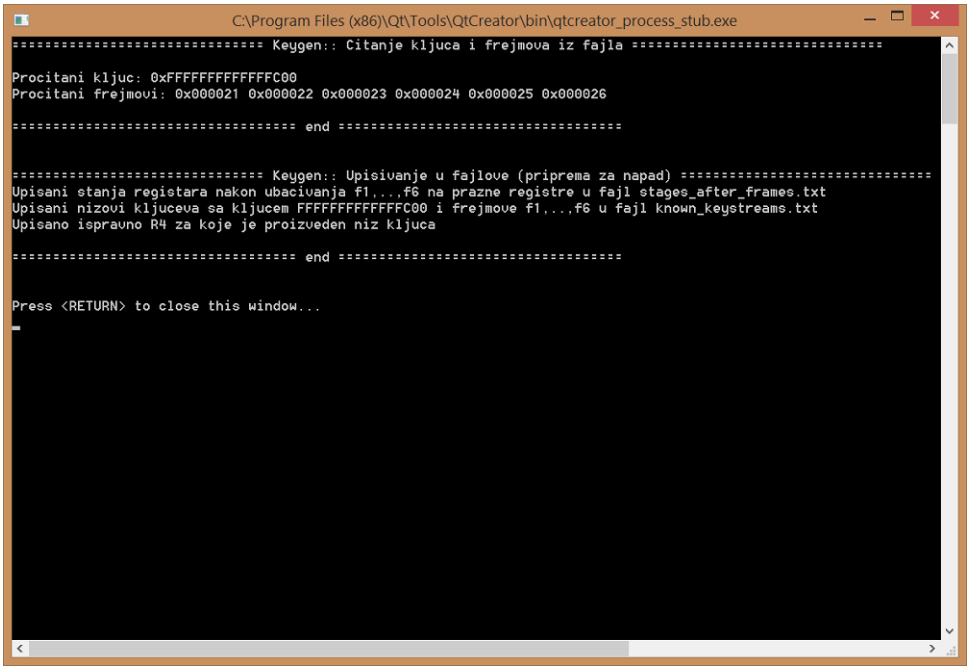
Ovo je potrebno uraditi za oba programa (A52Attack i A52KeystreamGeneration). Promena aktivnog projekta se vrši tako što se desnim klikom na projekat izabere opcija Set "NazivProjekta" as Active Project.
Nakon ovih podešavanja potrebno je vratiti se na program izborom kartice Edit sa leve strane. Program se prevodi opcijom Build Project, klikom na ikonicu koja se nalazi u dnu sa leve strane prozora. Program se pokreće klikom na zelenu strelicu koja se takođe nalazi u dnu prozora sa leve strane. Uvek se prevodi ili pokreće aktivan projekat. Program se može prevesti prečicom sa tastature Ctrl + B, a pokrenuti prečicom Ctrl + R. Program se automatski kompajlira pre pokretanja.

9.5.3 Pokretanje programa

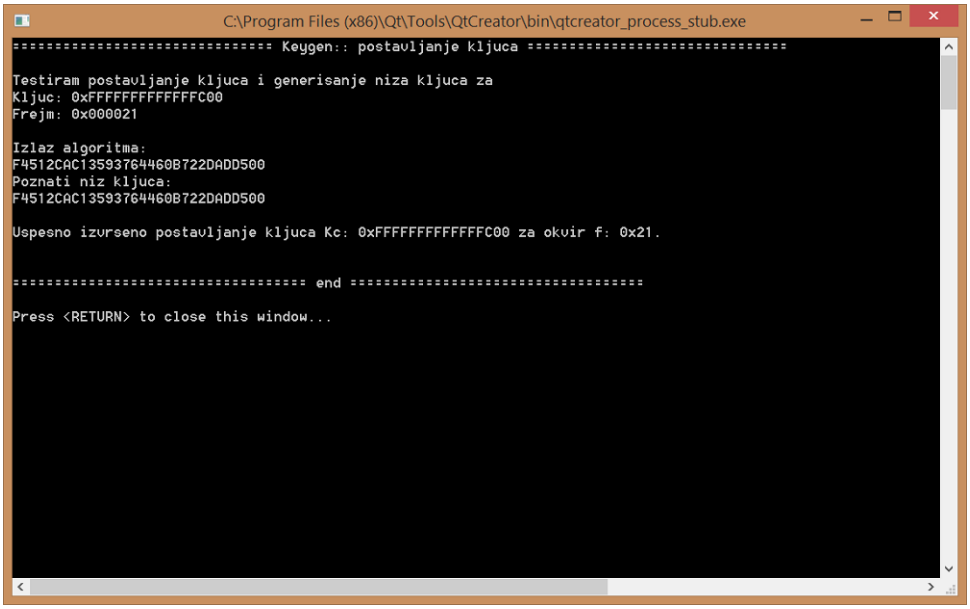
Prvo je potrebno pokrenuti program A52KeystreamGeneration, kao što i je objašenjeno na početku ovog poglavlja. Za pokretanje ovog programa potrebna je datoteka *key_and_frames.txt*, čiji sadržaj izgleda ovako:

```
ffffffffffffc00
000021
000022
000023
000024
000025
000026
```

Prvi red označava ključ u heksadekadnom zapisu, koji se ubacuje u algoritam. Ostalih šest redova, označavaju okvire $f_1, \ldots, f_6$, koji su takođe u heksadekadnom zapisu. Ovi okviri se koriste za generisanje stanja registra kada se ubace na prazne registre i za genererisanje šest poznatih niza ključa. Kada se program pokrene, prvo se ispisuje pročitani sadržaj datoteke *key_and_frames.txt*, a zatim se na osnovu rada algoritma A5/2 prvo generišu i upisuju u datoteku *stages_after_frames.txt* stanja registra nakon ubacivanja svakog od okvira na prazne registre. Zatim se za svaki okvir generiše niz ključa i izlaz tog programa se upisuje u datoteku *known_keystreams.txt*. Za potrebe testiranja napravljena je još jedna datoteka *correctR4.txt* u koju se upisuje stanje registra R4, nakon ubacivanja ključa i okvira f_1. Za to R4, sistem jednačina ima konzistentno rešenje. Datoteka se takođe nalazi u direktorijumu *myOutputFiles*. Generisanje niza ključa i stanja registra za šest okvira prikazano je na slici 11.
Testiranje ispravnosti algoritma A5/2 za generisanje niza ključa implementirano u metodi testKeysetup() prikazano je na slici 12.
Takođe implementirana je i funkcionalnost koja potvrđuje osobinu linearnosti (6.1.4) u postavci ključa u metodi testLinearity (klasa Keygen.cpp), sa ulaznim parametrima K_c i f_1. Izlaz te metode prikazan je na slici 13.
Nakon što se pripreme podaci za napad, pokreće se program A52Attack. Najpre se učitavaju podaci iz direktorijuma *myOutputFiles*, koji su generisani prethodnim programom, a zatim se ti isti podaci inicijalizuju, tj. upisuju se u odgovarajuće promenljive klase Attack.cpp.
Pored toga vrši se inicijalizacija linearnih pomeračkih registara koji sadrže vektor afinih kombinacija (klasa LFSR). Nakon što petlja u programu dođe do ispravnog R4 i pronađe se konzistento rešenje sistema, ulazi se u deo programa koji nalazi ključ na osnovu trenutnih stanja registra. Nakon što je ključ pronađen, prekida se dalja pretraga za moguće R4 i rezultat se ispisuje na standardni izlaz. Prosečno vreme za izvršavanje napada za jedno R4 traje 0.7 sekundi. Stoga, u najgorem slučaju za proveru svih 2^{16} mogućih R4, izvršavanje napada bi trajalo blizu 13 sati.

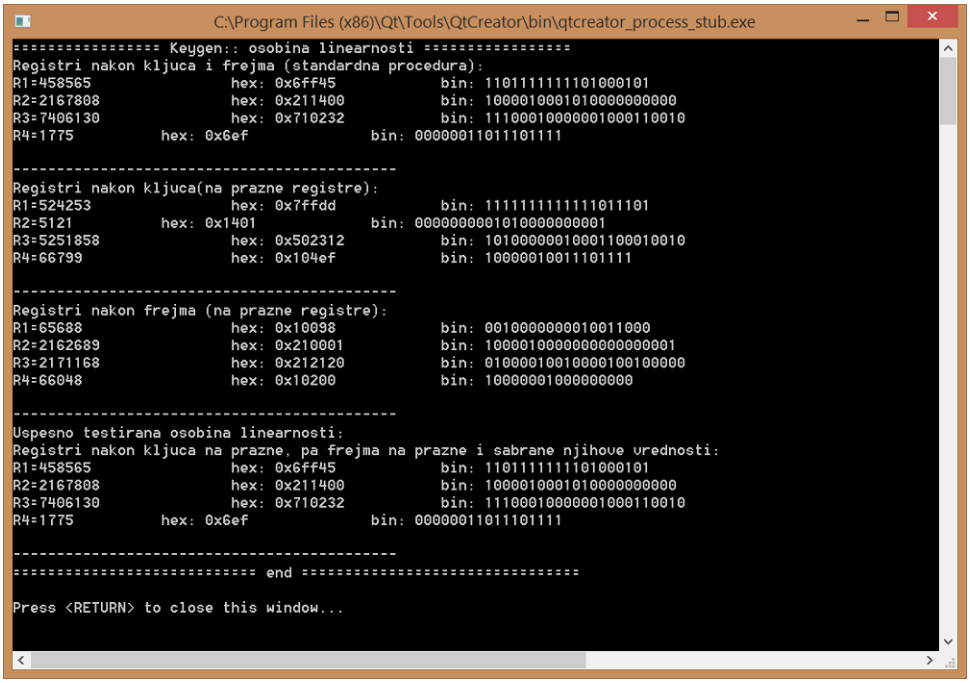


Slika 11: Generisanje poznatih nizova ključa i stanja registra nakon ubacivanja okvira na prazne registre



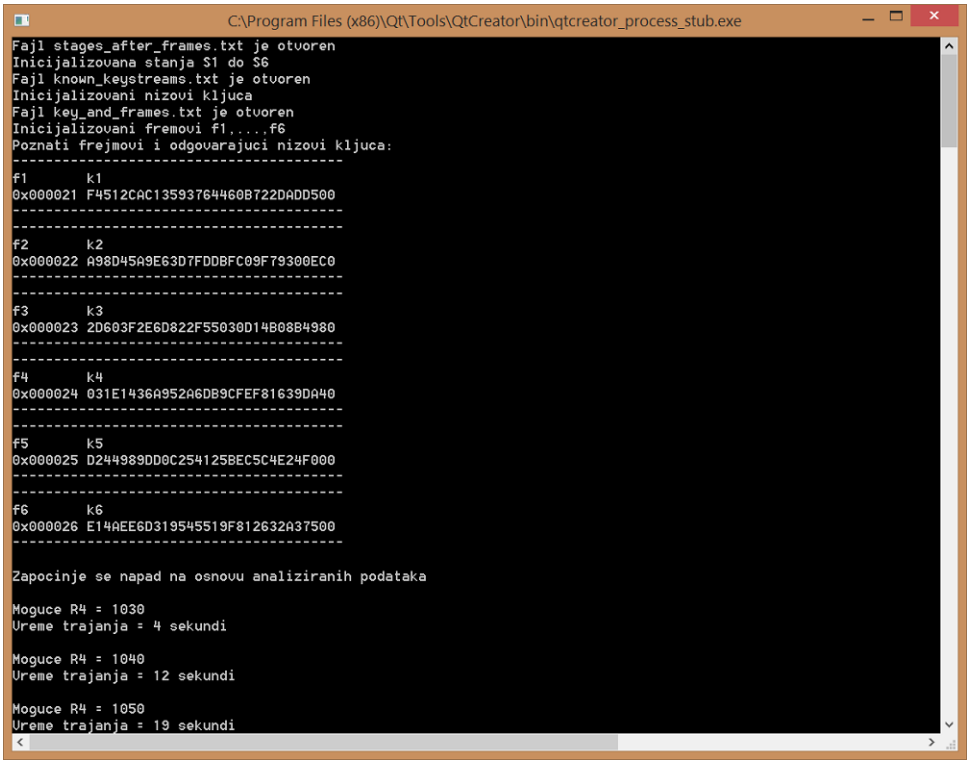
Slika 12: Testiranje rada algoritma A5/2

Pored ovih ispisa, u datoteku *myFilesForAnalyze/resene_matrica.txt*, upisuje se matrica nakon rešavanja sistema, nakon što se pronađe konzistento rešenje sistema.

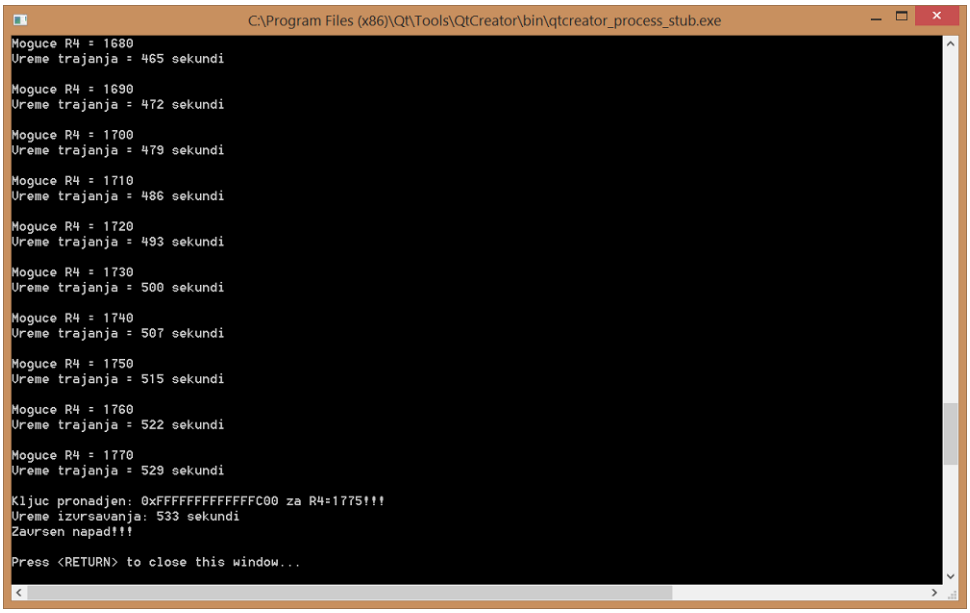


Slika 13: Testiranje osobine linearnosti

Početak izvšavanja napada prikazano je na slici 14, dok je na slici 15. prikazan trenutak nalaženja ključa i ispis istog.



Slika 14: Početak napada A52



Slika 15: Trenutak u kojem je pronađen ključ K_c

10 Zaključak

"Svaki lanac je jak samo onoliko koliko i njegova najslabija karika"

Skoro svaki algoritam za šifrovanje ima neku manu. Potrebno je tu manu pronaći i na njoj bazirati napade. U ovom radu može se videti da je realizovan napad [1] koji pokazuje da algoritam A5/2 ima manu, koja omogućuje napad.

U radu je realizovan algoritam za napad na A5/2 sa poznatim otvorenim tekstom zasnovanog na neoptimizovanom napadu sa poznatim otvorenim tekstom koji su implementirali Barkan, Biham i Keller. Pored toga opisan je GSM, kao sistem u okviru kojeg se šifra primenjivala. Programski su realizovani algoritmi za šifrovanje, odnosno dešifrovanje, kao i algoritam za napad na A5/2 sa poznatim otvorenim tekstom.

Barkan, Biham i Keller su zaključili na osnovu eksperimenata da su za napad dovoljna proizvoljna četiri okvira. Ovo bi moglo da bude jedno od unapređenje programa, tj. korišćenje manjeg broj okvira i njima odgovarajućih šifrata, u cilju smanjivanja količine materijala koji se koristi za napad. Na ovaj način smanjuje se vreme izvršavanja napada.
Drugo unapređenje programa, može biti implementacija napada na A5/2 sa poznavanjem samo šifrata koji su u svoj radu opisali Barkan, Biham i Keller. Napad se zasniva na kontrolama parnosti koji se u otvoreni tekst ubacuju pre šifrovanja. Cilj je izgraditi sistem jednačina na osnovu tih saznanja i rešiti ga na isti način kao što je to predstavljeno u ovom radu.

Literatura

[1] E.Barkan, E.Biham, N.Keller: Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication, Journal of Cryptology, 21 (3): 392–429 (2003)

[2] Ian Goldberg, David Wagner, Lucky Green. The (Real-Time) Cryptanalysis of A5/2. Rump session of Crypto'99, (1999)

[3] M. Živković: Kriptografija, Matematički fakultet, Beograd, (2000)

[4] GSM (Global System for Mobile Communications)
http://www.gsma.com

[5] ETSI (European Telecommunications Standards Institute)
http://www.etsi.org/

[6] European Telecommunications Standards Institute (ETSI), *Digital cellular telecommunications system (Phase 2+); Physical layer on the radio path; General description*, TS 100 573 (GSM 05.01), http://www.etsi.org.

[7] European Telecommunications Standards Institute (ETSI), *Digital cellular telecommunications system (Phase 2+); Channel Coding*, TS 100 909 (GSM 05.03), http://www.etsi.org.

[8] A52HackTool, http://www.npag.fr/project-a52hacktool