SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Pregled kriptografskih algoritama

Petar Afrić

JMBAG: 0036469979

Voditelj: prof. dr. sc. Marin Golub

Zagreb, Svibanj, 2014

Sadržaj

1. Uvod

1.1 Važnost kriptografije

Već od samih početaka ljudske civilizacije pojavila se potreba za sigurnom razmjenom poruka između ljudi odnosno ljudskih grupa. Elementi kriptografije bili su prisutni još u staroj grčkoj stoga ne čudi da je riječ kriptografija grčkog podrijetla te znači tajnopis. Spartanci su u 5. stoljeću prije Krista koristili kriptosustav zvan skital. Skital se sastojao od štapa na kojem je bila namotana vrpca pergamenta po kojoj se okomito pisala poruka. Nakon što je poruka bila napisana pergament bi se uklonio s štapa te bi preostala vrpca s izmiješanim slovima na sebi. Jedini način za pročitati poruku bio je namotavanje vrpce na štap jednake debljine. Rimski car i vojskovođa Gaj Julije Cezar šifrirao je svoje poruke tako da bi slovo zamijenio onim koje se nalazi 3 mjesta ispred njega u abecedi uz pretpostavku da je abeceda cikličke naravi. Danas se sve šifre temeljene na ovakvom modelu enkripcije nazivaju Cezarovim šiframa. Takve šifre se još nazivaju i monoalfabetskim šiframa jer svako slovo ima točno predodređen znak s kojim se zamjenjuje. S vremenom su nastale i polialfabetske šifre kod kojih se slovo mijenja s nekim od m mogućih znakova ovisno o načinu šifriranja. Primjer takve šifre je Vigenéreova šifra kod koje se slovo zamjenjuje slovom koje se nalazi k mjesta ispred njega, a k se izračunava kao ostatak dobiven prilikom dijeljenja pozicije slova u tekstu i ključa m. Daljnji razvoj kriptografije donio je zamjenu blokova slova među prvima od takvih sistema je Playfairova šifra koja zamjenjuje parove slova. Lester Hill je 1929. Godine osmislio kriptosustav koji je omogućio zamjenu m slova izvornog teksta s m slova šifrata. Ukoliko duljina teksta nije djeljiva s m potrebno ga je nadopuniti kako bi se provela podjela na blokove. Potom se konstruira invertibilna matrica m×m. U svakom bloku svakom slovu se odredi njegov numerički ekvivalent koji je jednak rednom broju slova u abecedi minus jedan. Nakon toga se blok, koji predstavlja vektor s m vrijednosti, množi s matricom m, a u dobivenom vektoru se dobivene vrijednosti mijenjaju ostatkom njihova dijeljenja s brojem slova u abecedi. Novo dobivene vrijednosti se prevode u alfabetski oblik. Ovaj način šifriranja naziva se Hillova šifra. Vrijedno je napomenuti da se tokom povijesti nisu svi načini šifriranja temeljili na supstituciji elemenata izvornog teksta s elementima šifrara. Postojali su i transpozicijski načini šifriranja čiji je temelj bilo u zamjeni međusobnog položaja elemenata izvornog teksta. Primjer naprave koja je omogućavala ovakvo šifriranje je Jeffersonov kotač. On se sastojao od 26 pokretnih cilindara s 26 slova na sebi. Jedan od redaka sadržavao je smisleni tekst dok je bilo koji od preostalih 25 redaka mogao poslužiti kao šifrat. Šifrat bi se poslao osobi s identičnim kotačem koja bi u jednom od redaka namjestila šifrat i nakon toga među preostalim redcima potražila onaj s smislenom porukom. Tokom drugog svjetskog rata korišteni su strojevi za kriptiranje ENIGMA i M-209. Enigma se sastojala od tipkovnice s 26 tipki i zaslona od 26 žaruljica, 3 mehanička rotora i električne prespojne ploče. Pritiskom na tipku upalila bi se žaruljica koja je predstavljala šifrat tog slova. Lako se može primijetiti kako je siguran prijenos informacija bio izuzetno bitan kroz civiliziranu ljudsku povijest. U današnjem dobu globalizacije informacije postaju važnije nego ikada prije. Zajedno s njihovim značajem raste i potreba za njihovom sigurnom razmjenom. Iz tog razloga u zadnjih 40-ak godina došlo je do rapidnog razvoja kriptografije[2].

1.2 Osnovni kriptografski pojmovi

Kriptografija je moderna znanost, bazirana na primijenjenoj matematici, koja se bavi proučavanjem i primjenom tehnika sigurne komunikacije u prisutnosti treće strane[1][2]. Cilj kriptografije je razvoj i implementacija algoritama čijom primjenom se postiže zadovoljavanje sigurnosnih zahtijeva:

- Povjerljivost-Informacije u sustavu mogu biti na raspolaganju samo ovlaštenim osobama.
- Raspoloživost-Informacije moraju biti na raspolaganju ovlaštenim osobama bez obzira na okolnosti.
- Besprijekornost-Samo ovlaštene osobe smiju manipulirati podacima
- Autentičnost-Sustav mora biti u stanju jednoznačno prepoznati ovlaštene osobe.
- Autorizacija-Ovlaštenim osobama dopušta se pristup samo do onih informacija koje su im potrebne za obavljanje njihove uloge u sustavu.
- Neporecivost-Korisnici ne mogu opovrgavati svoje akcije u sustavu.

Navedene sigurnosne zahtjeve postavljeni su kako bi spriječili: prisluškivanja, prekidanja, promjene sadržaja informacija, izmišljanja informacija, lažno predstavljanje te poricanje. Ukoliko dvije sobe razmjenjuju poruke odnosno informacije te se na njihov komunikacijski kanal to jest put kojim putuje informacija postavi treća osoba koja poruke čita dolazi do prisluškivanja. Ako treća osoba uđe u komunikacijski kanal ona može prekinuti komunikaciju između prvih dviju osoba, mijenjati sadržaje poruka koje putuju komunikacijskim kanalom te uspostaviti komunikaciju s jednom od osoba predstavljajući se kao ona druga te tako slati maliciozne poruke. Takve poruke mogu sadržavati i štetne programe. Zloćudne programe možemo podijeliti u četiri skupine:

- Virusi-programski odsječci koji se infiltriraju u postojeće programe te štetno djeluju.
- Crvi-cjeloviti programi koji djeluju destruktivno.
- Trojanski konj-program koji naočigled obavlja koristan posao, ali sadrži i funkcije štetna djelovanja.
- Rootkiti-programi koji svoje djelovanje skrivaju nadomještajući sustavske procese i podatke.

Kriptiranje omogućuje zadovoljavanje svih sigurnosnih zahtijeva osim raspoloživosti. Kriptiranje je proces sustavnog prevođenja razumljivog teksta u šifrirani oblik, a dekriptiranje je njemu obrnut proces. Sustavi koji omogućuju kriptiranje nazivaju se kriptosustavi. Kriptosustave dijelimo u dvije osnovne skupine:

- Simetrični kriptosustavi-za kriptiranje i dekriptiranje koriste jednake ključeve.
- Asimetrični kriptosustavi-za kriptiranje i dekriptiranje koriste različite ključeve.

2. Simetrični kriptografski algoritmi

2.1 Osnovne značajke simetričnih kriptosustava

Glavna odlika simetričnih kriptosustava je korištenje jednog ključa za kriptiranje i dekriptiranje. Korisnici ovakvog sustava moraju prije komunikacije razmijeniti ključ

Tablica 2-1 Kriptiranje XOR-om

Poruka	1	0	1	1	0	1
Ključ	0	1	1	0	0	1
Kriptirana poruka	1	1	0	1	0	0

Šifriranje se najčešće provodi primjenom operacije isključivo ili između blokova izvornog teksta i ključa šifriranja, kao što je pokazano u tablici 2-1 preuzetoj iz [1] Ipak samom primjenom operacije isključivo ili sustav postaje ranjiv na napade "poznavanja izvornog teksta". Ukoliko je poznat izvorni tekst i njegov kriptirani oblik operacijom isključivo ili između njih dvoje dobiva se ključ šifriranja. Ovaj sigurnosni rizik moguće je ublažiti tako da se prilikom svakog kriptiranja koristi različiti ključ. Takav način kriptiranja se naziva jednokratnom bilježnicom. Princip jednokratne bilježnice je nepraktičan jer zahtijeva od korisnika razmjenu teoretski beskonačnog broja ključeva. Kako bi bilo moguće koristit samo jedan ključ razvijeni su kompleksni algoritmi za kriptiranje[1][2].

2.2 Data Encryption Standard/DES

Data Encryption Standard algoritam osmislio je IBM-ov kriptografski tim 1973. godine. 1976. godine nakon nekih promjena prihvaćen je kao standard[1][2]. Algoritam kriptira poruke u blokovima od 64 bita koristeći 56 bitni ključ K. Izvorni tekst se početno podjeli na blokove od 64 bita. Ukoliko je potrebno tekst se na kraju nadopuni tako da i posljednji blok bude 64 bitne duljine. Svaki blok se zasebno kriptira primjenom idućih koraka algoritma:

1. Izvorni blok se permutira primjenom specifične inicijalne permutacije *IP*.

$$IP(BLOK) \tag{2.1}$$

2. Rezultat permutacije se dijeli na lijevi i desni dio, svaki veličine 32 bita.

$$IP(BLOK)=L_0D_0 \tag{2.2}$$

3. Na dobivene pod-blokove se primjenjuje šesnaest iteracija funkcije transformacije $F_1()$. K_i su nizovi od 48 bitova koji se dobivaju permutacijom ključa *K*.

$$F_1(L_{i-1}, D_{i-1})=\{ \quad (2.3)$$
$$L_i=D_{i-1}$$
$$D_i=L_{i-1} \oplus f(D_{i-1},K_i) \text{ ,i e } \{1,...,16\} \ \}$$

4. Konačan šifrat dobiva se primjenom inverzne inicijalne permutacije IP^{-1}

$$IP^{-1}(D_{16}L_{16}) \quad (2.4)$$

Lako je primijetiti kako su funkcija $f(D_{i-1},K_i)$ i određivanje među-ključeva *Ki* ključni dijelovi DES algoritma te je potrebno njihovo detaljnije razmatranje. Računanje među-ključeva obavlja se izvođenjem jednostavnog algoritma, a rezultati se pohranjuju u tablicu ključeva. Originalni ključ *K* zapravo se sastoji od 64 bita. Svaki osmi bit je paritetne prirode i postavlja se tako da njegov byte sadrži neparan broj jedinica. Prilikom izvođenja algoritma ovi bitovi se zanemaruju. Algoritam je moguće podijeliti na tri glavne iteracije:

I. Zanemarujući paritetne bitove ključ se permutira primjenom specifične permutacije $PC_1()$.

$$PC_1(K) \quad (2.5)$$

II. Rezultat permutacije dijeli se na lijevu i desni dio, svaki veličine 28 bita.

$$PC1(K)=C_0E_0 \quad (2.6)$$

III. Na dobivene pod-blokove se primjenjuje šesnaest iteracija funkcije transformacije $F_2()$ svaka dajući jedan među-ključ kao rezultat.

$$F_2(C_{i-1},E_{i-1})=\{ \quad (2.7)$$
$$C_i=LS_i(C_{i-1})$$
$$D_i=LS_i(D_{i-1})$$
$$K_i=PC_2(C_iD_i),$$
$$LS_i()=\{\text{"Ciklički pomak lijevo za n mjesta",}$$
$$i=\{1,2,9,16, n=1$$
$$\text{ostalo, } n=2\} \ \}$$
$$PC_2()=\{\text{"Fiksna kompresijska permutacija"}\} \ \}$$

Ključevi dobiveni ovim postupkom sudjeluju u izvođenju funkcije $f(D_{i-1},K_i)$ koje se sastoji od četiri koraka:

1) Kako bi oba operanda bili jednake duljine D_{i-1} se proširuje na 48 bita korištenjem funkcije proširenja $E(D_{i-1})$. Funkcija $E(D_{i-1})$ šesnaest bitova duplicira.

$$E(D_{i-1}) \tag{2.8}$$

2) Nad operandima K_i *i* $E(D_{i-1})$ provodi se funkcija isključivo ili, te na taj način dobiva niz B_i koji se sastoji od osam šest-bitnih pod-nizova $B_i = B_{i1} \ldots B_{i8}$.

$$B_i = E(D_{i-1}) \oplus K_i \tag{2.9}$$

3) Na dobiveni niz B_i primjenjuje se osam matrica dimenzija 4×16 koje sadrže elemente od 0 do 15,a ih nazivamo *S-kutijama*. Svaka se primjenjuje na točno jedan pod-niz. Od šest bitova pod-niza $B_{ij}=b_{ij1}b_{ij2}b_{ij3}b_{ij4}b_{ij5}b_{ij6}$ $b_{ij1}b_{ij6}$ određuju binarni zapis retka matrice, a $b_{ij2}b_{ij3}b_{ij4}b_{ij5}$ određuju binarni zapis stupca matrice. Element koji se nalazi na njihovom presjecištu tumačimo kao binarni broj duljine četiri C_{ij} te ga uzimamo kao šifru za B_{ij}.

$$C_{ij}=S_j\text{-}kutija[b_{ij1}b_{ij6}, b_{ij2}b_{ij3}b_{ij4}b_{ij5}], \quad j=\{1,\ldots,8\} \tag{2.10}$$

$$C_i=C_{i1}C_{i2}C_{i4}C_{i5}C_{i6}C_{i7}C_{i8}$$

4) Dobiveni niz bitova $C_i=C_{i1}C_{i2}C_{i4}C_{i5}C_{i6}C_{i7}C_{i8}$ duljine je 32-bita. Sada se još na njega primjenjuje konačna permutacija $P()$.

$$P(C_i) \tag{2.11}$$

Korištene *S-kutije* ne mogu biti bilo kako ispunjene već postoji pet kriterija koji moraju biti zadovoljeni:

i. Svaki redak mora sadržavati sve brojeve od 0 do 15.
ii. Niti jedna *S-kutija* ne smije biti linearna ili afina funkcija ulaznih podataka.
iii. Promjena jednog bit ulaznog podatka nakon primijene *S-kutije* rezultira promjenom barem dva bit izlaznog podatka.
iv. Za svaku *S-kutiju* i svaki ulazni podatak B_{ij} rezultati od $S[B_{ij}]$ i $S[B_{ij} \oplus 001100]$ se razlikuju u barem dva bita
v. Za svaku *S-kutiju* i svaki ulazni podatak B_{ij} i sve e,f e $\{0, 1\}$ vrijedi $S[B_{ij}] \neq S[B_{ij} \oplus 00ef00]$.

Završna permutacija $P()$ ima zadatak povećanja difuzije kriptosustava te mora zadovoljavati tri kriterija:

1. Četiri izlazna bita iz svake *S-kutije* moraju utjecati na šest različitih *S-kutija* u idućoj rundi.
2. Četiri izlazna bit iz svake *S-kutije* u *i*-toj rundi su distribuirana tako da dva od njih utječu na središnje bitove, a dva na krajnje bitove u (*i*+1)-voj rundi.
3. Za dvije *S-kutije* S_k *i* S_j mora vrijediti da ako neki izlazni bit od S_j utječe na neki od središnjih bitova S_k u idućoj rundi, onda niti jedan od izlaznih bitova S_k ne utječe na središnje bitove od S_j.

DES algoritam ima svojstvo da mala promjena ulaznih podataka vuče velike promjene u izlaznim podacima. Ovo svojstvo ponekad se naziva "efekt lavine" i izuzetno je poželjno u kriptografiji jer povećava otpornost na napade diferencijalnom kriptoanalizom.

2.3 Triple Data Encryption Standard/3DES

Triple Dana Encryption Standard odnosno 3DES nije ništa više nego primjena DES algoritma 3 puta koristeći tri različita ključa K_1, K_2 *i* K_3.

$$3DES(BLOK,K_1, K_2,K_3)=DES(DES^{-1}(DES(BLOK, K_1), K_2), K_3) \quad (2.12)$$

Ovako implementiran algoritam omogućuje korištenje 3DES-a poput DES ako se za sva tri ključa odabere identična vrijednost[1][2].

2.4 Advanced Encryption Standard/AES

Godine 1997. NIST je raspisao natječaj za razvoj novog enkripcijskog standarda. DES odnosno 3DES naprosto nije zadovoljavao sve zahtjeve tehnoloških tokova,a i sigurnost DES sustava je postajala upitna. 3DES-u je za kriptiranje potrebno 48 iteracija za postići sigurnost koja bi se vjerojatno mogla postići u 32 iteracije, 64-bitni blokovi nisu uvijek praktični za rad i prespor je za neke primjene, kao što je digitalna obrada video signala. Novi algoritam mora je zadovoljavati sljedeće zahtjeve:

- Morao je biti simetričan.
- Mora je kriptirati 128-bitne blokove.
- Mora je raditi s ključevima duljine 128, 192 ili 256 bitova.
- Izvorni tekst algoritma treba je biti javno dostupan.

Pobjednik natječaja bio je Rijandel te tako dobio naziv Advanced Encryption Standard[1][2]. Za potrebe kriptiranja AES koristi konačno Galoisovo polje oblika $GF(2^8)$ to jest polje konačnog broja elemenata gdje su elementi polinomi oblika $a_7x^7+ a_6x^6+ a_5x^5+ a_4x^4+ a_3x^3+ a_2x^2+ a_1x+ a_0$, $a_i=\{0,1\}$. Nad poljem su definirane operacije zbrajanja i množenja modulo fiksni ireducibilni polinom $x^8+ x^4+ x^3+x+1$. 8-bitne podatci se prikazuju u obliku polinoma npr. podatku 10011101 odgovara polinom $x^8+ x^5+x^4+ x^3+1$. AES kriptira blokove od 128 bitova odnosno 16 bajtova uz pomoć ključa duljine 128 bitova. Svaki bajt bloka interpretira se kao element

$$57_H * 83_H = C1_H \tag{2.13}$$

$$57_H = 01010111_2 = x^6 + x^4 + x^2 + x + 1$$

$$83_H = 10000011_2 = x^7 + x + 1$$

$$(x^6 + x^4 + x^2 + x + 1) * (x^7 + x + 1) =$$

$$x^{13} + \_ + x^{11} + \_ + x^9 + x^8 + x^7 + \_ + \_ + \_ + \_ + \_ + \_ + \_$$

$$\oplus \_ + \_ + \_ + \_ + \_ + \_ + x^7 + \_ + x^5 + \_ + x^3 + x^2 + x + 1$$

$$\oplus \_ + \_ + \_ + \_ + \_ + \_ + \_ + x^6 + \_ + x^4 + \_ + x^2 + x + 1$$

$$x^{13} + \_ + x^{11} + \_ + x^9 + x^8 + \_ + x^6 + x^5 + x^4 + x^3 + \_ + \_ + 1$$

$$(x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1) / (x^8 + x^4 + x^3 + x^2 + x + 1) = x^5 + x^3$$

$$x^{13} + \_ + x^9 + x^8 + x^6 + x^5$$

$$\_ + x^{11} + \_ + \_ + \_ + \_ + x^4 + x^3 + 1$$

$$\_ + x^{11} + \_ + \_ x^7 + x^6 + \_ + x^4 + x^3$$

$$\_ + \_ + \_ + x^7 + x^6 + \_ + \_ + \_ + 1 = C1_H$$

4×4 matrice oblika Galiosova polja. Tu matricu nazivamo AES-blok. Algoritam kriptiranja bloka se odvija u 10 iteracija, a iteracije se sastoje od 4 dijela:

1. Subbytes zamjenjuje svaki element AES-bloka s njegovim GF(2^8) inverzom te potom na svaki bit bloka primjenjuje fiksu afinu transformaciju. Transformaciju je moguće prikazati pomoću S-kutije. Prikazujući elemente AES-bloka u heksadecimalnom obliku prvi broj moguće je tumačiti kao oznaku reda, a drugi kao oznaku stupca S-kutije. Element se zamjenjuje s onim elementom S-kutije koji se nalazi na presjecištu označenog reda i stupca.

$$b_i^{'} = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i \tag{2.14}$$
$$i = (1,2,...,8), c = 01100011$$

2. Shiftrows ciklički posmiče redove novo dobivenog bloka ulijevo za i mjesta, gdje je i broj reda nad kojim se vrši operacija.
3. Mixcolums miješa stupce AES-bloka. Elementi stupca se tumače kao polinom oblika $a_{3i}x^3 + a_{2i}x^2 + a_{1i}x^1 + a_{0i}$ te GF(2^8)-množe polinomom $03x^3+01x^2+01x+02$. Konačan rezultat dobiva se primjenom modulo x^4+1 na rezultat množenja polinoma. U posljednjoj iteraciji miješanje stupaca se izostavlja.
4. Addroundkey provodi XOR operaciju nad AES blokom s međuključem odgovarajuće runde. Ovaj korak provodi se dodatno prije prve iteracije.

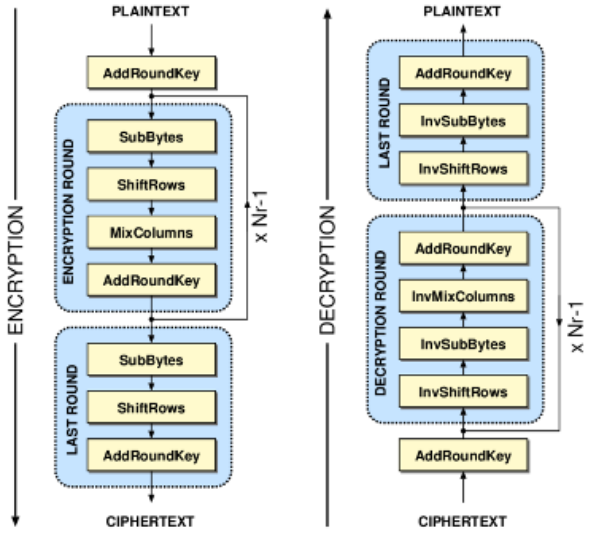
Konstrukciju međuključeva za posljednji dio algoritma može se podijeliti na dva dijela:

1) Proširenje ključa koje se ostvaruje pomoću XOR-a i cikličkog pomaka. Prva četiri bajta proširenog ključa predstavljaju izvorni ključ. Daljnji bajtovi r_i se konstruiraju kao XOR r_{i-1} i r_{i-4}. Ukoliko se radi o iteraciji u kojoj je i višekratnik od 4 prije XOR-a

se primjenjuje rotacija bajta jedno mjesto ulijevo, na svaki bajt riječi se primjenjuje S-kutija te XOR s riječi [$02^{(i-4)/4}$,00,00,00]. Ključ se proširuje do 44 bajta.

2) Iz proširenog ključa međuključevi se biraju tako da prva četiri bajta proširenog ključa predstavljaju prvi međuključ, druga četiri drugi međuključ itd.

Za razliku od DES algoritma kod AES-a su jasno definirana pravila konstrukcija S-kutija. S-kutija se generira određujući multiplikativni inverz dobivenog broja iz GF(2^8). Cilj S-kutija je što veća otpornost sustava na diferencijalnu i linearnu kriptoanalizu dok miješanje stupaca i posmak redova služe kako bi povećali difuziju sustava. Dekriptiranje šifranta odvija se na isti način kao i šifriranje izvornog teksta. Jedina razlika je što se koriste inverzi operacija te se primjenjuju obrnutim redoslijedom. Slika 2-1 grafički prikazuje kriptiranje i dekriptiranje pomoću AES algoritma. Slika je preuzeta s adrese: http://www.iis.ee.ethz.ch/~kgf/acacia/fig/aes.png .



Slika 2-1 AES

3. Asimetrični kriptografski algoritmi

3.1 Osnovne značajke asimetričnih kriptosustava

Za razliku od simetričnih kriptosustava gdje se kriptiranje i dekriptiranje provodi uz primjenu samo jednog ključa kod asimetričnih kriptosustava svaki postupak ima svoj ključ. Ovakav način kriptiranja zasniva se na teoriji brojeva te je za njegovo razumijevanje potrebno poznavanje njenih osnovnih koncepata[1][2][4].

- Djeljivost-a je djeljiv s d ako je a višekratnik od d. Najveći takav d je a, a najmanji 1. Ove djelitelje nazivamo trivijalnim djeliteljima. Preostali djelitelji se nazivaju faktori broja a.
- Teorem dijeljenja kaže da za svaki cijeli broj a i prirodni broj n postoje brojevi q i r tako da vrijedi: a=q*n+r.
- Ekvivalentnost po modulu tj. kongruentnost brojeva kaže da je broj a kongurentan broju b po modulu n ako vrijedi: a mod n=b mod n.
- Brojevi a i b su relativno prosti ukoliko nemaju zajedničkih faktora.
- Eulerova phi funkcija. Skup $Z_n=\{0,1,\ldots,n-1\}$ je prsten u kojem su definirane operacije zbrajanja, oduzimanja i množenja po modulu n. Z_n^* je podskup od Z_n koji se sastoji od elemenata koji su relativno prosti u odnosu na n. Broj takvih elemenata u skupu Z_n^* naziva se njegova kardinalnost i jednaka je Eulerovoj phi tj totient funkciji φ(n). Ako je n prosti broj p tada je φ(p)=p-1. Ako n nije prosti broj, $n = p_1^{e1} * p_2^{e2} * ... * p_{k-1}^{ek-1} * p_k^{ek}$ onda vrijedi:

$$\varphi(n)=n*(1-\frac{1}{p_1})*(1-\frac{1}{p_2})*...*(1-\frac{1}{p_k}) \tag{3.1}$$

Ako n=q*p vrijedi:

$$\varphi(n)=n*(1-\frac{1}{p})*(1-\frac{1}{q})=(p-1)*(q-1) \tag{3.2}$$

- Modularno potenciranje. $D=b^a \bmod n$, gdje je $a_m a_{m-1} a_{m-2} \ldots a_1 a_0$ binarni prikaz broja a, moguće je izračunati algoritmom:

```
for(i=m,d=1;i>=0;i--){
    d=(d*d)%n;
    if(a[i]==1){
    d=(d*b)%n;
    }}
```
(3.3)

- Eulerov teorem kaže da za svaki prirodni broj n>1 vrijedi:

$$a^{\varphi(n)} \bmod n = 1, \text{ za svaki a element od } Z_n^* \quad (3.4)$$

- Mali Fermatov teorem kaže da za proste brojeve p vrijedi:

$$a^{p-1} \bmod p = 1, \text{ za svaki a element od } Z_n^* \quad (3.5)$$

Što za posljedicu ima:

$$a^p \bmod p = a, \text{ za svaki a element od } Z_p \quad (3.6)$$

- Ako je a osnovni korijen od Z_n^* i b element od od Z_n^* postoji x tako da vrijedi:

$$a^x \bmod n = b \quad (3.7)$$

Broj x se naziva diskretni logaritam tj. indeks broja b (mod n) u odnosu na bazu a.

- Kineski teorem ostatka kaže da za $n=n_1 * n_2 * \ldots * n_{k-1} * n_k$, gdje su svi dvočlani podskupovi faktora relativno prosti, struktura od Z_n je jednaka Kartezijevu produktu $Zn_1 * Zn_2 * \ldots * Z_{nk.}$ U praktičnoj primjeni to znači da za $n_1=p$ i $n_2=q$ za bilo koja dva cijela broja x i a vrijedi:

$$x \bmod p = a \quad (3.8)$$
$$x \bmod q = a$$

ako i samo ako vrijedi:

$$x \bmod n = a \quad (3.9)$$

Uz pomoć navedenih koncepata teorije brojeva konstruiraju se dva različita ključa, ali ipak matematički povezana. Jedan ključ je javni i služi za enkripciju izvornog teksta, dok je drugi tajni odnosno privatni. Bilo tko može javnim ključem kriptirati tekst, ali ga samo osoba s privatnim ključem može lakoćom dekriptirati, bilo kome drugom bilo bi potrebno ulaganje neisplativo velikih napora. To je osnovni koncept asimetrične kriptografije.

3.2 RSA kriptosustav

Sigurnosni algoritam RSA razvijen je 1977. godine[1][2]. Ime je dobio kao kombinaciju imena svojih tvoraca. Algoritam se zasniva na teškoći faktorizacije velikih prirodnih brojeva iako se tokom samog kriptiranja i dekriptiranja koristi modularno potenciranje. RSA kriptosustav definiran je kao:

$$n = p * q, \text{ gdje su p i q prosti brojevi} \tag{3.10}$$

$$P = C = Z_n \text{ te vrijedi:}$$

$$K = \{(n, p, q, d, e) : n = p * q, (d * q) \bmod \varphi(n) = 1 \}$$

$$\text{Za i } e \ K:$$

$$e_i(x) = x^e \bmod n, \quad d_i(y) = y^d \bmod n, \quad x, y \, e \, Z_n$$

Vrijednosti n i e su javni ključ, a vrijednosti p,q i d su tajni ključ.

Funkcije $e_i(x)$ i $d_i(y)$ su međusobno inverzne tj. $e_i(d_i(x))=x$. Provođenje RSA algoritma može se podijeliti u 3 dijela.

1) Generacija ključeva. Odabiru se dva velika prosta broja p i q, tako da je $p,q>10^{150}$. Brojevi p i q ne smiju biti sličnih vrijednosti jer to olakšava njihovo određivanje. Isto tako je poželjno da $p \pm 1$ i $q \pm 1$ sadrže barem jedan veliki prosti faktor jer postoje efikasne metode faktoriziranja brojeva koji sadrže samo male proste faktore. Nakon određivanja p i q izračunava n=p*q, a potom $\varphi(n)$. Odabire se broj $e<\varphi(n)$, takav da je relativno prost u odnosu na $\varphi(n)$. Uz pomoć e i $\varphi(n)$ se izračunava d, tako da vrijedi:

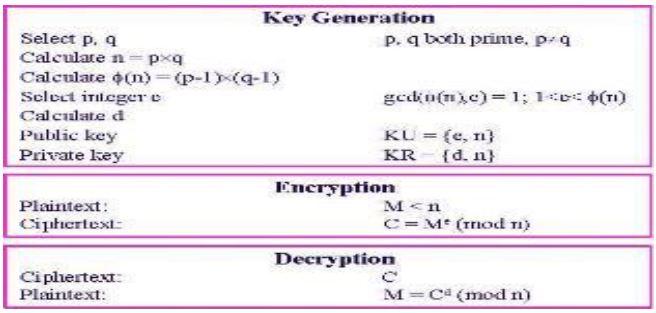
$$e*\text{d mod } \varphi(n)= 1 \tag{3.11}$$

2) Šifriranje. Poruka x<n se kriptira javnim ključem.

$$\text{y}= x^e \bmod n \tag{3.12}$$

3) Dešifriranje. Šifrant y se dekriptira privatnim ključem.

$$\text{x}= y^d \bmod n \tag{3.13}$$



Slika 3-1 RSA

Slika 3-1 prikazuje osnovne korake prilikom generacije ključa, enkripcije i deskripcije pomoću RSA algoritma, a preuzeta je s adrese:

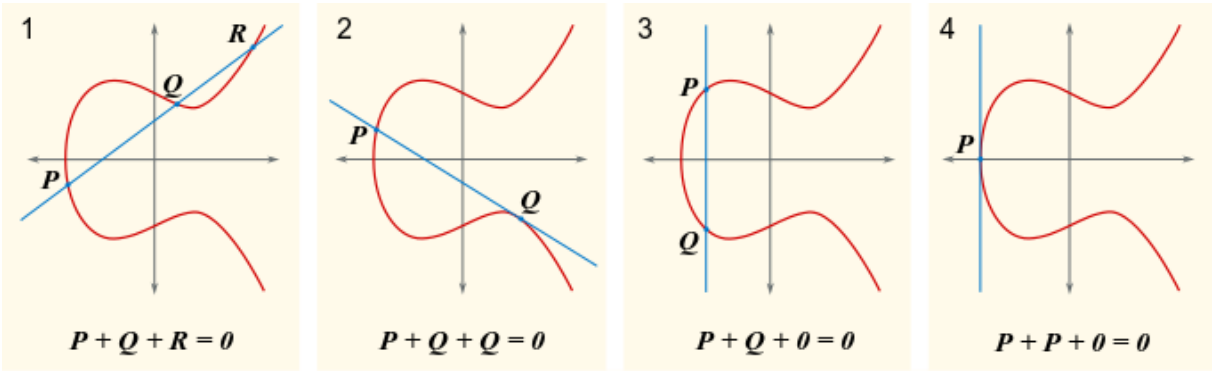
http://www.arl.wustl.edu/~jl1/education/cs502/images/rsa_algorithm.JPG Za efikasnost RSA sustava važne su činjenice da je faktorizacija brojeva od već 1024-bita neizvediva u razumnom vremenu, a modularno potenciranje veoma efikasno izvedivo uz pomoć prethodno navedenog algoritma. Razlika složenosti navedenih postupaka ključ je uspjeha RSA algoritma.

3.3 Elliptic curve cryptography / ECC

Kriptografija pomoću eliptičnih krivulja ušla je u masovnu uporabu 2004. godine. Ovaj tip kriptografije bazira se na algebarskoj strukturi eliptičnih krivulja u konačnim poljima[2][6]. Za razliku od RSA sustava koji svoju sigurnost bazira na problemu faktorizacije velikih prirodnih brojeve kod eliptičnih krivulja sigurnost je bazirana na problemu nalaženja diskretnog logaritma nasumičnog elementa eliptične krivulje. Za trenutne kriptografske potrebe uzima se eliptična krivulja čiji elementi zadovoljavaju jednadžbu[3.14], zvanu kratka Weierstrassova formula, zajedno s točkom u beskonačnosti, a one su definirane nad poljem K karakteristike različite od dva ili tri:

$$y^2 = x^3 + ax + b \qquad (3.14)$$

$$a, b \in K$$



Slika 3-2 Operacije nad eliptičnim krivuljama

Navedeni polinom ne smije imati višestruke korijene. Važno svojstvo eliptičnih krivulja je da se nad njima mogu definirati operacije uz koje one postaju Abelove grupe. Operacije nad eliptičnim krivuljama ilustrirane su na slici 3-2 koja je preuzeta s adrese: http://upload.wikimedia.org/wikipedia/commons/thumb/c/c1/ECClines.svg/680px-ECClines.svg.png .Glavna prednost ECC kriptografije je manja veličina ključa čime se smanjuju memorijske potrebe te povećava brzina prijenosa. Odnos veličina ključeva prilikom primjene ECC kriptografije, RSA algoritma i AES algoritma prikazan je na slici 3-3 koja je preuzeta s adrese: http://m.eet.com/media/1076009/0206LambertTbl01.gif . 256-bitni ECC ključ pruža sigurnost ekvivalentnu 3072-bitnom RSA ključu. Najčešće korišteni ECC algoritmi današnjice su:ECDH, ECES, MQV i Menezes-Vanstone.

Elliptic-Curve Digital Signature Algorithm (ECDSA)

NIST Guidelines for Public Key Sizes for AES			
ECC key size (bits)	RSA key size (bits)	Key size ratio	AES key size (bits)
163	1,024	1:6	
256	3,072	1:12	128
384	7,680	1:20	192
512	15,360	1:30	256

Supplied by NIST to ANSIX9F1

Slika 3-3 Veličine ključeva jednake sigurnosne razine

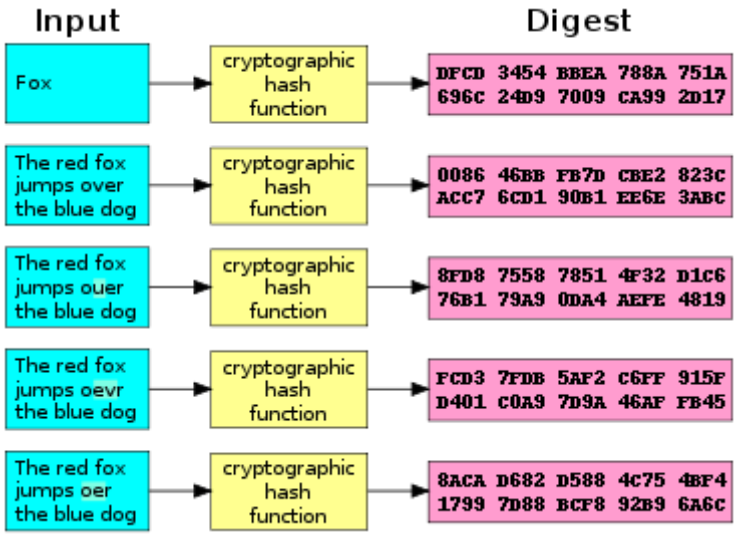
4. Funkcije sažimanja

4.1 Osnovne značajke funkcija sažimanja

Funkcije sažimanja/ hash funkcije ulazni niz podataka transformiraju u bitovni niz fiksne duljine tako da i najmanje promjene u ulaznom nizu gotovo u potpunosti mijenjaju izlazni bitovni niz[1][2][3]. Izlazni nizovi najčešće se prikazuju u heksadecimalnom obliku. Idealne hash funkcije zadovoljavaju četiri glavna uvjeta:

- Lako je generirati hash tj. izlazni bitovni niz, za svaku poruku bez obzira na njenu duljinu.
- Nemoguće je u razumnom vremenu iz hash-a generirati izvornu poruku.
- Nemoguće je promijeniti poruku bez da se promijeni hash.
- Nemoguće je u naći dvije poruke koje imaju isti hash.

Njihove primjene zalaze i van područja računalne sigurnosti pa su tako često korištene za uspoređivanje i brzo pretraživanje. U kriptografiji hash funkcije se najviše koriste za potrebe digitalnih potpisa i autentifikacije. Slika 4-1 pokazuje osnovnu ideju rada hash funkcije, a preuzeta je s adrese: http://en.wikipedia.org/wiki/Cryptographic_hash_function



Slika 4-1 Rad hash funkcije

4.2 SHA-2

Sigurnosni hash algoritam tj. SHA-2 je zapravo naziv za nekoliko algoritama: SHA-224,SHA-256, SHA-384, SHA-512, SHA-512/256 [2][3]. Njihove oznake označavaju duljine njihovih hash-eva. Algoritam je napravila Američka nacionalna sigurnosna agencija 2001. godine nakon što je otkriven sigurnosni propust u njegovom prethodniku SHA-1. Navedeni SHA-2 algoritmi međusobno se malo razlikuju stoga je za razumijevanje njihova rada dovoljna analiza jednog od njih, npr. SHA-256. Izvođenje SHA-256 algoritma moguće je podijeliti u sedam dijelova:

1. Prvo se inicijalizira 8 hash vrijednosti, h_0 do h_7. Inicijalizacije predstavljaju frakcionalne dijelove drugih korijena prvih osam prostih brojeva.

$$\sqrt{2}_{10} \approx 1{,}41421356237_{10} = 1.6\text{A09E667}_{16} \quad (4.1)$$
$$h_0 = 6\text{A09E667}_{16}$$

2. Inicijalizira se polje od 64 cjelobrojnih konstanti, k_0 do k_{63}. Konstante se izračunavaju kao frakcionalni dijelovi trećih krojena prvih 64 prosta broja.

$$\sqrt[3]{2}_{10} \approx 1{,}25992105_{10} = 1.428\text{A2F98}_{16} \quad (4.2)$$
$$k_0 = 428\text{A2F98}$$

3. Tekstu na koji će biti primijenjen hash se dodaje bit 1 te n bitova 0 tako da duljina ukupne poruke pri dijeljenju s 512 daje ostatak 448. Potom se poruci dodaje 64-bitovni zapis koji izražava duljinu poruke bez prethodno dodanih bitova. Na ovaj način dobiva se poruka čija je duljina cjelobrojni višekratnik od 512.
4. Poruka se dijeli na 512-bitne blokove. Za svaki blok se alocira polje od 64 32-bitne riječi. Blok se upisuje u prvih šesnaest elemenata polja te potom proširuje u preostalih 48 elemenata polja sljedećim algoritmom:

```
for(i=16;i<64;i++){                                          (4.3)
s0= rotiraj_x_u_desno_za_n(w[i-15],7) ^
rotiraj_x_u_desno_za_n(w[i 15] ,18) ^
shiftaj_x_u_desno_za_n(w[i-15],3);
s1=rotiraj_x_u_desno_za_n(w[i-2],17) ^
rotiraj_x_u_desno_za_n(w[i-2],19)^
shiftaj_x_u_desno_za_n(w[i-2],10);
w[i]=w[i-16]+s0+w[i-7]+s1;
 }
```

5. Nad rezultatom prethodnog koraka se potom provodi kompresijska funkcija:

```
for(i=0;i<64;i++){                                           (4.4)
S1=rotiraj_x_u_desno_za_n(h[4],6)^
rotiraj_x_u_desno_za_n(h[4],11)^
rotiraj_x_u_desno_za_n(h[4],25);
ch= (h[4] & h[5] ) ^ (~h[4] & h[6]);
temp1= h[7] +S1+ch+k[i]+w[i];
S0=rotiraj_x_u_desno_za_n(h[0],2)^
rotiraj_x_u_desno_za_n(h[0],13)^
```

```
rotiraj_x_u_desno_za_n(h[0],22);
maj=(h[0] & h[1] ) ^ (h[0] & h[2] ) ^ (h[1] & h[2] );
temp2=S0+maj;
h[7]= h[6];
h[6]= h[5];
h[5]= h[4];
h[4]= h[3]+temp1;
h[3]= h[2];
h[2]= h[1];
h[1]= h[0];
h[0]=temp1+temp2;
} // h[i]= copy_of(h_i) ; k[i]= k_i;
```

6. Nakon kompresijske funkcije se originalnim hash vrijednostima $h_{0,\ldots}$, h_7 pridodaju izračunate hash vrijednosti h[0],…, h[7]. Tako da je h_i= h[i]+h_i, i *e* {1,…7}.
7. Konačna hash vrijednost dobiva se spajanjem hash vrijednosti.

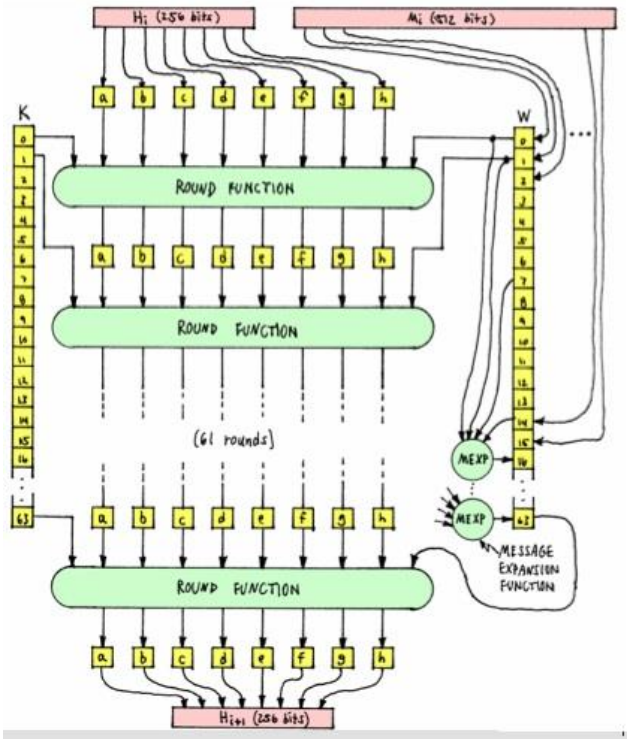
$$HASH= h_0h_1h_2h_3h_4h_5h_6h_7 ; \quad (4.5)$$

SHA-224 razlikuje se od SHA-265 utoliko što su inicijalne vrijednosti hash-a $h_0,\ldots,h_7$ drugačije te se na kraju zanemaruje h_7. U ovom slučaju se za inicijalne vrijednosti uzimaju drugih 32-bita decimalnog dijela drugog korijena od devetog do šesnaestog prostog broja.

SHA-512 radi s 1024-bitnim blokovima, inicijalne hash vrijednosti $h_0,\ldots,h_7$, cjelobrojne konstante, kojih ovdje ima 80, $k_0,\ldots, k_{79}$ i riječi $w_0,\ldots,w_{79}$ prikazuju se s 64-bitnom preciznošću, kompresijska funkcija ima 80 iteracija te koristi drugačije iznose pomaka te prilikom dodavanja duljine poruke duljina se prikazuje u 128-bitnoj preciznosti.

SHA-384 radi poput SHA-512 uz razliku inicijalnih vrijednosti hash-a te na kraju zanemaruje h_6 i h_7. Za inicijalne vrijednosti hash-a uzimaju se decimalni dijelovi drugih korijena od devetog do šesnaestog prostog broja.

SHA-512/256 radi poput SHA-512 samo što inicijalne vrijednosti hash-a generira funkcija SHA-512/t IV generation. Izlazni hash se skraćuje na 256-bita odbacivanjem hash-ev h_4, h_5, h_6 i h_7. Slika 4-2 prikazuje rad SHA-2 algoritma, a preuzeta je s adrese: http://j-7system.blog.so-net.ne.jp/2014-01-14 .



Slika 4-2 SHA-2 algoritam

4.3 SHA-3

Iako SHA-2 algoritam pruža adekvatnu sigurnosnu zaštitu postoje teorijski rizici. Iz tog razloga je Američki nacionalni institut za standarde i tehnologiju 2007. godine raspisao natječaj za razvoj novog hash algoritma. 2012. godine kao pobjednik natječaja objavljen je Keccak algoritam te je tako postao SHA-3.[3][5][7] Važno je napomenuti da SHA-3 nije razvijeni odnosno unaprijeđeni SHA-2 algoritam. SHA-3 koristi spužvastu funkciju za obradu podataka. Spužvaste funkcije su algoritmi s konačnim brojem stanja koji obrađuju nizove proizvoljne konačne duljine te generiraju nizove proizvoljne konačne duljine. Spužvaste se funkcije sastoje od tri dijela:

1. Memorije stanja, S koja se sastoji od b bitova. Bitove memorije stanja moguće je logički podijeliti u dvije skupine. Na bitove bitrata R i bitove kapacitet C.
2. Funkcije f(S) koja permutira memoriju stanja.
3. Funkciju proširenja P(x), gdje je x ulazni tekst. Funkcija izvorni tekst proširuje tako da njegova duljina postane cjelobrojni višekratnik bitrata R.

Izvođenje spužvastih funkcija moguće je podijeliti u tri koraka:

I. U prvom koraku inicijalizira se početno stanje S u nula i izvorni tekst se proširuje uz pomoć funkcije P(x).

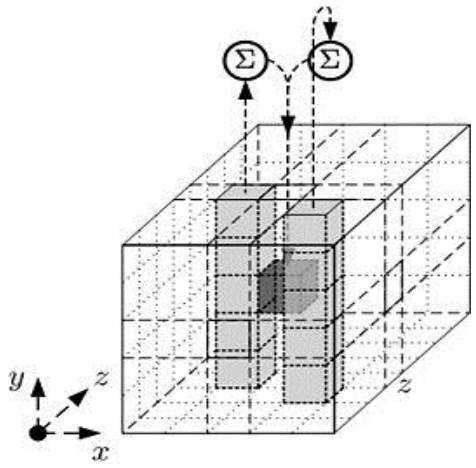
II. Drugi korak ima onoliko iteracija koliko puta je izvorni tekst veći od bitrata R. Za svaki blok veličine bitrata bitrat i taj blok se XOR-aju i izvede se $S_i=f(S_{i-1})$.

III. U zadnjem koraku dolazi do ispisa. Bitrat memorije stanja se ispisuje. Ako je traženo još bitova za ispis odvija se $S_i=f(S_{i-1})$ i nakon toga se bitrat ponovo

ispisuje. Ukoliko tražena duljina izlaznog niza nije višekratnik duljine bitrata prilikom zadnjeg ispisa odbacuju se bitovi viška.

Prilikom izvođenja C dio memorije stanja tj. kapacitetni bitovi se mijenjaju ovisno o funkciji f(S). Kada se spužvaste funkcije koriste za hash otpornost hash-a na kolizije i otpornost na izračunavanje originala ovise o veličini kapaciteta C. SHA-3 algoritam prilikom proširenja ulaznog teksta dodaje jednu jedinicu, potom odgovarajući broj nula i na kraju još jednu jedinicu. Algoritam radi s 64-bitnim riječima stoga se njegova memorija stanja prikazuje kao matrica dimenzija 5*5*64. f(S) funkcija SHA-3 algoritma, zvana još i Keccak f-permutacijom sastoji se od pet pod-operacija koje se izvršavaju u 24 iteracije, a označavaju se s θ, ρ, π, χ i ι.

1) θ funkcija, prikazana slikom 4-3 preuzetoj iz [5] izračunava paritet svakog stupa matrice te nad svakim elementom matrice provodi XOR s XOR-om pariteta dvaju predodređenih stupaca. Matematički je funkciju moguće zapisati kao:

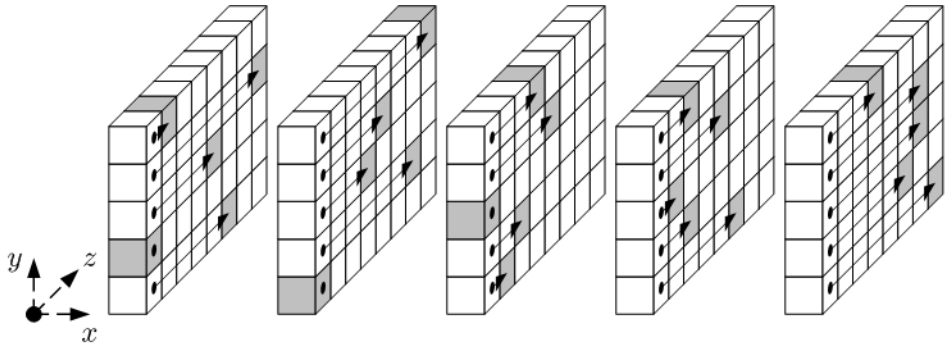


Slika 4-3 θ funkcija

$$a[i][j][k] = a[i][j][k] \oplus \\ paritet(a[i-1][j][k],\ j = \{0,...,4\}) \oplus \\ paritet(a[i+1][j][k-1],\ j = \{0,...,4\}) \tag{4.6}$$

2) ρ funkcija rotira svaku od 64-bitnih riječi za različiti trokutasti broj mjesta. Rad funkcije prikazan je na slici 4-4 preuzetoj iz [5]. Trokutasti brojevi su brojevi oblika Br= $\sum_{i=1}^{k} k, k \in N$ Matematički je postupak funkcije moguće zapisati kao:

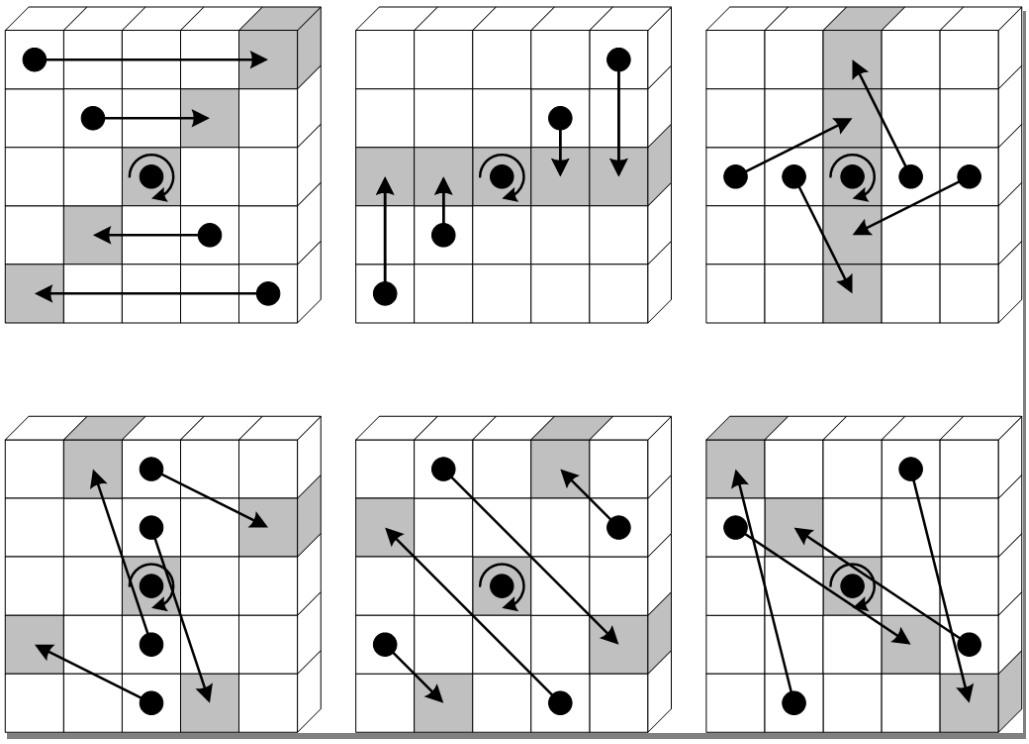
$$a[i][j][k] = a[i][j]\left[k - \frac{(t+1)*(t+2)}{2}\right], \text{za t=0,…,24 i } \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^t \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} i \\ j \end{pmatrix} \tag{4.7}$$



Slika 4-4 ρ funkcija

3) π je transpozicija stanja. π funkcija prikazana je na slici 4-5 preuzetoj iz [5] Matematički ju je moguće zapisati kao:

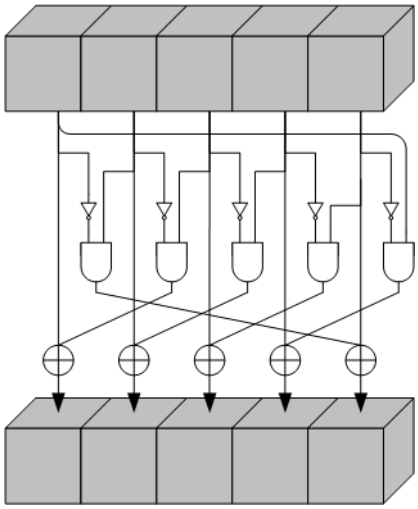
$$a[i][j] = a[i'][j'], \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^t \begin{pmatrix} i' \\ j' \end{pmatrix} = \begin{pmatrix} i \\ j \end{pmatrix} \tag{4.8}$$



Slika 4-5 π funkcija

4) χ je jedina nelinearna operacija u SHA-3. Svaki element matrice XOR-a se s XOR-om pariteta dvaju predodređenih redova kao što je prikazano na slici 4-6 preuzetoj iz [5]. Matematički je operaciju moguće prikazati kao:

$$a[i][j][k] = a[i][j][k] \oplus \neg a[i][j+1][k] \,\&\, a[i][i+2][k] \tag{4.9}$$

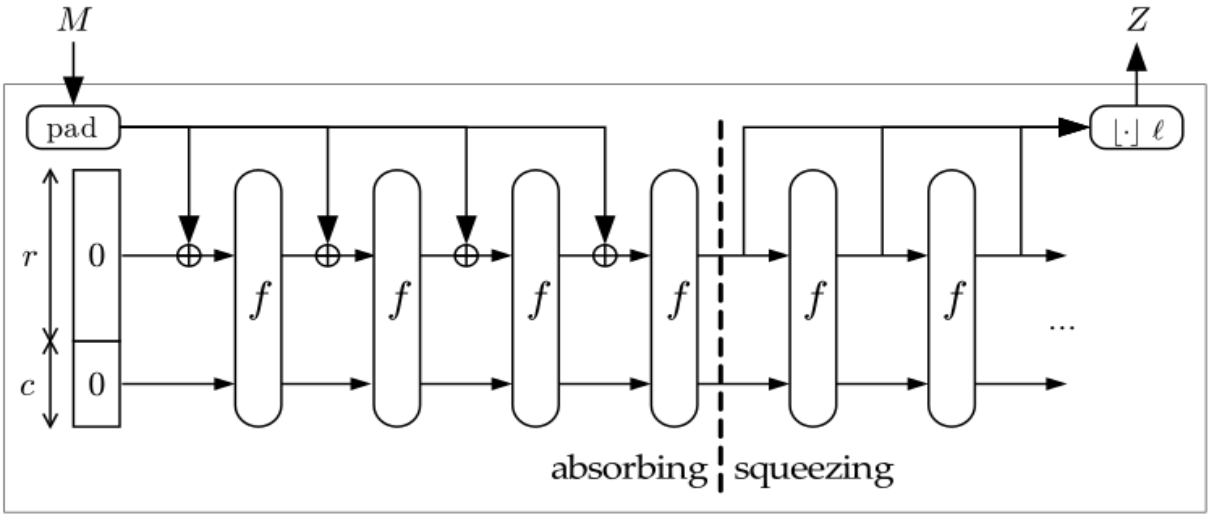


Slika 4-6 χ funkcija

5) ι operacija svaki element XOR-a s konstantnom vrijednošću RC[z_r]. Matematički se prikazuje kao:

$$a = a \oplus RC[z_r] \tag{4.10}$$
$$RC[z_r][0][0][2^y - 1] = rc[y + 7z_r], za$$
$$y = 0,...,6$$
$$rc[t] = (i^t \bmod i^8 + i^6 + i^5 + i^4 + 1) \bmod i,$$
$$inačn$$
$$RC[z_r] = 0$$

Prilikom ispisa SHA-3 duljina dobivenog hash-a je uvijek manja od duljine ulaznog bloka i čitavi izlazni niz se generira u jednoj iteraciji. Slika 4-7 prikazuje osnovnu ideju rada spužvaste funkcije, a preuzeta je s adrese: http://os2.zemris.fer.hr/algoritmi/hash/sha3/2013_kraljevic/keccak.html .



Slika 4-7 Spužvasta funkcija

5. Kriptoanaliza

5.1 Osnove kriptoanalize

Kriptoanaliza je znanost koja se bavi analizom sigurnosti kriptografskih algoritama[1][2][3][7]. Znanja kriptoanalize često bivaju korištena u pokušajima ugrožavanja sigurnosti. Jedna od osnovnih pretpostavki kriptoanalize, zvana Shannon-ova pretpostavka, nalaže da napadač poznaje sustav odnosno da je algoritam kriptiranja poznat. Ovisno o količini informacija kojom napadač raspolaže moguće je napraviti podjelu u kriptoanalizi.

- Poznavanje samo šifranta.
- Poznavanje izvornog teksta i njemu odgovarajućeg šifranta.
- Poznavanje točno određenog izvornog teksta i njegovog šifranta.
- Poznavanje više šifranata izvornog teksta koji se razlikuju zbog korištenja različitog ključa kriptiranja. Iako su sami ključevi kriptiranja nepoznati napadač zna za nekakav logički odnos među njima.

Napade na sigurnosni sustav moguće je dijeliti i s obzirom na resurse potrebne za njihovo izvođenje. Osnovni resursi su:

- Vrijeme. Zbog različitih brzina i načina izvođenja algoritama na različitim procesorima za vrijeme se ne uzima mjera sekunde već broj osnovnih operacija procesora koje je potrebno izvesti prilikom izvođenja algoritma.
- Memorija. Različiti napadi zahtijevaju različite količine memorijskog prostora za svoje izvođenje.
- Informacije. Različiti napadi zahtijevaju različite količine informacija o sustavu kojeg nastoje ugroziti. Za količinu potrebnih informacija često se uzima količina izvornog teksta i količina šifranta potrebna za ugrožavanje sustava.

Svaki sustav kriptiranja moguće je probiti primjenom grube sile odnosno isprobavanjem svih mogućih kombinacija. Takav napad često nije od praktične koristi jer ga je nemoguće izvesti u realnom vremenu. U akademskim krugovima probijanje sustava definira se kao bilo koji uspješan napad niže složenosti od napada primjenom grube sile tj. brute force napada. Ipak nije ni svaki takav napad od praktične koristi. Ukoliko napad zahtjeva nerazumne količine vremena, memorije i/ili informacija mogućnost njegova stvarnog izvođenja je nikakva. Probijanja sustava moguće je podijeliti s obzirom na količinu informacija dobivenih probojem

- Totalnim probojem- slučaj kada napadač otkrije tajni ključ šifriranja.
- Globalnom dedukcijom- slučaj kada napadač sazna funkcijski ekvivalent algoritma ali ne sazna tajni ključ korišten prilikom šifriranja.
- Lokalnom dedukcijom- slučaj kada napadač otkrije dodatne količine izvornog teksta i/ili šifranta.
- Informacijska dedukcija- napadač sazna Shannonovske informacije o izvornom tekstu i/ili šifrantu.
- Prepoznavanje algoritma- napadač prepozna način šifriranja.

Prilikom testiranja sigurnosti sustava proboji se pokušavaju ostvariti na oslabljenim inačicama sustava. Takve inačice često su originalni algoritmi s pokojom uklonjenom iteracijom. Ovakva testiranja predvidjela su probijanje kriptografskih algoritam DES, MD5 i SHA-1. Obične zamjenske šifre često je moguće ugroziti frekvencijskom analizom. U prirodnim jezicima određena slova i nakupine slova pojavljuje se učestalije od drugih. Ukoliko šifrant ne prikriva takve statističke odnose moguće je s određenom dozom sigurnosti pretpostaviti značenja pojedinih dijelova šifranta te pomoću toga probiti šifru. Ukoliko se više poruka kriptira koristeći jednake ključeve smanjuje se sigurnost kriptiranja. Poznavanje više poruka istog ključa kriptiranja povećava mogućnost otkrivanja tajnog ključa, a ukoliko se otkrije više takvih ključeva povećava se mogućnost razotkrivanja principa generiranja ključeva. Za različite kriptografske postupke primjenjuju se različiti kripto-analitički algoritmi. Za simetrične kriptografske algoritme najčešće se primjenjuju diferencijalna i linearna kriptoanaliza. Probijanje asimetričnih algoritama svodi se na rješavanje kompleksnih matematičkih problema kao što su faktorizacija velikih prirodnih brojeva i određivanje diskretnih algoritama. Hash algoritmi se mogu ugroziti s Rođendanskim napadom i Rainbow table algoritmom.

5.2 Kriptoanaliza simetričnih algoritama

Metode kriptoanalize simetričnih algoritama međusobno se dosta razlikuju. Unatoč tome sve one zasnivaju se na ideji pronalaženja pravilnosti prilikom kriptiranja koje bi otkrile dodatne informacije o postupku enkripcije. Dekriptiranje ključa ili među-ključa glavni je cilj simetrične kriptoanalize. Jednom kad je ključ poznat šifrant se jednostavno dekriptira te se dobije izvorni tekst. Linearna, diferencijalna i MITM kriptoanaliza često su primjenjivane metode kriptoanalize simetričnih blokovskih kriptosustava [7].

5.2.1 Diferencijalna kriptoanaliza

Iako su diferencijalnu analizu prvi javno opisali Eli Biham i Adi Shamir 1990. godine postoje indikacije da je postupak bio poznat IBM-ovom DES timu još 1974. godine te su vodili računa o njemu prilikom konstrukcije S-kutija i P permutacije[2][7]. Po postupku diferencijalna kriptoanaliza spada u skupinu poznavanja određenog izvornog teksta i njegova šifranta iako postoje verzije koje u nekih algoritama omogućuju napad poznavanjem samo izvornog teksta ili samo šifranta. Ideja diferencijalne kriptoanalize je usporedba XOR-a dvaju izvornih tekstova L_0R_0, $L_0^*R_0^*$ i njihovih šifranata nastalih istim ključem. XOR izvornih tekstova $L_0^{'}R_0^{'} = L_0^*R_0^* \oplus L_0R_0$ naziva se input tj. ulazni XOR, a šifranata output tj. izlazni XOR, a zajedno čine diferencijal. Na primjeru DES algoritma moguće je sagledati ideju postupka diferencijalne kriptoanalize te ju razložiti u tri koraka:

1. S-kutija označava se s S_j, j e {1, …, 8}. Par 6-bitnih ulaznih nizova u S-kutiju označava se s B_j, B_j^*, a par 4-bitnih izlaznih nizova s $S_j(B_j)$ i $S_j(B_j^*)$. Za svaki par ulaznih nizova jednakog input XOR-a izračunava se output XOR njihovih šifranata. Zbog šest bitova za svaki input XOR postoji 2^6=64 para. Činjenica da output XOR-ovi nisu uniformno distribuirani između 16 mogućih vrijednosti osnova je diferencijalna napada.
2. Za $j \in \{1,...,8\}$, $B_j^{'} = B_j \oplus B_j^*$ i $C_j^{'}$ definira se :

$$IN_j\left(B_j^{'}, C_j^{'}\right) = \left\{B_j \in Z_2^6 : S_j\left(B_j\right) \oplus S_j\left(B_j \oplus B_j^{'}\right) = C_j^{'}\right\}, \quad (5.1)$$
$$N_j\left(B_j^{'}, C_j^{'}\right) = \left|IN_j\left(B_j^{'}, C_j^{'}\right)\right|$$

Podsjetimo se da vrijedi $B_i = E(D_{i-1}) \oplus K_i$ za ulazni podatak S-kutije u i-toj rundi. Zbog provedenih operacija vrijedi:

$$B \oplus B^* = \left(E \oplus K_i\right) \oplus \left(E^* \oplus K_i\right) = E \oplus E^* \quad (5.2)$$

Važno je primijetiti kako input XOR ne ovisi o međuključu.

3. B, E, B^*, E^*, K zapisuju se kao:

$$B = B_1B_2B_3B_4B_5B_6B_7B_8 \quad (5.3)$$
$$E = E_1E_2E_3E_4E_5E_6E_7E_8$$
$$B^* = B_1^*B_2^*B_3^*B_4^*B_5^*B_6^*B_7^*B_8^*$$
$$E^* = E_1^*E_2^*E_3^*E_4^*E_5^*E_6^*E_7^*E_8^*$$
$$K = K_1K_2K_3K_4K_5K_6K_7K_8$$

Potom se definira:

$$test_j\left(E_j, E_j^*, C_j^{'}\right) = \left\{B_j \oplus E_j : B_j \in IN_j\left(E_j^{'}, C_j^{'}\right)\right\}, \quad (5.4)$$

gdje je

$$E_j^{'} = E_j \oplus E_j^*$$
$$C_j^{'} = S_j\left(B_j\right) \oplus S_j\left(B_j^*\right)$$

Za navedeni skup jednadžbi vrijedi:

$$K_j \in test_j\left(E_j, E_j^*, C_j^{'}\right) \quad (5.5)$$

Primjenom ovog algoritma dobiva se međuključ što znatno olakšava postupak određivanja cjelokupnog ključa.

5.2.2 Linearna kriptoanaliza

Linearna kriptoanaliza je napad poznavanjem izvornog teksta i njegova šifranta [2][7]. Ona se zasniva na ideji da iako bitovi ključa nisu linearne funkcije neki od njih mogu se dobro aproksimirati linearnim funkcijama. Postupak linearne kriptoanalize moguće je razložiti na sljedeći način:

I. Bitovi izvornog teksta, šifranta i ključa označavaju se kao:

$$P[1], P[2], P[3], ...izvorni\_tekst \quad (5.6)$$
$$C[1], C[2], C[3], ...šifrant$$
$$K[1], K[2], K[3], ...ključ$$

Uvodi se i oznaka:

$$A[i_1,...,i_k] = A[i_1] \oplus ... \oplus A[i_k] \quad (5.7)$$

II. Nakon toga traže se linearne jednadžbe oblika:

$$P[i_1,...,i_a] \oplus C[j_1,..., j_b] = K[k_1,...,k_e] \quad (5.8)$$

koje za nasumični izvorni tekst i šifrant vrijede s vjerojatnošću $p \neq 0.5$. Vjerojatnost desne strane dobiva se njenim izračunavanjem za veliki broj parova izvorni tekst, šifrant.

III. Ako je u više od pola slučajeva na lijevoj strani nula i $p > 0.5$ za vrijednost se uzima nula, a za $p > 0.5$ se uzima jedan. Na taj način dobiva se linearna jednadžba u bitovima ključa. Uz dovoljno velik broj takvih jednadžbi postaje moguće odrediti ključ.

IV. Kako bi metoda bila uspješna kod DES algoritma preporuča se uzimanje $2 * |p - 0.5|^{-2}$ parova izvornog teksta i šifranta.

Linearnu kriptoanalizu prvi je opisao Mitsuru Matsui 1993. godine kao teoretski napad na DES algoritam. Napad zahtijeva 2^{47} poznatih izvornih tekstova što ga čini nepraktičnim za upotrebu.

5.2.3 Meet in the middle napad

MITIM napad osmišljen je 1977 godine[7]. MITIM napad ne ovisi o unutrašnjoj strukturi kriptografskog sistema na koji se primjenjuje već je bitno da se prilikom šifriranja na izvorni tekst primjenjuje kompozicija više funkcije šifriranja s različitim ključevima kako bi se dobio šifrant. Za izvođenje ovog napada napadaču je potrebna mogućnost kriptiranja i dekriptiranja te posjedovanje izvornog teksta i njemu korespondentnog šifranta. Izvorni tekst *P* šifrira se pomoću funkcije *E* i ključeva K_1,K_2 kako bi se dobio šifrant *C*. Dobiveni šifrant dešifrira se primjenom deskripcijske funkcije *D* i početnih ključeva K_1,K_2.

$$C=E(E(P,K_1),K_2) \quad (5.9)$$
$$P=D(D(C,K_2),K_1)$$

Napadač izračunava $E(P,K_1)$ za sve moguće ključeve K_1 i $D(C,K_2)$ za sve moguće ključeve K_2. Sve identične vrijednosti dobivene ovim postupkom mogu otkriti točne ključeve šifriranja. Dobivene vrijednosti moguće je dodatno potvrditi provodeći postupak nad nekoliko parova

izvornih tekstova i šifranata. Ova vrsta napad jedan je od razloga zašto je DES algoritam bio zamijenjen s 3DES algoritmom.

5.3 Kriptoanaliza asimetričnih algoritama

Kriptoanaliza većine asimetričnih algoritama današnjice svodi se na rješavanje matematičkih problema faktorizacije velikih prirodnih brojeva i/ili pronalaženje diskretnog logaritma [2][7]. Algoritme faktorizacije moguće je podijeliti na one bazirane na metodi specijalne namjene i na opće metode. Metode specijalne namjene najčešće se koriste za faktoriziranje brojeva malih faktora dok opće metode imaju opću primjenu te se često koriste u kriptoanalizi odnosno kriptografiji. Složenost metoda opće namjene ovisi o redu veličine broja koji se faktorizira.

5.3.1 Pollardova p-1 metoda

Pollardova p-1 metoda razvijena je 1974 godine te spada u specijalne metode faktorizacije. Zasnovana je na Malom Fermatovom teoremu. Ukoliko je *n* broj koji treba faktorizirati i *p* neki njegov prosti faktor tada vrijedi:

$$a^m \bmod p=1 \text{ za svaki višekratnik od } p-1 \tag{5.10}$$

Pronalaskom *m* uz pomoć (a^m*-1,n*) moguće je identificirati faktor od *n.* Pronalazak višekratnika od *p-1* moguće je efikasno napraviti ukoliko se on sastoji samo od malih prostih faktora. Pretpostavi se da su svi faktori broja *p-1* manji ili jednaki *B* te da su sve potencije prostih brojeva koji dijele broj *p-1* manje od *B*. Tada je *m* najmanji zajednički višekratnik brojeva od 1 do *B.* Ova metoda u najgorem slučaju nije ništa lošija od običnog dijeljenja. Efikasnost ove metode izuzetno je ovisna o karakteristici broja *p-1*[2].

5.3.2 Fermatova metoda faktorizacije

Najefikasniji algoritam faktorizacije brojeva je NFS(Number field Sive) koji je moguće promatrati kao unaprijeđenu metodu kvadratnog sita, koja je varijanta metode faktorske baze odnosno Fermatove faktorizacije[2]. Fermatova faktorizacija zasniva se na činjenici:

$$n = ab \tag{5.11}$$

$$t = \left\lceil \frac{a+b}{2} \right\rceil^2$$

$$s = \left\lceil \frac{a-b}{2} \right\rceil^2$$

$$n = t^2 - s^2$$

$$a \approx b \Rightarrow t >> s, t > \sqrt{n}$$

Algoritam počinje provjeru od broja $t = \lfloor \sqrt{n} \rfloor + 1$ te traži $s = \sqrt{t^2 - n}, s \in N$. Ukoliko ne uspije poveća t za jedan i ponavlja se dok ne nađe takav broj. Jednom kad nađe brojeve t i s iz njih izračuna a i b te je broj n faktoriziran. Još neke opće metode faktorizacije su: Eulerova metoda, Dixonova metoda i metoda isključivanja.

5.3.3 Algoritam Indeks-calculus

Problem pronalaženja diskretnog logaritma je za neki prosti broj p i elemente a i b naći x tako da vrijedi:

$$a^x = b(\text{mod } p) \tag{5.12}$$

Jedan od efikasnih algoritama za rješavanje ovog problema je Indeks-calculus algoritam[7]. On služi za izračunavanje diskretnog logaritma u poljima oblika F_p i F_{2^p}, gdje je p prost broj. Osnovna ideja ovog algoritma temelji se na :

$$\exists x, y \in GF(p)^* \tag{5.13}$$
$$\prod_{i=1}^{m} x_i = \prod_{j=1}^{m} y_j \Rightarrow \sum_{i=1}^{m} \log_g x_i = \sum_{j=1}^{m} \log_g y_j (\text{mod } p-1)$$

Uz dovoljan broj navedenih jednadžbi i relativno malen broj elemenata x_i i y_j sustav je rješiv.

Algoritam je moguće podijeliti u 5 koraka:

1. Isprazni se lista relacija
2. Za k={1,2…} uz pomoć cjelobrojne faktorizacije prilagođenje za glatke brojeve faktorizira se a^k mod p po faktorskoj bazi. Sprema se svaki pronađeni k i njemu odgovarajuću faktorizaciju kao relaciju. Ako je novo dodana relacija linearno nezavisna od prethodno dodanih relacija zadržava se. Jednom kad se skupi r+1 takva relacija prestaje traženje novih relacija. r je veličina faktorske baze.
3. Formira se matrica čiji redovi predstavljaju relacije.
4. Matrica se svede na reducirani kanonski oblik odnosno prvi red je diskretni logaritam od -1 drugi od 2 itd.
5. Za s={0,1,2…} pokuša se faktorizirati a^s*b mod p po faktorskoj bazi. Kad se pronađe faktorizacija vraća se $x = f_0 \log_a(-1) + ... + f_3 \log_a(p_r) - s.$

Moderni asimetrični kriptosustavi se od navedenih postupaka brane pažljivim odabirom operanada tako da je algoritme nemoguće izvesti u realnom vremenu.

5.4 Kriptoanaliza funkcija sažimanja

Kako se hash algoritmi dosta koriste za ostavljanje digitalnog potpisa i autentifikacije napadi su često usmjereni na realizaciju mogućnosti zloporabe digitalnog potpisa te

mogućnost lažne autentifikacije. Dvije poznate vrste napada na hash funkcije su rođendanski napad i Rainbow table[7].

5.4.1 Rođendanski napad

Rođendanski napad na hash funkciju traži kolizije u hash funkciji, a vrši se na sljedeći način:

1. Pripremaju se dvije inačice dokumenta. Jedna dobronamjerna i jedna zlonamjerna.
2. Za svaku od inačica pripremi se mnogo varijanti istog smisla. Primjer izrada više varijanti istog dokumenta prikazan je na slici 5-1.

> Suprotno ljudskoj intuiciji sasvim/vrlo je lako/jednostavno napraviti/izraditi velik broj verzija/varijacija istog teksta bez promjene/mijenjanja smisla/značenja.
>
> Gornja rečenica ima 2^6 tj. 64 različite verzije.

Slika 5-1 Izrada više tekstova istog značenja

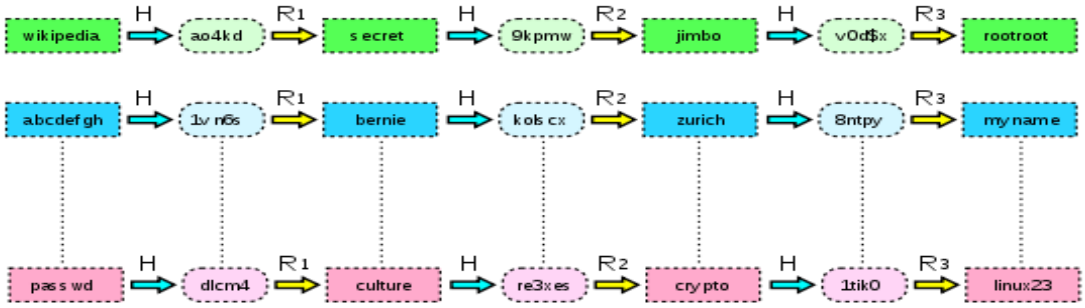
3. Potom je potrebno pronaći dvije inačice, jednu dobronamjernu i jednu zlonamjernu, koje imaju isti hash.
4. Zbog toga što inačice imaju isti hash imaju i isti potpis. Dovoljno je nekome dati dobronamjerni dokument da ga potpiše i kasnije je moguće tvrditi da je potpisao zlonamjerni dokument.

Rođendanski napad moguće je spriječiti promjenom dokumenta prije njegova potpisivanja jer mu se na taj način mijenja vrijednost hash-a.

5.4.2 Tablica Rainbow

Druga vrsta napada na hash funkciju je Rainbow table [7]. Radi se o prethodno pripremljenoj tablici koja od hash-a generira izvorni tekst. Prilikom autentifikacije passwordi se hashiraju te uspoređuju s prethodno spremljenim unosom odnosno njegovim hashem. Ukoliko su hashevi isti pristup biva odobren. Ukoliko netko ukrade hash vrijednosti njihovim unošenjem prilikom autentifikacije neće ući u sustav zato jer će sustav hashirati te vrijednosti i vratiti kako password ne odgovara korisniku. Kako bi uz pomoć hash vrijednosti kradljivac ušao u sustav potrebno mu je iz nje razaznati izvorni tekst passworda ili naći password istog hasha. Tako što moguće je idućim algoritmom:

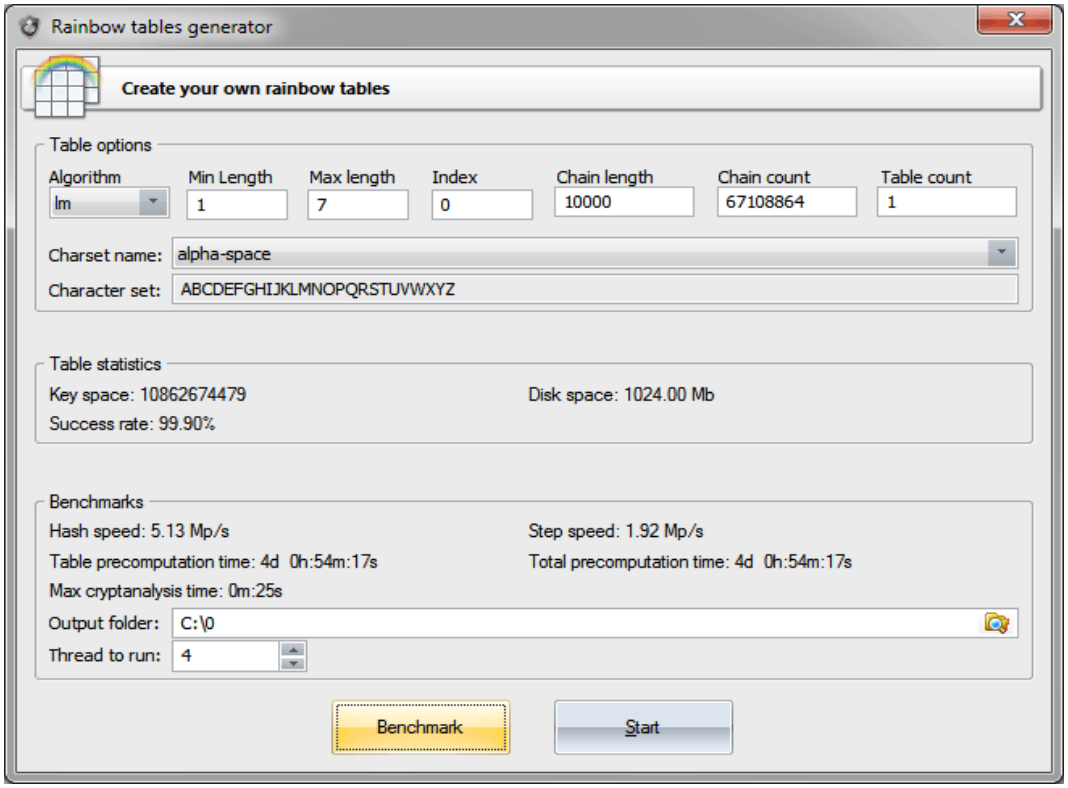
1. Definira se redukcijska funkcija R koja za dobiveni hash generira tekst P. R ne mora biti inverz hash funkcije.
2. Uzastopnom primjenom hash funkcije i redukcijske funkcije formiraju se hash lanci. Primjer hash lanca prikazan je na slici 5-2 preuzetoj s adrese: http://en.wikipedia.org/wiki/Rainbow_table .



Slika 5-2 Hash lanac

3. Početak i kraj svakog hash lanca sprema se u tablicu.
4. Uzima se hash vrijednost h za koju se želi dobiti izvorni tekst ili neki njemu iste hash vrijednosti. Na vrijednost h se uzastopno primjenjuje redukcijska funkcija R i hash funkcija H dok se ne pojavi vrijednost jednaka jednoj od krajnjih vrijednosti u tablici.
5. Uzima se početna vrijednost koja odgovara dobivenoj krajnjoj vrijednosti te se razvija hash lanac. Postoji dobra šansa da lanca sadrži vrijednost h. Ona vrijednost koja prethodi h je traženi password.

Rainbow table algoritam radi nešto drukčije od navedenog, ali zasniva se na istoj ideji. Kod njega se koristi niz više redukcijskih funkcija te ukoliko dva lanca u istoj iteraciji postignu istu hash vrijednost jedan se izbacuje jer je duplikat.



Slika 5-3 Sučelje programa koji radi rainbow tablice

Slika 5-3 prikazuje sučelje programa koji generira rainbow tablice, a preuzeta je s: http://www.passcape.com/windows_password_recovery_rainbow_table_generator . Način

obrane od Rainbow table algoritma je korištenjem relativno velikog nasumičnog dodatka prilikom svakog hashiranja. Dodatak osigurava jedinstvenost svakog hashiranja.

$$DH(password)=H(H(password)+Dodatak) \quad (5.14)$$

6. Zaključak

Kriptografski algoritmi s vremenom će se sigurno još razvijati, a skupa s njima i kriptoanalitički postupci njihova razmatranja. Čitanjem ovog seminara stječu se uvidi u osnovne koncepte i algoritme kriptografije i kriptoanalize. Njihovo razumijevanje ključ je daljnjeg izučavanja ovih područja.

Kriptografiju i kriptoanalizu moguće je promatrati kao dvije strane istog novčića. Razvoj prve povlači potrebu za razvojem druge i obrnuto. Razumno je pretpostaviti kako se procesorska snaga računala bude razvijala da će se tako razvijati i kriptografija odnosno kriptoanaliza. U svijetu u kojem vrijednost informacija te brzina i sigurnost njihova prijenosa rastu iz dana u dan ove znanosti sasvim sigurno imaju svoje mjesto i budućnost.

7. Sažetak

Ovaj seminarski rad sadrži jednostavan pregled i opis rada osnovnih kriptografskih algoritama te osnovne ideje i postupke kriptoanalize. U prvom dijelu seminara napravljen je kratak povijesni pregled kriptografije kako bi se pokazalo njen značaj u ljudskom društvu i objašnjeni su osnovni kriptografski pojmovi potrebni za razumijevanje seminara. Razrada se sastoji od tri odjeljka koji se bave različitim skupinama kriptografskih algoritama te pobliže opisuju neke od pripadnika skupina. Obrađeni algoritmi su:

- Simetrični algoritmi:
  - Data Encryption Standard
  - Triple Data Encryption Standard
  - Advanced Encryption Standard
- Asimetrični algoritmi:
  - RSA kriptosustav
  - Elliptic curve cryptography
- Hash algoritmi:
  - SHA-2
  - SHA-3

Razrada sadrži i četvrti dio koji prvo opisuje osnove kriptoanalize te potom osnovne postupke kriptoanalize po kriptografskim skupinama.

- Kriptoanaliza simetričnih algoritama
  - Diferencijalna kriptoanaliza
  - Linearna kriptoanaliza
  - Meet in the middle napad
- Kriptoanaliza asimetričnih algoritama
  - Pollardova p-1 metoda
  - Fermatova metoda faktorizacije
  - Algoritam Indeks-calculus
- Kriptoanaliza hash algoritama
  - Rođendanski napad
  - Tablica Rainbow

Seminar završava kratkim zaključkom koji razmatra buduće mjesto kriptografije u društvu.

8. Literatura

[1] Leo Budin, Marin Golub, Domagoj Jakobović, Leonard Jelenković; Operacijski sustavi; Element; Zagreb 2011;

[2] Andrej Dujella, Marcel Maretić; Kriptografija; Element; Zagreb 2007;

[3] Fakultet elektrotehnike i računarstva; Radovi iz područja računalne sigurnosti; 2013; http://os2.zemris.fer.hr/index.php?show=start

[4] Darko Žubrinić; Uvod u diskretnu matematiku; Element; Zagreb 2012

[5] Guido Bertoni, Joan Daemen, Michaël Peeters, Gilles Van Assche; The Keccak reference; http://keccak.noekeon.org/

[6]CARNet; Korištenje eliptičnih krivulja u kriptografiji; http://www.cert.hr/sites/default/files/CCERT-PUBDOC-2006-09-169.pdf

[7] Rainbow table, SHA3, Birthday attack, Differential cryptanalysis, Linear cryptanalysis, Meet in the middle attack; http://hr.wikipedia.org/wiki/Glavna_stranica