Rojstnodnevni napad

[Seminarska naloga pri predmetu Kriptografija in Teorija Kodiranja 2]

Naredil: Andraž Žagar

Mentor: prof. dr. Aleksandar Jurišić

Kazalo

Uvod

Skozi zgodovino kriptografije se je vedno vil dialog med tistimi, ki hočejo podatke zaščititi ter tistimi, ki želijo priti do zaščitenih podatkov in zato iščejo luknje v zaščiti in jih nato izkoriščajo. Tako so se ob uvedbi zgoščevalnih funkcij kmalu rodile verzije »brute force« napadov, ki iščejo trke teh funkcij. Ker so omenjeni napadi za sodobno uporabo zelo počasni, so se začele razvijati različice teh napadov, ki izkoriščajo zakonitosti verjetnosti in/ali statistike v svojo prid. En izmed teh napadov je tudi tako imenovani »Rojstnodnevni napad«, ki izkorišča zakonitosti, ki jih matematično opiše rojstnodnevni paradoks.

Ta seminarska naloga bo predstavila prednosti in slabosti rojstnodnevnega napada, matematiko v ozadju generičnega rojstnodnevnega paradoksa in delovanje omenjenega napada pri reševanju problema diskretnega algoritma, iskanju trkov zgoščevalnih funkcij ter iskanju ključev kriptografskega algoritma DES . Nato se bo poglobila v preurejanje osnovnih algoritmov v obliko, primerno za paralelno izvajanje ter paralelno izvajanje na GPU in predstavila dva nova algoritma za izvajanje paralelnega napada.

I. Paradoks rojstnega dne

Kot je bilo omenjeno že v uvodu, rojstnodnevni napad za izvedbo izkorišča matematični pojav rojstnodnevnega paradoksa. Zato moramo, če želimo razumeti rojstnodnevni napad, pregledati delovanje samega rojstnodnevnega paradoksa.

Rojstnodnevni paradoks prihaja iz teorije verjetnosti in nam pove, da če naključno in s ponavljanjem izbiramo elemente iz množice moči N, lahko po $1.2\sqrt{N}$ izbirah pričakujemo, da bomo izbrali že izbrani element (več o tej vrednosti in njen izvor je zapisano v referenci [1]).

Čeprav rojstnodnevni paradoks nima težke matematične podlage, vseeno raje najprej poglejmo njegov efekt na osnovnem problemu učencev v učilnici, in ga nato posplošimo in razširimo na problem iskanja trkov.

1 Osnovni problem

Pri osnovnem problemu imamo v učilnici n učencev, zanima pa nas, kakšna je verjetnost, da imata vsaj dva od učencev na isti dan rojstni dan (definicija vzeta po referenci [2]). Za osnovni problem predvidevajmo, da leto ni prestopno (ima vedno 365 dni). Da nam kasneje ne bo treba ponovno uvajati poimenovanja, bomo že v osnovnem problemu verjetnost, da imata vsaj dva učenca na isti dan rojstni dan, poimenovali P(trk).

Do izračuna omenjene verjetnosti je najbolje pristopiti z nasprotno verjetnostjo (torej, P(trk) = 1 – P(X)), kjer je X dogodek, da nima noben par učencev v razredu na isti dan rojstni dan.

Ker pri dogodku X ne smeta imeti dve osebi nikoli istega rojstnega dne, mora imeti vsaka naslednja oseba en dan manj na izbiro:

$$P(X) = \prod_{t=0}^{n-1}\left(1 - \frac{t}{365}\right) \tag{1.1}$$

$$P(X) = 1 \cdot \left(1 - \frac{1}{365}\right) \cdot \left(1 - \frac{2}{365}\right) \cdot \ldots \cdot \left(1 - \frac{n-2}{365}\right) \cdot \left(1 - \frac{n-1}{365}\right) \tag{1.2}$$

$$P(X) = \frac{365 \cdot 364 \cdot 363 \cdots (365 - n - 1)}{365^n} \tag{1.3}$$

$$P(X) = \frac{n! \cdot \binom{365}{n}}{365^n} \tag{1.4}$$

Torej je

$$P(trk) = 1 - \frac{n! \cdot \binom{365}{n}}{365^n} \tag{1.5}$$

za n učencev v sobi. Če je število učencev v sobi večje od 365, potem je P(trk) = 1.

Če to enačbi (1.1) in (1.5) posplošimo za N dni v letu, dobimo

$$P(trk) = 1 - \prod_{t=0}^{n-1}\left(1 - \frac{t}{N}\right) \tag{1.6}$$

$$P(trk) = 1 - \frac{n! \cdot \binom{N}{n}}{N^n} \tag{1.7}$$

Kjer velja da, če je n ≥ N je P(trk)=1.

2 Učinkovito računanje aproksimacije

Ker osnovna matematična definicija rojstnodnevnega paradoksa vsebuje precej računsko zahtevne operacije in tudi zaradi velikosti števil uporabljenih v računanju (n! in podobno), so se naredile aproksimacije, ki dokaj dobro aproksimirajo verjetnost trka z računsko precej manj zahtevnimi operacijami.

Za aproksimacijo lahko uporabimo dve bolj znani aproksimaciji, Taylorjevo in Poissonovo. Pri obeh aproksimacijah bomo predvidevali, da je 0 ≥ n ≥ N saj je drugače verjetnost očitna in za računanje verjetnosti ne potrebujemo aproksimacije.

2.1 Aproksimacija s Taylorjevo vrsto

Za izvedbo aproksimacije verjetnosti trka pri BP s Taylorjevo vrsto želimo zamenjati posamezne faktorje multipliukacije z prvimi nekaj členi Taylorjeve vrste. Preostali členi bodo nato predstavljali napako aproksimacije, samo aproksimacijo pa se bo lahko izračunalo bistveno hitreje kot točno verjetnost.

Če vzamemo Taylorjevo razširitev eksponentne funkcije $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!}\cdots$ in aproksimacijo $1 + x \approx e^x$ tako, da zanemarimo elemente $\frac{x^2}{2!} + \frac{x^3}{3!}\cdots$, potem lahko aproksimiramo posamezne faktorje znotraj P(X) tako, da določimo tak x, da bo veljalo

$$e^x \approx 1 - \frac{t}{N} \tag{1.8}$$

torej

$$x = -\frac{t}{N} \tag{1.9}$$

Sedaj lahko izračunamo $\tilde{P}(X)$ kot aproksimacijo $P(X)$:

$$P(X) = \prod_{t=0}^{t-1}\left(1 - \frac{t}{N}\right) \approx \prod_{t=0}^{t-1}\left(e^{-\frac{t}{N}}\right) = P(X) \tag{1.10}$$

$$\tilde{P}(X) = 1 \cdot e^{-\frac{1}{N}} \cdot e^{-\frac{2}{N}} \cdot \ldots \cdot e^{-\frac{n-2}{N}} \cdot e^{-\frac{n-1}{N}} \tag{1.11}$$

$$\tilde{P}(X) = e^{-\left(\frac{n(n-1)}{2N}\right)} \tag{1.12}$$

Iz česar sledi, da je

$$\tilde{P}(trk) = 1 - e^{-\left(\frac{n(n-1)}{2N}\right)} \tag{1.13}$$

Ker smo na začetku naredili predpostavko, da je $e^x \approx 1 + \frac{t}{N}$ in smo ignorirali preostanek Taylorjeve vrste, bo napaka enaka

$$\varepsilon = \left| \prod_{t=1}^{n-1} \left(\frac{\left(\frac{t}{N}\right)^2}{2!} + \frac{\left(\frac{t}{N}\right)^3}{3!} \cdots \right) \right| = \left| \prod_{t=1}^{n-1} \left(e^{\frac{t}{N}} - \left(1 + \frac{t}{N}\right) \right) \right| < \left| \prod_{t=1}^{n-1} \left(e^{\frac{t}{N}} - 1 \right) \right| \tag{1.14}$$

In ker vemo, da v našem primeru $e^{\frac{t}{N}}$ predstavlja verjetnost, ki ima vrednost največ 1 (če je t ≥ N, potem ne potrebujemo aproksimacije, da bi ugotovili, da je P(trk)=1), potem vemo, da za vrednost iz enačbe (1.14) vedno velja:

$$\varepsilon < \left| \prod_{t=1}^{n-1} \left(e^{\frac{t}{N}} - 1 \right) \right| < \left| e^{\frac{n-1}{N}} - 1 \right| \tag{1.15}$$

Iz enačbe (1.15) lahko vidimo, da napaka raste eksponentno s številom n, in bo majhna le za majhen n.

2.2 Aproksimacija s Poissonovo porazdelitvijo

Če na problem gledamo iz vidika parov na mesto iz vedika celotne množice, je verjetnost, da sta si elementa v paru različna enaka $\frac{N-1}{N}$. Pri izbiranju n elementov, je kombinacij dveh elementov C(n, 2), kjer C(n, 2) predstavlja binomski koeficient $\binom{n}{2}$ in ga uporabljamo zaradi preprostejšega zapisa. Da nebi bilo trka, morajo vsi elementi znotraj parov elementov biti različni, kar nam poda verjetnost

$$P(X) = \prod_{i=1}^{C(n,2)} \left(\frac{N-1}{N} \right) = \left(1 - \frac{1}{N}\right)^{C(n,2)} \tag{1.16}$$

To verjetnost lahko predstavimo tudi kot binomsko porazdelitev

$$Pr(K = k) = \binom{n'}{k} p^k (1-p)^{n'-k} \tag{1.17}$$

Če upoštevamo, da je k=0 torej velja

$$Pr(K = 0) = \binom{n'}{0} p^0 (1-p)^{n'} = (1-p)^{n'} \tag{1.18}$$

Hitro se lahko ugotovi, da je za k=0 P(X) binomska porazdelitev s $p = \frac{1}{N}$ in $n' = C(n, 2)$.

Sedaj lahko uporabimo izračunane podatke za določitev aproksimacije s pomočjo Poissonove porazdelitve. Če je verjetnost p majhna (v našem primeru je $p = \frac{1}{N}$ in z razsežnostjo problema postaja vse manjša) in število poskusov n' veliko, potem se začne binomska porazdelitev vesti zelo podobno Poissonovi z λ=np:

$$\binom{n'}{k} p^k (1-p)^{n'-k} \approx \frac{e^{-n'p}(n'p)^k}{k!} \tag{1.19}$$

V našem primeru je torej

$$\left(1 - \frac{1}{N}\right)^{C(n,2)} \approx e^{-\frac{C(n,2)}{N}} \tag{1.20}$$

In tako je

$$\tilde{P}(trk) = 1 - e^{-\frac{n(n-1)}{2N}} \tag{1.21}$$

3 Prehod na rojstnodnevni napad

Sedaj, ko vidimo kako oceniti verjetnost trka pri podanima N ter n, lahko to teorijo uporabimo tudi v praksi. Tako bomo pregledali v naslednjih poglavjih različne napade, ki uporabljajo za osnovo paradoks rojstnega dne.

II. Iskanje diskretega logaritma

Posplošen problem diskretnega algoritma (GDLP) je problem, ki se ogromno uporablja v »reševanju« kriptosistemov z javnimi ključi, vse odkar sta ga prvič predlagala W.Diffie in M.E.Hellman v njunem sistemu izmenjave ključev. Sam problem je v iskanju rešitve eksponentne enačbe $a^x = b$ v grupi G.

Od takrat se je razvilo že precej algoritmov za reševanje DLP. Med algoritmi zareševanje GDLP so najbolj znani »Mali korak – veliki korak«, Shanks-ov algoritem in Pollardov »Kangaroo algoritem« [3], za namene napada na DLP znotraj točno določene muliplikativne grupe pa sta najbolj znana predstavnika »Index calculus« in Pohlig-Hellman-ov algoritem. Za dodatne algoritme si lahko bralec pregleda [4].

1 Pristop z rojstnodnevnim napadom

1.1 Opis postopka

Predpostavimo, da je G grupa znotraj katere iščemo diskretnegi logaritem, $N = |G|$ je red grupe G in $g \in G$ generator te grupe.

Naj bo $S = \{y_i = g^{x_i} | 1 \leq i \leq m\}$ množica m znanih vrednosti, za katere želimo izračunati vrednost diskretnega logaritma. Pri iskanju vrednosti x_i za vsak $y_i \in S, 1 \leq i \leq m$ vzamemo n različnih števil $r_j^{(i)}, 1 \leq j \leq n$, kjer je $n = \left\lceil N^{\frac{1}{2}} \right\rceil + 1$. Kaj več o izbiri števila n si lahko preberete v [5]. Nato iz teh števil naredimo množico

$$S_i = \left\{ y_i^{r_j^{(i)}} \middle| 1 \le j \le n \right\}, 1 \le i \le m \qquad (2.1)$$

Tej množici na ničti indeks dodamo še množico n potenc generatorja, kjer je $r_j^{(i)}$ naključno izbrano število med 1 in n.

$$S_0 = \left\{ g^{r_j^{(0)}} \middle| 1 \le j \le n \right\} \qquad (2.2)$$

Sedaj določimo

$$\Omega = \bigcup_{0 \le i \le m} S_i \qquad (2.3)$$

če predpostavimo še, da je k celo število, za katerega velja $0 \le k \le m$, in da je $p_k(\Omega)$ verjetnost da je znotraj množice Ω vsaj k trkov, velja

$$p_k(\Omega) \ge 1 - e^{-(m+1)^2+\varepsilon} \cdot \frac{(m+1)^{2 \cdot k}}{k!} \qquad (2.4)$$

kjer je

$$\varepsilon < \frac{2 \cdot (T-k)^3}{3 \cdot N^2} \qquad (2.5)$$

In je

$$T = |\Omega| \le (m+1) \cdot n \qquad (2.6)$$

Ker je presek dobljene množice očitno linearna kombinacija začetnih vrednosti, oz. x_i, lahko na presek gledamo kot na linearno enačbo privatnih ključev

$$(y_i^{r_s^{(i)}} = y_j^{r_t^{(j)}}) = (r_s^{(i)} x_i = r_t^{(j)}) mod\ N \qquad (2.7)$$

$$(y_i^{r_s^{(i)}} = y_j^{r_t^{(0)}}) = (r_s^{(i)} x_i = r_t^{(0)}) mod\ N \qquad (2.8)$$

Torej, v primeru da je teh m linearnih enačb neodvisnih, bodo odkrite vse začetne vrednosti (torej tiste, ki sestavljajo množico S)

1.1.1 Dokaz

Za namen dokaza in za preprostejši zapis le-tega najprej uvedimo nekaj novih oznak. Naj predstavljata oznaki $[x]^r = x\,(x+1)\cdots(x+r-1)$ in $[x]_r = x\,(x-1)\cdots(x-r+1)$. Za lažji zapis zapisujemo binomski koeficient $\binom{n}{k}$ kot C_n^k.

Za vsako celo število i, kjer je $0 \le i \le m$ z λ_i označimo število vseh podmnožic velikosti T z N različnimi elementi in i trki, torej

$$\lambda_i = C_N^{T-i} \cdot \frac{[T-i]^i}{i!} = C_N^{T-i} \cdot C_{T-1}^i \tag{2.9}$$

Iz tega lahko sklepamo

$$\begin{aligned} p_k &\geq 1 - \sum_{0 \leq i \leq k} \frac{\lambda_i}{\frac{[N]^T}{T!}} \geq 1 - \sum_{0 \leq i \leq k} [N]_{T-i} \cdot [T]_i \cdot \frac{C_{T-i}^i}{[N]^t} \\ &\geq 1 - \prod_{i \leq i \leq t-k} \left(1 - \frac{2i}{N+1}\right) \cdot \sum_{0 \leq i \leq k} \frac{(m+1)^{2i}}{i!} \\ &\geq 1 - e^{\sum_{i \leq i \leq T-k} -\frac{2i}{N+i}} \cdot \frac{(m+1)^{2k}}{k!} \\ &\geq 1 - e^{\sum_{1 \leq i \leq T-k} \left(-\frac{2i}{N} + \frac{2i^2}{N^2}\right)} \cdot \frac{(m+1)^{2k}}{k!} \\ &\geq 1 - e^{-(m+1)^2 + \frac{2(T-k)^3}{3N^2}} \cdot \frac{(m+1)^{2k}}{k!} \end{aligned} \tag{2.10}$$

1.2 Primer

V priloženem programu je za poljubne parametre narejeno iskanje trkov na prej opisani način. Sam predel z napadom na GDLP lahko uporabnik najde na zavihku imenovanem »GDLP -> Diskretni logaritem«, aktivira pa ga s klikom na gumb »Začni poskus«. Pri privzetih parametrih (N = 68909, g=41, m=2) se da dokaj hitro opaziti, da je število najdenih parametrov večino časa dovolj visoko za rešitev dobljene linearne enačbe.

III. Iskanje trka pri zgoščevalnih funkcijah

Zgoščevalne funkcije so bile prvič omenjene leta 1968, ko je Wilkes prvič omeni napravo, ki jo je ustvaril Needham, čeprav ni javno znano, kako točno je Needhamova naprava delovala. Eden prvih, ki so opisali praktični dizajn pa je bil Purdy – njegov dizajn je temeljil na polinomih nad končnih poljih. Poleg njega so bili znani še Evans, Kantrowitz in Weiss, ki so ustvarili vsak svojo metodo, ki so temeljile na bločnih šifrah. Vse te, začetne, metode niso bile namenjene zgoščevanju, kot so današnje – namenjene so bile v večini za uporabo v shemi varovanja gesel.

Od takrat je seveda nastalo tudi precej algoritmov za iskanje trkov pri zgoščevalnih funkcijah, med katerimi dandanes prevladuje različica rojstnodnevnega napada, kjer se vrednosti funkcij držijo na javno dostopnih strežnikih.

1 Pristop z rojstnodnevnim napadom

1.1 Opis postopka

Napada na zgoščevalno funkcijo z rojstnodnevnim napadom se lotimo na precej bolj enostaven način kot pa napada na GDLP.

Najprej definirajmo funkcijo kot preslikavo $h: D \rightarrow R$, kjer sta domena D in razpon R končni množici. Ta funkcija je zgoščevalna funkcija takrat, ko velja $|D| > |R|$. Naš cilj je s pomočjo rojstnodnevnega napada najti trk, torej taka $x, y \in D | x \neq y \wedge h(x) = h(y)$.

Najprej izberemo q točk $x_1, \dots, x_q$ iz D, in izračunamo $y_i = h(x_i)$ za $i = 1, \dots, q$. Napad se vzame kot uspešen, če obstaja tak par i, j, da velja $y_i = y_j \wedge i \neq j$. Če je napad neuspešen, ga ponovimo. Za napad na različne zgoščevalne funkcije obstaja več različnih načinov izbiranja začetnih q točk. V našem programskem primeru je uporabljen naključen izbor točk. Za lažjo referenco bomo v prihodnje klicali število q »število poskusov«

Naj bo $C_h(q)$ verjetnost, da rojstnodnevni napad na zgoščevalno funkcijo $h: D \to R$ uspe najti trk znotraj q poskusov. Zanimivo je, kako ta funkcija raste s q. Znano je, da je

$$C_h(q) \approx \binom{q}{2} \cdot \frac{1}{r} \qquad (3.1)$$

kjer je $r = |R|$ velikost razpona funkcije h in $q \leq O(\sqrt{r})$. To nam pove, da je trk pričakovan po približno $r^{\frac{1}{2}}$ poskusih. Bolj točno, pove nam, da se lahko najde prvi trk v zgoščevalni funkciji, ki ima izhod velikosti m bitov, v približno $2^{\frac{m}{2}}$ poskusih.

1.1.1 Dokaz

Kot dokaz lahko uporabimo preprosto matematiko v definiciji rojstnodnevnega paradoksa, saj se tukaj uporabljajo ravno tam izpeljane enačbe.

1.2 Vpliv »ravnotežja« funkcije na rojstnodnevni napad

Ko smo prej omenjali zgoščevalno funkcijo, smo imeli v mislih popolno funkcijo, ki preslika vsak element na poljubno mesto in od katere izhodne vrednosti imajo diskretno enakomerno porazdelitev. Za tako zgoščevalno funkcijo je izbira poskusnih vrednosti $x_1, \dots x_q$ dokaj nepomembna, torej lahko izbiramo naključno.

Taka zgoščevalna funkcija pa v realnosti ne obstaja. Vse zgoščevalne funkcije imajo neko »ravnotežje« - stopnjo regularnosti (definicija po »Balance measure« iz reference [5]). Ta stopnja regularnosti je definirana na naslednji način:

1.2.1 Ravnotežna mera

Na razpon R zgoščevalne funkcije $h: D \to R$ gledamo kot na sklop $r \geq 2$ točk $R_1, \dots R_r$. Za $i = 1, \dots, r$, naj bo $h^{-1}(R_i)$ inverzna preslikava R_i čez h, kar pomeni, da je $R_i = \{x \in D | \, h(x) = R_i\}$ in naj bo $d_i = |h^{-1}(R_i)|$ velikost zaloge vrednosti inverzne funkcije od h. Naj bo $d = |D|$ velikost domene. Sedaj lahko definiramo ravnotežje funkcije h kot

$$\mu(h) = log_r \left[\frac{d^2}{d_1^2 + \dots + d_r^2} \right] \qquad (3.2)$$

Po tej definiciji, bo ravnotežje funkcije h največje – 1 – takrat , ko bo h regularna (torej bo $d_i = \frac{d}{r}$ za vsak i), medtem ko bo ravnotežje najmanjše – 0 – takrat, ko bo h constantna (torej bo $d_i = d$ za neki i, in $d_j = 0$ za vse $j \neq i$).

S to definicijo lahko sedaj ponovno poskusimo izračunati verjetnost, da rojstnodnevni napad na zgoščevalno funkcijo $h: D \to R$ uspe najti trk znotraj q poskusov:

$$C_h(q) = \binom{q}{2} \cdot \frac{1}{r^{\mu(h)}} \tag{3.3}$$

iz česar sledi, da je poskusov v resnici okrog $r^{\frac{\mu(h)}{2}}$, kar zna biti precej manj kot smo na začetku predpostavili.

Prej smo predpostavili, da je zgoščevalna funkcija regularna, torej $\mu(h) = 1$ in število poskusov$q = r^{\frac{1}{2}}$. Seveda pri funkcijah z manjšo stopnjo regularnosti oz. z manjšim ravnotežjem je potrebnih preceh manj poskusov, na primer, če ima zgoščevalna funkcija $\mu(h) = 0$, torej je h konstantna funkcija, najde napad trk v času $O(1)$. Kot primer lahko vzamemo še funkcijo, ki ima mero ravnotežja enako $\mu = 0.5$. Ta najde trk v približno $r^{\frac{1}{4}}$ poskusih.

1.2.2 Dokaz

Dokaz lahko izpeljemo s pomočjo uvedbe teorema, ki poda zgornjo in spodnjo mejo na $C_h(q)$, ki sta povezana med sabo z istimi faktorji.

TEOREM

Naj bo $h: D \to R$ zgoščevalna funkcija. Naj bo $d = |D|$ in $r = |R|$ ter predpostavimo, da velja $d > r \geq 2$.

Naj bo $\alpha \geq 0$ katerokoli realno število. Potem za poljubno celo število $q \geq 2$ velja

$$\left(1 - \frac{\alpha^2}{4} - \alpha\right) \cdot \binom{q}{2} \cdot \left[\frac{1}{r^{\mu(h)}} - \frac{1}{d}\right] \leq C_h(q) \leq \binom{q}{2} \cdot \left[\frac{1}{r^{\mu(h)}} - \frac{1}{d}\right] \tag{3.4}$$

Če želimo, da bo držala tudi trditev o spodnji meji, mora veljati tudi

$$q \leq \alpha \cdot \left(1 - \frac{r}{d}\right) \cdot r^{\frac{\mu(h)}{2}} \tag{3.5}$$

DOKAZ TEOREMA

Naj bo $p = r^{-\mu(h)} - \frac{1}{d}$

Naj $[q]_2$ predstavlja množico vseh dvo-elementnih podmnožic množice$[q]$. Spomnimo se, da napad izbere naključna števila $x_1, \ldots, x_q$ iz domene D. Povežemo katerokoli dvo-elementno množico $I = \{i, j\} \in [q]_2$ z naključno spremenljivko X_I, ki zavzame vrednost 1 takrat, ko x_i, x_j predstavljata trk (torej $x_i \neq x_j$ in $h(x_i) = h(x_j)$), drugače pa 0. Naj bo $X = \sum_{I \in [q]_2} X_I$ število trkov (pri tem načinu štetja trkov, v primeru, da ima n različnih točk iswto vrednost funkcije, skupaj prispevajo $\frac{n(n-1)}{2}$ v vsoti X). Za poljubni $I \in [q]_2$ imamo

$$E[X_I] = \Pr[X_I = 1] = \sum_{i=1}^{r} \frac{d_i(d_i - 1))}{d^2} = \sum_{i=1}^{r} \frac{d_i^2}{d^2} - \sum_{i=1}^{r} \frac{d_i}{d^2} = r^{-\mu(h)} - \frac{1}{d} = p \tag{3.6}$$

$$E[X] = \sum_{i=1} E[X_I] = \binom{q}{2} p \tag{3.7}$$

Zgornja meja teorema je le proprosta aplikacija Markove neenakosti in dobljene enačbe:

$$C_h(q) = \Pr[X \geq 1] \leq \frac{E[X]}{1} = \binom{q}{2} p \qquad (3.8)$$

Preostane nam še dokaz spodnje meje teorema. Naj bo $[q]_{2,2}$ množica vseh dvo-elementnih podmnožic $[q]_2$. Preko principa vključitve in izključitve velja

$$\begin{aligned} C_h(q) &= Pr\left[V_{I\in[q]_2} X_I = 1\right] \\ &\geq \sum_{I\in[q]_2} Pr\left[X_I = 1\right] - \sum_{\{I,J\}\in[q]_{2,2}} Pr\left[X_I = 1 \wedge X_j = 1\right] \end{aligned} \qquad (3.9)$$

Vemo da velja

$$\sum_{I\in[q]_2} \Pr[X_I = 1] = \sum_{I\in[q]_2} E[X_I] = E[X] = \binom{q}{2} p \qquad (3.10)$$

Sedaj moramo le še dokazati, da velja

$$\sum_{\{I,J\}\in[q]_{2,2}} \Pr[X_I = 1 \wedge X_j = 1] \leq -\left(\frac{\alpha^2}{4} + \alpha\right)\binom{q}{2} p \qquad (3.11)$$

Naj bo E taka množica vseh $\{I,J\} \in [q]_{2,2}$, da velja $I \cap J = \emptyset$ in naj bo N množica vseh $\{I,J\} \in [q]_{2,2}$, da velja $I \cap J \neq \emptyset$. Potem

$$\begin{aligned} &\sum_{\{I,J\}\in[q]_{2,2}} Pr\left[X_I = 1 \wedge X_J = 1\right] \\ &\qquad = \sum_{\{I,J\}\in E} Pr\left[X_I = 1 \wedge X_J = 1\right] \\ &\qquad + \sum_{\{I,J\}\in N} Pr\left[X_I = 1 \wedge X_J = 1\right] \end{aligned} \qquad (3.12)$$

kjer prvi del desne strani enačbe označimo z S_E, drugi del pa z S_N. Sedaj trdimo, da velja

$$S_E \leq \binom{q}{2} \frac{1}{4} \alpha^2 p \qquad (3.13)$$

$$S_N \leq \binom{q}{2} \alpha p \qquad (3.14)$$

Najprej dokažemo trditev glede S_E (enačba 3.13)

$$S_E = \sum_{\{I,J\}\in E} Pr[X_I = 1] \cdot Pr\left[X_J = 1\right] = |E| \cdot p^2 \qquad (3.15)$$

$$S_E = \frac{1}{2}\binom{q}{2}\binom{q-2}{2}p^2 = \binom{q}{2}p\left[\frac{1}{2}\binom{q-2}{2}p\right] = \binom{q}{2}p\frac{q^2-5q+6}{4}p \qquad (3.16)$$

$$S_E < \binom{q}{2}pq^2\frac{p}{4} \le \binom{q}{2}p\alpha^2 r^{\mu(h)}\frac{p}{4} \le \frac{1}{4}\alpha^2\binom{q}{2}p \qquad (3.17)$$

Sedaj dokažemo še trditev glede S_N (enačba 3.14)

$$S_N = \sum_{\{I,J\}\in N} Pr[X_I = 1 \wedge X_J = 1] = |N| \cdot \sum_{i=1}^{r}\frac{d_i(d_i-1)^2}{d^3} < \frac{|N|}{d^3}\cdot\sum_{i=1}^{r} d_i^3 \qquad (3.18)$$

Če izračunamo N in ga vstavimo v enačbo

$$\begin{aligned} |N| &= \frac{1}{2}\binom{q}{2}\binom{q}{2} - \frac{1}{2}\binom{q}{2} - \frac{1}{2}\binom{q}{2}\binom{q-2}{2} \\ &= \binom{q}{2}\cdot\left[\frac{1}{2}\binom{q}{2} - \frac{1}{2}\binom{q-2}{2} - \frac{1}{2}\right] \end{aligned} \qquad (3.19)$$

$$|N| = \binom{q}{2}\cdot\left[\frac{q(q-1)}{4} - \frac{(q-2)(q-3)}{4} - \frac{1}{2}\right] = \binom{q}{2}\cdot(q-2) \qquad (3.20)$$

$$S_N < \binom{q}{2}\cdot q\cdot\left[\frac{1}{d^3}\cdot\sum\nolimits_{i=1}^{r} d_i^3\right] \qquad (3.21)$$

Da podamo zgornjo mejo, gledamo na $d_1, \dots, d_r$ kot na spremenljivke optimizacijskega problema maksimizacije vsote $d_1^3 + \cdots + d_r^3$ pod omejitvami $\sum_{i=1}^{r} d_i^2 = d^2 \cdot r^{-\mu(h)}$. Rešitev tega problema je v točki $d_1 = d \cdot r^{-\frac{\mu(h)}{2}} \wedge d_i = 0$ za vsak $i = 2, \dots, r$, kar pomeni, da

$$\sum_{i=1}^{r} d_i^3 \le d^3 r^{-\frac{3\mu(h)}{2}} \qquad (3.22)$$

Če vstavimo to v prejšnjo enačbo, dobimo

$$S_N < \binom{q}{2}\cdot q\cdot\left[\frac{1}{d^3}\cdot\sum\nolimits_{i=1}^{r} d_i^3\right] \le \binom{q}{2}\cdot q\cdot\frac{1}{d^3}\cdot d^3 r^{-\frac{3\mu(h)}{2}} = \binom{q}{2}\cdot q\cdot r^{-\frac{3\mu(h)}{2}} \qquad (3.23)$$

$$S_N < \binom{q}{2}\alpha\left(1-\frac{r}{d}\right)r^{\frac{\mu(h)}{2}} r^{-\frac{3\mu(h)}{2}} \le \binom{q}{2}\alpha\left(1-\frac{r}{d}\right)r^{-\mu(h)} \le \binom{q}{2}\alpha p \qquad (3.24)$$

1.3 Primer

V priloženem programu je za poljubne parametre narejeno iskanje trkov na prej opisani način. Sam predel z napadom na zgoščevalne funkcije lahko uporabnik najde na zavihku imenovanem »Zgoščevalne funkcije«, aktivira pa ga s klikom na gumb »Začni poskus«. Parametri programa so razpon zgoščevalne funkcije (v bitih), vrsta vgrajene zgoščevalne funkcije in število zaporednih poskusov. Na voljo so štiri funkcije:

- Naključna zgoščevalna funkcija, kjer se vsakemu elementu ob prvem klicu naključno dodeli fiksna vrednost v razponu, torej vrednost $x_i \in \{0,1\}^{|R|}$.
- Konstantna zgoščevalna funkcija, ki vrača vedno isto vrednost in se v programu nahaja zgolj zaradi praktične izvedbe vseh omenjenih zgoščevalnih funkcij.
- Realna zgoščevalna funkcija, ki je vgrajena v MS Visual Studio 2011Beta, in se uporablja za shranjevanje elementov v zgoščevalne tabele. Ta funkcija je prilagojena za program tako, da se vedno vzame le zadnjih |R| bitov, da je primerjava z drugimi funkcijami konsistentna.
- Neuravnovešena naključna funkcija, ki se vede podobno kot naključna, le da z izbrano verjetnostjo element preslika v eno izmed že znanih vrednosti in s tem obteži nekatere vrednosti, medtem ko »razbremeni« ostale

IV. Napad na DES

DES ali Data Encryption Standard je šifra, ki so jo leta 1976 ZDA izbrale za federalni standard za procesiranje informacij (Federal Information Processing Standard). Od takrat se je DES razširil po celem svetu. Skozi čas se je sicer našlo nekaj teoretičnih šibkosti te šifre, ampak te slabosti so večinoma nemogoče za izkoristiti v praksi. Zanimivo je to, da kljub temu, da ima kriptoanaliza DES med bločnimi šiframi daleč največ izdanih informacij, a zanjo je še vedno najbolj izvedljiv ravno »brute-force« napad. Sicer obstajajo drugi napadi, ki so teoretično časovno manj zahtevni kot »brute-force« napadi, saj izkoriščajo znane kriptoanallitične lastnosti DES, vendar za izvedbo teh algoritmov bi potrebovali nerealno količino znanega »plain«-teksta (za razlagko si naj bralec prebere referenco [7]):

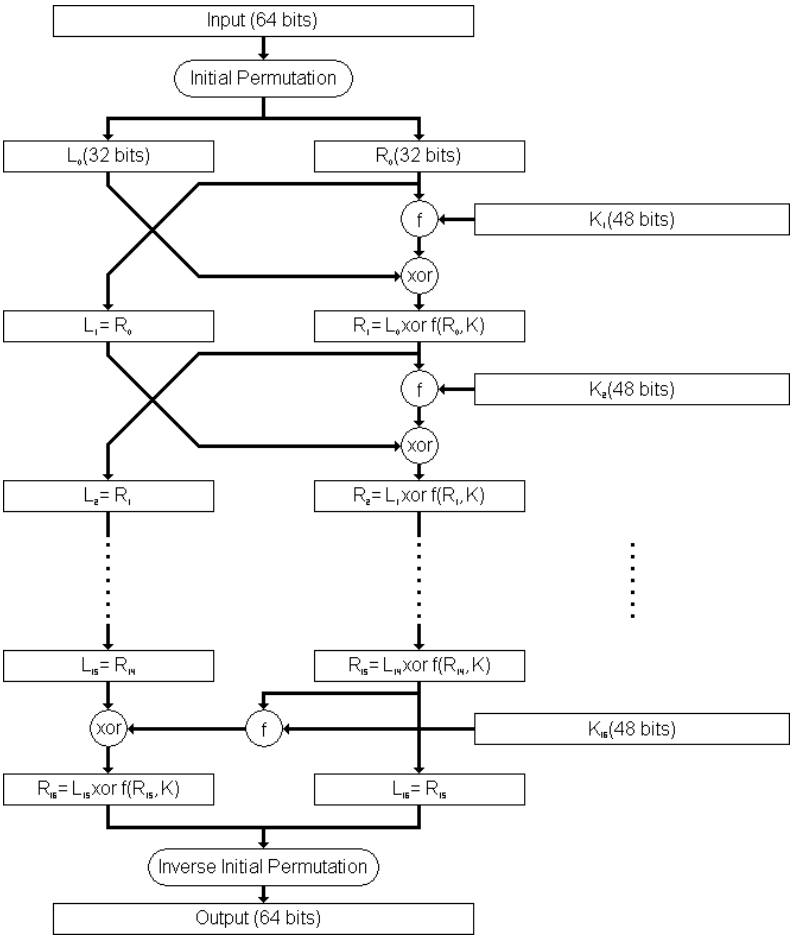
- Daviedov napad, ki je specializiran napad na DES, in ga je predlaga D. Davies (kasneje sta ga izboljšala Biham in Biryukov). Daviedov napad potrebuje 2^{50} znanih plain-tekstov in ima verjetnost uspeha 51%
- Diferencialna kriptoanaliza, ki sta jo objavljena A.Shamir in E.Biham leta 1990. Da zlomi celoten ključ (vseh 16 podključev) potrebuje se do 2^{47} izbranih plain-tekstov
- Linearna kriptoanaliza, ki jo je objavil M.Matsui leta 1992. Ta metoda za odkritje ključa potrebuje 2^{43} izbranih plain-tekstov

1 Data Encryption Standard

DES procesira bloke čistega teksta velikosti n = 64 bitov, in nam kot rezultat vrne 64 bitni blok, ki predstavlja šifro. Efektivna velikost skritega ključa je K = 56 bitov, zato je ključ K definiran kot 64 bitni, od katerih se lahko 8 bitov (zadnji bit vsakega 8-bitnega podbloka) uporabljajo kot paritetni biti. DES uporablja v šifriranju in dešifriranju Fiestelovo funkcijo f, ki se izvaja v 4 korakih:

- Rezširitev – 32 bitni polblok se razširi na 48 bitov z uporabo razširitvene permutacije (najprej jo bomo označili z E ali E-box) tako, da duplicira polovico bitov polbloka
- Mešanje ključev – rezultat rezširitve se združi s podključem preko XOR operacije. Podključ se dobi iz glavnega ključa z uporabo urnika ključev
- Zamenjava – v tem koraku se blok razdeli na osem 6 bitnih kosov, kjer vsak izmed teh blokov zamenja svojih šest bitov z štirimi biti glede na nelinearno transformacijo, ki je zapisana v vpogledni tabeli. Brez tega koraka bi bil DES linearen in lahko zlomljiv. Ta del bomo imenovali S-box transformacija
- Permutacija – na koncu se 32 izhodnih bitov še permutira s fiksno permutacijo

Celotna shema DES je razvidna iz slike »Slika 1 - Potek kodiranja z DES«



Slika 1 - Potek kodiranja z DES

2 Pristop z rojstnodnevnim napadom

2.1 Opis postopka

Najprej se spomnimo, da je iz samega zakodiranega teksta dobiti R_{16} in L_{16} trivialna naloga. Po zadnjen krogu DES, velja:

$$R_{16} = 1_{15} \oplus f(R_{15}, K_{16}), \qquad L_{16} = R_{15} \tag{4.1}$$

Iz česar sledi, da je

$$f(L_{16}, K_{16}) = R_{16} \oplus L_{15} \tag{4.2}$$

Seveda nam L_{15} in K_{16} nista znana.

Trk, ki ga mi iščemo, prihaja od dveh zakodiranih tekstov c in c', ki uporabljata isti ključ K_{16} izpolnjuje pogoj $R'_{16} = R_{16} \wedge L'_{16} \neq L_{16} \wedge L'_{15} = L_{15}$. Če je ta domneva resnična, velja tudi

$$f\left(L'_{16}, K_{16}\right) = f(L_{16}, K_{16}) \tag{4.3}$$

Če sedaj vzamemo še razširitveno funkcijo E iz f, lahko definiramo EL_{16} in K_{16} kot

$$EL_{16} = EL_{16}[1] \| EL_{16}[2] \| EL_{16}[3] \| EL_{16}[4] \| EL_{16}[5] \| EL_{16}[6] \| EL_{16}[7] \| EL_{16}[8] \quad (4.4)$$

$$K_{16} = K_{16}[1] \| K_{16}[2] \| K_{16}[3] \| K_{16}[4] \| K_{16}[5] \| K_{16}[6] \| K_{16}[7] \| K_{16}[8] \quad (4.5)$$

kjer je vsak $EL_{16}[j], K_{16}[j], j = 1, \dots, 8$ blok, dolg 6-bitov in operator || določuje združevanje dveh nizov (vzeto po [8]). Tako bodo vhodni podatki za vsak S-box enaki

$$S[i]\left(EL_{16}[i] \oplus K_{16}[i]\right) = S[i](EL'_{16}[i] \oplus K_{16}[j]) \quad (4.6)$$

Par $(EL_{16}[i] \oplus K_{16}[i], EL'_{16}[i] \oplus K_{16}[i])$ je le trk nelinearne funkcije S[i], na katero lahko gledamo kot naključno ali psevdo-naključno funkcijo, zato potebujemo za iskanje tega trka evalverati le 2^4 argumentov. Možnih vrednosti ključa $K_{16}[i]$, za katere velja da je $EL_{16}[i] \neq EL'_{16}[i]$ je le 2^2. Če pa je $EL_{16}[i] = EL'_{16}[i]$ pa je lahko ključ katerakoli 6-bitna vrednost. Za vsako izmed $S[j], j = 1, \dots, 8$ moramo integrirati vsak niz a dolžine 6 bitov z $EL_{16}[j], EL'_{16}[j]$. S pomočjo prej napisane enačbe lahko določimo vse kandidate za $K_{16}[j]$.

2.2 Opis rojstnodnevnega napada

Rojstnodnevni napad poteka lahko v naslednjih korakih.

1. Najprej zberemo primerna 64 bitna zakodirana sporočila, ki so bila zakodirana s pomočjo istega ključa K. Za vsako sporočilo izračunamo primerna L_{16} in R_{16}. Izberemo sporočila z enakim R_{16} in jih označimo z oznako $C_{R_{16},K}$.
2. Izračunamo kandidate za vsak $K_{16}[j], j = 1, \dots, 8$ in naključno izberemo dva zakodirana sporočila $c, c' \in C_{R_{16},K}$. Nato integriramo vsak 6-bitni niz a z $EL_{16}[j], EL'_{16}[j]$. Kandidate za ključ $K_{16}[j]$ določimo s preverjanjem $S[j](EL_{16}[j] \oplus a) \stackrel{?}{=} S[j](EL'_{16}[j] \oplus a)$
3. Če nismo našli nobenega kandidata za $K_{16}[i], i = 1, \dots, 8$ se vrnemo na 2. Korak
4. Iz kandidatov $K_{16}[1], \dots, K_{16}[8]$ določimo kandidate za K_{16}
5. S pomočjo K_{16} in »urnika ključev« določimo kandidate za K
6. Če imamo podan t.i. plaintext in primereno zakodirano besedilo lahko s pomočjo evalvacij poskusimo razločiti ključ od drugih kandidatov
7. Če nam ne uspe razločiti ključa, se vrnemo na 2. korak.

V. Paralelna izvedba rojstnodnevnega napada

Zaradi visokega napredka na področju strojne opreme in nagibanja tehnologij k porazdeljenim arhitekturam, se v zadnjih nekaj letih restrukturira starejše algoritme, da lahko tečejo na večih precesorjih (CPU) ali pa celo na grafični kartici (GPU), ki je namenjena hitremu preračunavanju. Trerba se je seveda zavedati omejitev, ki jih tak način kreiranja algoritmov prinaša in tudi arhitekture, na katerih lako delujejo taki algoritmi. Da ne bomo v preveč časa zapravili s pojasnjevanjem osnovnih konceptov paralelnega programiranja, si lahko bralec prebere [9], kjer je celoten koncept primerno opisan. V tem poglavju bomo za CPU in GPU posebej opisali po eno metodo, metodo vzporednih napadov ter metodo napadov s skupno tabelo.

1 Paralelizem na nivoju CPU

Paralelizem na nivoju CPU je v primerjavi z nekaterimi, bolj osnovnimi, relativno mlada praksa, saj se je zečal razvijate s prihodom večjedrnih procesorjev. Kljub temu, da ne zniža same stopnje časovne

zahtevnosti, lahko znatno pospeši program, in zato se praksa paralelizacije algoritmov dandanes vedno bolj uporablja.

1.1 Konstrukcija metode – Metoda Vzporednih Napadov

Kot omenjeno v poglavju »Iskanje diskretega logaritma«, ki opisuje osnovno metodo iskanja trkov v GDLP, lahko izračunamo verjetnost uspešnosti navadnega rojstnodnevnega napada na GDLP na

$$p(trk) = 1 - \frac{n!\binom{H}{n}}{H^n} \tag{5.1}$$

Pri metodi vzporednih napadov bomo istočasno izvajali P manjših napadov, kjer P predstavlja število procesorjev oz. računalnikov, ki izvajajo ta napad. Verjetnost za uspešnost tega napada je pri m poskusih v posameznih podnapadih enaka

$$P_{mvn} = 1 - \left(\frac{m!\binom{H}{m}}{H^m}\right)^p \tag{5.2}$$

Čeprav algoritem ni popolnoma paraleliziran, je izvajanje tega algoritma precej hitrejše kot izvajanje osnovnega algoritma. Za približno isto verjetnost trka kot pri osnovnem napadu potrebujemo

$$m \geq \frac{\sqrt{p} + \sqrt{p + 4n(n-1)}}{2\sqrt{p}} \tag{5.3}$$

poskusov.

1.1.1 Dokaz

Za dokaz verjetnosti zadetka bomo uporabljali aproksimasijo enačbe 5.2 s Taylorjevo vrsto, saj je iskanje trkov problem z zelo majhnimi verjetnostimi trkov in je ravno zato ta aproksimacija dokaj natančna.

Najprej definiramo naš problem. Iščemo tak m, da bo veljalo

$$1 - \frac{n!\binom{H}{n}}{H^n} = 1 - \left(\frac{m!\binom{H}{m}}{H^m}\right)^p \tag{5.4}$$

torej, da bo verjetnost zadetkov v originalnem in paralelnem algoritmu ista.

Ko to prevedemo na taylorjevo vrsto, dobimo

$$e^{-\frac{n(n-1)}{2H}} = e^{-\frac{m(m-1)}{2H}p} \tag{5.5}$$

Iz tega lahko izpeljemo

$$n(n-1) = m(m-1)\,p \tag{5.6}$$

$$m^2 - m - \frac{n(n-1)}{p} = 0 \tag{5.7}$$

Ko rešimo to preprosto enačbo, dobimo dve rešitvi

$$m_{1,2} = \frac{\sqrt{p} \pm \sqrt{p + 4n(n-1)}}{2\sqrt{p}} \tag{5.8}$$

Ker je n > 1, in zato velja

$$\sqrt{p + 4n(n-1)} > 0 \tag{5.9}$$

in ker vemo, da je mora biti m>0, lahko uporabimo samo pozitivno rešitev, torej

$$m_{1,2} = \frac{\sqrt{p} + \sqrt{p + 4n(n-1)}}{2\sqrt{p}} \tag{5.10}$$

1.1.2 Prednosti

Največja prednost tega algoritma je kombinacija njegove časovne in prostorske zahtevnosti. Ker se nitim ni potrebno sinhronizirati do samega konca izvajanja ne izgubljamo časa z nepotrebnim medsebojnim čakanjem. Prostorska zahtevnost je premo sorazmerna s prostorsko zahtevnostjo osnovnega napada in ker vsaka nit drži svoj »pomnilnik« je algoritem možno izvajati na večih računalnikih in tako razporediti zahteve po prostoru na več računalnikov. Dodatno nam občutek porazdeljenosti da odsotnost odvečnih sinhronizacij.

Zaradi omenjenih prednostih je ta algoritem odličen za porazdeljen napad.

1.1.3 Slabosti

Očitna slabost tega algoritma je število primerov, ki je more vsak računalnik obdelati, saj s številom procesorjev sicer pada, a vedno počasneje. Če bi osnovni algoritem obdelal isto število primerov, kot vsi procesorji/računalniki pri paralelnem algoritmu skupaj, bi precej hitreje bila verjetnost zadetka precej večja, kot pri paralelnem algoritmu.

1.1.4 Možne izboljšave

Močna možna izboljšava bi bila, če bi si procesorji/računalniki med sabo izmenjevali več informacij, kot si jih pri našem paralelnem algoritmu. Sicer bi porabili čas, za izmenjavo informacij, a pri pravilno formiranih sporočilih bi lahko precej hitreje našli trk.

1.2 Primer

V priloženem programu je ta algoritem implementiran pod istim zavihkom kot osnovni napad, če uporabnik izbere opcijo »Računaj Paralelno«. Ker je število procesorjev v tem primeru dovolj majhno (torej se program ne poganja na oblaku ali porazdeljeno na večih računalnikih), se za število pregledanih primerov izbere $\frac{n}{L}$, kjer je n število pregledanih primerov v originalnem algoritmu, L pa število procesorjev, ki jih ima uporabnik na računalniku (število procesorjev se določi avtomatsko). Če ima uporabnik na razpolago le 1 procesor (torej je L=1), ne bo opazil razlike med osnovnim in paralelnim algoritmom.

2 Paralelizem na nivoju GPU

Zaradi same količine procesnih enot, ki se nahajajo na grafičnih karticah, je uporaba GPU vedno močnejša v eksperimentalnih paralelnih algoritmih. Zato lahko algoritem, v primeru da je dovolj dobro paraleliziran, z uporabo GPU v primerjavi s CPU deluje hitreje z visokim faktorjem pospešitve (na primer do 100x hitrejše množenje matrik pri povprečnih grafičnih karticah).

2.1 Konstrukcija metode – Metoda Napadov s Skupno Tabelo

Problem, ki se pojavi pri paralelizaciji algoritmov na GPU je zelo majhna količina hitrega pomnilnika, ki je na voljo. Tako prenosi med navadnim in GPU pomnilnikom lahko vzamejo relativno veliko časa, zato treba paziti, da bo le-teh čim manj.

V algoritmu »Metoda Napadov s Skupno Tabelo« (Shared Table Attack - STA) bomo sklepali, da si vsi procesorji delijo nek skupni pomnilnik (kar seveda drži, če ima računalnik le eno grafično kartico).

Algoritem STA

```
Vnosna polja:
      gen = generator grupe
      n = velikost tabele
      y = iskano število
      N = |G|
      i = število iteracij

Izhodno polje:
      Zabeleženi trki

1 : TX = int[n]
2 : izvajaj vzporedno:
3 :     x = izberi naključno celo število
4 :     r = (y^x % N) % n
5 :     če je TX[r] prazen:
6 :         zakleni za uporabo TX[r]
7 :         TX[r] = x
8 :     ponavljaj dokler se TX ne neha polniti
9: izvajaj vzporedno:
10:     ponavljaj n-krat:
10:         x = izberi naključno celo število
11:         r = gen^x % N
12:         y' = y^TX[r] % N
13:         če je r = y', zabeleži trk
14:     ponavljaj i-krat
15: vrni zabeležene trke
```

Algoritem 1 - Metoda Napadov s Skupno Tabelo

Pri tej obliki napada nadoknadimo prostor z ogromno računsko zmogljivostjo. Namesto da bi držali veliko izračunov (kot na primer pri osnovnem napadu) in s tem dvigovali verjetnost trka, raje naredimo ogromno količino izračunov in iščemo trke v manjši množici.

2.1.1 Prednosti

Zaradi izkoriščanja velike procesorske moči grafičnih kartic, je ta napad zelo hiter in bo z nadaljnim razvojem GPU-jev postal še hitrejši. Dobra lastnost tega napada je tudi dejstvo, da ne porabi veliko pomnilnika, ampak za sam napad uporablja le pomnilnik grafične kartice in s tem ne izgublja odvečnega časa za prenos podatkov med diskom in hitrim pomnilnikom.

2.1.2 Slabosti

Slaba stran tega algoritma je, da čeprav je močno paraleliziran, se še vedno lahko izvaja le na enem računalniku, saj uporablja za izvajanje deljen pomnilnik – vsi procesorji dostopajo do skupne tabele, v kateri držijo preračunane vrednosti. Dodatna slaba lastnost algoritma je pomnilnik, ki je nekoliko bolj omejen kot pri uporabi CPU.

2.1.3 Možne izboljšave

Zanimiva izboljšava bi bila, če bi STA združili s prej omenjenim MVN, torej da bi več računalnikov istočasno izvajalo STA in si izmenjevalo informacije glede iskanja trkov.

2.2 Primer

V priloženem programu je v istem zavihku kot osnovni napad in večnitni napad dosegljiv tudi večnitni napad s pomočjo GPU, do katerega se pride, če uporabnik izbere možnost »Računaj na GPU«. Sama izvedba tega algoritma je le osnovna, saj nimamo na uporabo najnovejših grafičnih kartic, ki podpirajo tudi podatkovne tipe kot so double in long, tako da v primeru lahko uporabnik išče trke le v majhnih grupah (ranga 32767 ali manj).

Za uporabo GPU program uporablja knižnjico C++ AMP, ki v VS2011Beta preko jezika C++ omogoča izvajanje preprostih ukazov na GPU.

VI. Zaključek in diskusija

1 Zaključek

V tej seminarski nalogi smo pregledali različne uporabe rojstnodnevnega napada oz. bolj splošno, rojstnodnevnega paradoksa v različnih kriptografičnih napadih. Nato smo se poglobili v področje paralelizacije rojstnodnevnega napada in predstavili dva algoritma, kjer ima vsak izmed njiju svojo stopnjo porazdeljenosti in način delovanja ter sinhroniziranja večih procesov.

2 Diskusija

Čeprav so rojstnodnevni napadi še vedno praktično neizvedljivi, postajajo vedno boljši načini za izvajanje le-teh. Kot primer lahko vzamemo tudi napad MVN, ki ne sloni na računski moči enega procesorja, temveč na množici računalnikov (vsak s svojim pomnilnikom), ki istočasno izvajajo napad. Tako bi lahko ta napad klicali tudi DBA oz. »Distribuirani Rojstnodnevni Napad« (Distributed Birthday Attack), saj ga lahko istočasno izvaja cela množica računalnikov in s tem je za uspeh potrebno bistveno manj časa.

VII. Literatura

[1] Hong-Hsin Chen, Birthday Problem and Birthday Attack, October 26, 2010

[2] M. Girault, R. Cohen in M. Campana: A Generalized birthday attack, LNCS 330, Eurocrypt 1988

[3] Paul C. van Oorschot in Michael J. Wiener, Bell-Northern Research, Parallel Collision Search with Application to Hash Functions and Discrete Logarithms

[4] Kevin S. McCurley, The Diuscrete Logarithm Problem, Proceedings of Sy,posia in Applied Mathematics, Volume 42, 1990

[5] Li An-Ping, Birthday attack to discrete logarithm, Beijing 100080 P.R. China

[6] M. Bellare in T. Kohno, Hash function balance and its impact on birthday attacks, Advances in Cryptology, EUROCRYPT '04, lecture Notes in Computer Science Vol. 3027, C. Cachin and J. Camenisch ed., Springer-Verlag, 2004.

[7] E.Biham in A.Biryukov, An Improvement of Davies' Attack on DES, Journal of Cryptology, 10(3), 195-206, 1997

[8] Zhengjun Cao, How to Launch A Birthday Attack Against DES, 2008

[9] Guy E. Blelloch in Bruce M. Maggs, Parallel Algorithms, School of Computer Science, Carnegie Mellon University

[10] Marcus Peinado in Ramarathnam Venkatesan, Highly parallel cryptographic attacks

[11] Spletna knižnjica znanja http://msdn.microsoft.com/en-us/library (za programski del)