Kriptoanaliza šifre CMEA

Aleksandar Kirćanski
Master teza
Mentor: Prof. Miodrag Živković
Matematički fakultet, Beograd

Jul, 2007.

Odbrana 27.07.2007
Komisija:
1. M. Živković (mentor)
2. Ž. Mijajlović
3. P. Janičić

Predgovor

Predmet ove teze je razbijanje šifre CMEA nezavisno od prethodno objavljenih rezultata, kao i računarska implementacija formulisanog napada. Šifra CMEA (Cellular Message Encryption Algorithm) se koristila, a možda ponegde i danas koristi, za šifrovanje kontrolnih poruka u mobilnoj telefoniji u Americi.

Rad koji je prethodio ovoj tezi obuhvatao je kriptoanalizu oslabljenih šifarskih algoritama, kao što je DES od 4 i 6 rundi, modifikovanih 8 rundi algoritma RC5 itd [9]. Početna ideja prilikom biranja teme za ovu tezu bila je da se nakon oslabljenih šifri, pokuša sa samostalnim razbijanjem nekog od algoritama u onom obliku u kom se koristio u praksi.

Prilikom izbora šifre za tu svrhu, bilo je potrebno naći algoritam čija mera nesigurnosti odgovara našim potrebama. Sa jedne strane, odmah su otpali "veliki" algoritmi čija bi kriptoanaliza bila isuviše teška i neizvesna, a sa druge strane, preslabi algoritmi kao što su algoritmi klasične kriptografije takođe nisu dolazili u obzir. Sa obzirom da je mobilna telefonija kako u Evropi tako i van nje poznata po nesigurnim šiframa, odlučili smo se da tu potražimo algoritam koji nam je potreban. Izboru šifre CMEA išlo je u prilog i to što je njena kriptoanaliza navedena kao zadatak u [8], tekstu koji predlaže jednu moguću metodologiju upućivanja u veštinu kriptoanalize.

Do rezultata gotovo identičnih onim koji su već objavljeni, došli smo u skoro svim segmentima kriptoanalize, osim u jednom. Naime, napad "susreta u sredini" ("meet in the middle attack" na engleskom jeziku) nismo otkrili iznova, već implementirali i detaljno obrazložili, konsultujući prethodne rezultate. U pitanju je opšti napad koji je oktriven krajem sedamdesetih godina, pri analizi dvostrukog i trostrukog DES-a [6]. Osnovna ideja napada "susreta u sredini" je rudimentarna i verovatno se primenjuje u efikasnom rešavanju problema i van kriptografije.

Na osnovu iskustva koje smo stekli u susretu sa situacijom u kojoj je bilo potrebno primeniti napad "susreta u sredini" i kasnijeg čitanja gotovog rezultata [2], zaključujemo da je nalaženje rešenja, odnosno tehnike "susreta u sredini", zahtevalo operativno znanje iz oblasti konstrukcije i analize algoritama, predstavljene na primer u [5].

Sadržaj

Glava 1

Uvod

Postoji više definicija kriptografije u opticaju. Na osnovu jedne od njih, ova oblast tiče se, ukratko, komuniciranja u prisustvu neprijatelja. Druga opisuje kriptografiju kao nauku o matematičkim tehnikama vezanim za razne aspekte sigurnosti informacija, kao što su poverljivost i integritet podataka, autentifikacija kako podataka tako i entiteta (osoba koje komuniciraju).

Iako postoji koliko i ljudska potreba za sigurnom komunikacijom, kriptografija se kao moderna matematička disciplina formirala u prošlom veku. Pritom, u periodu do šezdesetih godina dvadesetog veka većina istraživanja u ovoj oblasti odvijala su se u vojne i druge državne svrhe. Pojava računara i drugih komunikacionih sistema šesdesetih i sedamdesetih godina donosi i potrebu za sigurnim komuniciranjem u okviru privatnog sektora, što otvara širok krug istraživanja u ovoj oblasti u okviru akademske zajednice. Otkriće teorije informacija četrdesetih godina prošlog veka, proglašenje standarda za šifrovanje nevojnih vladinih komunikacija u Americi 1977. godine (DES), otkriće koncepta šifre sa javnim i tajnim ključem, kao i metoda za razmenu ključa preko nebezbednog kanala neki su od važnih momenata koji su se desili u drugoj polovini dvadesetog veka.

Između ostalog, kriptografija se bavi dizajnom šifarskih algoritama raznih vrsta. Za kriterijum uspešnosti šifara uzimaju se rezultati kriptoanalize, poddiscipline koja se bavi njihovim razbijanjem. U tom smislu, dizajn algoritama i njihovo razbijanje su se razvijali paralelno i predstavljali jedno drugom izazov. Tako na primer, već pomenuti algoritam DES, koji je sada zastareo i zamenjen novim standardom, algoritmom AES, 80-tih i 90-tih godina poslužio je kao inspiracija za otkriće mnogih kriptoanalitičkih napada.

Danas postoji više algoritama za koje nisu pronađeni efikasni kriptoanalitički napadi i za koje se veruje da su sigurni. Ova činjenica ide u prilog tezi da je samim tim i kriptoanaliza došla u neku vrstu slepe ulice, što je navelo istoričara kriptografije Dejvida Kana da izjavi da je "kriptoanaliza mrtva" [7], uz opasku da će se budući napadi na šifre oslanjati na slabosti realizacija šifarskih algoritama, novonastalih mogućnosti tzv. kvantnih kompjutera i dr.

Međutim, situacija je daleko od toga da slabe šifre više nisu u opticaju. Na

primer, osamdesetih i devedesetih godina prošlog veka otkriveni su realistični, efikasni napadi na dva algoritma koji su predloženi kao zamena za DES: Madryga i FEAL. Dalje, efikasni napadi nađeni su i za algoritme A5/1 i A5/2, predviđene za šifrovanje u okviru mobilne telefonije implementirane u skladu sa GSM[1] standardom. Ovi i drugi slučajevi kriptoanalize navode na zaključak da među komercijalnim šiframa i šiframa u okviru industrije još uvek postoje nesigurne šifre, a time i da je kriptoanaliza aktuelna oblast.

1.1 Kriptografski standard za mobilne komunikacije i šifra CMEA

Osamdesetih godina proslog veka, većina mobilnih komunikacija u Americi bila je nezaštićena u kriptografskom smislu. Poverljivost podataka zasnivala se isključivo na visokoj ceni uređaja za prijem digitalnih podataka iz etra. Međutim, kako je vreme prolazilo, cena ovih skenera je opadala, da bi u jednom trenutku postala dostupna i pojedincima. Time se javila potreba za kriptografskom zaštitom, što je rezultiralo kriptografskim standardom za mobilne komunikacije, realizovanim od strane radne grupe u okviru Telekomunikacione Industrijske Asocijacije (TIA).

Za razliku od ostalih delova ovog standarda koji su služili za šifrovanje glasa, autentifikaciju i drugo, algoritam CMEA koristio se za šifrovanje kontrolnih poruka. Među kontrolne poruke jedne sesije komuniciranja mobilnim telefonom spadaju na primer DTMF[2] tonovi cifara biranih pre, ali i tokom razgovora. Celokupan sadržaj ovog standarda koji je pretrpeo više revizija, opisan je u [1].

Jedan realističan napad na CMEA opisan je 1997. godine u [2]. Time je otvorena mogućnost praćenja telefonskih brojeva koje korisnik mobilnog telefona poziva, kao i eventualnih cifara-smernica automatskim telefonskim sekretaricama, PIN brojeva kreditnih kartica, brojeva bankovnih računa i sličnih osetljivih podataka. Godine 2000. izlazi novi standard, dopunjen sa dva značajno sigurnija naslednika CMEA: SCEMA (Secure Cellular Encryption Module Algorithm) i ECMEA (Enhanced Cellular Message Encryption Algorithm). Međutim, kako CMEA kao jedan od mogućih algoritama nije eliminisan iz standarda iz 2000. godine i kako izbor šifarskog algoritma zavisi isključivo od operatera centrale, postoji mogućnost da još uvek opstaju mesta na kojim se ova šifra koristi.

Ostali delovi pomenutog standarda iz 1992., npr. deo za šifrovanje glasovne komunikacije, razbijeni su još u prvoj polovini devedesetih godina.

[1] GSM je skraćenica od "Global System for Mobile Telephony" i predstavlja trenutno najviše korišćeni standard za mobilnu telefoniju. Kada se pojavio, glavne prednosti ovog standarda bile su visok digitalni kvalitet glasa i mogućnost jeftinih alternativa telefonskom razgovoru (na primer, SMS poruke)

[2] Mobilna telefonija koristi Dual-Tone Multi Frequency signalizaciju, u kojoj se komande centrali, na primer biranje cifara, zadaju istovremenim emitovanjem dve različite frekvencije u liniju. Tako, pritiskanjem cifre 1 na tastaturi, telefon emituje ton sastavljen od 697Hz i 1209Hz.

1.2 Predmet teze

Ova teza obuhvata:

(a) *Pokušaj kriptoanalize CMEA nezavisno od [2].* Ovaj zadatak zapravo predstavlja vežbu čiji je cilj usavršavanje kriptoanalitičke veštine. Naime, prema tekstu [8], koji daje metodologiju ovladavanja ovom veštinom, kaže se sledeće:

> Jedini način da se nauči kriptoanaliza je kroz praksu. Potrebno je samostalno raditi na razbijanju šifre za šifrom, otkrivajući nove tehnike i menjajući već postojeće. Čitanje već postojećih rezultata iz kriptoanalize pomaže, ali ne može biti zamena za iskustvo samostalnog kriptoanalitičkog rada.

U [8] dat je i niz šifara koje mogu da posluže za vežbu. Među njima je i CMEA, pod brojem 6.23.

Što se tiče rezultata, osim preuzimanja ideje i specifične primene napada "susreta u sredini" iz [2], u analizi smo samostalno došli do skoro identičnih rezultata kao u tom radu. Napad izložen u ovoj tezi uzima $13-18$ parova (otvoreni tekst, šifrat), obavlja se u $2^{24}-2^{32}$ operacija koristeći najviše 2^{24} 32-bitnih memorijskih lokacija.

(b) *Implementaciju napada na jeziku C++.* U vezi zadatka iz stavke (b), kažimo to da implementaciju činimo elegnatnijom i kraćom tako što u jednoj fazi napada konstruišemo logičke formule čije je ispitivanje zadovoljivosti ekvivalentno razbijanju šifre, a onda koristimo javno dostupni SAT rešavač MiniSat. Tačnije, koristimo MiniSat-ovu C++ biblioteku, o čemu će takođe biti reči kasnije. Koliko je nama poznato, do sada javno dostupnom nije učinjena ni jedna implementacija napada na algoritam CMEA.

Dakle, ova teza sadrži detaljan opis napada na jedan od oblika CMEA algoritma, zajedno sa elegantnom implementacijom. Teza se tako može koristiti kao dodatak radu [2].

Ostatak dokumenta organizovan je na sledeći način. U glavi 2 dajemo osnovne pojmove iz konteksta u kome radimo. U glavi 3 dajemo specifikaciju šifre, početne komentare i osvrt na [2]. U glavi 4 delimo napad na dve faze koje potom detaljno opisujemo. Zaključak i smernice za dalji rad su u glavi 5.

Glava 2

Osnovni pojmovi

U ovom poglavlju dajemo osnovne pojmove iz konteksta u kome radimo. Prva dva odeljka posvećeni su kriptografiji i kriptoanalizi, a treći logičkim formulama, problemu SAT i programu MiniSat.

2.1 Kriptografija

Pretpostavimo da osobe A i B žele da komuniciraju tako da niko osim njih samih ne može da sazna sadržaj te komunikacije. Jedan od mogućih načina da to realizuju je primenom kriptografije.

Sadržaj njihove komunikacije, odnosno, poruke koje razmenjuju nazivaćemo *otvorenim tekstom*. Proces transformacije, odnosno maskiranja otvorenog teksta nazivaćemo *šifrovanjem*. Rezultat šifrovanja otvorenog teksta zvaćemo *šifratom*. Proces suprotan šifrovanju zvaćemo *dešifrovanjem*.

Pod *kriptografskim algoritmom* ili *šifrom* podrazumevamo funkciju koja transformiše otvoreni tekst u šifrat i obratno. Kriptografski algoritam zavisi od parametra koji ćemo nazivati *ključem*. Skup svih mogućih ključeva nazivaćemo *prostorom ključeva*. Razlog zašto su moderni kriptografski algoritmi parametrizovani ključem je što bi se u suprotnom sigurnost algoritma zasnivala isključivo na tajnosti specifikacije algoritma.

U opštem slučaju, ključevi za šifrovanje i dešifrovanje ne moraju biti isti. Proces šifrovanja otvorenog teksta P ključem K_1 da bi se dobio šifrat C zapisujemo kao

$$C = E_{K_1}(P)$$

a proces dešifrovanja odgovarajućim ključem K_2 sa

$$P = D_{K_2}(C)$$

Činjenicu da su šifrovanje i dešifrovanje međusobno inverzne operacije sada možemo zapisati kao

$$P = D_{K_2}(E_{K_1}(P))$$

Kriptografske algoritme u kojima se ključ za dešifrovanje efikasno izračunava na osnovu ključa za šifrovanje nazivamo *simetričnim*. U praksi, to se najčešće svodi na jednakost ključeva za šifrovanje i dešifrovanje.

Simetrične algoritme možemo uslovno podeliti na dve kategorije. Neki od ovih algoritama otvoreni tekst šifruju bit po bit, a neki blok po blok, gde pod blokovima podrazumevamo grupe bita. Prve nazivamo *lančanim*, a druge *blokovskim* šiframa.

2.2 Kriptoanaliza

Osnovni cilj kriptografije predstavlja očuvanje tajnosti poruke. Neovlašteni pokušaj određivanja sadržaja otvorenog teksta nekog šifrata, odnosno pokušaj određivanja tog sadržaja bez poznavanja ključa, nazivaćemo *napadom*. Nauka ili veština koja izučava tehnike napada na šifre naziva se *kriptoanaliza*. Osobu koja sprovodi napad nazivaćemo *napadačem* ili *kriptoanalitičarem*. U zavisnosti od vrste ulaznih podataka koji se koriste pri napadu, razlikujemo:

- *Napad na osnovu šifrata.* U ovom scenariju, kriptoanalitičar poseduje šifrat više poruka, šifrovanih pod istim ključem. Cilj kriptoanalitičara je da odredi otvoreni tekst od što više datih šifrata, ili, još bolje, da odredi ključ pod kojim je šifrovanje izvršeno.

  Ulaz: $C_1 = E_K(P_1), C_2 = E_K(P_2), \ldots C_n = E_K(P_n)$

  Izlaz: $P_1, P_2, \ldots P_n$ ili ključ K ili algoritam za određivanje P_{n+1} na osnovu $C_{n+1} = E_k(P_{n+1})$

- *Napad na osnovu parova (otvoreni tekst, šifrat).* U ovom, za napadača povoljnijem scenariju, on ne samo da poseduje šifrate, već i njima odgovarajuće otvorene tekstove. Njegov zadatak je da odredi ključ pod kojim su poruke šifrovane, ili algoritam po kome se drugi šifrati mogu dešifrovati u otvoreni tekst.

  Ulaz: $(P_1, C_1 = E_K(P_1)), (P_2, C_2 = E_K(P_2)), \ldots (P_n, C_n = E_K(P_n))$

  Izlaz: ključ K ili algoritam za određivanje P_{n+1} na osnovu $C_{n+1} = E_k(P_{n+1})$

- *Napad na osnovu izabranog otvorenog teksta.* Kriptoanalitičar ne samo da poseduje parove (otvoreni tekst, šifrat), već je u mogućnosti da izabere otvorene tekstove čije će šifrate poznavati. Ovaj scenario je još povoljniji za napadača, jer otvara mogućnost odabira grupe otvorenih tekstova koji zajedno daju najviše informacija. Cilj napadača je rekonstrukcija ključa, ili određivanje algoritma koji dešifruje bilo koji šifrat.

  Ulaz: $(P_1, C_1 = E_K(P_1)), (P_2, C_2 = E_K(P_2)), \ldots (P_n, C_n = E_K(P_n))$ pri čemu napadač bira $P_1, P_2, \ldots P_n$

  Izlaz: ključ K ili algoritam za određivanje P_{n+1} na osnovu $C_{n+1} = E_k(P_{n+1})$

Trivijalni napad na osnovu parova je tzv. *potpuna pretraga*. U tom napadu, vrši se šifrovanje datih otvorenih tekstova svim ključevima iz prostora ključeva i proverava koji od ključeva odgovara istovremeno svim zadatim parovima. Stoga, dovoljna veličina prostora ključeva je potreban uslov sigurnosti šifre. Primetimo da je potpunu pretragu moguće izvesti i kao napad na osnovu šifrata, ukoliko se ima pretpostavka o statističkim zakonitostima otvorenog teksta. U tom slučaju, vrši se dešifrovanje datih šifrata i beleže se oni ključevi čiji odgovarajući otvoreni tekst zadovoljava zadate statističke zakonitosti. Uobičajena dužina ključa simetrične šifre danas iznosi od 64 do 128 bita, što potpunu pretragu čini neizvodljivom u praksi.

Složenost napada opisuje se različitim parametrima:

- *Količinom ulaznih podataka.* Na primer, u slučaju napada na osnovu šifrata, ovaj parametar može predstavljati broj bajtova šifrata potrebnih da bi se napadom odredio ključ .
- *Brojem operacija.* Na primer, može biti potrebno 2^{60} operacija šifrovanja da bi se izvršio napad.
- *Veličinom prostora za skladištenje.* U nekim napadima potrebno je skadištiti obrađene informacije radi daljih obrada.

Tako, napad potpune pretrage obično zahteva relativno mali broj parova (otvoreni tekst, šifrat), približno $2^{|K|}$ operacija, gde je sa $|K|$ označen broj svih ključeva i zanemariv prostor za skladištenje. Postoje napadi u kojima je je moguće smanjiti jedan od navedenih parametara na račun drugih. Na primer, u tzv. "susret u sredini" napadu, o kome će biti reči u ovoj tezi, broj operacija smanjuje se na račun prostora za skladištenje.

Otkrivanje uspešnog napada na kriptografski algoritam ne podrazumeva nalaženje praktično izvodljivog načina da se dođe do ključa ili otvorenog teksta datog šifrata, već otkriće bilo koje slabosti šifre koja pokazuje da se šifra ne ponaša onako kako su njeni autori zamislili i na osnovu koje je moguće razbiti šifru sa složenošću manjom od složenosti potpune pretrage. Ovo može značiti da je umesto 2^{128} operacija za razbijanje potrebno 2^{120} operacija, što je praktično neizvodljivo. Ipak, činjenica da se šifra ne ponaša onako kako su njeni autori zamislili otvara mogućnost da će se u budućnosti napad poboljšati i učiniti je i praktično nesigurnom.

2.3 Zadovoljivost logičkih formula i SAT rešavač MiniSat

Kako u jednom delu napada na šifru CMEA konstruišemo logičke iskazne formule čije je ispitivanje zadovoljivosti ekvivalentno razbijanju tog dela šifre, u ovom odeljku navodimo osnovne pojmove koji se tiču problema zadovoljivosti i programa MiniSat koji služi za njegovo rešavanje. Za definicije osnovnih pojmova iz logike, kao što su iskazna formula, valuacija, interpretacija kao i za detaljniji prikaz ove problematike, videti [3].

Za iskaznu logičku formulu kažemo da je *zadovoljiva* ukoliko postoji valuacija u kojoj je ta formula tačna. Problem određivanja da li je neka iskazna formula zadovoljiva označava se sa SAT. Osim što je poznat po tome što je prvi problem za koji je dokazano da je NP-kompletan, SAT problem je značajan i po tome što ima primenu u mnogim oblastima računarstva, kao što je teorija algoritama, veštačka inteligencija, dizajn hardvera itd.

Imajući u vidu da se trenutno ne zna da li su klase P i NP-kompletnih problema jednake, ne zna se ni da li postoji algoritam za rešavanje SAT problema koji je polinomijalne složenosti. Svi danas poznati postupci za rešavanje ovog problema su eksponencijalne složenosti. Davis/Logemann/Lovelandova procedura, metod rezolucije i metod tabloa su neki od tih postupaka. Međutim, vreme potrebno za rešavanje formula koje konstruišemo u ovoj tezi na personalnom računaru je zanemarivo, tako da ovde nećemo ulaziti u problematiku rešavanja teških instanci SAT problema.

Postoji veći broj javno dostupnih, besplatnih realizacija SAT rešavača. Za korišćenje u ovoj tezi, odlučili smo se za MiniSat. U pitanju je minimalistički SAT rešavač čiji javno dostupni kôd sadrži samo 600 linija (ne računajući komentare). Veliku popularnost stekao je zahvaljujući tome što je jednostavan za izmenu, visoko efikasan i lak za integraciju u softverske sisteme. U nastavku dajemo primer korišćenja MiniSat-a. Celokupna dokumentacija ovog programa data je u [4].

Za iskaznu logičku formulu kažemo da je u konjunktivnoj normalnoj formi ako je oblika

$$A_1 \wedge A_2 \wedge \ldots \wedge A_n$$

pri čemu je svaka od formula A_i $(1 \leq i \leq n)$ disjunkcija iskaznih slova ili njihovih negacija. Kao i većina drugih SAT rešavača, MiniSat na ulazu uzima formule u KNF-u. Pritom, formulu je moguće zadati na dva načina:

- Preko datoteke u DIMACS, gzip-ovanom DIMACS ili BCNF formatu. Ukratko ćemo opisati DIMACS format kroz primer datoteke:

```
c formula.cnf: Primer datoteke u DIMACS formatu
p cnf 3 2
1 -3 0
2 3 -1 0
```

Komentari počinju slovom `c`, kao što je slučaj sa prvom linijom gornje datoteke. Nakon toga, potrebno je definisati broj klauza i broj iskaznih slova formule, što se radi komandom `p cnf n m` gde je `n` broj iskaznih slova a `m` broj klauza. Svaka sledeća linija predstavlja po jednu klauzu. Iskazna slova identifikuju se brojevima od `1` do `n`. Znak minus ispred broja označava negaciju odgovarajućeg iskaznog slova. Svaki red koji opisuje klauzu završava se brojem `0`. Gornja datoteka opisuje formulu $(p_1 \vee \neg p_3) \wedge (p_2 \vee p_3 \vee \neg p_1)$. Primer poziva MiniSat-a:

```
# minisat formula.cnf rezultat.txt
```

Kao prvi i drugi argument pri pozivu programa prosleđuju se ime datoteke sa formulom i ime datoteke u koju će se upisati rezultat, redom.

- Pozivajući funkcije iz MiniSat-ove C++ biblioteke. Sledi primer C++ programa koji ispituje zadovoljivost gornje formule:

```
//  sat.c, primer koriscenja MiniSat API-ja
#include <Solver.h>
#include <iostream>

int main()
{
    Solver solver;   // Deklarisemo instancu resavaca-a
    vec<Lit> lits;   // Definisemo klauzu

    // Najavljujemo da ce biti tri iskazna slova
    // (predstavljena sa 0, 1, i 2)

    solver.newVar();
    solver.newVar();
    solver.newVar();

    // Dodajemo prvu binarnu klauzu

    solver.addBinary(Lit(0), ~Lit(2));

    // Definisemo drugu klauzu, sa tri iskazne promenljive i
    // dodajemo je u formulu

    lits.clear();
    lits.push( Lit(1) );
    lits.push( Lit(2) );
    lits.push( ~Lit(0) );
    solver.addClause(lits);

    solver.simplifyDB();

    // Proveravamo da li je zadovoljiva, i ako jeste, vracamo
    // nadjenu zadovoljavajucu valuaciju

    if (!solver.okay() || !solver.solve())
    {
        std::cout << "Formula nezadovoljiva";
    }
    else
    {
```

```
            std::cout << "Zadovoljiva: ";
            for (unsigned int i=0; i<solver.model.size(); i++)
                std::cout << solver.model[i].toInt() << " ";
            std::cout << std::endl;
        }

    }
```

Program se može kompajlirati korišćenjem sledećeg `Makefile`-a:

```
MINISAT=./MiniSat_v1.14
MINISAT_LIB=$(MINISAT)/Solver.o
INCLUDE=-I $(MINISAT)

all: sat

sat.o: sat.cpp
        g++ -g -c $(INCLUDE) sat.cpp
sat: sat.o
        g++ -g $(MINISAT_LIB) -lz sat.o -o sat
```

U implementaciji datoj u ovoj tezi, koristili smo drugi navedeni način upotrebe MiniSat-a. Primetimo da prikazani postupak nalazi jednu zadovoljavajuću valuaciju logičke iskazne formule. Da bi se našle sve ovakve valuacije, potrebno je uzastopno izvršavati traganje za zadovoljavajućim valuacijama, uz dodavanje klauza koje negiraju do tada nađene valuacije. Postupak se ponavlja dok formula ne postane protivrečna.

Glava 3

Specifikacija šifre CMEA i postojeći napad

U daljem tekstu koristićemo sledeće simbole i oznake:

- Znakovi $+, -$ označavaju sabiranje, odnosno oduzimanje po modulu 256, a znakovi $\oplus, \vee, \wedge$ bitske operacije XOR-ovanja, disjunkcije i konjunkcije.
- Ako je X niz bitova, sa $X^{(i)}$ označavamo i-ti bit od X. Sa $inv(X, i)$ označavamo X sa invertovanim i-tim bitom.

3.1 Specifikacija

Radi se o "maloj" blokovskoj šifri, optimizovanoj za 8-bitne procesore, varijabilne dužine ulaza i 64-bitnog ključa. U zavisnosti od veličine ulaza koja može biti proizvoljnih $n \geq 2$ bajtova, algoritam CMEA uzima različite oblike, pri čemu u praksi najčešće $n \in \{2, 3, 4, 5, 6\}$. U ovoj tezi obrađujemo slučaj $n = 2$. Uvedimo sledeću notaciju:

- Otvoreni tekst dužine n bajta označavaćemo sa $P_0P_1 \dots P_{n-1}$.
- Ključ dužine 8 bajta ozačavaćemo sa $K = K_0 \dots K_7$.
- Šifrat dužine n bajta označavaćemo sa $C_0C_1 \dots C_{n-1}$.

Sledeći kôd opisuje tranformaciju $P_0P_1 \dots P_{n-1}$ u $C_0C_1 \dots C_{n-1}$, sa parametrom $K_0 \dots K_7$:

$y_0 = 0$
for $i = 0 \dots n-1$
  $P_i' = P_i + T(y_i \oplus i)$
  $y_{i+1} = y_i + P_i'$

for $i = 0 \dots \lfloor \frac{n}{2} \rfloor - 1$

$$P_i'' = P_i' \oplus (P'_{n-1-i} \vee 1) \tag{3.1}$$

$z_0 = 0$
for $i = 0 \ldots n-1$
$z_{i+1} = z_i + P_i''$
$C_i = P_i'' - T(z_i \oplus i)$

Pritom, funkcija $T : Z_{256} \to Z_{256}$ definisana je sledećim izrazom:

$$T(x) = C(((C(((C(((C((x \oplus K_0) + K_1) + x) \oplus K_2) + K_3) + x) \oplus$$
$$\oplus K_4) + K_5) + x) \oplus K_6) + K_7) + x$$

Funkcija $C : Z_{256} \to Z_{256}$ ne zavisi od ključa i zadata je tabelom 1.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	d9	23	5f	e6	ca	68	97	b0	7b	f2	0c	34	11	a5	8d	4e
1	0a	46	77	8d	10	9f	5e	62	f1	34	ec	a5	c9	b3	d8	2b
2	59	47	e3	d2	ff	ae	64	ca	15	8b	7d	38	21	bc	96	00
3	49	56	23	15	97	e4	cb	6f	f2	70	3c	88	ba	d1	0d	ae
4	e2	38	ba	44	9f	83	5d	1c	de	ab	c7	65	f1	76	09	20
5	86	bd	0a	f1	3c	a7	29	93	cb	45	5f	e8	10	74	62	de
6	b8	77	80	d1	12	26	ac	6d	e9	cf	f3	54	3a	0b	95	4e
7	b1	30	a4	96	f8	57	49	8e	05	1f	62	7c	c3	2b	da	ed
8	bb	86	0d	7a	97	13	6c	4e	51	30	e5	f2	2f	d8	c4	a9
9	91	76	f0	17	43	38	29	84	a2	db	ef	65	5e	ca	0d	bc
10	e7	fa	d8	81	6f	00	14	42	25	7c	5d	c9	9e	b6	33	ab
11	5a	6f	9b	d9	fe	71	44	c5	37	a2	88	2d	00	b6	13	ec
12	4e	96	a8	5a	b5	d7	c3	8d	3f	f2	ec	04	60	71	1b	29
13	04	79	e3	c7	1b	66	81	4a	25	9d	dc	5f	3e	b0	f8	a2
14	91	34	f6	5c	67	89	73	05	22	aa	cb	ee	bf	18	d0	4d
15	f5	36	ae	01	2f	94	c3	49	8b	bd	58	12	e0	77	6c	da

Tabela 3.1: $C(x)$ jednako je vrednosti polja (m, n), gde su m, n određeni sa 4 bita veće, odnosno manje težine x. Npr: $C(90) = C(01011010) = C(5, 10) = 5f$.

Predmet ove teze je CMEA za $n = 2$. U tom slučaju, otvoreni tekst i šifrat su dužine 16 bita. Obeležavamo ih sa P_0P_1 i C_0C_1, redom. Kôd (3.1) uzima sledeći oblik:

$P_0' = P_0 + T(0)$
$P_1' = P_1 + T(P_0' \oplus 1)$

$$P_0'' = P_0' \oplus (P_1' \vee 1), P_1'' = P_1' \tag{3.2}$$

$C_0 = P_0'' - T(0)$
$C_1 = P_1'' - T(P_0'' \oplus 1)$

Specifikacija funkcije T i dužina ključa K ne zavise od n.
Komentari: Možemo primetiti sledeće.

- Proces dešifrovanja identičan je procesu šifrovanja, odnosno $CMEA$ je inverzna sama sebi.

- Za svaki par (P_0P_1, C_0C_1) važi $P_0^{(8)} \oplus C_0^{(8)} = (P_0' - T(0))^{(8)} \oplus (P_0'' - T(0))^{(8)} = (P_0'^{(8)} \oplus P_0''^{(8)}) = (P_0' \oplus P_0'')^{(8)} = (P_1' \vee 1)^{(8)} = 1$. Stoga, svaki šifrat automatski otkriva jedan bit otvorenog teksta.

- U tabeli 1 se ne pojavljuje 92 od mogućih 256 vrednosti, što znači da C nije "na" preslikavanje. Ovu činjenicu koristićemo u obe faze napada.

3.2 Postojeći napad

Kao što smo već rekli, realističan napad na šifru CMEA objavljen je u [2], 1997. godine. U pitanju je tzv. napad na osnovu zadatih parova, odnosno napad u kome se pretpostavlja da kriptoanalitičar raspolaže nekim brojem parova oblika (otvoreni tekst, šifrat). Preciznije, u [2] objavljeni su napadi na CMEA za:

- $n = 2$ (CMEA sa ulaznim blokom dužine 16 bita) na osnovu 4 para (otovoreni tekst, šifrat). Složenost: $2^{29} - 2^{37}$.

- $n = 3$ (CMEA sa ulaznim blokom dužine 24 bita) na osnovu $50-80$ parova (otvoreni tekst, šifrat). Složenost: $2^{24} - 2^{32}$.

Napomenimo i to da napad na CMEA za $n = 3$ predstavlja prirodno proširenje napada za $n = 2$, odnosno da se osnovna ideja ne menja. Slično, sa $n = 3$ prirodno se prelazi na $n = 4$, odnosno, opštije, sa n na $n + 1$. Ipak, ne može se odmah reći da je na taj način idejama iz [2] šifra razbijena za svako n. Naime, vremenski interval potreban za izvođenje napada raste zajedno sa n, pa se može desiti da za dovoljno veliko n taj interval nadmaši vreme potrebno za napad potpune pretrage ključeva.

Glava 4

Napad na šifru CMEA

U ovoj glavi dajemo detaljan prikaz napada. Prvo poglavlje služi da u osnovnim crtama predstavi napad, odnosno njegovu prvu i drugu fazu. Drugo, odnosno treće poglavlje detaljno obrađuje ove dve faze.

4.1 Plan napada

Cilj nam je da na osnovu unapred zadatog skupa parova otvoreni tekst-šifrat odredimo sve ključeve koji date otvorene tekstove transformišu u njima odgovarajuće šifrate. Ukoliko je zadati skup parova dovoljno veliki, ključ će biti jedinstven[1].

Šifra CMEA predstavlja kompoziciju dve transformacije gde je prva opisana kôdom (3.2), a druga funkcijom T. Ove dve transformacije napadamo odvojeno, u dve faze:

- na osnovu zadatog skupa parova oblika (P_0P_1, C_0C_1) odrediti $T(x_i)$ u nekoliko tačaka x_i
- na osnovu dobijenih parova $(x_i, T_{x_i}), i = 1 \ldots n$, rekonstruisati ključ K.

U prvoj fazi, na osnovu poznatih parova i specifikacije šifre, formiramo logičke formule čije zadovoljavajuće valuacije otkrivaju informacije o vrednostima funkcije T u nekim tačkama. U implementaciji, pretragu za zadovoljavajućim valuacijama obavljamo koristeći MiniSat. Detaljno, prva faza napada opisana je u sledećem odeljku.

U drugoj fazi, ključno je to što je funkcija T je ranjiva na tzv. napad "susreta u sredini"[2], koji predstavlja uprošćenu pretragu kroz prostor svih ključeva. Pretraga je, osim tehnikom "susret u sredini" uprošćena i korišćenjem dodatne dve slabosti funkcije T. Ova faza napada opisana je u odeljku 3 ove glave.

Prelazimo na prvu fazu napada.

[1] Odnosno, jedinstveno određen do na 4 vrednosti, kako ćemo videti u daljem tekstu.
[2] "Meet in the middle" na engleskom jeziku

4.2 Rekonstrukcija nekoliko vrednosti funkcije T

Analiza: Neka je dat otvoreni tekst P_0P_1. Pretpostavimo da je $T(0)$ jednako nekoj fiksiranoj, poznatoj veličini A i uvedimo oznake za vrednosti koje uzima T u 2. i 6. redu kôda (3.2):

$$t_2 = T(P_0' \oplus 1)$$

$$t_6 = T(P_0'' \oplus 1)$$

Odgovor na pitanje koje veličine mogu uzeti t_2 i t_6 da bi šifrat bio jednak nekom C_0C_1 daje

Tvrđenje 1 *Neka je* $T : Z_{256} \to Z_{256}$ *i* $T(0) = A$. *Tada,*
$CMEA_T(P_0P_1) = C_0C_1 \Leftrightarrow$
$(t_2 = (C_0 + A) \oplus (P_0 + A) - P_1 \vee t_2 = inv((C_0 + A) \oplus (P_0 + A), 8) - P_1)$,
$t_6 = P_1 - C_1 + t_2$

Dokaz:
$CMEA_T(P_0P_1) = C_0C_1$
$\Leftrightarrow P_0'' - A = C_0, P_1'' - t_6 = C_1$
$\Leftrightarrow (P_0 + A) \oplus ((P_1 + t_2) \vee 1) - A = C_0,\ P_1 + t_2 - C_1 - t_6 = 0$
$\Leftrightarrow (t_2 + P_1) \vee 1 = (C_0 + A) \oplus (P_0 + A),\ t_6 = P_1 - C_1 + t_2$.
Imajući u vidu drugi komentar iz prethodne glave, jasno je da je poslednji izraz ekvivalentan desnoj strani ekvivalencije iz formulacije tvrđenja. □

Pretpostavimo sada da je kriptoanalitičar došao u posed skupa Π parova :

$$\Pi = \{(P_0P_1, C_0C_1) | P_0P_1 \in P, C_0C_1 = CMEA_{T_K}(P_0P_1))\}$$

gde je P skup otvorenih tekstova. Pretpostavimo takođe da je $|\Pi| = n$.

Koristeći prethodno tvrđenje, uz pretpostavku $T(0) = A$, na osnovu i-tog para skupa Π dobijamo iskaz

$$T_i(A) : (T(x) = X_1 \wedge T(y) = Y_1) \vee (T(x) = X_2 \wedge T(y) = Y_2)$$

pri čemu

$$\begin{aligned} x &= P_0' \oplus 1 \\ y &= P_0'' \oplus 1 \\ X_1 &= (C_0 + A) \oplus (P_0 + A) - P_1 \\ Y_1 &= P_1 - C_1 + (C_0 + A) \oplus (P_0 + A) - P_1 \\ X_2 &= inv((C_0 + A) \oplus (P_0 + A), 8) - P_1) \\ Y_2 &= P_1 - C_1 + inv((C_0 + A) \oplus (P_0 + A), 8) - P_1 \end{aligned}$$

Primetimo da uvek važi $X_1 \neq X_2$, $Y_1 \neq Y_2$. Ovakve iskaze, dobijene za sve parove skupa Π, stavljamo u konjunkciju. Dodajemo i posebnu klauzu za $T(0)$:

$$\Phi_A = T_1(A) \wedge \ldots \wedge T_n(A) \wedge (T(0) = A).$$

Konačno, formiramo sledeću disjunkciju:

$$\Phi = \Phi_0 \vee \Phi_1 \vee \ldots \vee \Phi_{255}. \tag{4.1}$$

Definicija 1 *Za funkciju T reći ćemo da* zadovoljava par (P_0P_1, C_0C_1) *ukoliko važi $C_0C_1 = CMEA_T(P_0P_1)$. Opštije, za funkciju T reći ćemo da* zadovoljava skup parova Π, *ukoliko zadovoljava svaki par iz* Π.

Koristeći prethodno tvrđenje, pokazuje se da važi:

Tvrđenje 2 *Funkcija T zadovoljava Π akko T zadovoljava iskaz Φ (4.1).*

Dokaz: Iz pretpostavke da T zadovoljava Π, primenom desnog smera prethodnog tvrđenja n puta, sledi da T zadovoljava $\Phi_{T(0)}$, a time i Φ. Obratno, iz pretpostavke da T zadovoljava Φ, sledi da zadovoljava i Φ_A, za neko $0 \leq A \leq 255$. To dalje znači da T zadovoljava svaki konjunkt iz Φ_A, pa primenom levog smera prethodne teoreme n puta, sledi da T zadovoljava Π. □

Na taj način svodimo problem određivanja funkcija T koje zadovoljavaju Π na problem određivanja funkcija T koje zadovoljavaju iskaz Φ. Primetimo, međutim, da se može desiti da Φ zadovoljava funkcija T takva da je $T(x) = y$, gde $T(x) - y$ predstavlja vrednost koje nema u tabeli C. U implementaciji ovakve vrednosti odbacujemo.

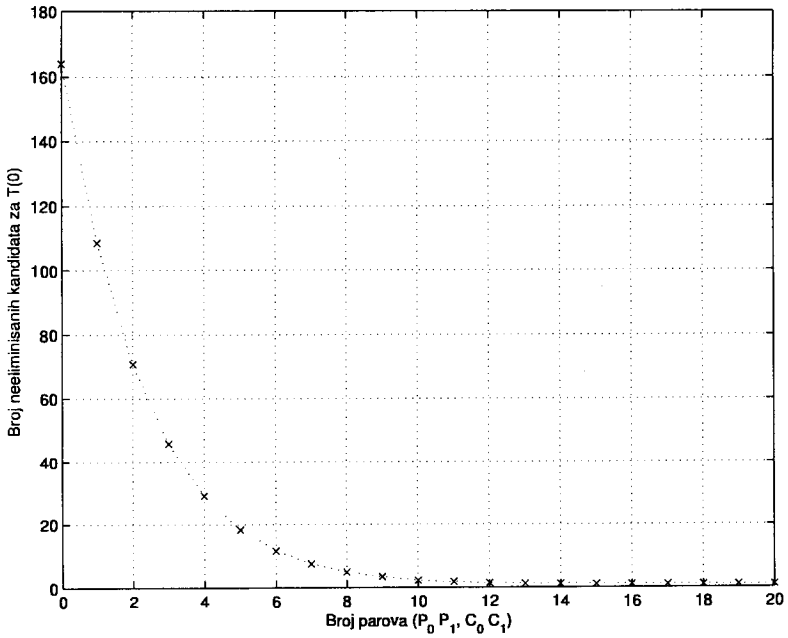
Grafik sa slike 4.1 dobijen je eksperimentalnim putem i prikazuje zavisnost broja neeliminsanih vrednosti $T(0)$ od broja parova n. Za svako $n = 1, 2, \ldots n$, eksperiment generisanja slučajnog ključa i n slučajnih parova, formiranja odgovarajuće formule Φ i na kraju brojanja neeliminsanih $T(0)$ izveden je 1000 puta. Kao vrednost funkcije grafika u tački n uzeta je aritmetička sredina ovih 1000 vrednosti. Primetimo da za 0 parova broj neeliminsanih $T(0)$ vrednosti nije 256 već 164. Naime, kako smo gore već napomenuli, funkcija C, a time ni vrednost $T(0)$, ne može uzeti 92 od 256 mogućih vrednosti.

Grafik sa slike 4.2 takođe je dobijen eksperimentalnim putem. Pod pretpostavkom da poznajemo pravu vrednost $T(0)$, prikazana je zavisnost broja jednoznačno određenih vrednosti T formulom $\Phi_{T(0)}$ i broja parova n. Broj ponavljanja eksperimenta takođe je iznosio 1000 puta.

Na osnovu grafika sa slike 4.1, vidi se da je, za 13-18 parova, broj kandidata $T(0)$, odnosno broj funkcija-kandidata T blizak 1. Na osnovu grafika slike 4.2, svaki od ovih kandidata biće jednoznačno određen u 13-18 tačaka. U nastavku ćemo videti da je to dovoljno da ključ K svake od ovih funkcija bude određen jednoznačno do na klasu ekvivalencije.

Implementacija

Ulaz za program koji implementira ovu fazu napada je skup parova Π. Izlaz sadrži skup vrednosti $0 \leq A \leq 255$ takvih da postoji funkcija T za koju važi $T(0) = A$, a zadovoljava svaki od parova iz Π. Za svako takvo A, izlaz takođe sadrži niz implikacija oblika $T(0) = A \Rightarrow T(x) = y$. Stoga, u programu treba:



Slika 4.1: Zavisnost broja neeliminisanih kandidata za $T(0)$ i broja parova n

- Na osnovu skupa Π odrediti iskaze Φ_A, $0 \leq A \leq 255$.
- Proveriti neprotivrečnost svakog od ovih iskaza.
- Za neprotivrečne Φ_A odrediti sve implikacije oblika $\Phi_A \Rightarrow T(x) = y$.

Poslednje dve stavke programski elegantno rešavamo koristeći SAT rešavač MiniSat. Prethodno, svaki od iskaza Φ_A, $0 \leq A \leq 255$ prevodimo u logičku formulu, na način opisan u nastavku.

Počnimo sa $T_i(A)$. Neka iskazno slovo $p_{x,y}$ označava iskaz $T(x) = y$. Iskaz $T_i(A)$ se onda prevodi u

$$(p_{x,X_1} \wedge p_{y,Y_1}) \vee (p_{x,X_2} \wedge p_{y,Y_2})$$

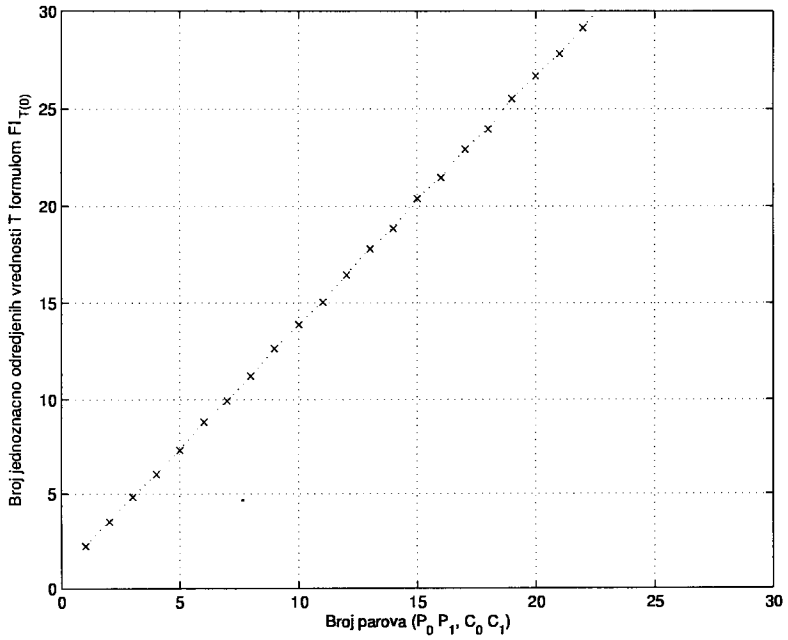
Međutim, potrebno je zadati i uslov da je T na datim tačkama funkcija, odnosno da ne može uzeti dve različite vrednosti. Tako dobijamo

$$((p_{x,X_1} \wedge p_{y,Y_1}) \vee (p_{x,X_2} \wedge p_{y,Y_2})) \wedge \neg(p_{x,X_1} \wedge p_{x,X_2}) \wedge \neg(p_{y,Y_1} \wedge p_{y,Y_2})$$

Transformisanjem formule u KNF, dobijamo

$$(p_{x,X_1} \vee p_{x,X_2}) \wedge (p_{x,X_1} \vee p_{y,Y_2}) \wedge (p_{x,X_2} \vee p_{y,Y_1})$$

$$(p_{y,Y_1} \vee p_{y,Y_2}) \wedge (\neg p_{x,X_1} \vee \neg p_{x,X_2}) \wedge (\neg p_{y,Y_1} \vee \neg p_{y,Y_2})$$



Slika 4.2: Zavisnost broja tačaka u kojima formula $\Phi_{T(0)}$ jednoznačno određuje vrednost T od broja parova n

Neposredno se proverava da prvi i četvrti konjunkt slede iz ostatka formule (na primer, metodom istinitosnih tabela). Njihovom eliminisanjem dobijamo logički ekvivalentnu formulu

$$(p_{x,X_1} \vee p_{y,Y_2}) \wedge (p_{x,X_2} \vee p_{y,Y_1}) \wedge (\neg p_{x,X_1} \vee \neg p_{x,X_2}) \wedge (\neg p_{y,Y_1} \vee \neg p_{y,Y_2})$$

Sada se konjunkcijom ovakvih formula formira logička formula koja odgovara Φ_A. Dalje, dodajmo ograničenja vezana za nemogućnost funkcije T da uzme sve vrednosti. U tom cilju dovoljno je dodati klauze oblika $\neg p_{x,y}$ za svako $p_{x,y}$ prisutno u formuli, a istovremeno takvo da je $T(x) = y$ neostvarivo, prepuštajući SAT rešavaču da izvrši potrebne propagacije u ostalim klauzama.

Dobijenu logičku formulu obeležimo sa $L(\Phi_A)$. Iz načina konstrukcije $L(\Phi_A)$ sledi:

- $L(\Phi_A)$ je neprotivrečna akko je iskaz Φ_A neprotivrečan.
- Svaka zadovoljavajuća valuacija $L(\Phi_A)$ određuje skup funkcija T koje zadovoljavaju Φ_A. Naime, svaka funkcija T za koju, ukoliko je $p_{x,y}$ tačno u datoj valuaciji, važi $T(x) = y$, zadovoljava Φ_A.

  I obratno, svaka funkcija T koja zadovoljava Φ_A može se na ovaj način dobiti nekom zadovoljavajućom valuacijom $L(\Phi_A)$.

Na svako od $L(\Phi_A)$, $A = 0 \ldots 255$ primenjujemo MiniSat (256, odnosno preskakanjem nemogućih $T(0)$, 164 puta). U zavisnosti od ishoda, postupamo na sledeći način:

- Formula je nezadovoljiva: odbacujemo pretpostavku $T(0) = A$
- Formula je zadovoljiva: proveravamo koja iskazna slova $p_{x,y}$ su tačna u svim zadovoljavajućim valuacijama. Ovu proveru izvodimo koristeći mogućnost MiniSat-a da proveri zadovoljivost formule pod pretpostavkom tačnosti ili netačnosti proizvoljnog iskaznog slova. Naime, $p_{x,y}$ će biti tačno u svim zadovoljavajućim valuacija formule ako i samo ako je formula uz pretpostavku o netačnosti $p_{x,y}$ nezadovoljiva.

  Za svako ovako određeno slovo $p_{x,y}$ zaključujemo da ukoliko je $T(0) = A$, onda važi $T(x) = y$. Stoga, ishod je oblika $T(0) = A, T(x_1) = y_1, T(x_2) = y_2, \ldots T(x_k) = y_k$ i predstavlja delimične informacije o funkciji T koja zadovoljava Π.

Vreme potrebno za izvršenje opisanog procesa na personalnom računaru je manje od jedne sekunde.

4.3 Rekonstrukcija ključa na osnovu vrednosti T

Pretpostavimo da je prethodni deo analize rezultirao otkrićem parova (x_i, Tx_i), $i = 1 \ldots n$. U nastavku tražimo sve ključeve $K \in \{0,1\}^{64}$ takve da važi $T_K(x_i) = Tx_i, i = 1 \ldots n$. Kako je prostor ključeva veličine 2^{64}, potpuna pretraga je u praksi neizvodljiva. Za skraćenje pretrage koristićemo sledeće tri tehnike:

- Preskakanje ekvivalentnih ključeva
- Eliminacije nekih vrednosti K_6K_7 na osnovu nemogućih izlaza C
- Napad susreta u sredini.

Sledeća tri odeljka sadrže detaljan opis ovih tehnika.

Preskakanje ekvivalentnih ključeva

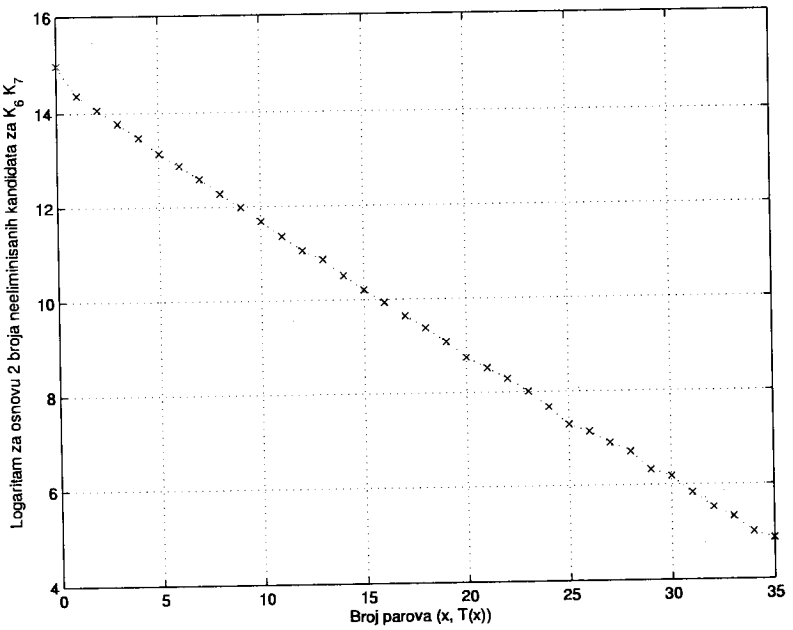
Prvo, uvedimo jednu opštu definiciju.

Definicija 2 *Za ključeve K i K' neke simetrične šifre reći ćemo da su* ekvivalentni *ukoliko funkcije šifrovanja za K i K' uzimaju jednake vrednosti na svim otvorenim tekstovima.*

Postojanje ekvivalentnih ključeva generalno je nepoželjna osobina. Međutim, ono što direktno kompromituje sigurnost šifre je postojanje ekvivalenata *svakog*

ključa K, koji su osnovu K još pritom efikasno izračunljivi. Upravo je to slučaj sa T^3.

Naime, ako je K proizvoljan ključ, lako je videti da za proizvoljno x, istovremeno komplementiranje $K_0^{(1)}, K_1^{(1)}$ ne menja $T_K(x)$. Isto važi i za $K_iK_{i+1}, i = 2, 4, 6$. Na taj način, zajedno sa K, dobijamo ukupno $2^4 = 16$ međusobno ekvivalentnih ključeva. Kako je gore navedena relacija očigledno relacija ekvivalencije, možemo govoriti o klasama ekvivalencije ključeva, kojih prema prethodnom ima najviše 2^{60}.



Slika 4.3: Zavisnost broja kandidata za K_6K_7 koji izdržavaju "test nemogućih" izlaza C i broja parova $(x, T(x))$.

Pretragu je sada dovoljno izvršiti po predstavnicima klasa, npr. fiksirajući $K_0^{(1)}, K_2^{(1)}, K_4^{(1)}, K_6^{(1)}$ na 0.

Eliminacija nekih K_6K_7 na osnovu nemogućih izlaza C

Koristeći činjenicu da $C : Z_{256} \to Z_{256}$ nije "na", neke od netačnih vrednosti K_6, K_7 jednostavno se eliminišu.

Naime, opisno rečeno, pretpostavku $K_6 = k_6, K_7 = k_7$ odbacujemo, ukoliko "odmotavanje $T(x)$ do odgovarajućeg izlaza C" daje vrednosti koje C nije mogla da proizvede. Preciznije, obeležimo sa $C^{-1}(y) = \{x|C(x) = y\}$ i posmatrajmo

[3]Odnosno, transformacija T posmatrana kao simetrična šifra ima upravo takve slučajeve ekvivalentnih ključeva.

skup

$$\{(c - k_7) \oplus k_6 - x_1 | c \in C^{-1}(Tx_1 - x_1)\}$$

indukovan parom (x_1, Tx_1). Ukoliko među vrednostima ovog skupa nema nijedne koja može biti izlaz iz C, odbacujemo kandidate k_6, k_7. Postupak ponavljamo za sve parove (x_i, Tx_i) i za svih 2^{15} kandidata k_6k_7. Na slici 4.3, prikazani su rezultati eksperimentalne ocene uspešnosti ovog metoda.

Napad "susreta u sredini"

Ovaj tip napada prvi put je objavljen krajem sedamdesetih godina, u [6]. Pretpostavimo da se proces šifrovanja neke simetrične šifre može predstaviti kompozicijom dve funkcije:

$$C = E_{K''}(e_{K'}(P)) \tag{4.2}$$

pri čemu su K' i K'' međusobno nezavisni podključevi ključa $K = (K', K'')$.

Ako je dato n parova (P, C), umesto potpunom pretragom, koja bi zahtevala najviše $n * 2^{|K'|+|K''|}$ operacija, možemo postupiti prema sledećem algoritmu:

- Formirati niz vrednosti $(K', (e_{K'}(P_1), \ldots, e_{K'}(P_n)))$ za svih $2^{|K'|}$ podključeva
- Za svaki od $2^{|K''|}$ podključeva K'':
  - Izračunati $(E^{-1}_{K''}(C_1), \ldots, E^{-1}_{K''}(C_n))$
  - Proveriti da li se ova vrednost nalazi u nizu: ako da, zabeležiti odgovarajući ključ $K = (K', K'')$.

Na ovaj način svi traženi ključevi nalaze se uz $n * (2^{|K'|} + 2^{|K''|})$ operacija šifrovanja i $2^{|K''|}$ operacija nalaženja elementa niza. Međutim, ova vremenska ušteda ide na račun prostora, jer je potrebno $n * 2^{|K'|}$ memorijskih polja da bi se ona realizovala.

U nastavku, dajemo efikasnu primenu SUS napada na funkciju T, preuzetu iz [2]. U svakom slučaju, prilagođavanje opšteg SUS napada na T iziskuje sledeće komentare:

- Inverz jedne runde funkcije T ne mora biti jednoznačno određena vrednost. Stoga, u nastavku, sa $E^{-1}_{K''}(Tx))$ označavaćemo skup inverza, a ne jedinstven inverz. Sada, provera iz drugog koraka SUS napada, da li je ovaj inverz prisutan u heš tabeli prerasta u proveru da li je bilo koji od inverza iz skupa $E^{-1}_{K''}(Tx))$ prisutan u heš tabeli.
- Postavlja se pitanje koju od mogućih dekompozicija (4.2) izabrati. Sa obzirom da je vremenska ušteda najveća u slučaju kada je $|K'| = |K''|$, a većina vrednosti za K_6K_7 eliminisana tehnikom eliminacije na osnovu nemogućih izlaza C, logično je "preseći" između K_2 i K_3, odnosno uzeti $K' = (K_0, K_1, K_2)$ i $K'' = (K_3, K_4, K_5, K_6, K_7)$.

Pre nego što navedemo sam algoritam, napomenimo da je zarad smanjenja pretrage za još 8 bita umesto kompozicijom od dve funkcije, T predstavljena kompozicijom na tri funkcije. Iz postupka biće jasno na koji način se ostvaruje ova ušteda.

Neka su dati parovi (x_i, Tx_i), $i = 1 \ldots 4$. Razlaganje T koje ćemo koristiti je $T(x) = E_{K_4K_5K_6K_7}(S_{K_3}(e_{K_0K_1K_2}(x)))$, gde je

- $e_{K_0K_1K_2}(x) = (C((x \oplus K_0) + K_1) + x) \oplus K_2$
- $S_{K_3}(y) = y + K_3$
- $E_{K_4K_5K_6K_7}(z) = C(((C(((C(z) + x) \oplus K_4) + K_5) + x) \oplus K_6) + K_7) + x$

Pošto će iz konteksta biti jasno o kojim se podključevima radi, pisaćemo kratko E, S i e. Sledi algoritam:

- Za svako $K_0K_1K_2$, takvo da je $K_0^{(1)} = K_2^{(1)} = 0$
  - Izračunati $(e(x_1), e(x_2), e(x_3), e(x_4))$
  - Tekuću vrednost $K_0K_1K_2$ staviti u tabelu veličine 2^{24} na poziciju $n = ((e(x_1) - e(x_4), e(x_2) - e(x_4), e(x_3) - e(x_4))$.
- Za svako $K_4K_5K_6K_7$, takvo da je $K_4^{(1)} = K_6^{(1)} = 0$.
  - Izračunati sve vrednosti skupa

$$Z = \{(z_1, z_2, z_3, z_4) | z_i \in E^{-1}(Tx_i)), i = 1 \ldots 4\}$$

  - Za svaku od nepraznih pozicija $m = (z_1 - z_4, z_2 - z_4, z_3 - z_4)$ gde $(z_1, z_2, z_3, z_4) \in Z$, zabeležiti odgovarajući ključ

$$(K_0, K_1, K_2, E^{-1}(Tx_1) - e(x_1), K_4, K_5, K_6, K_7)$$

Korektnost dobijenih kandidata sledi iz činjenice da će četvorka brojeva (e_1, e_2, e_3, e_4) biti jednaka četvorki $(E_1 + K_3, E_2 + K_3, E_3 + K_3, E_4 + K_3)$ za neko K_3 ako i samo ako je $(e_1 - e_4, e_2 - e_4, e_3 - e_4) = (E_1 - E_4, E_2 - E_4, E_3 - E_4)$. U tom slučaju, $K_3 = E_1 - e_1$.

Tako dobijamo kandidate za ključ koje dalje eliminišemo proverom zadovoljavaju li i ostale parove. Broj potrebnih operacija grubo iznosi $2^{24} - 2^{32}$. Eksperimentalni rezultati pokazuju da je u prethodnom odeljku predlozenih 13-18 kandidata dovoljno da ključ bude određen jednoznačno do na klasu ekvivalencije.

Implementacija

Ulaz za ovu fazu napada je niz parova (x_i, Tx_i), $i = 1 \ldots n$. Izlaz su svi ključevi $K \in \{0,1\}^{64}$ takvi da $T_K(x_i) = Tx_i, i = 1 \ldots n$, za koje važi $K_0^{(1)} = K_2^{(1)} = K_4^{(1)} = K_6^{(1)} = 0$.

Struktura podataka koju koristimo za skladištenje elemenata je heš tabela. Problem kolizija rešavamo metodom odvojenog nizanja, u kome svako polje tabele zapravo predstavlja listu, koja u slučaju kolizije ima više od jednog člana.

Da bismo opravdali izbor heš tabele kao strukture podataka za čuvanje međuvrednosti T, izveli smo sledeći eksperiment. Za svaku od 100 četvorki (x_1, x_2, x_3, x_4) koje smo slučajno generisali, formirali smo odgovarajuću heš tabelu listi i izračunali prosečnu dužinu neprazne liste u okviru svake tabele. Prosek ovih 100 prosečnih dužina iznosio je 6.875, na osnovu čega zaključujemo da je heš tabela dovoljno efikasna struktura podataka u ovom slučaju.

Na PC računaru sa procesorom od 1.2 Ghz i 256 Mb memorije vreme izvršavanja ovog procesa iznosi 1-5 minuta.

Glava 5

Zaključci i dalji rad

U ovoj tezi prikazali smo kako je moguće razbiti šifru CMEA za $n = 2$ (gde je n broj bajtova ulaza), uz pretpostavku o posedovanju $13 - 18$ parova (otvoreni tekst, šifrat). Implementacija napada izvedena je na jeziku C++, uz korišćenje SAT rešavača MiniSat i efikasno se izvršava na kućnom računaru. Autoru ovog teksta nije pozanto da je ranije objavljena implementacija napada na CMEA za bilo koje n. Iako je ovde pokriven samo slučaj $n = 2$, deo programa koji predstavlja napad na funkciju T ne zavisi od n i može se iskoristiti u eventualnim budućim implementacijama napada na CMEA za $n > 2$.

Analizu i razbijanje šifre izveli smo nezavisno od [2], uz ogradu da "susret u sredini" tehniku, koju smo ovde primenili na funkciju T, nismo razvili samostalno, već samo implementirali. Konstatujemo da smo došli do skoro identičnih rezultata kao u [2], pri čemu tamošnji napad zahteva 4 para, $2^{29} - 2^{37}$ operacija i 2^{24} 32-bitnih memorijskih lokacija, a napad prikazan u ovom tekstu $13 - 18$ parova, $2^{24} - 2^{32}$ operacija i 2^{24} 32-bitnih memorijskih lokacija. Dakle, istom tehnikom napada, na osnovu većeg broja parova, ostvarujemo manji broj operacija.

Kao što je već rečeno na kraju glave 2, opisani napad može se prirodno proširiti na $n > 2$. Lako se vidi da je na osnovu kôda (3.1) koji odgovara proizvoljnom n moguće formirati iskaz koji odgovara iskazu (4.1), odnosno iskaz koji određuje sve funkcije T koje zadovoljavaju date parove otvorenih tekstova i šifrata. Kako je to urađeno u slučaju $n = 3$, može se videti u [2]. Analogno je i za ostale n. Međutim, problem je što veličina odgovarajućeg iskaza, kao i vreme potrebno da se on reši, rastu sa n. Kako druga faza napada uvek ostaje ista, dalji rad mogao bi se sastojati od:

- Evaluacije vremenskih resursa i količine ulaznih podataka potrebnih za rešavanje iskaza analognih (4.1) za $n > 2$, odnosno $n > 3$ (slučaj $n = 3$ je rešen u [2] i zahteva 50-80 parova (otvoreni tekst, šifrat) i, zajedno sa drugom fazom napada, zahteva $2^{24} - 2^{32}$ operacija).

- Pokušaja primene SAT rešavača na logičke formule koje odgovaraju iskazu formiranom CMEA za $n > 2$, a analognom (4.1), iz razloga elegantnog

programerskog rešenja, odnosno prepuštanja zadatka "rešavanja" odgovarajućih iskaza SAT rešavaču.

U vezi sa drugom stavkom, problem je što se broj klauza i iskaznih promenljivih povećava. Maksimalan broj iskaznih promenljivih je $164^2 = 26896$, koliko ima mogućih iskaza oblika $T(x) = y$, gde $x, y \in Z_{256}$.

Literatura

[1] Telecommunications Industry Association, TR45.0.A, *Common Cryptographic Algorithms*, June 1995, Rev B

[2] D. Wagner, B. Schneier and J. Kelsey. *Cryptanalysis of the cellular message encryption algorithm*, Advances in Cryptology - CRYPTO'97, volume 1294 of Lecture Notes in Computer Science, pages 526-537. Springer-Verlag, 1997.

[3] Predrag Janičić, *Matematička logika u računastvu*, Matematički Fakultet, 2007. Beograd

[4] Niklas Eén, Niklas Sörensson *An Extensible SAT-solver*, Theory and applications of Satisfiability testing, number 2919 in Lecture Notes in Artificial Intelligence Series, p502-518, Springer Berlin/Heidelberg, 2004.

[5] M. Živković, *Primene računara*, Matematički fakultet, 2004. Beograd

[6] Whitfield Diffie, Martin Hellman *Exhaustive Cryptanalysis of the NBS Data Encryption Standard*, IEEE Computer 10(6), June 1977, pp74-84

[7] David Kahn, *Govor održan na pedesetogodišnjici postojanja National Security Agency (NSA)*, Amerika, 2002.

[8] Bruce Schneier, `http://www.schneier.com/paper-self-study.pdf` , elektronska publikacija, 1998.

[9] Aleksandar Kirćanski, `http://alas.matf.bg.ac.yu/~mr98224`, elektronska publikacija, 2006.

Thesis overview

Encryption algorithm CMEA (Cellular Message Encryption Algorithm) is a symmetric block cipher that was published in 1992, as a part of mobile communications standard, released by Telecomunications Industry Association in the US. It was not meant to protect voice communication. It's purpose was to protect sensitive control data, such as the digits dialed by the cell phone user.

The cipher was broken by Wagner, Schneier and Kelsey in 1997 [2]. CMEA can take any number of bytes n, where $n \geq 2$. For different n, CMEA takes different forms. The attack presented in [2] aimed at CMEA for $n = 2$ and $n = 3$, and could naturally be extended to $n > 3$ (although with unknown time complexity and amount of needed data). In the well-known cryptanalysis autotutorial by Schneier [8], which proposes several ciphers to be broken independently of known results for the purpose of obtaining skills in cryptanalysis, CMEA is listed as problem 6.23.

Two main tasks were conducted in this thesis:

- Cryptanalysis of CMEA for $n = 2$ independently of [2]
- Implementation of devised attack

The purpose of the first task was in cryptanalysis skills refinment by using existing and eventually inventing new techniques in this field [8]. Apart from consulting [2] for one important idea (and that is "meet-in-the-middle" attack which we did not reinvent), we independently rediscovered results presented in [2]. The attack presented in that paper requires 4 (plaintext, ciphertext) pairs, $2^{29} - 2^{37}$ operations and 2^{24} 32-bit memory locations. The attack presented in this thesis requires more pairs and less operations: 13-18 (plaintext, ciphertext) pairs, $2^{24} - 2^{32}$ operations and 2^{24} 32-bit memory locations.

As for the second task, an elegant implementation that uses SAT solver MiniSat is given (MiniSat is used in the first phase of an attack, where we transform the problem into a SAT problem). The attack given here requires around 13-18 (plaintext, ciphertext) pairs.