МАТЕМАТИЧКИ ФАКУЛТЕТ
УНИВЕРЗИТЕТ У БЕОГРАДУ

Проналажење колизија код криптографских хеш функција

Магистарски рад

Марко Царић

Ментор: проф. др. Миодраг Живковић

Садржај

Глава 1

Увод

Предмет овог рада је анализа основних поступака проналажења колизија у фамилији MD4 криптографских хеш функција, односно прецизније код алгоритама MD4 и SHA-0. Алгоритму MD4 посвећена је већа пажња, из чега је проистекао поступак за аутоматску претрагу путања промена битова уколико се две поруке разликују у једном биту. Овај поступак (описан у поглављу 3.6) уједно представља оригинални допринос рада; у циљу провере коректности је тај поступак у одређеним случајевима програмски имплементиран. Исправност поступка потврђена је примерима нових колизија.

У раду је анализа слабости хеш функција усмерена на диференцијални напад, за који се показало да мање или више успешно проналази колизије код свих постојећих хеш функција. Најпре су приказани поступци Ванг и Ју за напад на алгоритамMD4. Такође је предложено поступак за проналажење колизије уколико се две поруке разликују у једном биту. Пошто диференцијални напад кроз алгоритам прати развој почетне разлике две поруке, коришћене су означене разлике, као алат који описује стање разлика у сваком кораку алгоритма. Напад на алгоритам SHA-0 приказан је у основној (Шобон, Жу) и усавршеној форми (Бихам, Чен); при томе се не прати кретање разлика између порука у облику означених разлика, већ се посматра вероватноћа са којом се долази до колизије.

Искористио бих ову прилику да поменем особе које су ми директно или индиректно помогле у изради ове тезе. Захваљујем се свом ментору, професору Миодрагу Живковићу, на помоћи и издвојеном времену у току којег сам усмераван у правцу идеје која је у раду приказана. Искрено се надам да ће и наставак овог рада бити обликован под његовим надзором. Такође сам захвалан професору Предрагу Јаничићу на корисним саветима и сугестијама. Неизоставно се захваљујем професору Драгану Плескоњићу, који ме је својом речју и делом инспирисао да прихватим изазове који истовремено могу бити и тешки и лепи. Професору, пуковнику Ранку Дашићу дугујем више него

дубоку захвалност за безброј драгоцених стручних и животних савета. Захвалан сам свом драгом оцу Мирку на немирном духу, неопходном да се иза понуђене површности уочи дубљи смисао. Захвалан сам мојој вољеној Катарини и слатким зврковима Марији и Зоји, који ми сваки дан чине посебно лепим. Захваљујем се свима који су ми на било који начин помогли или олакшали рад на овој тези. Овај рад, пре свих, посвећујем мојој драгој мајци Зоји, која на жалост није могла физички да подели сву моју радост поводом завршетка основних и последипломских студија.

Глава 2

Криптографске хеш функције

Намена Хеш алгоритма је добијање поруке фиксне дужине од било ког основног текста (енгл. plaintext). На тај начин добијени сажетак, одосно хеш вредност, репрезентује полазну поруку. Уколико се полазна порука промени и њена хеш вредност (енгл. message digest, hash) не би требало да остане иста. На тај начин, хеш алгоритам се може користити за проверу интегритета поруке. Пошто је скуп свих могућих порука већи од скупа свих могућих хеш вредности фиксне дужине, постоји могућност да две различите поруке дају исту хеш вредност (колизија). Ако се од 512-битног улазног блока добија рецимо 160-битна хеш вредност, у просеку чак $2^{512}/2^{160}$ порука даје исту хеш вредност. Међутим, није једноставно ефективно пронаћи две такве поруке. Иако је неефикасно тражити колизију грубом силом, постоје методе за њено брже проналажење. Хеш функције се деле на оне које зависе, односно не зависе од тајног кључа. Овај рад бави се хеш функцијама без кључа.

Нека је $B = \{0, 1\}$.

Дефиниција 2.0.1. *Нека је $X = B^*$ низ битова произвољне дужине, а $Y = B^n$ за неко $n \geq 1$. Функција $h : X \to Y$ назива се хеш функција. Слика (енгл. image) $y = h(x)$ од неке поруке (енгл. preimage) $x \in X$ назива се сажетак, извод односно хеш вредност од x.*

2.1 Особине хеш функција

Уобичајена претпоставка је да хеш функција треба да задовољава следеће услове [25]:

Дефиниција 2.1.1. *Хеш функција поседује једносмерност (енгл. one way, preimage resistance) ако се из поруке x лако налази хеш вредност $H(x)$, али не и обрнуто.*

За једносмерну хеш функцију дужине n, напад грубом силом на једносмерност свео би се у просеку на 2^{n-1} покушаја (у смислу: потребно је у просеку формирати 2^{n-1} различитих порука док се не пронађе нека са хеш вредношћу $H(x)$).

Дефиниција 2.1.2. *Хеш функција поседује слабу отпорност на колизију (енгл. second preimage resistance, weak collision resistance) ако је за дато x практично немогуће пронаћи $y \neq x$ тако да је $H(x) = H(y)$.*

За хеш дужине n, налажење поруке са истом хеш вредношћу као задата порука, захтева у просеку 2^{n-1} покушаја.

Дефиниција 2.1.3. *Хеш функција поседује јаку отпорност на колизију (енгл. (strong) collision resistance) ако је немогуће наћи било који пар различитих порука x и y, тако да важи $H(x) = H(y)$.*

За хеш дужине n, налажење две поруке са истом хеш вредношћу захтева у просеку $2^{n/2}$ покушаја (рођендански напад).

Може се показати (под разумљивим претпоставкама) да ако је функција јако отпорна на колизију, онда је и слабо отпорна на колизију. Због тога се обично од хеш функција захтева јака отпорност на колизију.

2.2 Рођендански напад

Од интереса је следећи проблем. Ако је дата порука x и хеш функција H, са n могућих хеш вредности и фиксираним излазом $H(x)$ и ако је H примењена на k улаза (порука), колико треба узети порука (k) тако да са вероватноћом $1/2$ барем за једну поруку y важи $H(y) = H(x)$?

За случајно изабрано y, вероватноћа да важи $H(y) = H(x)$ је $1/n$. Обрнуто, вероватноћа да је $H(x) \neq H(y)$ износи $1 - 1/n$. Ако се генерише k случајних вредности за поруку y, вероватноћа да за сваку од њих важи $H(x) \neq H(y)$ износи $(1 - 1/n)^k$. Одатле следи да вероватноћа да бар за једно y важи $H(y) = H(x)$ износи $1 - (1 - 1/n)^k$. Ако се узме у обзир да за мале вредности a важи апроксимација $(1-a)^k \approx 1-ka$ тада важи $1 - (1 - 1/n)^k \approx 1 - (1 - k/n) = k/n$. Ако је поменута вероватноћа $1/2$, онда важи $k = n/2$. Специјално, за m-тобитни хеш, број могућих излаза је 2^m, па је тражена вредност $k = 2^m/2 = 2^{m-1}$.

Претходно разматрање односи се на тражење поруке која даје исти хеш као и задата порука. Интуитивно је јасно да је вероватноћа налажења овакве поруке мала (за n-тобитни хеш је 2^{1-n}). Међутим, овај случај се често поистовећује са случајем када се траже било које две поруке које производе исти хеш, што је далеко лакши задатак. Налажење оваквог пара порука чини колизију у хеш алгоритму, а сам чин налажења (напад) назива се рођендански напад. Назив потиче од решавања аналогног проблема приликом тражења било ког пара ученика у резреду који су рођени истог датума. Вероватноћа да у

одељењу од k ученика ни која два не буду рођена истог датума износи $V_n^k/\overline{V}_n^k = n!/((n-k)!n^k) = 365!/((365-k)!365^k)$ Следи да вероватноћа да су барем два ученика рођена истог датума износи $1 - 365!/((365-k)!365^k)$. Занимљиво је да се већ за $k = 23$ ученика може очекивати са вероватноћом преко $1/2$ да су нека два рођена истог датума. Постоји аналогија: налажење две поруке са истом хеш вредношћу одговара налажењу два ученика која су рођена истог датума. Слично претходном резоновању, ако постоји n хешева, вероватноћа да међу k порука долази до колизије износи $1-n!/((n-k)!n^k) = 1-(n(n-1)(n-2)\cdots(n-k+1)/n^k) = 1-(((n-1)/n)((n-2)/n)\cdots((n-k+1)/n)) = 1-((1-1/n)(1-2/n)\cdots(1-(n-k+1)/n))$. Будући да важи $(1-x) \leq e^{-x}, x \geq 0$ (што се лако показује пошто је права $1-x$ тангента функције e^{-x} у тачки $x = 0$), претходни израз се претвара у: $P_{kolizije} = 1-(e^{-1/n}e^{-1/n}\cdots e^{-(k-1)/n}) = 1-e^{-(1/n+2/n+\cdots+(k-1)/n)} = 1-e^{-(k(k-1)/2n)}$. Пошто је важан случај када је $P_{kolizije} > 1/2$, из $1/2 = 1-e^{-(k(k-1)/2n)}$ следи $\ln 2 = k(k-1)/(2n)$. Ако се замени производ $(k(k-1)$ са k^2 добија се $k = \sqrt{2n\ln 2} = 1.17\sqrt{n} > \sqrt{n}$. Дакле, ако постоји n могућих хешева, може се очекивати колизија са вероватноћом блиском $1/2$, уколико се генерише приближно $\sqrt{n}$ порука. Хеш функција се сматра *академски разбијеном* ако се колизија може пронаћи израчунавањем мањег броја вредности него приликом рођенданског напада.

2.3 Рођенданski напад у пракси

У вези са практичном применом рођенданског напада, потребно је размотрити следећи проблем: у скупу од n различитих бројева $\{1, ..., n\}$ издвајају се случајним избором два скупа од по $k \leq n$ елемената ($X = \{x_1, x_2, ..., x_k\}$ и $Y = \{y_1, y_2, ..., y_k\}$). Колика је вероватноћа да постоји пресек између наведених скупова?

За фиксирано x_1, $P(x_1 = y_1) = 1/n$, односно $P(x_1 \neq y_1) = 1 - 1/n$. Следи да важи $\forall i,\ i = 1, ..., k,\ P(x_1 \neq y_i) = (1-1/n)^k$, односно $\exists i,\ i = 1, ..., k,\ P(x_1 = y_i) = 1-(1-1/n)^k$. Другим речима, $P(\{x_1\} \cap Y = \emptyset) = (1-1/n)^k$ одакле следи $P(X \cap Y = \emptyset) = ((1-1/n)^k)^k = (1-1/n)^{k^2}$. Следи $P(X \cap Y \neq \emptyset) = 1-(1-1/n)^{k^2}$.
Комбиновањем са неједнакошћу $(1-x) < e^{-x}, x > 0$ доија се да је $1-(1-1/n)^{k^2} > 1-(e^{-1/n})^{k^2}$ односно $P(X \cap Y \neq \emptyset) > 1-e^{-k^2/n}$.
Од посебног интереса је вредност k за коју важи $P(X \cap Y \neq \emptyset) > 0.5$. Из $0.5 = 1-e^{-k^2/n}$ следи $2 = e^{k^2/n}$ тј. $\ln 2 = k^2/n$, па је тражена вредност $k = \sqrt{\ln 2 \cdot n} = 0.83\sqrt{n} < \sqrt{n}$.

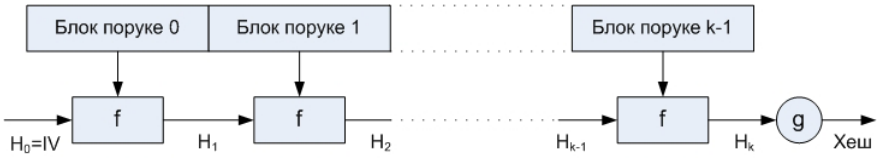
Дакле, да би два k-точлана подскупа скупа од 2^n елемената имала непразан пресек са вероватноћом преко $1/2$, потребно је да буде $k \approx 2^{n/2}$.

Претходни резултат је важан за одговор на питање како у пракси формирати две произвољне поруке тако да дају исту хеш вредност

дужине n, а да сачувају свој смисао. То се може постићи избором две произвољне поруке x и y. За сваку поруку формира се додатних $2^{n/2}$ порука тако што се на $n/2$ места у тексту по потреби дода размак (чиме се не мења смисао поруке). Тада се са вероватноћом већом од $1/2$ може очекивати да ће неки пар порука (једна настала од поруке x, а друга од поруке y) дати исту хеш вредност.

2.4 Меркле - Дамгард конструкција

Меркле и Дамгард ($Merkle - Damgård$) ([26]) формулисали су општу шему за формирање хеш алгоритама. Да би се направио хеш алгоритам, обично се полази од *сажимајуће* функције $f : B^{m+t} \to B^t$ ($compress$) за коју се претпоставља да има све три наведене особине. Функција f може се искористити за добијање хеш функције. Претпоставимо да је порука разбијена у блокове M_1, M_2, $\ldots M_k \in B^t$. Ако дужина поруке није дељива са m, онда се последњи блок допуњује до дужине m. Задат је унапред договорени блок од t битова (иницијализациони вектор IV). Тада је $H(M) = H_k$, где је $H_0 = IV$ и $H_i = F(H_{i-1}, M_i)$, $i = 1, 2, \ldots, k$.



Слика 2.1. Итеративна обрада блокова поруке хеш функцијом

Ако се уместо фиксираног IV користи тајни кључ, хеш функција се назива *аутентикациони код поруке* или MAC (енгл. message authentication code). MAC је дакле хеш функција са тајним кључем.

Прецизније, нека важе следеће ознаке:

$|x|$ - дужина поруке x у битовима,
$\|$ - надовезивање(конкатенација),
$\lceil x \rceil$ - најмањи цео број већи или једнак од x

Нека $x \leftarrow y$ означава доделу тренутне вредности промењиве y промењивој x. Тако, ако је у неком тренутку $x = 4$, онда после доделе $x \leftarrow x + 1$ промењива x има нову вредност 5.

Под функцијом сажимања (компресије) подразумевамо функцију $compress : B^{m+t} \to B^m$ где је $t > 0$. Функција сажимања користи се за конструкцију хеш функције $\bigcup_{i=m+t+1}^{\infty} B^i \to B^m$ отпорне на колизију. Нека је $t > 1$. Елемент из скупа $\bigcup_{i=m+t+1}^{\infty} B^i$ је низ састављен од нула или јединица. Нека је $|x| = n \geq m + t + 1$. Порука x може се изразити

као надовезивање $x_1 \parallel x_2 \parallel ... \parallel x_k$, где је $|x_1| = |x_2| = ... = |x_{k-1}| = t-1$ и $|x_k| = t-1-d$, где је $k = \lceil n/(t-1) \rceil$ и $0 \leq d \leq t-2$. Наредни кôд описује Меркле-Дамгардову конструкцију хеш функције одређене функцијом сажимања; конструкција слична овој је примењена на комплетну фамилију хеш функција MD .

Алгоритам Меркле-Дамгард
(за дату функцију компресије $\bigcup_{i=m+t+1}^{\infty} B^i \to B^m$ где је $t \geq 1$)

$n \leftarrow |x|$
$k \leftarrow \lceil n/(t-1) \rceil$
$d \leftarrow k(t-1) - n$
for $i \leftarrow 1$ **to** $k-1$
    $y_i \leftarrow x_i$
$y_k \leftarrow x_k \parallel 0^d$
$y_{k+1} \leftarrow (t-1)$-битна бинарна репрезентација d
$z_1 \leftarrow 0^{m+1} \parallel y_1$
$g_1 \leftarrow compress(z_1)$
for $i \leftarrow 1$ **to** k
    $z_{i+1} \leftarrow g_i \parallel 1 \parallel y_{i+1}$
    $g_{i+1} \leftarrow compress(z_{i+1})$
$h(x) \leftarrow g_{k+1}$
return $h(x)$

Овде је $y(x) = y_1 \parallel y_2 \parallel ... \parallel y_{k+1}$. Блок y_k добијен је из x_k додавањем d нула с десне стране, па је и он дужине $t-1$. Такође, на y_{k+1} надовезан је потребан број нула са десне стране, па је и он дужине $t-1$. Значи да важи $|y_i| = t-1$ за $1 \leq i \leq k+1$. Важно је напоменути да је пресликавање $x \to y(x)$ инјекција, јер је то неопходно да би итеративна хеш функција била отпорна на колизију. Низ y_k формиран је од низа x_k као низ блокова исте дужине. Низ z_k добија се од чланова низа y_k и на њега се примењује функција сажимања. При томе важи следећа теорема:

Теорема 2.4.1. *Ако је функција сажимања* $compress : B^{m+t} \to B^m$ *где је* $t > 1$ *отпорна на колизију, онда је и функција* $\bigcup_{i=m+t+1}^{\infty} B^i \to B^m$ *добијена претходним алгоритмом такође отпорна на колизију.*

2.5 Хеш функција MD4

Алгоритам MD4 [9] је први алгоритам фамилије MD4 (MD4, MD5, SHA-0, SHA-1, SHA-2, RIPEMD, HAVAL). Иако се MD4 више не користи, и даље је предмет анализе јер се тиме усавршавају принципи криптоанализе других хеш функција. Обрнуто, принципи примењени на разбијању рецимо алгоритма SHA, накнадно су тестирани на алгоритму MD4. "Диференцијални напад" (енгл. Differential attack) који

је коришћен при разбијању алгоритма MD4 је најефикаснији алат за тражење колизија код хеш алгоритама. У тачкама 3.4 и 3.5 биће детаљно приказани диференцијални напади на MD4.

2.5.1 Опис алгоритма

Алгоритам MD4 заснива се на сажимајућој функцији $f : B^{128} \times B^{512} \to B^{128}$. Нека је $M \in B^{512}$ и $M = (m_0, m_1, ..., m_{15})$ где је $m_i \in B^{16}$, $i = 0, 1, ..., 15$.

Нека се блок од 4 бита назива нибл, блок од осам битова бајт и блок од 32 бита реч. У току израчунавања функције f, непрестано се ажурира стање четири 32-битна регистра a, b, c, d. На почетку важи $a = a_0 = 0x67452301$, $b = b_0 = 0xEFCDAB89$, $c = c_0 = 0x98BADCFE$, $d = d_0 = 0x1025476$. Ознака $0x$ означава да иза ње следи хексадецимални број. При томе се ниблови $0000, 0001, \ldots, 1111$ представљају цифрама $0, 1, \ldots, F$.

За $e \in B^{32}$, $1 \leq n \leq 31$, нека $e \ll n$ означава цикличку ротацију e за n бита улево. На пример, ако је $e = 01000000000000000111111111111111$, онда је после ротације $e \ll 3$, $e = 00000000000001111111111111111010$.

Нека $+$ означава сабирање бројева n из скупа $\{n \mid 0 \leq n \leq 2^{32} - 1\}$ по модулу 2^{32}. Сваком таквом броју једнозначно одговара елемент из скупа B^{32}.

Нека су F, G и H три функције $B^{32^3} \to B^{32}$ дефинисане изразима:

$$F(X, Y, Z) = XY \vee \bar{X}Z = (X \wedge Y) \vee (\neg X \wedge Z)$$

$$G(X, Y, Z) = XY \vee XZ \vee YZ = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$$

$$H(X, Y, Z) = X \oplus Y \oplus Z$$

Овде је $\bar{1} = 0$, $\bar{0} = 1$ ((или $\neg$) негација); $\vee$ је дисјункција (или), а множење (односно $\wedge$) је конјункција (и). Операције над речима изводе се истовремено са свим њиховим одговарајућим битовима. На пример, нека се F примени на три бајта (уместо на три речи). Нека је $X = 00001111$, $Y = 00110011$, $Z = 01010101$, онда је $F(X, Y, Z) = 01010011$.

X	Y	Z	XY	$\bar{X}Z$	$XY \vee \bar{X}Z$
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	0	0
0	1	1	0	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	1	0	1
1	1	1	1	0	1

Примећује се да за 4 од 8 могућих једнобитних улаза, F има вредност 0, а за остала 4 улаза има вредност 1. Ово важи и за функције G и H.
Функција F такође се назива IF, пошто за $Y \neq Z$ излаз функције зависи од промењљиве X.
Функција G такође се назива MAJ пошто већинско стање променљивих даје излаз функције.
Функција H такође се назива XOR.

Израчунавање сажимајуће функције f описује се следећим корацима:

Алгоритам $f(a, b, c, d, M)$

$(a_0, b_0, c_0, d_0) \leftarrow (a, b, c, d)$
  рунда 1 {израчунавају се $a_i, d_i, c_i, b_i,\ i = 1, 2, 3, 4\}$
  рунда 2 {израчунавају се $a_i, d_i, c_i, b_i,\ i = 5, 6, 7, 8\}$
  рунда 3 {израчунавају се $a_i, d_i, c_i, b_i,\ i = 9, 10, 11, 12\}$
return $(a_{12} + a_0, b_{12} + b_0, c_{12} + c_0, d_{12} + d_0)$

Алгоритам MD4 најпре за задати улаз x формира низ $m_0 m_1 ... m_{N-1}$ од $N = 0 \mod 16$ речи (односно $N = \lceil (|x|+65)/512 \rceil$ 512-битних блокова) на следећи начин:

1. $k \leftarrow (447 - |x|) \mod 512,\ 0 \leq k < 512$
2. нека је b 64-битна бинарна репрезентација дужине $|x|$
3. $x \leftarrow x \parallel 1 \parallel 0^k \parallel b$

Другим речима, на поруку се дописује јединица и потребан број нула тако да њена дужина буде облика $512 \cdot l + 448$. Поступак добијања 128-битног сажетка описује се следећим кôдом:

Алгоритам MD4 (уопштено)

продужавање поруке x и формирање N блокова
for $i \leftarrow 0$ **to** $N/16 - 1$
  $M \leftarrow (x_{16i},\ x_{16i+1}, ...,\ x_{16i+15})$
  $(a, b, c, d) \leftarrow f(a, b, c, d, M)$
return (a, b, c, d)

Речи a, b, c, d су регистри или ланчане променљиве (енгл. chaining variables). На почетку, они су иницијализовани фиксираним константним вредностима. Пошто послуже да се обради један 512-битни блок, после сабирања са својим почетним регистрима a_0, b_0, c_0, d_0 они чине улаз за обраду следећег 512-битног блока. После обраде последњег 512-битног блока, конкатенација $(a + a_0) \parallel (b + b_0) \parallel (c + c_0) \parallel (d + d_0)$ је израчуната хеш вредност.

Алгоритам MD4 (детально)

$a \leftarrow 0x67452301$
$b \leftarrow 0xEFCDAB89$
$c \leftarrow 0x98BADCFE$
$d \leftarrow 0x10325476$
for $i \leftarrow 0$ **to** $N/16 - 1$
  for $j \leftarrow 0$ **to** 15
    $m_j \leftarrow M_{16i+j}$
  $aa \leftarrow a$
  $bb \leftarrow b$
  $cc \leftarrow c$
  $dd \leftarrow d$
  Рунда 1 {израчунавају се $a_i, d_i, c_i, b_i,\ i = 1, 2, 3, 4\}$
  Рунда 2 {израчунавају се $a_i, d_i, c_i, b_i,\ i = 5, 6, 7, 8\}$
  Рунда 3 {израчунавају се $a_i, d_i, c_i, b_i,\ i = 9, 10, 11, 12\}$
  $a \leftarrow a + aa$
  $b \leftarrow b + bb$
  $c \leftarrow c + cc$
  $d \leftarrow d + dd$
return : $H \leftarrow a \parallel b \parallel c \parallel d$

Рунда 1:

$a_1 \leftarrow (a_0 + F(b_0, c_0, d_0) + m_0) \lll 3$
$d_1 \leftarrow (d_0 + F(a_1, b_0, c_0) + m_1) \lll 7$
$c_1 \leftarrow (c_0 + F(d_1, a_1, b_0) + m_2) \lll 11$
$b_1 \leftarrow (b_0 + F(c_1, d_1, a_1) + m_3) \lll 19$
$a_2 \leftarrow (a_1 + F(b_1, c_1, d_1) + m_4) \lll 3$
$d_2 \leftarrow (d_1 + F(a_2, b_1, c_1) + m_5) \lll 7$
$c_2 \leftarrow (c_1 + F(d_2, a_2, b_1) + m_6) \lll 11$
$b_2 \leftarrow (b_1 + F(c_2, d_2, a_2) + m_7) \lll 19$
$a_3 \leftarrow (a_2 + F(b_2, c_2, d_2) + m_8) \lll 3$
$d_3 \leftarrow (d_2 + F(a_3, b_2, c_2) + m_9) \lll 7$
$c_3 \leftarrow (c_2 + F(d_3, a_3, b_2) + m_{10}) \lll 11$
$b_3 \leftarrow (b_2 + F(c_3, d_3, a_3) + m_{11}) \lll 19$
$a_4 \leftarrow (a_3 + F(b_3, c_3, d_3) + m_{12}) \lll 3$
$d_4 \leftarrow (d_3 + F(a_4, b_3, c_3) + m_{13}) \lll 7$
$c_4 \leftarrow (c_3 + F(d_4, a_4, b_3) + m_{14}) \lll 11$
$b_4 \leftarrow (b_3 + F(c_4, d_4, a_4) + m_{15}) \lll 19$

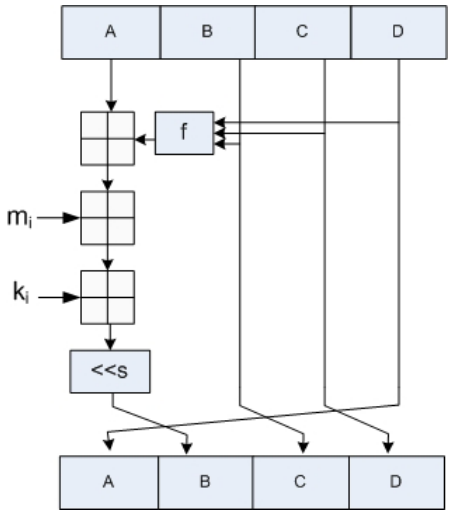
Рунда 2:

$a_5 \leftarrow (a_4 + G(b_4, c_4, d_4) + m_0 + 0x5A827999) \lll 3$
$d_5 \leftarrow (d_4 + G(a_5, b_4, c_4) + m_4 + 0x5A827999) \lll 5$
$c_5 \leftarrow (c_4 + G(d_5, a_5, b_4) + m_8 + 0x5A827999) \lll 9$

$b_5 \leftarrow (b_4 + G(c_5, d_5, a_5) + m_{12} + 0x5A827999) \lll 13$
$a_6 \leftarrow (a_5 + G(b_5, c_5, d_5) + m_1 + 0x5A827999) \lll 3$
$d_6 \leftarrow (d_5 + G(a_6, b_5, c_5) + m_5 + 0x5A827999) \lll 5$
$c_6 \leftarrow (c_5 + G(d_6, a_6, b_5) + m_9 + 0x5A827999) \lll 9$
$b_6 \leftarrow (b_5 + G(c_6, d_6, a_6) + m_{13} + 0x5A827999) \lll 13$
$a_7 \leftarrow (a_6 + G(b_6, c_6, d_6) + m_2 + 0x5A827999) \lll 3$
$d_7 \leftarrow (d_6 + G(a_7, b_6, c_6) + m_6 + 0x5A827999) \lll 5$
$c_7 \leftarrow (c_6 + G(d_7, a_7, b_6) + m_{10} + 0x5A827999) \lll 9$
$b_7 \leftarrow (b_6 + G(c_7, d_7, a_7) + m_{14} + 0x5A827999) \lll 13$
$a_8 \leftarrow (a_7 + G(b_7, c_7, d_7) + m_3 + 0x5A827999) \lll 3$
$d_8 \leftarrow (d_7 + G(a_8, b_7, c_7) + m_7 + 0x5A827999) \lll 5$
$c_8 \leftarrow (c_7 + G(d_8, a_8, b_7) + m_{11} + 0x5A827999) \lll 9$
$b_8 \leftarrow (b_7 + G(c_8, d_8, a_8) + m_{15} + 0x5A827999) \lll 13$

Рунда 3:

$a_9 \leftarrow (a_8 + H(b_8, c_8, d_8) + m_0 + 0x6ED9EBA1) \lll 3$
$d_9 \leftarrow (d_8 + H(a_9, b_8, c_8) + m_8 + 0x6ED9EBA1) \lll 9$
$c_9 \leftarrow (c_8 + H(d_9, a_9, b_8) + m_4 + 0x6ED9EBA1) \lll 11$
$b_9 \leftarrow (b_8 + H(c_9, d_9, a_9) + m_{12} + 0x6ED9EBA1) \lll 15$
$a_{10} \leftarrow (a_9 + H(b_9, c_9, d_9) + m_2 + 0x6ED9EBA1) \lll 3$
$d_{10} \leftarrow (d_9 + H(a_{10}, b_9, c_9) + m_{10} + 0x6ED9EBA1) \lll 9$
$c_{10} \leftarrow (c_9 + H(d_{10}, a_{10}, b_9) + m_6 + 0x6ED9EBA1) \lll 11$
$b_{10} \leftarrow (b_9 + H(c_{10}, d_{10}, a_{10}) + m_{14} + 0x6ED9EBA1) \lll 15$
$a_{11} \leftarrow (a_{10} + H(b_{10}, c_{10}, d_{10}) + m_1 + 0x6ED9EBA1) \lll 3$
$d_{11} \leftarrow (d_{10} + H(a_{11}, b_{10}, c_{10}) + m_9 + 0x6ED9EBA1) \lll 9$
$c_{11} \leftarrow (c_{10} + H(d_{11}, a_{11}, b_{10}) + m_5 + 0x6ED9EBA1) \lll 11$
$b_{11} \leftarrow (b_{10} + H(c_{11}, d_{11}, a_{11}) + m_{13} + 0x6ED9EBA1) \lll 15$
$a_{12} \leftarrow (a_{11} + H(b_{11}, c_{11}, d_{11}) + m_3 + 0x6ED9EBA1) \lll 3$
$d_{12} \leftarrow (d_{11} + H(a_{12}, b_{11}, c_{11}) + m_{11} + 0x6ED9EBA1) \lll 9$
$c_{12} \leftarrow (c_{11} + H(d_{12}, a_{12}, b_{11}) + m_7 + 0x6ED9EBA1) \lll 11$
$b_{12} \leftarrow (b_{11} + H(c_{12}, d_{12}, a_{12}) + m_{15} + 0x6ED9EBA1) \lll 15$



Слика 2.2. Понашање регистара у два узастопна корака

Сваки 512-битни блок поруке пролази кроз приказане три рунде, при чему су улазни регистри a_0, b_0, c_0, d_0 иницијализовани почетним константама. Надовезани збирови улазних и излазних регистара a_{12}, b_{12}, c_{12}, d_{12} дају хеш за тај блок поруке.
Корисно је приметити да се у опису алгоритма четири низа регистара, могу заменити једним који пролази кроз 48 корака:

$$a_i \leftarrow (a_{i-4} + f_i(a_{i-1}, a_{i-2}, a_{i-3}) + m_{w_i} + c_i) \ll s_i \quad 0 \leq i \leq 47 \tag{2.1}$$

У том случају резултат примене функције f постаје:

$$(a, b, c, d) \leftarrow (a_{-1}, a_{-2}, a_{-3}, a_{-4}) + (a_{47}, a_{46}, a_{45}, a_{44})$$

i	f_i	c_i
0...15	$IF(x, y, z)$	$0x00000000$
16...31	$MAJ(x, y, z)$	$0x5a827999$
32...47	$XOR(x, y, z)$	$0x6ed9eba1$

Табела 2.3. Преглед примењених функција и константи по рундама

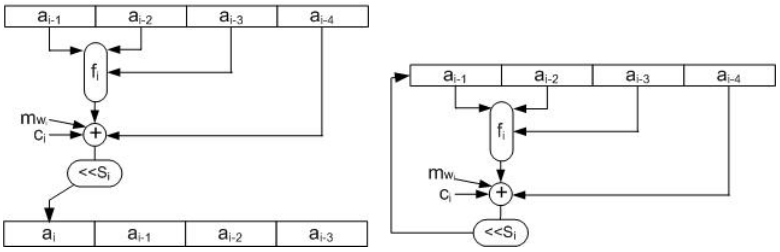
i	w_i
0...15	$0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15$
16...31	$0, 4, 8, 12, 1, 5, 9, 13, 2, 6, 10, 14, 3, 7, 11, 15$
32...47	$0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15$

Табела 2.4. Редослед учешћа улазних речи у корацима три рунде

i	s_i
0...15	$3, 7, 11, 19, 3, 7, 11, 19, 3, 7, 11, 19, 3, 7, 11, 19$
16...31	$3, 5, 9, 13, 3, 5, 9, 13, 3, 5, 9, 13, 3, 5, 9, 13$
32...47	$3, 9, 11, 15, 3, 9, 11, 15, 3, 9, 11, 15, 3, 9, 11, 15$

Табела 2.5. Редослед цикличких помераја по рундама

Наредна слика илуструје један од 48 корака који се извршавају приликом израчунавања вредности функције сажимања f једним тактом померачког регистра:



Слика 2.1. Померачки регистар у једном кораку алгоритма MD4

Напомена 2.5.1. Порука се увек допуњује. Она се најпре допуњује тако да за њену дужину важи дужина$-64 \equiv 448 \pmod{512}$. Допуњавање се чак и ако је дужина поруке $448 \pmod{512}$: тада се додаје једна јединица и 511 нула. Нека је b дужина поруке пре допуњавања. Бинарна представа броја b дописује се на крај поруке, после чега дужина поруке постаје умножак од 512. На пример, нека је дужина оригиналне поруке 1700 бита. Пошто је $1770 \equiv 164 \pmod{512}$ и $448 - 164 = 284$, додаје се једна јединица и 283 нуле. Сада је дужина поруке 1984 бита, $1984 \equiv 448 \equiv -64 \pmod{512}$. Даље, пошто је $1700 = (11010100100)_2$, поруци се дописује

00000000000000000000000000000000000000000000000000000011010100100

тако да је сада дужина поруке $1984 + 64 = 2048 = 4 \cdot 512$.

Напомена 2.5.2. MD4 користи "little endian" архитектуру. То значи да ако је $a_1a_2a_3a_4$ реч састављена од четри бајта, сваки a_i можемо третирати као цео број (бинарно представљен) у распону 0,...,255 при чему реч представља број $a_4 2^{24} + a_3 2^{16} + a_2 2^8 + a_1$. Напоменимо да избор архитектуре представљања редоследа бајтова не утиче на сам алгоритам будући да важи следеће: претпоставимо да поруке $M = m_0m_1...m_{63}$ и $M' = m'_0m'_1...m'_{63}$ (m_i је бајт) чине колизију у MD4 алгоритму уз коришћење "big endian" архитектуре (реч $a_1a_2a_3a_4$ представља број $a_1 2^{24} + a_2 2^{16} + a_3 2^8 + a_4$). Тада поруке $m_3m_2m_1m_0m_7m_6m_5m_4...$ $m_{63}m_{62}m_{61}m_{60}$ и $m'_3m'_2m'_1m_0m'_7m_6m'_5m'_4...m'_{63}m'_{62}m'_{61}m'_{60}$ чине колизију уз коришћење "little endian" архитектуре.

2.5.2 Преглед напада

Радови Добертина (Dobbertin) [31] инспирисали су Кинескињу Ванг (Wang) за разбијање практично свих алгоритама из фамилије MD4 [1-4]. У [1], приказан је поступак добијања колизије у MD4 алгоритму

са вероватноћом већом од 2^{-6}. Каснијом анализом Наито (Naito) [8] показује да се поменута вероватноћа може повећати на скоро сигуран догађај. Са вероватноћом 1, Ју (Yu) [5] добија колизију, па се њен резултат може сматрати нападом на јаку отпорност изабране "слабе" поруке; такође, двоструко смањује број потребних услова у односу на рад [1]. Потом, Шлафер (Schläffer) даје алгоритам за аутоматско тражење колизије [27,28]. Објединивши добијене путање ([1] и [5]), Ју и Ванг дају пример четвороструке колизије, уз генерални шаблон за добијање вишеструке колизије у алгоритму MD4 [30]. Анализирајући понашање интерне колизије у трећој рунди, Сасаки (Sasaki) добија најбржу колизију [7], уз почетну разлику која већину услова групише у прву рунду.

2.6 Хеш функција SHA − 0

Алгоритам SHA-0 [18] створен је у NSA (енгл. National Security Agency) по узору на хеш алгоритме MD4 и MD5, које је дизајнирао Роналд Ривест. Услед показане слабости, учињена је минорна промена у дизајну SHA-0 алгоритма, чиме је створен побољшани наследник - SHA-1 [19]. Временом су се појавиле и друге варијанте SHA алгоритма, али је највећу употребу задржао SHA-1. Сваку поруку, ма колика она била, SHA-0 алгоритам претвара у 160 битова. То је еквивалентно надовезивању пет 32-битних речи. Идеја SHA-0, као и сваког хеш алгоритма, јесте да било која промена на оригиналној поруци узрокује промену и на сажетку, тј. хешу те поруке. SHA-0 функционише тако што оригиналну поруку дели у блокове од по 512 битова. Уколико је последњи блок краћи, додаје му се јединица (бит), потребан број нула битова (енгл. padding) и последњих 64 бита који бинарно репрезентују величину полазне поруке.

2.6.1 Опис алгоритма

Алгоритам SHA-0 , слично алгоритму MD4 , састоји се од три корака:

1. Иницијализација пет регистара A, B, C, D, E константама:
   - $A = 0x67452301$
   - $B = 0xefcdab89$
   - $C = 0x98badcfe$
   - $D = 0x10325476$
   - $E = 0xc3d2e1f0$

2. За сваки 512-битни блок поруке копирати регистре A, B, C, D, E у регистре AA, BB, CC, DD, EE на које се примењује функција сажимања која редом производи нове регистре AA', BB', CC', DD', EE'. На почетне регистре A, B, C, D, E додају се (по модулу 2^{32}) нове вредности AA', BB', CC', DD', EE'

3. Надовезивањем регистара $A \,||\, B \,||\, C \,||\, D \,||\, E$ добија се 160-битни излаз за дати блок поруке.

У циљу описа функције сажимања, улазни 512-тобитни блок поруке дели се на 16 32-битних речи $(W_0, ..., W_{15})$. Ових 16 проширује се на 80 речи на основу рекурентног израза:

$$W_i = W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} \ \forall i,\ 16 \leq i \leq 79$$

Ових 80 речи користе се да у наредних 80 рунди промене регистре $A_i, B_i, C_i, D_i, E_i,\ 1 \leq i \leq 80$. Нека је иницијално стање регистара означено са $\langle A_0, B_0, C_0, D_0, E_0 \rangle$. Нека ROL_i означава ротацију у лево за i места, а ADD сабирање по модулу 2^{32}:

$$ADD(U, V, W, X, Y) = U + V + W + X + Y \ mod\ 2^{32}$$

Тада је 80 рунди функције сажимања дефинисано са:

$for\ i = 0\ to\ 79$
$A_{i+1} = ADD(W_i, ROL_5(A_i), f_i(B_i, C_i, D_i), E_i, K_i)$
$B_{i+1} = A_i$
$C_{i+1} = ROL_{30}(B_i)$
$D_{i+1} = C_i$
$E_{i+1} = D_i$

При томе се, зависно од редног броја рунде, користе следеће функције f_i и константе K_i:

i-та рунда	Функција f_i		Константа K_i
0 - 19	IF	$(X \wedge Y) \vee (\neg X \wedge Z)$	$0x5a87999$
20 - 39	XOR	$X \oplus Y \oplus Z$	$0x6ed9eba1$
40 - 59	MAJ	$(X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$	$0x8f1bbcdc$
60 - 79	XOR	$X \oplus Y \oplus Z$	$0xca62c1d6$

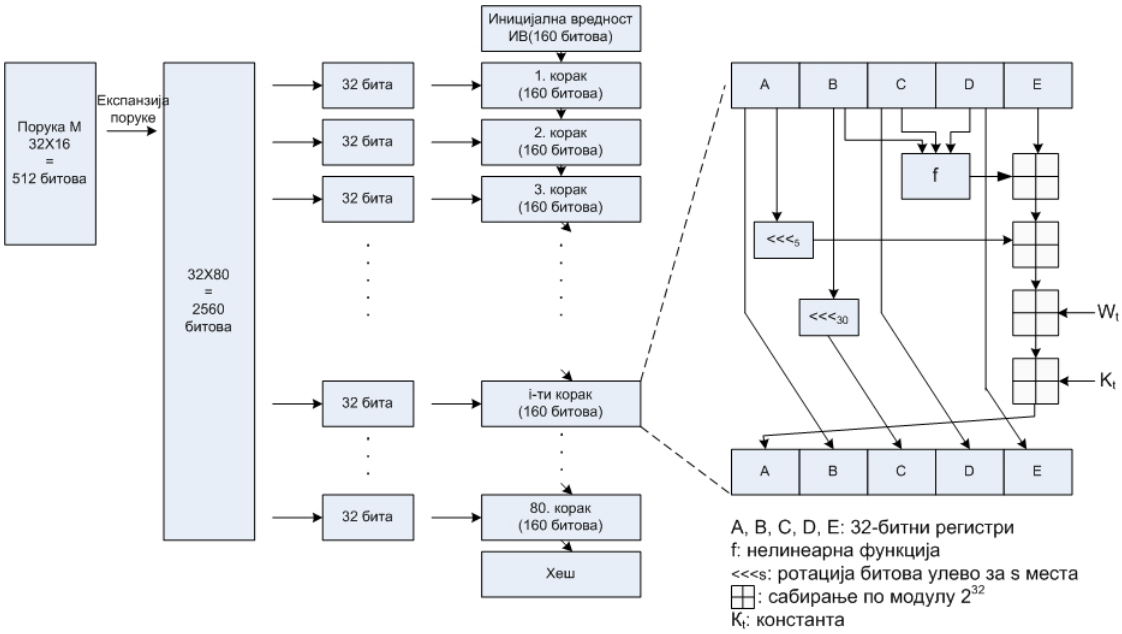
Табела 2.6. Функције f_i и константе K_i кроз рунде

Излаз функције сажимања је петорка регистара $(A_{80}, B_{80}, C_{80}, D_{80}, E_{80})$. Под колизијом се сматрају два различита улаза $(W_0, .., W_{15})$ и $(W'_0, ..., W'_{15})$ који за исте почетне вредности $(A_0, B_0, C_0, D_0, E_0)$ дају исти излаз.

Једина разлика алгоритма SHA-1 у односу на SHA-0 је постојање додатне ротације у процесу експанзије поруке:

$$W_i = ROL_1(W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16}) \ \forall i,\ 16 \leq i \leq 79$$

Слика 2.1 илуструје алгоритам израчунавања вредности сажимајуће функције.



Слика 2.1. Структура алгоритма SHA-{0,1}

2.6.2 Преглед напада

Шобон (Chauband) и Жу (Joux) отварају серију напада на SHA-0, налажењем колизије за поједностављену верзију алгоритма. Они такође описују начин проналажења колизије на правој верзији са тежином напада 2^{61} [11]. Бихам (Biham) и Чен (Chen) увођењем технике неутралних битова спуштају тежину напада на 2^{56} [12]. Такође показују да се "скоро-колизија" може користити у циљу налажења колизије. Ванг налази колизију са тежином напада 2^{39} [3], а Наито унапређује њен напад на 2^{36} [21]. Мануел (Manuel) и Пејрин (Peyrin) налазе колизију са тежином напада 2^{33} [32].

Глава 3

Проналажење колизија код MD4

У овој глави описан је диференцијални напад на алгоритам MD4 . У одељцима 3.1 и 3.2, уводе се ознаке везане за анализу диференцијалног напада на MD4. У одељку 3.3, уведен је појам интерне колизије. У одељку 3.4 детаљно се описује напад који је пронашла Ванг [1]. У складу с тим, доказују се поједине теореме чији се утицај протеже кроз цео рад. Потом је у одељку 3.5 детаљно приказан напад који је Ју [5].У одељку 3.6, у циљу аутоматизације поступка [5], предложен је начин за аутоматско проналажење колизије у алгоритму MD4, под условом да се две поруке разликују у једном биту. На крају, у одељку 3.7, објашњен је појам вишеструке колизије.

3.1 Означене разлике

У наставку се користе следеће ознаке:

1. Нека су $M = (m_0, m_1, ..., m_{15})$ и $M' = (m'_0, m'_1, ..., m'_{15})$ две 512-битне поруке. Елементи ова два вектора су 32-битне речи m_i.

2. Нека 32-битни регистри a_i, d_i, c_i, b_i представљају излазе из корака $4i-3, 4i-2, 4i-1, 4i$, за $1 \leq i \leq 12$ приликом примене алгоритма на поруку M.

3. Нека 32-битни регистри a'_i, d'_i, c'_i, b'_i представљају излазе из корака $4i-3, 4i-2, 4i-1, 4i$, за $1 \leq i \leq 12$ приликом примене алгоритма на поруку M'.

4. Нека је $\Delta m_i = m'_i - m_i$.

5. Нека $a_{i,j}, d_{i,j}, c_{i,j}, b_{i,j}$ представљају j-ти бит респективно вредности a_i, d_i, c_i, b_i, где је први бит - најмање значајни бит, а 32-ги бит - најзначајнији бит.

6. Нека су $x_i[j]$ и $x_i[-j]$ (x може бити један од регистара a, b, c, d) резултујуће вредности речи x_i када јој се промени j-ти бит. При томе, $x_i[j]$ значи да се вредност j-тог бита променила са нуле на јединицу, док $x_i[-j]$ значи да се вредност j-тог бита променила са јединице на нулу.

7. Нека се $x_i[\pm j_1, \pm j_2, ..., \pm j_l]$ добија тако што су се промениле вредности битова са назначеним индексима речи. Знак + значи да се бит променио са нуле на јединицу, а знак − да се бит променио са јединице на нулу.

8. Уз претходне ознаке, нека је x_i' нова, промењена вредност регистра x_i. тј. $x_i' = x_i[\pm j_1, \pm j_2, ..., \pm j_l]$ или краће $x_i[\pm j_1, \pm j_2, ..., \pm j_l]$. У случају да нема промене, подразумеваће се да важи ознака $x_i' = x_i[\ ] = x_i$ и у даљем тексту неће бити навођена.

Дефиниција 3.1.1. *Под појмом колизионе разлике (енгл. collision differential) подразумева се почетна разлика у порукама M и M' у циљу добијања колизије.*

Дефиниција 3.1.2. *Под појмом диференцијалног пута (енгл. differential path) подразумевају се све настале разлике у регистрима између две поруке које се јављају кретањем кроз хеш алгоритам, а проузроковане су почетном колизионом разликом.*

Дакле, колизиона разлика диктира диференцијални пут. Важно је напоменути да диференцијални пут такође зависи и од тренутних вредности регистара почетне поруке. Одговарајућом модификацијом почетне поруке постижу се одговарајућа стања регистара, чиме се диференцијални пут усмерава да две поруке са почетном колизионом разликом дају исту хеш вредност (производе колизију).
Напад се састоји од три корака:

1. Проналажење колизионе разлике у којој поруке M и M' производе колизију.

2. Извођење скупа довољних услова који обезбеђују диференцијални пут.

3. Модификација случајно изабране поруке M тако да промењена порука задовољава углавном све услове који обезбеђују диференцијални пут.

Следе формалне дефиниције ових појмова.([6,27])

Дефиниција 3.1.3. *Означена разлика (енгл. Signed Difference)* Δx *две 32-битне речи је вектор*

$$\Delta x = \Delta(x', x) = (\delta x_{32}, ..., \delta x_1), \text{ где је } \delta x_j = x'_j - x_j \in \{-1, 0, 1\}, 1 \leq j \leq 32$$

Дефиниција 3.1.4. *Хемингова тежина (енгл. Hamming weight) означене разлике* $w_H(\Delta x)$ *је број не нултих елемената* Δx*:*

$$w_H(\Delta x) = \sum_{j=1}^{32} |\delta x_j|$$

Дефиниција 3.1.5. *Под Хемингговим растојањем (енгл. Hamming distance) речи* x *и* x' *подразумева се Хемингова тежина њихове означене разлике.*

Означена разлика је 32-битни вектор коме су већина елемената нуле, па се може краће записати издвајањем само места са промењеним битовима.

Дефиниција 3.1.6. *Означена разлика* Δx *са Хемингговом тежином* $w = w_H(\Delta x) \geq 0$ *може се записати на следећи начин:*

$$\Delta x = \Delta[d_1, d_2, ..., d_w] \text{ где је}$$

$$d_i = \begin{cases} -j & , \ \delta x_j = -1 \\ j & , \ \delta x_j = 1 \end{cases}$$

Корисно је приметити да модуларна разлика не одређује једнозначно означену разлику јер је на пример:

$$2^{i-1} = \Delta[i] = \Delta[i+1, -i]$$

али обрнуто важи:

$$\Delta x = \Delta(x', x) = \Delta[d_1, d_2, ..., d_w] \Rightarrow x' - x = \sum_{i=1}^{w} sign(d_i) \cdot 2^{|d_i|-1} \ mod \ 2^{32}$$

На основу означене разлике $\Delta x = \Delta(x', x) = (\delta x_{32}, ..., \delta x_1) = \Delta[d_1, d_2, ..., d_w]$ могу се одредити вредности одговарајућих битова у регистрима x' и x:

$$x_{|d_i|} = \begin{cases} 0 & , \ sign(d_i) = 1 \\ 1 & , \ sign(d_i) = -1 \end{cases} \qquad x'_{|d_i|} = \begin{cases} 1 & , \ sign(d_i) = 1 \\ 0 & , \ sign(d_i) = -1 \end{cases}$$

Јасно је да се не може десити $\Delta x_j = 1$ ако је $x_j = 1$, или $\Delta x_j = -1$ ако је $x_j = 0$.

Лема 3.1.1. *Нека су почетни услови у регистру* x *означене разлике* Δx*, са Хемингговом тежином* $w = w_H(\Delta x)$ *међусобно независни. Тада је вероватноћа:*

$$P(\Delta x) = P(\Delta[d_1, d_2, ..., d_w] = 2^{-w}.$$

Пример 3.1.1. За фиксиране вредности битова $x_{27} = 1, x_{15} = 0, x_3 = 1$, важи:

$$\Delta x = -2^{26} + 2^{14} - 2^2 = \Delta[-27, 15, -3] =$$

$$(0,0,0,0,0,-1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,-1,0,0)$$

ако је задовољено $x'_{27} = 0, x'_{15} = 1, x'_3 = 0$, па је $P(\Delta[-27, 15, -3]) = 2^{-3}$.

3.2 Операције са означеним разликама

У операције са означеним разликама спадају сабирање, ротација и промена чланова. Основна намена означених разлика јесте формирање диференцијалног пута.

Нека је s_j j-ти бит суме, а c_j одговарајући бит преноса збира $\Delta x + \Delta y$. Тада је

$$\Delta z = \Delta x + \Delta y = (\delta x_{32}, ..., \delta x_1) + (\delta y_{32}, ..., \delta y_1) = (s_{32}, ..., s_1)$$

где је $c_0 = 0$ и

$$s_j = \begin{cases} -1 & , \ \delta x_j + \delta y_j + c_j \in \{-3, -1\} \\ 0 & , \ \delta x_j + \delta y_j + c_j \in \{-2, 0, 2\} \\ 1 & , \ \delta x_j + \delta y_j + c_j \in \{1, 3\} \end{cases}$$

$$c_{j+1} = \begin{cases} -1 & , \ \delta x_j + \delta y_j + c_j \in \{-3, -2\} \\ 0 & , \ \delta x_j + \delta y_j + c_j \in \{-1, 0, 1\} \\ 1 & , \ \delta x_j + \delta y_j + c_j \in \{2, 3\} \end{cases}$$

Специјално, збир две означене разлике $\Delta x = \Delta[dx_1]$ и $\Delta y = \Delta[dy_1]$ са Хеминговом тежином $w_H(\Delta x) = w_H(\Delta y) = 1$ може се краће записати на следећи начин:

$$\Delta[dx_1] + \Delta[dy_1] = \begin{cases} \Delta[\,] & , \ dx_1 = -dy_1 \\ \Delta[sign(dx_1)(|dx_1| + 1)] & , \ dx_1 = dy_1 \\ \Delta[dx_1, dy_1] & , \text{ иначе} \end{cases}$$

Потребно је нагласити да као последица сабирања по модулу 2^{32} следи

$$\Delta[32] + \Delta[32] = \Delta[-32] + \Delta[-32] = \Delta[-32] + \Delta[32] = \Delta[\,]$$

Пример 3.2.1. Следећи пример илуструје сабирање две означене разлике са Хеминговом тежином $w = 1$:

$$\Delta[24] + \Delta[-24] = \Delta[\,]$$
$$\Delta[24] + \Delta[24] = \Delta[25]$$
$$\Delta[-24] + \Delta[-24] = \Delta[-25]$$
$$\Delta[-24] + \Delta[25] = \Delta[25, -24]$$

Пример 3.2.2. Следећи пример илуструје сабирање две означене разлике са произвољном Хеминговом тежином:

$$\Delta[32,-27,15,-1]+\Delta[32,27,16,15,4]=\Delta[17,4,-1]$$

Ротација 32-битне означене разлике $\Delta x=(\delta x_{32},...,\delta x_1)$ за s места односно битова, обавља се тако што се на сваки бит δx_j примењује:

$$\delta x_j \ll s=\begin{cases} \delta x_{(j+s)\ mod\ 32} & ,\ j+s\neq 32 \\ \delta x_{32} & ,\ j+s=32 \end{cases}$$

односно ротација $\Delta x=\Delta[d_1,d_2,...,d_w]$ добија се ротацијом сваког члана:

$$\Delta[\tilde{d}_i]\ll s=\Delta[d_i],\quad d_i=\begin{cases} sign(\tilde{d}_i)((|\tilde{d}_i|+s)\ mod\ 32) & ,\ |\tilde{d}_i|+s\neq 32 \\ sign(\tilde{d}_i)\cdot 32 & ,\ |\tilde{d}_i|+s=32 \end{cases}$$

Пример 3.2.3. Ако се означена разлика $\Delta x=\Delta[32,-27,-3]$ ротира за 9 битова добија се:

$$\begin{aligned} \Delta x\ll s &= \Delta[32,-27,-3]\ll 9 \\ &= \Delta[(32+9\ mod\ 32),-(27+9\ mod\ 32),-(3+9\ mod\ 32)] \\ &= \Delta[9,-4,-12]=\Delta[-12,9,-4] \end{aligned}$$

Промена неког елемента d_i означене разлике, услед преноса може утицати на следећи елемент:

$$\Delta[d_1,...,d_i,...,d_w]=\begin{cases} \Delta[d_1,...,-d_i,...,d_w] & ,\ |d_i|=32 \\ \Delta[d_1,...,-d_i,...,d_w]+\Delta[d_i]\ll 1 & ,\ |d_i|\neq 32 \end{cases}$$

Да би се добила сва могућа проширења настала променом бита d_i, потребно је рекурзивно применити претходну формулу на све постојеће, као и на новопромењене елементе у означеној разлици. Број елемената који су промењени под утицајем преноса назива се *дубина проширења*.

Пример 3.2.4. Нека је дата означена разлика $\Delta x=\Delta[-11,9]$ и нека је дубина проширења једнака 2. Елемент који се проширује означава се са $\overleftarrow{d_i}$.

$$\begin{array}{rll} \Delta x\to\Delta[-11,\overleftarrow{9}] & \to\Delta[-11,\overleftarrow{10},-9] & \to\Delta[-10,-9] \\ & \to\Delta[-\overleftarrow{11},10,-9] & \to\Delta[-12,11,10,-9] \\ \to\Delta[-\overleftarrow{11},9] & \to\Delta[-\overleftarrow{12},11,9] & \to\Delta[-13,12,11,9] \\ & \to\Delta[-12,11,\overleftarrow{9}] & \to\Delta[-12,11,10,-9] \end{array}$$

Све претходне репрезентације означене разлике Δx могу се приказати сортиране по Хеминговој тежини:

$$\begin{aligned} \Delta x &= \Delta[-11,9]=\Delta[-10,-9] \\ &= \Delta[-11,10,-9]=\Delta[-12,11,9] \\ &= \Delta[-12,11,10,-9]=\Delta[-13,12,11,9] \end{aligned}$$

Непосредно се види да није увек могуће наћи јединствену репрезентацију означене разлике са минималном Хеминговом тежином, будући да важи:

$$\Delta[j+1, j] = \Delta[j+2, -j]$$

Уколико се означена разлика Δx са Хеминговом тежином $w = w_H(\Delta x)$ ротира после проширења, и уколико се ротација обавља преко бита на позицији 32, није могуће ту означену разлику поново вратити на Хемингову тежину пре проширења.

Пример 3.2.5. $\Delta[12]$ после проширења и ротације није могуће поново вратити на Хемингову тежину пре проширења:

$$\Delta[12] \ll 19 = \Delta[13, -12] \ll 19 = \Delta[-32, 1] \neq \Delta[32] = \Delta[12] \ll 19.$$

3.3 Интерна колизија

Дефиниција 3.3.1. *Под појмом нулте разлике (енгл. zero difference) у кораку i, подразумева се следеће стање разлика сва четири регистра које производе поруке M и M':*

$$(\Delta a_{i-3}, \Delta a_i - 2, \Delta a_{i-1}, \Delta a_i) = (0, 0, 0, 0)$$

Дефиниција 3.3.2. *Под појмом интерне колизије (енгл. internal, local collision) у низу корака $(i, i+1, ..., n)$ подразумева се појава нулте разлике у корацима $i-1$ и n, уз истовремене ненулте разлике у међукорацима $(i, ..., n-1)$.*

Лема 3.3.1. *Нека је $W_i = m_{w_i}$ i-та реч поруке. Ако у кораку i важи $(\Delta a_{i-3}, \Delta a_{i-2}, \Delta a_{i-1}, \Delta a_i) = (0, 0, 0, 0)$, тада важе следећа два тврђења:*

$$\begin{aligned} \Delta a_{i+1} = 0 &\iff \Delta W_{i+1} = 0 \\ \Delta a_{i-4} = 0 &\iff \Delta W_i = 0 \end{aligned}$$

Доказ. Ако би важило $\Delta a_{i+1} = 0$, то би била последња у низу од пет означених разлика које су једнаке нули. На њу дакле не утичу претходне четири разлике, тако да је једино може променити реч W_{i+1} у датом кораку. Будући да је $\Delta a_{i+1} = 0$ следи да је $\Delta W_{i+1} = 0$. Слично се доказује обрнута импликација.
Из претпоставке $\Delta a_{i-4} = 0$, идентично као у претходном разматрању следи да би пет узастопних означених разлика једнаких нули (завршно са кораком i), па је $\Delta W_i = 0$. □

Из претходне леме следи да унутрашња колизија може почети и завршити се само у кораку где је уведена колизиона разлика.

Теорема 3.3.1. *За сваку интерну колизију у низу корака* $(i, ..., n)$*, важи* $\Delta W_i \neq 0$ *и* $\Delta W_n \neq 0$.

Пример 3.3.1. Претпоставимо да означена разлика задовољава услове: $\Delta a_{i+1} \neq 0, \Delta a_{i+2} = 0, \Delta a_{i+3} = 0, \Delta a_{i+4} = 0, \Delta a_{i+5} = 0$ Нека интерна колизија почиње у кораку $i+1$, а завршава се у кораку $i+5$. Из леме 3.2. следи $\Delta a_{i+1} = 0 \iff \Delta W_{i+1} = 0$, па пошто је $\Delta a_{i+1} \neq 0$ следи $\Delta W_{i+1} \neq 0$. Из леме 3.2. следи $\Delta a_{i+1} = 0 \iff \Delta W_{i+5} = 0$, па пошто је $\Delta a_{i+1} \neq 0$ следи $\Delta W_{i+5} \neq 0$.

Важно је приметити да интерна колизија не може почети и завршити се у истом кораку. Ако би се то могло десити, рецимо у кораку i, тада би следило:

$$\text{пошто почиње} \Rightarrow \Delta a_{i-4} = \Delta a_{i-3} = \Delta a_{i-2} = \Delta a_{i-1} = 0$$
$$\text{пошто завршава} \Rightarrow \Delta a_{i-3} = \Delta a_{i-2} = \Delta a_{i-1} = \Delta a_i = 0$$

Пошто је интерна колизија уведена у i-том кораку, из теореме 3.1. следи $\Delta W_i \neq 0$. Међутим, пошто су пет узастопних разлика нулте, из леме 3.2. следило би $\Delta W_i = 0$, што је контрадикција.
Претходна чињеница указује да уколико се колизиона разлика своди на један бит разлике (самим тим на једну реч разлике која постоји у свакој рунди), интерна колизија мора почети у првој и завршити се у трећој рунди, јер би се у супротном морала и завршити и поново почети у истом кораку друге рунде.

Могући начин формирања интерне колизије у трећој рунди, уколико колизиона разлика укључује три речи, приказан је табели 3.1. Полазећи од израза $(a_i = (a_{i-4} + XOR(a_{i-3}, a_{i-2}, a_{i-1}) + m_{w_i} + c_i) \lll s_i)$ закључује се да је:

Корак	Δa_i	$\sum$	Δm_{w_i}	ΔXOR	Δa_{i-1}	Δa_{i-2}	Δa_{i-3}	Δa_{i-4}
$i-1$	0	0	0	0	0	0	0	0
i	$\Delta[32]$	$\Delta[32-s_i]$	$\Delta[32-s_i]$	0	0	0	0	0
$i+1$	$\Delta[32]$	$\Delta[32-s_{i+1}]$	$\Delta[32, 32-s_{i+1}]$	$\Delta[32]$	$\Delta[32]$	0	0	0
$i+2$	0	0	0	0	$\Delta[32]$	$\Delta[32]$	0	0
$i+3$	0	0	0	0	0	$\Delta[32]$	$\Delta[32]$	0
$i+4$	0	0	0	0	0	0	$\Delta[32]$	$\Delta[32]$
$i+5$	0	0	$\Delta[32]$	$\Delta[32]$	0	0	0	$\Delta[32]$
$i+6$	0	0	0	0	0	0	0	0

Табела 3.1. Простирање разлика у трећој рунди

Трећа колона $(\sum)$ означава суму свих постојећих означених разлика у датом кораку. Нека је претпоставка да у кораку $i-1$ постоји нулта разлика. Да би се у i-том кораку створила разлика $\Delta a_i = \Delta[32]$, довољно је да важи $\Delta m_{w_i} = \Delta[32 - s_i]$, јер та промена речи m_{w_i} уз

померај s_i изазива $\Delta a_i = \Delta[32]$. Сада се новонастала промена преноси на корак $i+1$, где изазива разлику $\Delta[32]$ у функцији XOR. Да би се анулирао утицај ове промене на регистар a_{i+1} уводи се нова разлика $\Delta m_{w_{i+1}} = \Delta[32]$. Да би се у следећем кораку постигле две промене у функцији XOR, потребно је у кораку $i+1$ увести још једну разлику $\Delta m_{w_{i+1}} = \Delta[32 - s_{i+1}]$, која изазива нову промену $\Delta a_{i+1} = \Delta[32]$. Тиме се добијају две разлике $\Delta[32]$ у два узастопна корака, чиме ће се очувати излаз функције XOR у наредна три корака. Тек у кораку $i+5$, потребно је поново увести разлику $\Delta m_{w_{i+5}} = \Delta[32]$ да би се анулирала једина преостала разлика у функцији XOR. Тиме се поново у кораку $i+6$ добија нулта разлика.

3.4 Поступак Ванг

Може се рећи да је кинескиња Ванг у протеклих пет година, у извесном смислу направила револуцију у криптоанализи хеш алгоритама. Паралелно са њеним истраживањем, појавили су се и други независни радови, али је ипак последњих година, највећи број радова инспирисан њеним идејама. У овом поглављу описан је први рад везан за криптоанализу алгоритма MD4 у коме је диференцијални напад практично реализован у свим својим етапама.

3.4.1 Основна идеја

Идеја диференционалног напада, какав је и овај, своди се на посматрање разлике између две поруке. Разлика подразумева одузимање порука (по модулу 2^{32}), а не ексклузивну дисјункцију (што јесте била почетна идеја код увођења појма диференцијалног напада). У циљу бољег разумевања означених разлика, разматра се ефекат преноса бита код сабирања (енгл. carry effect) на следећем примеру. Нека је нпр. разлика регистара $\Delta c_2 = c_2' - c_2 = -2^{18} + 2^{21}$, што значи да би очекивана ознака за ову промену била $c_2' = c_2[-19, 22]$. Ово следи из чињенице да одузимање од 32-битног бинарног броја вредности 2^{18}, у ствари значи промену његовог 19-тог бита на нулу, уколико је пре тога био јединица. Слично, додавањем вредности 2^{21}, 22-ги бит се поставља на јединицу, уколико је пре тога био нула. Дакле, разлика зависи од почетног стања променљиве c_2. У случају да важе услови $c_{2,19} = 0, c_{2,20} = 0, c_{2,21} = 1, c_{2,22} = 0$ десиће се промена $c_2' = c_2[19, 20, -21, 22]$. У даљем тексту наведена промена детаљно се објашњава; сада је битно приметити да однос између c_2' и c_2 зависи од тренутног стања регистра c_2, а самим тим од поруке M. То значи да се погодним избором битова у поруци M може утицати на однос између порука M и M'. Ово је основна идеја на којој се заснива поступак проналажења колизије између посматраних порука са вероватноћом између 2^{-2} и 2^{-6}.

За нелинеарну функцију прве рунде $F = (X \wedge Y) \vee (\neg X \wedge Z)$ важи:

Лема 3.4.1. *$F(X,Y,Z) = F(\neg X,Y,Z)$ ако и само ако важи $Y = Z$.*

Доказ. Услов $F(X,Y,Z) \oplus F(\neg X,Y,Z) = 0$ еквивалентан је чињеници да промена промењиве X не утиче на излаз функције F. Међутим, $F(X,Y,Z) \oplus F(\neg X,Y,Z) = ((X \wedge Y) \vee (\neg X \wedge Z)) \oplus ((\neg X \wedge Y) \vee (X \wedge Z)) = ((X \wedge Y) \vee (\neg X \wedge Y)) \oplus ((\neg X \wedge Z) \vee (X \wedge Z)) = (Y \wedge (\neg X \vee X)) \oplus (Z \wedge (\neg X \vee X)) = Y \oplus Z$, па негирање промењиве X не мења вредност функције F ако и само ако је $Y = Z$. □

Лема 3.4.2. *$F(X,Y,Z) = F(X,\neg Y,Z)$ ако и само ако важи $X = 0$.*

Доказ. За $X = 0$ је $F(X,Y,Z) \oplus F(X,\neg Y,Z) = ((X \wedge Y) \vee (\neg X \wedge Z)) \oplus ((X \wedge \neg Y) \vee (\neg X \wedge Z)) = (\neg X \wedge Z) \oplus (\neg X \wedge Z) = 0$.
За $X = 1$ је $F(X,Y,Z) \oplus F(X,\neg Y,Z) = ((X \wedge Y) \vee (\neg X \wedge Z)) \oplus ((X \wedge \neg Y) \vee (\neg X \wedge Z)) = (X \wedge Y) \oplus (X \wedge \neg Y) = Y \oplus \neg Y = 1$.
Дакле, негирање промењиве Y не мења вредност функције F ако и само ако је $X = 0$. □

Лема 3.4.3. *$F(X,Y,Z) = F(X,Y,\neg Z)$ ако и само ако важи $X = 1$.*

Доказ. За $X = 0$ је $F(X,Y,Z) \oplus F(X,Y,\neg Z) = ((X \wedge Y) \vee (\neg X \wedge Z)) \oplus ((\neg X \wedge Y) \vee (X \wedge \neg Z)) = (\neg X \wedge Z) \oplus (\neg X \wedge \neg Z) = Z \oplus \neg Z = 1$.
За $X = 1$ је $F(X,Y,Z) \oplus F(X,Y,\neg Z) = ((X \wedge Y) \vee (\neg X \wedge Z)) \oplus ((X \wedge Y) \vee (X \wedge \neg Z)) = (X \wedge Y) \oplus (X \wedge Y) = 0$.
Дакле, негирање промењиве Z не мења вредност функције F ако и само ако је $X = 1$. □

За нелинеарну функцију друге рунде $G(X,Y,Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)$ важи:

Лема 3.4.4. *$G(X,Y,Z) = G(\neg X,Y,Z)$ ако и само ако важи $Y = Z$.*

Доказ. За $Y = Z = 1$ следи $G(X,Y,Z) \oplus G(\neg X,Y,Z) = ((X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)) \oplus ((\neg X \wedge Y) \vee (\neg X \wedge Z) \vee (Y \wedge Z)) = 1 \oplus 1 = 0$.

За $Y = Z = 0$ следи $G(X,Y,Z) \oplus G(\neg X,Y,Z) = ((X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)) \oplus ((\neg X \wedge Y) \vee (\neg X \wedge Z) \vee (Y \wedge Z)) = 0$.

Из $Y \neq Z$ следи $G(X,Y,Z) \oplus G(\neg X,Y,Z) = ((X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z)) \oplus ((\neg X \wedge Y) \vee (\neg X \wedge Z) \vee (Y \wedge Z)) = ((X \wedge Y) \vee (X \wedge Z)) \oplus ((\neg X \wedge Y) \vee (\neg X \wedge Z)) = X \oplus \neg X = 1$.

Дакле, за $Y = Z$, негирање промењиве X не мења вредност функције G. □

Лема 3.4.5. *$G(X,Y,Z) = G(X,\neg Y,Z)$ ако и само ако важи $X = Z$.*

Доказ. Због симетричности функције G, доказ је аналоган као у леми 2.4. □

Лема 3.4.6. *$G(X,Y,Z) = G(X,Y,\neg Z)$ ако и само ако важи $X = Y$.*

Доказ. Због симетричности функције G, доказ је аналоган као у леми 2.4. □

За нелинеарну функцију треће рунде $H(X,Y,Z) = X \oplus Y \oplus Z$ важи:

Лема 3.4.7. $H(X,Y,Z) = \neg H(\neg X,Y,Z) = \neg H(X,\neg Y,Z) = \neg H(X,Y,\neg Z)$.

Доказ. Непосредно следи из особине функције H. □

Лема 3.4.8. $H(X,Y,Z) = H(\neg X,\neg Y,Z) = H(\neg X,Y,\neg Z) = H(X,\neg Y,\neg Z)$.

Доказ. Непосредно следи из особине функције H. □

У табели која следи описано је понашање функција F, G и H. На пример, $\Delta x = 0$ означава очување вредности бита x, $\Delta x = 1$ означава његову промену са нуле на јединицу, док $\Delta x = -1$ означава промену са јединице на нулу. Слично, $\Delta F = 0$ означава да излаз функције F није промењен, док $\Delta F = 1$ односно $\Delta F = -1$ означава да је промењен са нуле на јединицу односно са јединице на нулу. Уколико се у пољу налази цртица (-), у питању је немогућ излаз функције F. Уколико стоји јединица, конкретни излаз сигурно се дешава. Уколико стоји одређен услов, излаз функције се дешава уколико је тај услов испуњен. Аналогно важи и за функције G и H.

Δx	Δy	Δz	$\Delta F = 0$	$\Delta F = 1$	$\Delta F = -1$	$\Delta G = 0$	$\Delta G = 1$	$\Delta G = -1$
0	0	0	1	-	-	1	-	-
0	0	1	$x = 1$	$x = 0$	-	$x = y$	$x \neq y$	-
0	0	-1	$x = 1$	-	$x = 0$	$x = y$	-	$x \neq y$
0	1	0	$x = 0$	$x = 1$	-	$x = z$	$x \neq z$	-
0	1	1	-	1	-	-	1	-
0	1	-1	-	$x = 1$	$x = 0$	1	-	-
0	-1	0	$x = 0$	-	$x = 1$	$x = z$	-	$x \neq z$
0	-1	1	-	$x = 0$	$x = 1$	1	-	-
0	-1	-1	-	-	$y = 0$	1	-	-
1	0	0	$y = z$	$y = 1, z = 0$	$y = 0, z = 1$	$y = z$	$y \neq z$	-
1	0	1	$y = 0$	$y = 1$	-	-	1	-
1	0	-1	$y = 1$	-	$y = 0$	1	-	-
1	1	0	$z = 1$	$z = 0$	-	-	1	-
1	1	1	-	1	-	-	1	-
1	1	-1	1	-	-	-	1	-
1	-1	0	$z = 0$	-	$z = 1$	1	-	-
1	-1	1	1	-	-	-	1	-
1	-1	-1	-	-	1	-	-	1
-1	0	0	$y = z$	$y = 0, z = 1$	$y = 1, z = 0$	$y = z$	-	$y \neq z$

Δx	Δy	Δz	$\Delta F = 0$	$\Delta F = 1$	$\Delta F = -1$	$\Delta G = 0$	$\Delta G = 1$	$\Delta G = -1$
-1	0	1	$y = 1$	$y = 0$	-	1	-	-
-1	0	-1	$y = 0$	-	$y = 1$	-	-	1
-1	1	0	$z = 0$	$z = 1$	-	1	-	-
-1	1	1	-	1	-	-	1	-
-1	1	-1	1	-	-	-	-	1
-1	-1	0	$z = 1$	-	$z = 0$	-	-	1
-1	-1	1	1	-	-	-	-	1
-1	-1	-1	-	-	1	-	-	1

Табела 2.1. Приказ особина функција $F = IF(x, y, z) = (x \wedge y) \vee (\neg x \wedge z)$ и $G = MAJ(x, y, z) = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$

Δx	Δy	Δz	$\Delta H = 0$	$\Delta H = 1$	$\Delta H = -1$
0	0	0	1	-	-
0	0	1	-	$x = y$	$x \neq y$
0	0	-1	-	$x \neq y$	$x = y$
0	1	0	-	$x = z$	$x \neq z$
0	1	1	1	-	-
0	1	-1	1	-	-
0	-1	0	-	$x \neq z$	$x = z$
0	-1	1	1	-	-
0	-1	-1	1	-	-
1	0	0	-	$y = z$	$y \neq z$
1	0	1	1	-	-
1	0	-1	1	-	-
1	1	0	1	-	-
1	1	1	-	1	-
1	1	-1	-	-	1
1	-1	0	1	-	-
1	-1	1	-	-	1
1	-1	-1	-	1	-
-1	0	0	-	$y \neq z$	$y = z$
-1	0	1	1	-	-
-1	0	-1	1	-	-
-1	1	0	1	-	-
-1	1	1	-	-	1
-1	1	-1	-	1	-
-1	-1	0	1	-	-
-1	-1	1	-	1	-
-1	-1	-1	-	-	1

Табела 2.2. Приказ особина функције $H = XOR(x, y, z) = x \oplus y \oplus z$

Приликом тражења колизије за алгоритам MD4 полази се од поруке

M, $|M| = 512$, и тражи се порука M' тако да је $f(H_0, M) = f(H_0, M')$. Нека је $\Delta H_0 = H_0' - H_0 = 0 \xrightarrow{(M,M')} \Delta H = 0$, при чему важи следеће:

$$\Delta M = M' - M = (\Delta m_0, \Delta m_1, ..., \Delta m_{15})$$

$$\Delta m_1 = 2^{31}, \Delta m_2 = 2^{31} - 2^{28}, \Delta m_{12} = -2^{16}$$

$$\Delta m_i = 0, 0 \leq i \leq 15, i \neq 1, 2, 12.$$

Разлика $\Delta m_1 = 2^{31}$ може значити да је 32-и бит речи m_i' једнак јединици, а речи m_i једнак нули. Слично, $\Delta m_2 = 2^{31} - 2^{28}$ може значити да је сем у 32-ом биту дошло и до промене у 29-ом биту који је нула за поруку m_2', а јединица за поруку m_2. У речи m_{12}' 17-ти бит може бити нула, а у одговарајућој речи m_{12}, 17-ти бит може бити јединица. Овим је утврђена почетна разлика колизије, али нису једнозначно одређене и саме поруке M и M'.

3.4.2 Анализа кретања битова кроз кораке алгоритма

1. $a_1 \leftarrow ((a_0 + F(b_0, c_0, d_0) + m_0) \lll 3)$

   Пошто је $\Delta m_0 = 0$, није дошло ни до какве промене, па је $a_1^{'} = a_1$.

2. $d_1 \leftarrow ((d_0 + F(a_1, b_0, c_0) + m_1) \lll 7)$

   Из промене $\Delta m_1 = 2^{31}$ следи $d_{1,7} \leftarrow 1$, а да би то могло да се деси, услов је да у основној поруци пре тога важи $d_{1,7} = 0$. Наведена промена другачије се означава са $d_1' = d_1[7]$ и чак и ако није наведен апостроф, подразумева се да је то излаз поруке M' тј. да се у ланчаној променљивој d_1' седми бит поставио на један.

3. $c_1 \leftarrow ((c_0 + F(d_1, a_1, b_0) + m_2) \lll 11)$

   Из претходног корака преноси се промена $d_1[7]$. Да бисмо ту промену анулирали на основу особине функције F мора важити $a_{1,7} = b_{0,7}$. Уколико важе услови $c_{1,11} = 0$ и $c_{1,8} = 1$, промена $\Delta m_2 = 2^{31} - 2^{28}$ изазваће $c_{1,11}' \leftarrow 1$ под утицајем кружне ротације 32-ог бита за 11 и $c_{1,8}' \leftarrow 0$ под утицајем кружне ротације 29-ог бита за 11. Другим речима, долази до промене $c_1[-8, 11]$.

4. $b_1 \leftarrow ((b_0 + F(c_1, d_1, a_1) + m_3) \lll 19)$

   Из претходних корака преносе се промене $d_1[7]$ и $c_1[-8, 11]$.

   Да би се анулирала промена $c_1[-8, 11]$, морају важити услови $d_{1,8} = a_{1,8}$ и $d_{1,11} = a_{1,11}$.

   При томе увођење допунског услова $c_{1,7} = 1$ има за последицу да промена $d_1[7]$ проузрокује промену $b_1[26]$. Ово значи да мора важити услов $c_{1,7} = 1$, јер би се у супротном промена $d_1[7]$ такође анулирала.

5. $a_2 \leftarrow ((a_1 + F(b_1, c_1, d_1) + m_4) \ll 3)$

   Из претходних корака преносе се промене $d_1[7]$,$c_1[-8, 11]$ и $b_1[26]$. Потребни услови за њихово анулирање су редом $b_{1,7} = 1, b_{1,8} = 0, b_{1,11} = 0, c_{1,26} = d_{1,26}$. Следи да се регистар a_2 није променио, тј. $a_2' = a_2$.

6. $d_2 \leftarrow ((d_1 + F(a_2, b_1, c_1) + m_5) \ll 7)$

   Из претходних корака преносе се промене $d_1[7]$,$c_1[-8, 11]$ и $b_1[26]$.

   Промене $c_1[-8, 11]$ и $b_1[26]$ анулирају се редом условима $a_{2,8} = 1, a_{2,11} = 1, a_{2,26} = 0$.

   Промена $d_1[7]$ изазива промену $d_2[14]$ уз услов $d_{2,14} = 0$.

7. $c_2 \leftarrow ((c_1 + F(d_2, a_2, b_1) + m_6) \ll 11)$

   Из претходних корака преносе се промене $d_2[14]$,$c_1[-8, 11]$ и $b_1[26]$.

   Промене $d_2[14]$ и $b_1[26]$ анулирају се условима $a_{2,14} = b_{1,14}$ и $d_{2,26} = 1$.

   Промена $c_1[11]$ изазива промену $c_2[22]$ уз услов $c_{2,22} = 0$.

   Нека важе услови $c_{2,19} = 0, c_{2,20} = 0, c_{2,21} = 1$. Нека важи претпоставка да није дошло до промене $c_1[-8]$ тј. да је $c_{1,8} = 1$. Пошто важи претпоставка да је $c_{2,19} = 0$, из $c_{2,19} \leftarrow ((c_{1,8}+F(d_{2,8}, a_{2,8}, b_{1,8})+ m_{6,8}) \ll 11)$ следи $F(d_{2,8}, a_{2,8}, b_{1,8}) + m_{6,8} = 1$. А пошто ипак јесте дошло до промене, тј. $c_{1,8} = 0$, следи да је $c_{2,19}' = 1$ односно $c_2[19]$. Такође, да није дошло до промене, догодио би се пренос јединице при сабирању при чему треба да буде $c_{2,20} = 0$, па би то значило да из израза $c_{2,20} \leftarrow ((c_{1,9} + F(d_{2,9}, a_{2,9}, b_{1,9}) + m_{6,9}) \ll 11)$ следи $c_{1,9} + F(d_{2,9}, a_{2,9}, b_{1,9}) + m_{6,9} = 1$ (уз пренос јединице у следећи корак). Пошто јесте дошло до промене (и самим тим није дошло до преноса) следи $c_{2,20}' = 1$ тј. $c_2[20]$. Ако се опет претпостави да није дошло до промене (и самим тим је дошло до преноса), из $c_{1,9} + F(d_{2,9}, a_{2,9}, b_{1,9}) + m_{6,9} = 1$ и $c_{2,20} = 0$ следи да је дошло до преноса јединице у 21. корак и због претпоставке да важи $c_{2,21} = 1$, из $c_{2,21} \leftarrow ((c_{1,10} + F(d_{2,10}, a_{2,10}, b_{1,10}) + m_{6,10}) \ll 11)$ следи $c_{1,10} + F(d_{2,10}, a_{2,10}, b_{1,10}) + m_{6,10} = 0$, па у ситуацији када није дошло до преноса следи $c_{2,21}' = 0$ тј. $c_2[-21]$. Дакле, $c_2' = c_2[19, 20, -21, 22]$.

8. $b_2 \leftarrow ((b_1 + F(c_2, d_2, a_2) + m_7) \ll 19)$ Из претходних корака преносе се промене $d_2[14]$,$c_2[19, 20, -21, 22]$ и $b_1[26]$.

   Анулирање промене $c_2[19, 20, -21, 22]$ постиже се редом условима $d_{2,19} = a_{2,19}, d_{2,20} = a_{2,20}, d_{2,21} = a_{2,21}, d_{2,22} = a_{2,22}$.

   Услов $d_2[14]$ анулира се условом $c_{2,14} = 0$.

   Уз почетне услове $b_{2,13} = 1, b_{2,14} = 1, b_{2,15} = 0$ дешава се следеће: да није дошло до промене $b_1[26]$, важило би $b_{1,26} = 0$, па из израза $b_{2,13} \leftarrow ((b_{1,26} + F(c_{2,26}, d_{2,26}, a_{2,26}) + m_{7,26}) \ll 19)$ следи

$F(c_{2,26}, d_{2,26}, a_{2,26}) + m_{7,26} = 1$. Пошто јесте дошло до промене, следи $b'_{2,13} = 0$ тј. $b_2[-13]$ уз пренос јединице. Да није дошло до промене $b_2[-13]$, не би дошло ни до преноса јединице, па из претпоставке $b_{2,14} = 1$ и $b_{2,14} \leftarrow ((b_{1,27} + F(c_{2,27}, d_{2,27}, a_{2,27}) + m_{7,27}) \ll 19)$ следи $(b_{1,27} + F(c_{2,27}, d_{2,27}, a_{2,27}) + m_{7,27} = 1$. Али пошто је дошло до промене и до преноса биће $b'_{2,14} = 0$ ($b_2[-14]$) уз пренос јединице. Да није дошло до промене, тј. да нема преноса $b_2[-14]$, из претпоставке $b_{2,15} = 0$ и $b_{2,15} \leftarrow ((b_{1,28} + F(c_{2,28}, d_{2,28}, a_{2,28}) + m_{7,28}) \ll 19)$ следи $b_{1,28} + F(c_{2,28}, d_{2,28}, a_{2,28}) + m_{7,28} = 0$. Али, пошто јесте дошло до промене и према томе постоји пренос, следи $b'_{2,15} = 1$, односно промена $b_2[15]$. Дакле, $b'_2 = b_2[-13, -14, 15]$.

9. $a_3 \leftarrow ((a_2 + F(b_2, c_2, d_2) + m_8) \ll 3)$

Из претходних корака преносе се промене $d_2[14]$,$c_2[19, 20, -21, 22]$ и $b_2[-13, -14, 15]$.

Промена $b_2[-13, -14, 15]$ анулира се редом условима $c_{2,13} = d_{2,13}, c_{2,14} = d_{2,14}, c_{2,15} = d_{2,15}$.

Промена $c_2[19, 20, -21, 22]$ анулира се редом условима $b_{2,19} = 0, b_{2,20} = 0, b_{2,21} = 0, b_{2,22} = 0$.

Овде је важно приметити да је истовремено промењен 14. бит у регистрима b_2 и d_2, при чему су пре промене важили услови $b_{2,14} = 1, c_{2,14} = 0, d_{2,14} = 0$. Дакле, вредност функције F уместо постојеће $F(b_{2,14}, c_{2,14}, d_{2,14}) = F(1, 0, 0) = 0$, постаје $F(\neg b_{2,14}, c_{2,14}, \neg d_{2,14}) = F(0, 0, 1) = 1$, па је у изразу $a_{2,14} + F(b_{2,14}, c_{2,14}, d_{2,14}) + m_{8,14}$, 14. бит промењен на јединицу, што изазива промену $a_3[17]$.

10. $d_3 \leftarrow ((d_2 + F(a_3, b_2, c_2) + m_9) \ll 7)$

Из претходних корака преносе се промене $a_3[17]$,$d_2[14]$,$c_2[19, 20, -21, 22]$ и $b_2[-13, -14, 15]$.

Промена $a_3[17]$, анулира се условом $b_{2,17} = c_{2,17}$.

Уз услове $d_{3,20} = 0, d_{3,21} = 1, d_{3,22} = 1, d_{3,23} = 0, a_{3,13} = 1, a_{3,14} = 1$, $a_{3,15} = 1, c_{2,13} = d_{2,13}, c_{2,14} = 0, c_{2,15} = d_{2,15}$ посматра се простирање промене $b_2[-13]$. Ако се претпостави да није дошло до промене $b_2[-13]$, тј. да важи $b_{2,13} = 1$, из $d_{3,20} \leftarrow ((d_{2,13} + F(a_{3,13}, b_{2,13}, c_{2,13}) + m_{9,13}) \ll 7)$ следи $d_{2,13} + F(1, 1, c_{2,13}) + m_{9,13} = 0$, тј. $d_{2,13} + 1 + m_{9,13} = 0$ тј. $d_{2,13} + m_{9,13} = 1$ (при чему се дешава пренос јединице у следећи корак). То значи да уз промену $b_2[-13]$ следи $d_{2,13} + F(1, 0, c_{2,13}) + m_{9,13} = d_{2,13} + 0 + m_{9,13} = 1$, односно промена $d_3[20]$ без преноса јединице, па уз услов $d_{3,21} = 0$ следи промена $d_3[-21]$.

Промене $b_2[-14]$ и $d_2[14]$ међусобно се потиру уз услов $a_{3,14} \neq c_{2,14}$, а то је испуњено условима $a_{3,14} = 1$ и $c_{2,14} = 0$.

Испитујући промену $b_2[15]$, у случају да до ње није дошло, важи $b_{2,15} = 0$. Из услова $d_{3,22} = 1, a_{3,15} = 1$ и из $d_{3,22} \leftarrow ((d_{2,15} + F(a_{3,15}, b_{2,15}, c_{2,15}) + m_{9,15}) \ll 7)$ следи $1 = d_{2,15} + F(1, 0, c_{2,15}) + m_{9,15}$, односно $d_{2,15} + m_{9,15} = 1$. То значи да уз промену $b_2[15]$ следи $d_{2,15} + F(1, 1, c_{2,15}) + m_{9,15} = 0$, односно $d_3[-22]$ уз пренос јединице у следећи корак. Како важи услов $d_{3,23} = 0$, пренос из претходног корака изазива промену $d_3[23]$. Дакле, $d'_3 = d_3[20, -21, -22, 23]$.

Даље, потребно је испитати утицај промене $c_2[19, 20, -21, 22]$. Промена $c_2[22]$ анулира се условом $a_{3,22} = 1$. Нека важе услови $d_{3,26} = 1, a_{3,19} = 0, a_{3,20} = 0, a_{3,21} = 0, b_{2,19} = 0, b_{2,20} = 0, b_{2,21} = 0, c_{2,19} = 0, c_{2,20} = 0, c_{2,21} = 1$. Да није дошло до промене $c_2[19]$, тј. из $c_{2,19} = 0$ и из $d_{3,26} \leftarrow ((d_{2,19} + F(a_{3,19}, b_{2,19}, c_{2,19}) + m_{9,19}) \ll 7)$ следило би $1 = d_{2,19} + F(0, 0, 0) + m_{9,19}$, односно $d_{2,19} + m_{9,19} = 1$. То значи да у случају промене $c_2[19]$ следи $d_{2,19} + F(0, 0, 1) + m_{9,19} = d_{2,19} + 1 + m_{9,19} = 0$, односно $d_3[-26]$ уз пренос јединице у следећи корак. Слично, да није дошло до промене $c_2[20]$, тј. да важи $c_{2,20} = 0$, следило би $d_{3,27} \leftarrow ((d_{2,20} + F(0, 0, 0) + m_{9,20}) \ll 7)$, а уз промену $c_2[20]$, следи $d_{3,27} \leftarrow ((d_{2,20} + F(0, 0, 1) + m_{9,20}) \ll 7)$ тј. $d_{3,27} \leftarrow ((d_{2,20}+1+m_{9,20}) \ll 7)$. Пошто постоји пренос из претходног корака, вредност бита $d_{3,27}$ остаје иста, али уз пренос јединице у следећи корак у коме промена $c_2[21]$ смањује вредност функције F, али пренос јединице из претходног корака то анулира, тако да вредност бита $d_{3,28}$. Дакле, промена $c_2[19, 20, -21]$ изазива промену $d_3[-26]$.

11. $c_3 \leftarrow ((c_2 + F(d_3, a_3, b_2) + m_{10}) \ll 11)$

Из претходних корака преносе се промене $a_3[17]$,$d_3[20, -21, -22, 23, -26]$, $c_2[19, 20, -21, 22]$ и $b_2[-13, -14, 15]$.

Промена $a_3[17]$, анулира се условом $d_{3,17} = 0$.

Промена $b_2[-13, -14, 15]$ анулира се редом условима $d_{3,13} = 1, d_{3,14} = 1, d_{3,15} = 1$.

Промена $d_3[20]$ анулира се условом $a_{3,20} = b_{2,20}$, који се замењује условом $a_{3,20} = 0$ (пошто већ постоји услов $b_{2,20} = 0$). Промена $d_3[-21]$ анулира се условом $a_{3,21} = b_{2,21}$, који се замењује условом $b_{2,21} = 0$ (пошто већ постоји услов $a_{3,21} = 0$).

Промене $d_3[-22]$ и $c_2[22]$ међусобно се анулирају пошто важи $a_{3,22} \neq b_{2,22}$.

Промене $d_3[23, -26]$ анулирају се условима $a_{3,23} = b_{2,23}, a_{3,26} = b_{2,26}$

Промена $c_2[19]$, уз услов $c_{3,30} = 1$ и
$c_{3,30} \leftarrow ((c_{2,19} + F(d_{3,19}, a_{3,19}, b_{2,19}) + m_{10,19}) \ll 11)$ изазива промену $c_3[-30]$ уз пренос јединице. Због претходног преноса, промене $c_2[20]$ и $c_{3,31} \leftarrow ((c_{2,20} + F(d_{3,20}, a_{3,20}, b_{2,20}) + m_{10,20}) \ll 11)$ следи да вредност бита $c_{3,31}$ остаје не промењена, али опет уз пренос јединице

у следећи корак у коме се тај пренос и промена $c_2[-21]$ анилурају, па и вредност бита $c_{3,32}$ остаје иста.

12. $b_3 \leftarrow ((b_2 + F(c_3, d_3, a_3) + m_{11}) \ll 19)$ Из претходних корака преносе се промене $a_3[17]$,$d_3[20, -21, -22, 23, -26]$, $c_3[-30]$ и $b_2[-13, -14, 15]$.

    Промена $a_3[17]$ анулира се условом $c_{3,17} = 1$.

    Промена $c_3[-30]$ анулира се условом $d_{3,30} = a_{3,30}$.

    Промена $d_3[20, -21, -22, 23, -26]$ анулира се редом условима $c_{3,20} = 0, c_{3,21} = 0, c_{3,22} = 0, c_{3,23} = 0, c_{3,26} = 0$.

    Из услова $b_{2,13} = 1$ и $b_{3,32} = 0$ и из $b_{3,32} \leftarrow ((b_{2,13} + F(c_{3,13}, d_{3,13}, a_{3,13}) + m_{11,13}) \ll 19)$ следи $F(c_{3,13}, d_{3,13}, a_{3,13}) + m_{11,13} = 1$ уз пренос јединице. Уз промену $b_2[-13]$, следи промена $b_3[32]$. Ако се десила промена $b_2[-14]$, вредност четрнаестог бита се спушта на нулу, па је вредност без промене у односу на вредност са променом већа за два при чему је наравно излаз исти, али постоји разлика у преносу. Та разлика анулира се променом $b_2[15]$, што значи да битови $b_{2,14}$ и $b_{2,15}$ остају непромењени.

13. $a_4 \leftarrow ((a_3 + F(b_3, c_3, d_3) + m_{12}) \ll 3)$

    Из претходних корака преносе се промене $a_3[17]$,$d_3[20, -21, -22, 23, -26]$,$c_3[-30]$ и $b_3[32]$.

    У овом кораку појављује се разлика $\Delta m_{12} = -2^{16}$ и анулира се променом $a_3[17]$, што се на слици 3.1 представља црвеном стрелицом.

    Промена $b_3[32]$ анулира се условом $c_{3,32} = d_{3,32}$.

    Промена $c_3[-30]$ анулира се условом $b_{3,30} = 0$.

    Промена $d_3[20]$ узрокује промену $a_4[23]$ уз услов $b_{3,20} = 0$. Промене $d_3[-21]$ и $d_3[-22]$ анулирају се редом условима $b_{3,21} = 1$ и $b_{3,22} = 1$. Промена $d_3[23]$ узрокује промену $a_4[26]$ уз услов $b_{3,23} = 0$. Овај услов се може заменити условом $b_{3,23} = c_{3,23}$ пошто важи услов $c_{3,23} = 0$. Промена $d_3[-26]$ анулира се условом $b_{3,26} = 1$. Дакле, дошло је до промене $a_4[23, 26]$.

14. $d_4 \leftarrow ((d_3 + F(a_4, b_3, c_3) + m_{13}) \ll 7)$

    Из претходних корака преносе се промене $a_4[23, 26]$,$d_3[20, -21, -22, 23, -26]$,$c_3[-30]$ и $b_3[32]$.

    Промена $a_4[23]$ анулира се условом $b_{3,23} = c_{3,23}$

    Промене $a_4[26]$ и $d_3[-26]$ међусобно се анулирају пошто важи $b_{3,26} \neq c_{3,26}$.

    Промена $b_3[32]$ анулира се условом $a_{4,32} = 0$. Промена $c_3[-30]$ анулира се условом $a_{4,30} = 0$.

    Нека важе услови: $d_{4,27} = 1, d_{4,29} = 1, d_{4,30} = 0, d_{4,32} = 1$. Да није дошло до промене $d_3[20]$, тј. да је $d_{3,20} = 0$, из $d_{4,27} \leftarrow ((d_{3,20} +$

$F(a_{4,20}, b_{3,20}, c_{3,20}) + m_{13,20}) \ll 7)$ следи $F(a_{4,20}, b_{3,20}, c_{3,20}) + m_{13,20} = 1$. То значи да промена $d_3[20]$ изазива промену $d_4[-27]$ уз пренос јединице. Овај пренос анулира се променом $d_4[-28]$, тако да бит $d_{4,28}$ остаје непромењен. Да није дошло до промене $d_3[-22]$, тј. да је $d_{3,22} = 1$ из $d_{4,29} \leftarrow ((d_{3,22} + F(a_{4,22}, b_{3,22}, c_{3,22}) + m_{13,22}) \ll 7)$ следило би $F(a_{4,22}, b_{3,22}, c_{3,22}) + m_{13,22} = 0$, а пошто је дошло до промене, следи $d_4[-29]$. Да није дошло до промене $d_3[23]$, тј. да је $d_{3,23} = 0$, из $d_{4,30} \leftarrow ((d_{3,23} + F(a_{4,23}, b_{3,23}, c_{3,23}) + m_{13,23}) \ll 7)$ следило би $F(a_{4,23}, b_{3,23}, c_{3,23}) + m_{13,23} = 0$, а пошто је дошло до промене $d_3[23]$, следи промена $d_4[30]$.

15. $c_4 \leftarrow ((c_3 + F(d_4, a_4, b_3) + m_{14}) \ll 11)$

Из претходних корака преносе се промене $a_4[23, 26]$,$d_4[-27, -29, 30]$, $c_3[-30]$ и $b_3[32]$.

Промене $a_4[23, 26]$ анулирају се редом условима $d_{4,23} = 0, d_{4,26} = 0$.

Промена $b_3[32]$ анулира се условом $d_{4,32} = 1$.

Промене $d_4[-27, -29]$ анулирају се редом условима $a_{4,27} = b_{3,27}$ и $a_{4,29} = b_{3,29}$.

Уз услове $a_{4,30} = 1, b_{4,30} = 0$ промене $d_3[30]$ и $c_3[-30]$ међусобно се анулирају.

16. $b_4 \leftarrow ((b_3 + F(c_4, d_4, a_4) + m_{15}) \ll 19)$

Из претходних корака преносе се промене $a_4[23, 26]$,$d_4[-27, -29, 30]$ и $b_3[32]$.

Промене $a_4[23, 26]$ анулирају се редом условима $c_{4,23} = 0, c_{4,26} = 0$.

Промене $d_4[-27, -29, 30]$ анулирају се редом условима $c_{4,27} = 0, c_{4,29} = 0, c_{4,30} = 0$.

Промена $b_3[32]$ изазива промену $b_4[19]$ уз услов $b_{4,19} = 0$.

17. $a_5 \leftarrow ((a_4 + G(b_4, c_4, d_4) + m_0 + 5A827999) \ll 3)$

Из претходних корака преносе се промене $a_4[23, 26]$,$d_4[-27, -29, 30]$ и $b_4[19]$.

Промена $b_4[19]$ анулира се условом $c_{4,19} = d_{4,19}$.

Нека важе услови: $a_{5,26} = 1, a_{5,27} = 0, a_{5,29} = 1, a_{5,32} = 1$. Ако је пре промене $a_4[23]$ важио услов $a_{5,26} = 1$, онда ће после промене важити промена $a_5[-26]$ уз пренос јединице. Због тог преноса и услова $a_{5,27} = 0$, важи промена $a_5[27]$. Слично, из услова $a_{5,29} = 1$ и промене $a_4[26]$, следи промена $a_5[-29]$, уз пренос јединице који се у 30. биту анулира променом $d_4[-27]$. Претходно следи из чињенице да промена $d_4[-27]$ мења излаз функције $G(b_{4,27}, c_{4,27}, d_{4,27})$ пошто важе услови $b_{4,27} = 1, c_{4,27} = 0$ односно $b_{4,27} \neq c_{4,27}$. Због

услова $a_{5,32} = 1, b_{4,29} = 1, c_{4,29} = 0 (b_{4,29} \neq c_{4,29})$, промена $d_4[-29]$ изазива промену $a_5[-32]$.

Промена $d_4[30]$ анулирана је условима $b_{4,30} = 0, c_{4,30} = 0$ ($b_{4,30} = c_{4,30}$).

18. $d_5 \leftarrow ((d_4 + G(a_5, b_4, c_4) + m_4 + 5A827999) \lll 5)$

Из претходних корака преносе се промене $a_5[-26, 27, -29, -32]$, $d_4[-27, -29, 30]$ и $b_4[19]$.

Промена $b_4[19]$ анулира се условом $a_{5,19} = c_{4,19}$.

Промена $a_5[-26]$ анулира се условима $b_{4,26} = c_{4,26} = 1$.

Уз услове $b_{4,29} = 1, c_{4,29} = 0$, промене $a_5[-29]$ и $d_4[-29]$ не мењају вредност бита $d_{5,29}$, али у односу на ситуацију без промене, одузима се јединица, што се у следећем кораку анулира променом $d_4[30]$. Пошто промена неког бита у функцији G не мења њен излаз ако су преостала два бита једнака, промена $a_5[-32]$ анулира се условом $b_{4,32} = c_{4,32}$[1].

19. $c_5 \leftarrow ((c_4 + G(d_5, a_5, b_4) + m_8 + 5A827999) \lll 9)$

Из претходних корака преносе се промене $a_5[-26, 27, -29, -32]$ и $b_4[19]$.

Промена $b_4[19]$ анулира се условом $d_{5,19} = a_{5,19}$.

Промене $a_5[-26, 27, -29, -32]$ анулирају се редом условима $d_{5,26} = b_{4,26}, d_{5,27} = b_{4,27}, d_{5,29} = b_{4,29}, d_{5,32} = b_{4,32}$.

20. $b_5 \leftarrow ((b_4 + G(c_5, d_5, a_5) + m_{12} + 5A827999) \lll 13)$

Из претходних корака преносе се промене $a_5[-26, 27, -29, -32]$ и $b_4[19]$.

Промене $a_5[-26, 27, -29, -32]$ анулирају се редом условима $c_{5,26} = d_{5,26}, c_{5,27} = d_{5,27}, c_{5,29} = d_{5,29}, c_{5,32} = d_{5,32}$.

Промена $b_4[19]$ изазива промену $b_5[32]$ уз услов $b_{5,32} = 0$.

Разлика $\Delta m_{12} = -2^{16}$ изазива промену $b_5[-30]$ уз услов $b_{5,30} = 1$.

21. $a_6 \leftarrow ((a_5 + G(b_5, c_5, d_5) + m_1 + 5A827999) \lll 3)$.

Из претходних корака преносе се промене $a_5[-26, 27, -29, -32]$ и $b_5[-30, 32]$.

Промена $b_5[-30, 32]$ анулира се редом условима $c_{5,30} = d_{5,30}$, $c_{5,32} = d_{5,32}$.

Разлика $\Delta m_1 = -2^{31}$ и промена $a_5[-32]$ анулирају се међусобно.

[1] У оригиналном раду овај услов је пропуштен, али је на то званично скренута пажња у раду [8].

Уз услове[2] $a_{6,29} = 1, a_{6,30} = 0, a_{6,32} = 1$ промене $a_5[-26, 27, -29]$ изазивају промене $a_6[-29, 30, -32]$.

22. $d_6 \leftarrow ((d_5 + G(a_6, b_5, c_5) + m_5 + 5A827999) \ll 5)$

Из претходних корака преносе се промене $a_6[-29, 30, -32]$ и $b_5[-30, 32]$.

Промене $a_6[30, -32]$ и $b_5[-30, 32]$ анулирају се међусобно.

Промена $a_6[-29]$ анулира се условом $b_{5,29} = c_{5,29}$.

23. $c_6 \leftarrow ((c_5 + G(d_6, a_6, b_5) + m_9 + 5A827999) \ll 9)$

Из претходних корака преносе се промене $a_6[-29, 30, -32]$ и $b_5[-30, 32]$.

Промене $a_6[30, -32]$ и $b_5[-30, 32]$ анулирају се међусобно.

Промена $a_6[-29]$ анулира се условом $d_{6,29} = b_{5,29}$.

24. $b_6 \leftarrow ((b_5 + G(c_6, d_6, a_6) + m_{13} + 5A827999) \ll 13)$

Из претходних корака преносе се промене $a_6[-29, 30, -32]$ и $b_5[-30, 32]$.

Промена $a_6[-29]$ анулира се условом $d_{6,29} = c_{6,29}$.

Промене $a_6[30, -32]$ и $b_5[-30, 32]$ анулирају се међусобно уз услове $c_{6,30} \neq d_{6,30}, c_{6,32} \neq d_{6,32}$ (односно $c_{6,30} = d_{6,30} + 1, c_{6,32} = d_{6,32} + 1$).

25. $a_7 \leftarrow ((a_6 + G(b_6, c_6, d_6) + m_2 + 5A827999) \ll 3)$.

Из претходних корака преносе се промене $a_6[-29, 30, -32]$. Разлика $\Delta m_2 = 2^{31} - 2^{28}$ анулира промене $a_6[-29, 30, -32]$, тако што разлика 2^{31} анулира промену $a_6[-32]$, а двоструко спуштање 29. бита анулира се променом $a_6[30]$. Важно је приметити да је су све промене анулирале, па ће се следећа промена догодити тек у тридесетшестом кораку под утицајем разлике $\Delta m_{12} = -2^{16}$.

36. $b_9 \leftarrow ((b_8 + H(c_9, d_9, a_9) + m_{12} + 6ED9EBA1) \ll 15)$

Због утицаја разлике $\Delta m_{12} = -2^{16}$ следи промена $b_9[-32]$.

37. $a_{10} \leftarrow ((a_9 + H(b_9, c_9, d_9) + m_2 + 6ED9EBA1) \ll 3)$

Из претходних корака преноси се промена $b_9[-32]$. Услед утицаја разлике $\Delta m_2 = 2^{31} - 2^{28}$, анулира се промена $b_9[-32]$ и производи промена $a_{10}[-32]$.

38. $d_{10} \leftarrow ((d_9 + H(a_{10}, b_9, c_9) + m_{10} + 6ED9EBA1) \ll 9)$

Из претходних корака преносе се промене $b_9[-32]$ и $a_{10}[-32]$ које се међусобно анулирају.

[2] У оригиналном раду услов $a_{6,30} = 0$ је пропуштен, али је на то званично скренута пажња у раду [8].

39. $c_{10} \leftarrow ((c_9 + H(d_{10}, a_{10}, b_9) + m_6 + 6ED9EBA1) \ll 11)$

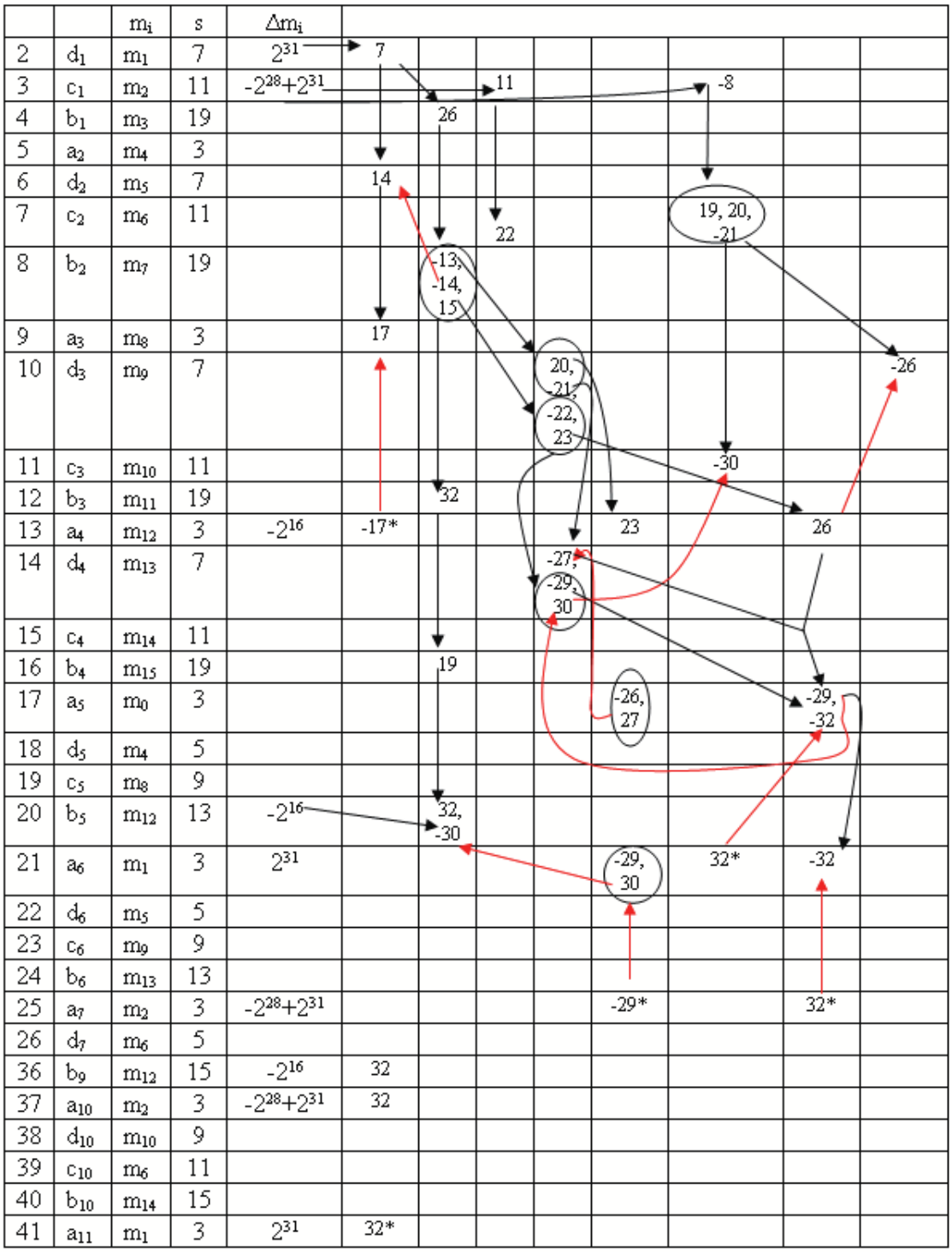
    Из претходних корака преносе се промене $b_9[-32]$ и $a_{10}[-32]$ које се међусобно анулирају.

40. $b_{10} \leftarrow ((b_9 + H(c_{10}, d_{10}, a_{10}) + m_{14} + 6ED9EBA1) \ll 15)$

    Из претходних корака преносе се промене $b_9[-32]$ и $a_{10}[-32]$ које се међусобно анулирају.

41. $a_{11} \leftarrow ((a_{10} + H(b_{10}, c_{10}, d_{10}) + m_1 + 6ED9EBA1) \ll 3)$

    Из претходних корака преноси се промена $a_{10}[-32]$ која се анулира разликом $\Delta m_1 = 2^{31}$.


m_i
s
Δm_i
d_1 m_1 7 2^{31}
c_1 m_2 11 $-2^{28}+2^{31}$
b_1 m_3 19
a_2 m_4 3
d_2 m_5 7
c_2 m_6 11
b_2 m_7 19
a_3 m_8 3
d_3 m_9 7
c_3 m_{10} 11
b_3 m_{11} 19
a_4 m_{12} 3 -2^{16}
d_4 m_{13} 7
c_4 m_{14} 11
b_4 m_{15} 19
a_5 m_0 3
d_5 m_4 5
c_5 m_8 9
b_5 m_{12} 13 -2^{16}
a_6 m_1 3 2^{31}
d_6 m_5 5
c_6 m_9 9
b_6 m_{13} 13
a_7 m_2 3 $-2^{28}+2^{31}$
d_7 m_6 5
b_9 m_{12} 15 -2^{16}
a_{10} m_2 3 $-2^{28}+2^{31}$
d_{10} m_{10} 9
c_{10} m_6 11
b_{10} m_{14} 15
a_{11} m_1 3 2^{31}
7
11
-8
26
14
19, 20, -21
22
-13, -14, 15
17
20, 21, -22, 23
-26
-30
32
-17*
23
26
-27, -29, 30
19
-26, 27
-29, -32
32, -30
32*
-32
-29, 30
-29*
32*

Слика 3.1. Кретање промена битова кроз алгоритам

3.4.3 Интерна колизија у трећој рунди

Као што је већ речено, под појмом интерне колизије подразумева се појава када у одређеном делу алгоритма одређена промена у поруци само тренутно изазива промену у регистрима, после чега бива анулирана. На тај начин, локално гледано, долази до колизије. Како се свих 16 речи поруке појављује у све три рунде, за добијање колизије неопходан је завршетак интерне колизије у трећој рунди. Уочава се да се три промењене речи у описаном нападу (m_1, m_2, m_{12}) у трећој рунди појављују у оквиру шест узастопних корака:

$b_9 \leftarrow (b_8 + H(c_9, d_9, a_9) + m_{12} + 6ED9EBA1) \ll 15$
$a_{10} \leftarrow (a_9 + H(b_9, c_9, d_9) + m_2 + 6ED9EBA1) \ll 3$
$d_{10} \leftarrow (d_9 + H(a_{10}, b_9, c_9) + m_{10} + 6ED9EBA1) \ll 9$
$c_{10} \leftarrow (c_9 + H(d_{10}, a_{10}, b_9) + m_6 + 6ED9EBA1) \ll 11$
$b_{10} \leftarrow (b_9 + H(c_{10}, d_{10}, a_{10}) + m_{14} + 6ED9EBA1) \ll 15$
$a_{11} \leftarrow (a_{10} + H(b_{10}, c_{10}, d_{10}) + m_1 + 6ED9EBA1) \ll 3$

Табела 3.2. Једнакости на основу којих је у трећој рунди пронађена интерна колизија

Претпоставка за ову конструкцију су почетне промене: $\Delta m_{12} = -2^{31-15} = -2^{16}, \Delta m_2 = -2^{31-3} + 2^{31} = 2^{31} - 2^{28}, \Delta m_1 = 2^{31}$. Промена $\Delta m_{12} = -2^{16}$ изазива у i-том кораку промену бита $b_{9,32}$. Промена $m_2 = 2^{31} - 2^{28}$ анулира промену бита $b_{9,32}$ и изазива нову промену бита $a_{10,32}$. У следећа три корака, промене $b_{9,32}$ и $a_{10,32}$ међусобно се анулирају, тако да регистри d_{10} и c_{10} остају непромењени. У последњем кораку, промене $a_{10,32}$ и $\Delta m_1 = 2^{31}$ међусобно се анулирају, што доводи до интерне колизије у шест корака. Важно је приметити да је до анулирања свих промена дошло само зато што је коришћен 32. бит, тј. бит највеће тежине (енгл. MSB - most significant bit). У супротном, могао би се приликом сваког покушаја анулирања десити пренос јединице са вероватноћом 1/2. Такође, треба приметити да је у i-том и $i+1$-ом кораку потребно поставити два услова, да приликом промене битова $b_{9,32}$ и $a_{10,32}$ под утицајем промена у речима m_{12} и m_2 не би дошло до преноса јединице. То значи да се интерна колизија догађа ако су испуњена та два услова, тј. са вероватноћом $1/2 \cdot 1/2 = 1/4$. Претходна анализа може се уопштити на произвољна четри регистра x, y, z, t и произвољне вредности корака i и бита j. Нека је s_i је померај у i-том кораку,а w_i индекс речи у i-том кораку. Ако се занемаре индекси регистара, нека у кораку $i-1$ важи веза:
$x = (x + H(y, z, t) + m_{w_{i-1}} + 6ED9EBA1) \ll s_{i-1}$ и нека у сваком наредном кораку x означава слободан регистар а регистри y, z, t редом први, други и трећи аргумент функције XOR. Тада важи табела 3.3:

корак	Δx	Δy	Δz	Δt	$\Delta m_{w_{i-1}}$
$i-1$	2^j	0	0	0	2^{j-s_i}
i	2^j	2^j	0	0	$2^{j-s_{i+1}}, 2^j$
$i+1$	0	2^j	2^j	0	0
$i+2$	0	0	2^j	2^j	0
$i+3$	2^j	0	0	2^j	0
$i+4$	0	2^j	0	0	2^j
$i+5$	0	0	0	0	0

Табела 3.3. Општи случај добијања интерне колизије у трећој рунди уз промену три речи

Алтернативни начин формирања интерне колизије ([7]), укључује промену пет (за разлику од досадашње три) речи, док се број корака смањује са шест на пет:

корак	Δx	Δy	Δz	Δt	$\Delta m_{w_{i-1}}$
$i-1$	2^j	0	0	0	2^{j-s_i}
i	0	2^j	0	0	2^j
$i+1$	0	0	2^j	0	2^j
$i+2$	0	0	0	2^j	2^j
$i+3$	2^j	0	0	0	2^j
$i+4$	0	0	0	0	0

Табела 3.4. Општи случај добијања интерне колизије у трећој рунди уз промену пет речи

Претходна интерна колизија подразумева једну промену бита у i-том кораку и четири анулирања битова у наредним корацима. То значи да уколико би се користио за бит промене $j = 32$, потребно је само у i-том кораку поставити услов, тако да нема преноса јединице. Дакле, до интерне колизије долази са вероватноћом $1/2$. Следећа почетна разлика порука ([7]) даје колизију тако што у трећој рунди ствара интерну колизију у пет корака и већину услова које треба задовољити оставља у првој рунди, што коришћењем просте модификације убрзава процес добијања колизије.

$$\Delta M = M' - M = (\Delta m_0, \Delta m_1, ..., \Delta m_{15})$$

$$\Delta m_0 = 2^{31-3} = 2^{28}, \Delta m_2 = 2^{31}, \Delta m_4 = 2^{31}, \Delta m_8 = 2^{31}, \Delta m_{12} = 2^{31}$$

$$\Delta m_i = 0, 0 \leq i \leq 15, i \neq 0, 2, 4, 8, 12.$$

3.4.4 Анализа напада

У табели која следи дат је преглед кретања битова поруке M' под поменутом разликом колизије. Прва колона означава корак у рундама (има их укупно 48, у свакој рунди по 16); друга колона означава

конкретну ланчану променљиву која се мења у датом кораку; трећа колона означава реч која се појављује у датом кораку; четврта колона означава кружни померај битова у лево; пета означава присуство почетне разлике колизије; шеста означава утицај разлике колизије на ланчану променљиву после кружног помераја или неког другог утицаја; седма представља промењену ланчану променљиву поруке M' у односу на одговарајућу ланчану променљиву поруке M. Специјално, празна поља у петој и шестој колони, као и изостављени кораци означавају да нема разлика између две поруке. Када се први пут појаве у низу четири узастопне ланчане променљиве такве да су исте за обе поруке M и M', направљена је интерна колизија. Из табеле се могу приметити две интерне колизије у корацима 2-25 и 36-41. Довољни услови за добијање колизије описани су у табели 3. То значи да ако порука M задовољава све дате услове, поруке M и M' изазваће колизију. У циљу бољег разјашњења појма *проширења промена битова*, следећи пример показује да разлика различитих парова једнобајтних порука може давати исти резултат.

```
M'   :   00010000  00100000  01000000  10000000
M    :   00000000  00010000  00110000  01110000
-----------------------------------------------
M'-M :   00010000  00010000  00010000  00010000
```

Дакле, разлика порука ($\Delta M = M' - M$) није довољна да би се сазнале саме поруке M и M'. Јасно је да ако на поруку M додамо разлику $\Delta M = 00010000$, новодобијена порука (M') *зависиће од претходног стања битова* на позицијама веће тежине (5,6,7 и 8). То значи да у наведеном примеру, разлика у једном биту (ΔM) у односу на поруку M, ствара нову поруку (M'), у којој се промене *проширују* у односу на стару поруку, променом 1,2,3 или 4 бита. Следећа табела илуструје кретање битова уз постављање услова у циљу добијања жељене путање. Колона *Разлика регистара* указује на почетну разлику регистара (аналогно разлици порука у претходном примеру), док колона *Излаз поруке* M' упућује на промене у излазном регистру у зависности од затечених стања битова (аналогно поруци M' у претходном примеру).

Корак	Излаз поруке M	Реч	Померај	Δm_i	Разлика регистара	Излаз поруке M'
1	a_1	m_0	3			a_1
2	d_1	m_1	7	2^{31}	2^6	$d_1[7]$
3	c_1	m_2	11	$-2^{28} + 2^{31}$	$-2^7 + 2^{10}$	$c_1[-8, 11]$
4	b_1	m_3	19		2^{25}	$b_1[26]$
5	a_2	m_4	3			a_2
6	d_2	m_5	7		2^{13}	$d_2[14]$
7	c_2	m_6	11		$2^{-18} + 2^{21}$	$c_2[19, 20, -21, 22]$
8	b_2	m_7	19		2^{12}	$b_2[-13, -14, 15]$

Корак	Излаз поруке M	Реч	Померај	Δm_i	Разлика регистара	Излаз поруке M'
9	a_3	m_8	3		2^{16}	$a_3[17]$
10	d_3	m_9	7		$2^{19}+2^{20}-2^{25}$	$d_3[20,-21,-22,23,-26]$
11	c_3	m_{10}	11		-2^{29}	$c_3[-30]$
12	b_3	m_{11}	19		2^{31}	$b_3[32]$
13	a_4	m_{12}	3	-2^{16}	$2^{22}+2^{25}$	$a_4[23,26]$
14	d_4	m_{13}	7		$-2^{26}+2^{28}$	$d_4[-27,-29,30]$
15	c_4	m_{14}	11			c_4
16	b_4	m_{15}	19		2^{18}	$b_4[19]$
17	a_5	m_0	3		$2^{25}-2^{28}-2^{31}$	$a_5[-26,27,-29,32]$
18	d_5	m_4	5			d_5
19	c_5	m_8	9			c_5
20	b_5	m_{12}	13	-2^{16}	$-2^{29}+2^{31}$	$b_5[-30,32]$
21	a_6	m_1	3	2^{31}	$2^{28}-2^{31}$	$a_6[-29,30,-32]$
22	d_6	m_5	5			d_6
23	c_6	m_9	9			c_6
24	b_6	m_{13}	13			b_6
25	a_7	m_2	3	$-2^{28}+2^{31}$		a_7
...	.....					.....
36	b_9	m_{12}	15	-2^{16}	2^{31}	$b_9[-32]$
37	a_{10}	m_2	3	$-2^{28}+2^{31}$	2^{31}	$a_{10}[-32]$
38	d_{10}	m_{10}	9			d_{10}
39	c_{10}	m_6	11			c_{10}
40	b_{10}	m_{14}	15			b_{10}
41	a_{11}	m_1	3	2^{31}		a_{11}

Табела 3.2. Колизиона путања - Ванг

Регистар	Довољни услови
a_1	$a_{1,7}=b_{0,7}$
d_1	$d_{1,7}=0, d_{1,8}=a_{1,8}, d_{1,11}=a_{1,11}$
c_1	$c_{1,7}=1, c_{1,8}=1, c_{1,11}=0, c_{1,26}=d_{1,26}$
b_1	$b_{1,7}=1, b_{1,8}=0, b_{1,11}=0, b_{1,26}=0$
a_2	$a_{2,8}=1, a_{2,11}=1, a_{2,26}=0, a_{2,14}=b_{1,14}$
d_2	$d_{2,14}=0, d_{2,19}=a_{2,19}, d_{2,20}=a_{2,20}, d_{2,21}=a_{2,21}, d_{2,22}=a_{2,22}, d_{2,26}=1$
c_2	$c_{2,13}=d_{2,13}, c_{2,14}=0, c_{2,15}=d_{2,15}, c_{2,19}=0, c_{2,20}=0, c_{2,21}=1, c_{2,22}=0$
b_2	$b_{2,13}=1, b_{2,14}=1, b_{2,15}=0, b_{2,17}=c_{2,17}, b_{2,19}=0, b_{2,20}=0, b_{2,21}=0, b_{2,22}=0$
a_3	$a_{3,13}=1, a_{3,14}=1, a_{3,15}=1, a_{3,17}=0, a_{3,19}=0, a_{3,20}=0, a_{3,21}=0, a_{3,23}=b_{2,23}$ $a_{3,22}=1, a_{3,26}=b_{2,26}$
d_3	$d_{3,13}=1, d_{3,14}=1, d_{3,15}=1, d_{3,17}=0, d_{3,20}=0, d_{3,21}=1, d_{3,22}=1, d_{3,23}=0$ $d_{3,26}=1, d_{3,30}=a_{3,30}$

c_3	$c_{3,17} = 1, c_{3,20} = 0, c_{3,21} = 0, c_{3,22} = 0, c_{3,23} = 0, c_{3,26} = 0, c_{3,30} = 1, c_{3,32} = d_{3,32}$
b_3	$b_{3,20} = 0, b_{3,21} = 1, b_{3,22} = 1, b_{3,23} = c_{3,23}, b_{3,26} = 1, b_{3,30} = 0, b_{3,32} = 0$
a_4	$a_{4,23} = 0, a_{4,26} = 0, a_{4,27} = b_{3,17}, a_{4,29} = b_{3,29}, a_{4,30} = 1, a_{4,32} = 0$
d_4	$d_{4,23} = 0, d_{4,26} = 0, d_{4,27} = 1, d_{4,29} = 1, d_{4,30} = 0, d_{4,32} = 1$
c_4	$c_{4,19} = d_{4,19}, c_{4,23} = 1, c_{4,26} = 1, c_{4,27} = 0, c_{4,29} = 0, c_{4,30} = 0$
b_4	$b_{4,19} = 0, b_{4,26} = c_{4,26} = 1, b_{4,27} = 1, b_{4,29} = 1, b_{4,30} = 0, b_{4,32} = c_{4,32}$
a_5	$a_{5,19} = c_{4,19}, a_{5,26} = 1, a_{5,27} = 0, a_{5,29} = 1, a_{5,32} = 1$
d_5	$d_{5,19} = a_{5,19}, d_{5,26} = b_{4,26}, d_{5,27} = b_{4,27}, d_{5,29} = b_{4,29}, d_{5,32} = b_{4,32}$
c_5	$c_{5,26} = d_{5,26}, c_{5,27} = d_{5,27}, c_{5,29} = d_{5,29}, c_{5,30} = d_{5,30}, c_{5,32} = d_{5,32}$
b_5	$b_{5,29} = c_{5,29}, b_{5,30} = 1, b_{5,32} = 0$
a_6	$a_{6,29} = 1, a_{6,32} = 1, a_{6,30} = 0$
d_6	$d_{6,29} = b_{5,29}$
c_6	$c_{6,29} = d_{6,29}, c_{6,30} = d_{6,30} + 1, c_{6,32} = d_{6,32} + 1$
b_9	$b_{9,32} = 1$
a_{10}	$a_{10,32} = 1$

Табела 3.3. Скуп довољних услова који воде до колизије

3.4.5 Модификација поруке

Уколико је задовољено свих 124 услова из табеле 3.3 поруку ћемо назвати "слабом". У том случају довољно је на њу додати разлику

$$\Delta M = M' - M = (\Delta m_0, \Delta m_1, ..., \Delta m_{15})$$

$$\Delta m_1 = 2^{31}, \Delta m_2 = 2^{31} - 2^{28}, \Delta m_{12} = -2^{16}$$

$$\Delta m_i = 0, 0 \leq i \leq 15, i \neq 1, 2, 12.$$

да би се добила нова порука M' која заједно са основном поруком M производи колизију. Међутим, случајним избором поруке , имајући у виду да 124 услова морају бити задовољена, у просеку, тек свака 2^{124}-та порука биће слаба. Ово је знатно слабије од рођенданског напада који у случају 128-битног сажетка износи 2^{64}.

3.4.6 Проста модификација

У циљу лакшег налажења "слабе" поруке, могуће је модификовати основну поруку M тако да важе услови везани за прву рунду. Овим се број потребних услова смањује на 25. На пример, m_1 се може модификовати са:

$d_1 \leftarrow d_1 \oplus (d_{1,7} \ll 6) \oplus ((d_{1,8} \oplus a_{1,8}) \ll 7) \oplus ((d_{1,1} \oplus a_{1,1}) \ll 10)$
$m_1 \leftarrow (d_1 \gg 7) - d_0 - F(a_1, b_0, c_0)$

Тиме су задовољени услови $d_{1,7} = 0, d_{1,8} = a_{1,8}, d_{1,11} = a_{1,11}$. Вредност седмог бита регистра d_1 померила се за шест места у лево до

седме позиције, где је примењена операција $\oplus$ над њим самим. Тиме је на седмом месту добијена нула. У случају да је $d_{1,8} \neq a_{1,8}$ следи $d_{1,8} \oplus a_{1,8} = 1$, па ће та јединица померањем на осмо место и применом операције $\oplus$ променити вредност осмог бита регистра d_1. У случају да је $d_{1,8} = a_{1,8}$ следи $d_{1,8} \oplus a_{1,8} = 0$, па ће та нула померањем на осмо место и применом операције $\oplus$ на постојећи бит $d_{1,8}$ сачувати његову вредност. Тиме се задовољава услов $d_{1,8} = a_{1,8}$. Идентично важи и у случају задовољења услова $d_{1,11} = a_{1,11}$.

3.4.7 Сложена модификација

Иако је вероватноћа 2^{-25} довољно мала да се грубом силом може наћи колизија, могуће је даље унапређење. Овај вид модификације поруке, користи се у циљу корекције услова у другој рунди и важан је и за криптоанализу алгоритама MD5, SHA-0 и посебно SHA-1. Основна идеја своди се на узастопну промену речи у првој рунди, тако да услови у првој рунди остану задовољени, а да промењена реч изазове промену одговарајућег бита у другој рунди (и самим тим задовољење услова друге рунде). На пример, претпоставимо да важи услов $a_{5,19} = \overline{c_{4,19}}$ и да желимо да га променимо (тако да буде $a_{5,19} = c_{4,19}$). Да би смо испунили циљ потребно је променити $a_{5,19}$, а то се постиже (због $a_{5,19} \leftarrow (a_{4,16} + G(b_{4,16}, c_{4,16}, d_{4,16}) + m_{0,16} + (5A827999)_{16}) \lll 3$) променом 16. бита у m_0. Ова промена, уз одговарајућу промену следеће узастопне четири речи m_1, m_2, m_3, m_4, изазива промену регистра a_1, али не и регистара који следе (d_1, c_1, b_1, a_2). Тиме се ова промена не простире даље.

Корак	Померај	Промењена реч m_i	Стање регистара
1	3	$m_0 \leftarrow m_0 \pm 2^{i-4}$	$a_1^{novo} = a_1[\pm i], b_0, c_0, d_0$
2	7	$m_1 \leftarrow (d_1 \ggg 7) - d_0 - F(a_1^{novo}, b_0, c_0)$	$d_1, a_1^{novo}, b_0, c_0$
3	11	$m_2 \leftarrow (c_1 \ggg 11) - c_0 - F(d_1, a_1^{novo}, b_0)$	$c_1, d_1, a_1^{novo}, b_0$
4	19	$m_3 \leftarrow (b_1 \ggg 19) - b_0 - F(c_1, d_1, a_1^{novo})$	$b_1, c_1, d_1, a_1^{novo}$
5	3	$m_4 \leftarrow (a_2 \ggg 3) - a_1^{novo} - F(b_1, c_1, d_1)$	a_2, b_1, c_1, d_1

Табела 3.4. Модификација бита $a_{5,i}$

Предходна табела приказује корекцију бита $a_{5,i}$ за $i = 19, 26, 27, 29, 32$ јер управо те битове треба променити да би се у другој рунди кориговали услови везани за регистар a_5. Иако је промењен бит $a_1[i]$, тиме није поремећен услов прве рунде $a_{1,7} = b_{0,7}$. У петом кораку $(a_2 \leftarrow ((a_1 + F(b_1, c_1, d_1) + m_4) \lll 3))$, истовремено су промењени a_1 и m_4, па се међусобно анулирају, не мењајући вредност регистра a_2. Сада се поставља питање како промена m_0, m_1, m_2, m_3, m_4 утиче на други рунду? m_0 је променило регистар a_1, при чему је промена регистра a_1 која се узгред десила потпуно небитна. Сама промена поруке је дозвољена јер се не изводи напад на јаку отпорност, већ се тражи колизија.

Промене m_1, m_2, m_3, m_4 могу додатно покварити услове из друге рунде, али их можемо поново кориговати понављањем сложене модификације. Применом просте или сложене модификације поруке, сви услови у првој и другој рунди биће кориговани.

У оригиналном раду ([1]), у циљу задовољења услова $c_{5,26} = d_{5,26}$ предложена је следећа модификација бита $c_{5,26}$
$(c_{5,26} \leftarrow (c_{4,17} + G(d_{5,17}, a_{5,17}, b_{4,17}) + m_{8,17} + (5A827999)_{17}) \ll 9)$:

Корак	Померај	Промењена реч m_i	Стање регистара	Додатни услов
6	7	$m_5 \leftarrow m_5 + 2^9$	$d_2[17], a_2, b_1, c_1$	$d_{2,17} = 0$
7	11		$c_2, d_2[17], a_2, b_1$	$a_{2,17} = b_{1,17}$
8	19		$b_2, c_2, d_2[17], a_2$	$c_{2,17} = 0$
9	3	$m_8 \leftarrow m_8 - 2^{16}$	$a_3, b_2, c_2, d_2[17]$	$b_{2,17} = 0$
10	11	$m_9 \leftarrow m_9 - 2^{16}$	d_3, a_3, b_2, c_2	

Табела 3.5 Модификација бита $c_{5,26}$

Потребно је приметити да у корацима 6 и 7, нема модификације одговарајућих речи m_6 и m_7, пошто се утицај промене $d_2[17]$ анулира одговарајућим условима прве рунде, тако да промена не утиче на излаз функција $F(d_2, a_2, b_1)$ и $F(c_2, d_2, a_2)$. С друге стране, услов у деветом кораку $b_{2,17} = 0$ не анулира утицај промене $d_2[17]$, већ то чини промена речи m_8. Промена речи m_9 анулира утицај промене $d_2[17]$, која у десетом кораку не утиче на вредност регистра d_3, а самим тим и на следеће регистре. *Тиме је промена речи m_8 у другој рунди изазвала жељену корекцију бита $c_{5,26}$, док је у првој рунди утицај те промене локализован.* Међутим, у случају да истовремено нису задовољени услови над битовима $d_{5,19}$ и $c_{5,26}$ $(d_{5,19} = a_{5,19}, c_{5,26} = d_{5,26})$, потребно је кориговати оба бита. Из $d_{5,19} \leftarrow ((d_{4,14}+G(a_{5,14}, b_{4,14}, c_{4,14})+m_{4,14}+(5A827999)_{14}) \ll 5)$ следи да је за анулирање промене $d_{5,19}$ потребно променити $m_{4,14}$, што ће због $a_{2,17} \leftarrow ((a_{1,14}+F(b_{1,14}, c_{1,14}, d_{1,14})+m_{4,14}) \ll 3)$ изазвати промену бита $a_{2,17}$. Тиме се ремети услов $a_{2,17} = b_{2,17}$, који је потребан за корекцију бита $c_{5,26}$. Дакле, описана ситуација дешава се у случају да су услови над битовима $d_{5,19}$ и $c_{5,26}$ истовремено незадовољени, а то се дешава са вероватноћом 1/4. Међутим, постоји начин да се и поред нарушавања услова $a_{2,17} = b_{2,17}$, анулира утицај промене $d_2[17]$ без коришћења додатног услова. Потребно је променити реч m_6 која одговара кораку у ком се појавио додатни услов $a_{2,17} = b_{2,17}$ тј. $m_6 \leftarrow (c_2 \gg 11) - c_1 - F(d_2^{novo}, a_2, b_1)$. Управо ова модификација свакако би се десила при корекцији бита $d_{6,29}$ уколико не задовољава услов $d_{6,29} = b_{5,29}$ (ово се дешава са вероватноћом 1/2):

Корак	Померај	Промењена реч m_i	Стање регистара	Додатни услов
6	7	$m_5 \leftarrow m_5 + 2^{23}$	$d_2^{novo} = d_2[\pm 31], a_2, b_1, c_1$	
7	11	$m_6 \leftarrow (c_2 \gg 11) - c_1 - F(d_2^{novo}, a_2, b_1)$	$c_2, d_2^{novo}, a_2, b_1$	
8	19	$m_7 \leftarrow (b_2 \gg 19) - b_1 - F(c_2, d_2^{novo}, a_2)$	$b_2, c_2, d_2^{novo}, a_2$	
9	3		$a_3, b_2, c_2, d_2^{novo}$	$b_{2,31} = 1$
10	11	$m_9 \leftarrow (d_3 \gg 7) - d^{novo} - F(a_3, b_2, c_2)$	d_3, a_3, b_2, c_2	

Табела 3.6. Модификација бита $d_{6,29}$

Другим речима, чак и ако је нарушен додатни услов $a_{2,17} = b_{2,17}$, уколико је бит $d_{6,29}$ коригован и бит $c_{5,26}$ биће адекватно коригован. На основу претходног, може се закључити да ће бит $c_{5,26}$ бити адекватно постављен (односно модификован) у следећим ситуацијама:

1. $c_{5,26}$ задовољава услов $c_{5,26} = d_{5,26}$ без потребе за корекцијом. То је испуњено са вероватноћом $1/2$.

2. $c_{5,26}$ је коригован, а $d_{5,19}$ је већ задовољавао потребан услов, па додатни услов $a_{2,17} = b_{2,17}$ није нарушен. То је испуњено са вероватноћом $1/4$.

3. Истовремено су $c_{5,26}$ и $d_{5,19}$ модификовани, па је нарушен додатни услов $a_{2,17} = b_{2,17}$, али је у седмом кораку регистар c_2 остао непромењен услед модификације $d_{6,29}$. То је испуњено са вероватноћом $1/4 \cdot 1/2 = 1/8$.

Дакле, вероватноћа успешног задовољења услова над битом $c_{5,26}$ износи $1/2 + 1/4 + 1/8 = 7/8$.

Сличан проблем јавља се у ситуацији када су битови $c_{5,32}$ и $c_{6,32}$ истовремено модификовани.

Корак	Померај	Промењена реч m_i	Стање регистара	Додатни услов
6	7	$m_5 \leftarrow m_5 + 2^{15}$	$d_2[23], a_2, b_1, c_1$	$d_{2,23} = 0$
7	11		$c_2, d_2[23], a_2, b_1$	$a_{2,23} = b_{1,23}$
8	19		$b_2, c_2, d_2[23], a_2$	$c_{2,23} = 0$
9	3	$m_8 \leftarrow m_8 - 2^{22}$	$a_3, b_2, c_2, d_2[23]$	$b_{2,23} = 0$
10	7	$m_9 \leftarrow m_9 - 2^{22}$	d_3, a_3, b_2, c_2	

Табела 3.7. Модификација бита $c_{5,32}$

Корак	Померај	Промењена реч m_i	Стање регистара	Додатни услов
7	11	$m_6 \leftarrow m_6 + 2^{11}$	$c_2[23], d_2, a_2, b_1$	$c_{2,23} = 0$
8	19		$b_2, c_2[23], d_2, a_2$	$d_{2,23} = a_{2,23}$
9	3		$a_3, b_2, c_2[23], d_2$	$b_{2,23} = 0$
10	7	$m_9 \leftarrow m_9 - 2^{22}$	$d_3, a_3, b_2, c_2[23]$	$a_{3,23} = 0$
11	11	$m_{10} \leftarrow m_{10} - 2^{22}$	c_3, d_3, a_3, b_2	

Табела 3.8. Модификација бита $c_{6,32}$

Приликом модификације бита $c_{5,32}$, неопходно је поставити додатни услов $c_{2,23} = 0$. У случају да се модификација бита $c_{6,32}$ обавља на принципу модификације бита $c_{5,32}$, из $c_6 \leftarrow ((c_5 + G(d_6, a_6, b_5) + m_9 + 5A827999) \ll 9)$ следило би да се утицај промене речи m_9 на прву рунду коригује променом речи m_6, m_9, m_{10} слично као у случају бита $c_{5,32}$. Из $c_{2,23} \leftarrow ((c_{1,12} + F(d_{2,12}, a_{2,12}, b_{1,12}) + m_{6,12}) \ll 11)$ следи да промена речи m_6 производи промену бита $c_2[23]$, чиме се ремети услов $c_{2,23} = 0$. Овај случај (истовремена модификација битова $c_{5,32}$ и $c_{6,32}$) дешава се са вероватноћом 1/4. Зато је потребно обавити нови начин модификације у коме ће се, слично као у случају бита $c_{5,26}$, корекција услова обавити без контрадикције.

Модификација бита $c_{5,29}$ на стари начин такође доводи до контрадикције услова, будући да се у циљу добијања интерне колизије у пет корака, мења вредност бита $d_{2,20}$, чиме се може нарушити услов $d_{2,20} = a_{2,20}$. Ово се дешава са вероватноћом 1/2. Зато је потребно обавити нови начин модификације у коме ће се, слично као у случају бита $c_{5,26}$, корекција услова обавити без контрадикције.

Корак	Померај	Промењена реч m_i	Стање регистара	Додатни услов
6	7	$m_5 \leftarrow m_5 + 2^{12}$	$d_2[20], a_2, b_1, c_1$	$d_{2,20} = 0$
7	11		$c_2, d_2[20], a_2, b_1$	$a_{2,20} = b_{1,20}$
8	19		$b_2, c_2, d_2[20], a_2$	$c_{2,20} = 0$
9	3	$m_8 \leftarrow m_8 - 2^{19}$	$a_3, b_2, c_2, d_2[20]$	$b_{2,20} = 0$
10	11	$m_9 \leftarrow m_9 - 2^{19}$	d_3, a_3, b_2, c_2	

Табела 3.9. Модификација бита $c_{5,29}$

На описан начин могу се успешно кориговати сви услови из прве и друге рунде, чиме се не обухватају само два услова из треће рунде. Имајући у виду анализу из [8], може се тачно израчунати вероватноћа успешне корекције поруке из оригиналног рада [1]. Вероватноћа задовољења два услова из треће рунде износи $1/2 \cdot 1/2 = 1/4$. Вероватноћа задовољења два изостављена услова ($a_{6,30} = 0$ и $b_{4,32} = c_{4,32}$) такође износи $1/2 \cdot 1/2 = 1/4$. Вероватноћа успешне модификације бита $c_{5,26}$ износи 7/8. Вероватноћа успешне модификације бита $c_{5,32}$ износи

3/4. Вероватноћа успешне модификације бита $c_{5,29}$ износи 1/2. Дакле, вероватноћа налажења колизије износи 21/1024. Степен над основом 2, који одговара овом броју добија се из: $2^x = 21/1024 \Rightarrow x = \log_2(21/1024) = \log_2 21 - \log_2 1024 = \log_2 3 + \log_2 7 - 10 = -5.60769$. Другим речима, вероватноћа налажења колизије приближно износи $2^{-5.60769}$ (што одговара полазној констатацији да се та вероватноћа налази између 2^{-2} и 2^{-6}).

3.5 Поступак Jу

У односу на оригинални рад [1], у коме је прво презентована идеја диференцијалног напада на MD4, у раду [5] дата је једноставнија путања која доводи до колизије. Тиме је број потребних услова које порука треба да задовољи у циљу добијања колизије, сведен са 124 на 62. Практично гледано, ово не представља суштинску добит, јер се већина услова анулира модификацијом поруке у првој и другој рунди. Међутим, поједностављена путања јасније илуструје идеју проналажења путање битова која води до колизије. Треба нагласити да је у овом случају слаба порука пронађена са вероватноћом 1 (што се постиже модификацијом свих услова произвољне поруке), па се за ту слабу поруку може наћи друга која даје исту хеш вредност, што се може посматрати и као напад на слабу отпорност на колизију.

3.5.1 Анализа кретања битова кроз кораке алгоритма

У овом случају почетна разлика порука гласи $\Delta M = M^{'} - M$ где је $M = (m_0, m_1, ..., m_{15})$, а $\Delta M = (0, 0, 0, 0, 2^{22}, 0, ..., 0)$, односно $\Delta m_4 = 2^{22}, \Delta m_i = 0, \forall i \neq 4$. У даљем тексту анализираће се кретање битова кроз кораке алгоритма.

1. $a_1 \leftarrow ((a_0 + F(b_0, c_0, d_0) + m_0) \lll 3)$

   Пошто је $\Delta m_0 = 0$, није дошло ни до какве промене, па је $a_1^{'} = a_1$

2. $d_1 \leftarrow ((d_0 + F(a_1, b_0, c_0) + m_1) \lll 7)$

   Пошто је $\Delta m_1 = 0$, није дошло ни до какве промене, па је $d_1^{'} = d_1$

3. $c_1 \leftarrow ((c_0 + F(d_1, a_1, b_0) + m_2) \lll 11)$

   Пошто је $\Delta m_2 = 0$, није дошло ни до какве промене, па је $c_1^{'} = c_1$

4. $b_1 \leftarrow ((b_0 + F(c_1, d_1, a_1) + m_3) \lll 19)$

   Пошто је $\Delta m_3 = 0$, није дошло ни до какве промене, па је $b_1^{'} = b_1$

5. $a_2 \leftarrow ((a_1 + F(b_1, c_1, d_1) + m_4) \lll 3)$

   Под утицајем промене 23. бита у речи m_4, због померања за 3 дошло је до промене $a_2[26]$. Да би се ова промена могла догодити, неопходно је да важи услов $a_{2,26} = 0$.

6. $d_2 \leftarrow ((d_1 + F(a_2, b_1, c_1) + m_5) \lll 7)$

   Из претходног корака преноси се промена $a_2[26]$. Да би се анулирао њен утицај на регистар d_2, неопходно је да важи услов $b_{1,26} = c_{1,26}$.

7. $c_2 \leftarrow ((c_1 + F(d_2, a_2, b_1) + m_6) \lll 11)$

   Из претходних корака преноси се промена $a_2[26]$. Да би се анулирао њен утицај на регистар c_2, неопходно је да важи услов $d_{2,26} = 0$.

8. $b_2 \leftarrow ((b_1 + F(c_2, d_2, a_2) + m_7) \lll 19)$

   Из претходних корака преноси се промена $a_2[26]$. Да би се анулирао њен утицај на регистар b_2, неопходно је да важи услов $c_{2,26} = 1$.

9. $a_3 \leftarrow ((a_2 + F(b_2, c_2, d_2) + m_8) \lll 3)$

   Из претходних корака преноси се промена $a_2[26]$. Ако би у овом кораку важио услов $a_{3,29} = 0$ следила би (због $a_2[26]$) промена $a_3[29]$. Међутим, ако су испуњени услови $a_{3,29} = 1$ и $a_{3,30} = 0$, тада важи следеће: $a_{3,29} = 1 \Rightarrow a_{2,26} + f(b_{2,26}, c_{2,26}, d_{2,26}) + m_{8,26} = 1$. Пошто је пре било каквих промена важило $a_{2,26} = 0$, следи да важи $f(b_{2,26}, c_{2,26}, d_{2,26}) + m_{8,26} = 1$. Пошто је дошло до промене $a_2[26]$ (тј. $a_{2,26} = 1$), следи (због $f(b_{2,26}, c_{2,26}, d_{2,26}) + m_{8,26} = 1$) $a_{3,29} = 0$ (тј. $a_3[-29]$) уз пренос јединице. Пошто је полазна претпоставка да је испуњен услов $a_{3,30} = 0$, следи да ће због преноса јединице важити $a_{3,30} = 1$ тј. $a_3[30]$. Дакле, догодила се промена $a_3[-29, 30]$.

10. $d_3 \leftarrow ((d_2 + F(a_3, b_2, c_2) + m_9) \lll 7)$

    Из претходног корака преноси се промена $a_3[-29, 30]$. Да би се анулирао њен утицај на регистар d_3, неопходно је да важе услови $b_{2,29} = c_{2,29}$ и $b_{2,30} = c_{2,30}$.

11. $c_3 \leftarrow ((c_2 + F(d_3, a_3, b_2) + m_{10}) \lll 11)$

    Из претходних корака преноси се промена $a_3[-29, 30]$. У овом кораку се уз услов $d_{3,29} = 1$ пушта утицај промене $a_3[-29]$ на регистар c_3, док се условом $d_{3,30} = 0$ анулира утицај промене $a_3[30]$. Дакле, промена $a_3[-29]$, због померања за 11, изазива промену $c_3[-8]$ при чему мора важити услов $c_{3,8} = 1$.

12. $b_3 \leftarrow ((b_2 + F(c_3, d_3, a_3) + m_{11}) \lll 19)$

    Из претходних корака преносе се промене $a_3[-29, 30], c_3[-8]$. Уз услове $c_{3,29} = 1, c_{3,30} = 1, d_{3,8} = a_{3,8}$ респективно се анулира утицај наведених промена на регистар b_3.

13. $a_4 \leftarrow ((a_3 + F(b_3, c_3, d_3) + m_{12}) \lll 3)$

    Из претходних корака преносе се промене $a_3[-29, 30]$ и $c_3[-8]$.

Услов $b_{3,8} = 0$ анулира промену $c_3[-8]$.

Ако се постави услов $a_{4,32} = 0$, он сугерише да је пре промене $a_3[-29]$, односно уз стање $a_{3,29} = 1$ важило $F(b_{3,29}, c_{3,29}, d_{3,29}) + m_{12,29} = 1$ уз пренос јединице. Ово и даље важи, јер сви битови у претходном изразу нису промењени. Међутим, пошто се десила промена $a_3[-29]$, тј. када вредност бита $a_{3,29}$ падне на нулу, следи да се због $F(b_{3,29}, c_{3,29}, d_{3,29}) + m_{12,29} = 1$ дешава промена $a_4[32]$ и то без преноса јединице. То значи да уз промену $a_3[-29]$, нема преноса јединице који би се без те промене догодио, па промена $a_3[30]$ замењује утицај преноса јединице из претходног бита, тако да вредност бита $a_{4,1}$ остаје непромењена. Дакле, уз услове $b_{3,8} = 0$ и $a_{4,32} = 0$, излазна промена у овом кораку је $a_4[32]$. Важно је приметити да је погодно изабран услов, $(a_{4,32} = 0)$, две промене битова $(a_3[-29, 30])$ сузио у једну $(a_4[32])$.

14. $d_4 \leftarrow ((d_3 + F(a_4, b_3, c_3) + m_{13}) \lll 7)$

Из претходних корака преносе се промене $a_4[32]$ и $c_3[-8]$ које се респективно анулирају условима $b_{3,32} = c_{3,32}$ и $a_{4,8} = 1$.

15. $c_4 \leftarrow ((c_3 + F(d_4, a_4, b_3) + m_{14}) \lll 11)$

Из претходних корака преносе се промене $a_4[32]$ и $c_3[-8]$.

Уз услов $d_{4,32} = 0$ анулира се промена $a_4[32]$.

Промена $c_3[-8]$, уз померај 11, изазива промену $c_4[-19]$ ако пре тога важи услов $c_{4,19} = 1$.

16. $b_4 \leftarrow ((b_3 + F(c_4, d_4, a_4) + m_{15}) \lll 19)$

Из претходних корака преносе се промене $a_4[32]$ и $c_4[-19]$ које се респективно анулирају условима $c_{4,32} = 1$ и $d_{4,19} = a_{4,19}$.

17. $a_5 \leftarrow ((a_4 + G(b_4, c_4, d_4) + m_0 + 5A827999) \lll 3)$

Из претходних корака преносе се промене $a_4[32]$ и $c_4[-19]$. Ово је први корак у другој рунди тако да се функција F замењује функцијом G у којој промена једног бита не мења излаз уколико су преостала два бита истог знака. Промена $c_4[-19]$ анулира се условом $b_{4,19} = d_{4,19}$, док промена $a_4[32]$ даље изазива промену $a_5[3]$ уз услов $a_{5,3} = 0$.

18. $d_5 \leftarrow ((d_4 + G(a_5, b_4, c_4) + m_4 + 5A827999) \lll 5)$

Из претходних корака преносе се промене $a_5[3]$ и $c_4[-19]$.

Промена $c_4[-19]$ анулира се условом $a_{5,19} = b_{4,19}$.

Промена $a_5[3]$ уз померај 5, изазива промену $d_5[8]$ уз услов $d_{5,8} = 0$. Овде је важно нагласити својство функције G у којој промена једног бита мења излаз, уколико су преостала два бита супротног знака. У овом случају, промена $a_5[3]$ мења излаз

функције G под условом да је задовољено $b_{4,3} \neq c_{4,3}$ односно $b_{4,3} = c_{4,3} + 1$.

Такође, у овом кораку је присутна разлика у полазној поруци $\Delta m_4 = 2^{22}$, што уз померај 5 изазива промену $d_5[28]$ уз услов $d_{5,28} = 0$.

19. $c_5 \leftarrow ((c_4 + G(d_5, a_5, b_4) + m_8 + 5A827999) \lll 9)$

Из претходних корака преносе се промене $a_5[3], c_4[-19]$ и $d_5[8, 28]$.

Промена $c_4[-19]$, уз померај 9, изазива промену $c_5[-28]$ уз услов $c_{5,28} = 1$.

Промена $a_5[3]$ анулира се условом $d_{5,3} = b_{4,3}$.

Промене $d_5[8, 28]$ анулирају се редом условима $a_{5,8} = b_{4,8}$ и $a_{5,28} = b_{4,28}$.

20. $b_5 \leftarrow ((b_4 + G(c_5, d_5, a_5) + m_{12} + 5A827999) \lll 13)$

Из претходних корака преносе се промене $a_5[3], c_5[-28]$ и $d_5[8, 28]$.

Промене $c_5[-28]$ и $d_5[28]$ међусобно се анулирају.

Промена $a_5[3]$ анулира се условом $c_{5,3} = d_{5,3}$.

Промена $d_5[8]$ анулира се условом $c_{5,8} = a_{5,8}$.

21. $a_6 \leftarrow ((a_5 + G(b_5, c_5, d_5) + m_1 + 5A827999) \lll 3)$.

Из претходних корака преносе се промене $a_5[3], c_5[-28]$ и $d_5[8, 28]$.

Промене $c_5[-28]$ и $d_5[28]$ међусобно се анулирају.

Промена $a_5[3]$ изазива промену $a_6[6]$ уз услов $a_{6,6} = 0$.

Промена $d_5[8]$ анулира се условом $b_{5,8} = c_{5,8}$.

22. $d_6 \leftarrow ((d_5 + G(a_6, b_5, c_5) + m_5 + 5A827999) \lll 5)$

Из претходних корака преносе се промене $a_6[6], c_5[-28]$ и $d_5[8, 28]$.

Промене $c_5[-28]$ и $d_5[28]$ међусобно се анулирају ако промена $c_5[-28]$ мења излаз функције G, а то се дешава ако важи $a_{6,28} \neq b_{5,28}$, односно $a_{6,28} = b_{5,28} + 1$.

Промена $a_6[6]$ анулира се условом $b_{5,6} = c_{5,6}$.

Промена $d_5[8]$ уз померај 5 изазива промену $d_6[13]$ уз услов $d_{6,13} = 0$.

23. $c_6 \leftarrow ((c_5 + G(d_6, a_6, b_5) + m_9 + 5A827999) \lll 9)$

Из претходних корака преносе се промене $a_6[6], c_5[-28]$ и $d_6[13]$.

Промена $d_6[13]$ анулира се условом $a_{6,13} = b_{5,13}$.

Промена $a_6[6]$ анулира се условом $d_{6,6} = b_{5,6}$.

Уколико би важио услов $c_{6,5} = 1$, промена $c_5[-28]$ изазвала би промену $c_6[-5]$, односно промена једног бита изазвала би такође промену једног бита. Међутим, претпоставља се да у поруци почетно важе услови $c_{6,5} = 0$ и $c_{6,6} = 1$. Тада, да није дошло до промене $c_5[-28]$ важило би $c_{5,28} = 1$, па би из једнакости $c_{6,5} \leftarrow ((c_{5,28} + G(d_{6,28}, a_{6,28}, b_{5,28}) + m_{9,28} + (5A827999)_{28}) \ll 9)$ следило да је $G(d_{6,28}, a_{6,28}, b_{5,28}) + m_{9,28} + (5A827999)_{28} = 1$ уз пренос јединице. Али, пошто је дошло до промене $c_5[-28]$ тј. пошто важи $c_{5,28} = 0$ (и $G(d_{6,28}, a_{6,28}, b_{5,28}) + m_{9,28} + (5A827999)_{28} = 1$), следи промена $c_6[5]$, при чему нема преноса јединице. Пошто нема преноса јединице, $c_{6,6}$ пада на нулу ($c_6[-6]$), ако је пре тога важио услов $c_{6,6} = 1$. Дакле, уз услове $c_{6,5} = 0$ и $c_{6,6} = 1$, промена $c_5[-28]$ изазива промену $c_6[5, -6]$.

24. $b_6 \leftarrow ((b_5 + G(c_6, d_6, a_6) + m_{13} + 5A827999) \ll 13)$

Из претходних корака преносе се промене $a_6[6], c_6[5, -6]$ и $d_6[13]$.

Промене $a_6[6]$ и $c_6[-6]$ међусобно се анулирају.

Промене $c_6[5]$ и $d_6[13]$ анулирају се редом условома $d_{6,5} = a_{6,5}$ и $b_{6,13} = c_{6,13}$.

25. $a_7 \leftarrow ((a_6 + G(b_6, c_6, d_6) + m_2 + 5A827999) \ll 3)$.

Из претходних корака преносе се промене $a_6[6], c_6[5, -6]$ и $d_6[13]$.

Промене $a_6[6]$ и $c_6[-6]$ међусобно се анулирају уз услов $b_{6,6} \neq d_{6,6}$ тј. $b_{6,6} = d_{6,6} + 1$.

Промена $c_6[5]$ анулира се условом $b_{6,5} = d_{6,5}$.

Промена $d_6[13]$ анулира се условом $b_{6,13} = c_{6,13}$.

26. $d_7 \leftarrow ((d_6 + G(a_7, b_6, c_6) + m_6 + 5A827999) \ll 5)$

Из претходних корака преносе се промене $c_6[5, -6]$ и $d_6[13]$.

Промене $c_6[5, -6]$ анулира се редом условима $a_{7,5} = b_{6,5}$ и $a_{7,6} = b_{6,6}$.

Промена $d_6[13]$ уз померај 5 изазива промену $d_7[18]$ уз услов $d_{7,18} = 0$.

27. $c_7 \leftarrow ((c_6 + G(d_7, a_7, b_6) + m_{10} + 5A827999) \ll 9)$

Из претходних корака преносе се промене $c_6[5, -6]$ и $d_7[18]$.

Промена $d_7[18]$ анулира се условом $a_{7,18} = b_{6,18}$.

Ако се постави услов $c_{7,14} = 1$ и ако се претпостави да се није десила промена $c_6[5, -6]$ тј. да важи $c_{6,5} = 0$ и $c_{6,6} = 1$, из $c_{7,14} \leftarrow ((c_{6,5} + G(d_{7,5}, a_{7,5}, b_{6,5}) + m_{14,5} + (5A827999)_5) \ll 9)$ следи $G(d_{7,5}, a_{7,5}, b_{6,5}) + m_{14,5} + (5A827999)_5 = 1$. Ако се ипак узме у обзир промена $c_6[5, -6]$, бит $c_{7,14}$ пада на нулу (тј. $c_7[-14]$) уз пренос јединице чији се ефекат анулира јер се догодила промена $c_6[-6]$.

28. $b_7 \leftarrow ((b_6 + G(c_7, d_7, a_7) + m_{14} + 5A827999) \lll 13)$

    Из претходних корака преносе се промене $c_7[-14]$ и $d_7[18]$. Промене $c_7[-14]$ и $d_7[18]$ анулирају се редом условима $d_{7,14} = a_{7,14}$ и $c_{7,18} = a_{7,18}$.

29. $a_8 \leftarrow ((a_7 + G(b_7, c_7, d_7) + m_3 + 5A827999) \lll 3)$.

    Из претходних корака преносе се промене $c_7[-14]$ и $d_7[18]$. Промене $c_7[-14]$ и $d_7[18]$ анулирају се редом условима $b_{7,14} = d_{7,14}$ и $b_{7,18} = c_{7,18}$.

30. $d_8 \leftarrow ((d_7 + G(a_8, b_7, c_7) + m_7 + 5A827999) \lll 5)$

    Из претходних корака преносе се промене $c_7[-14]$ и $d_7[18]$.

    Промена $c_7[-14]$ анулира се условом $a_{8,14} = b_{7,14}$.

    Промена $d_7[18]$ уз померај 5 изазива промену $d_8[23]$ уз услов $d_{8,23} = 0$.

31. $c_8 \leftarrow ((c_7 + G(d_8, a_8, b_7) + m_{11} + 5A827999) \lll 9)$

    Из претходних корака преносе се промене $c_7[-14]$ и $d_8[23]$.

    Промена $d_8[23]$ анулира се условом $a_{8,23} = b_{7,23}$.

    Промена $c_7[-14]$ уз услов $c_{8,23} = 1$ изазива промену $c_8[-23]$.

32. $b_8 \leftarrow ((b_7 + G(c_8, d_8, a_8) + m_{15} + 5A827999) \lll 13)$

    Из претходних корака преносе се промене $c_8[-23]$ и $d_8[23]$ које се међусобно анулирају.

33. $a_9 \leftarrow ((a_8 + H(b_8, c_8, d_8) + m_0 + 6ED9EBA1) \lll 3)$.

    Из претходних корака преносе се промене $c_8[-23]$ и $d_8[23]$ које се међусобно анулирају.

34. $d_9 \leftarrow ((d_8 + H(a_9, b_8, c_8) + m_8 + 6ED9EBA1) \lll 9)$

    Из претходних корака преносе се промене $c_8[-23]$ и $d_8[23]$ које се међусобно анулирају уколико излаз из функције H са јединице пада на нулу, а то ће се десити ако важи $a_{9,23} = b_{8,23}$.

35. $c_9 \leftarrow ((c_8 + H(d_9, a_9, b_8) + m_4 + 6ED9EBA1) \lll 11)$

    Из претходног корака преноси се промена $c_8[-23]$. Како је у овом кораку присутна реч m_4 и како полазна промена поруке гласи $\Delta m_4 = 2^{22}$, следи да се промене 23. бита у речи m_4 и у регистру c_8 међусобно анулирају. Тиме су добијена 4 узастопна непромењена регистра b_8, a_9, d_9, c_9 (тј. $\Delta b_8 = 0, \Delta a_9 = 0, \Delta d_9 = 0, \Delta c_9 = 0$) после којих се не јавља промењена реч m_4, што све скупа води до колизије.

3.5.2 Анализа напада

Претходна анализа кретања промена битова кроз алгоритам показује да прва промена настаје у петом кораку ($a_2[26]$) под утицајем промене у четвртој речи (m_4). Даље је значајна промена настала под утицајем речи m_4 у кораку 18 друге рунде ($d_5[28]$). На крају, битна је промена која у кораку 35 треће рунде ($c_8[-23]$) коригује утицај речи m_4 и промена $d_8[23]$ која коригује утицај промене $c_8[-23]$ на претходне три рунде (32,33,34). Да би се пронашла колизија, потребно је пронаћи путању од почетне промене $a_2[26]$ до промена $c_8[-23]$ и $d_8[23]$. Узгред, треба водити рачуна да се одговарајућим условима анулира и промена друге рунде $d_5[28]$. Из анализе из претходног одељка види се да постоје следеће путање:

$$a_2[26] \xrightarrow{3} a_3[-29, 30] \xrightarrow{3} a_4[32] \xrightarrow{3} a_5[3] \xrightarrow{5} d_5[8] \xrightarrow{5} d_6[13] \xrightarrow{5} d_7[18] \xrightarrow{5} d_8[23].$$

$$a_2[26] \xrightarrow{3} a_3[-29] \xrightarrow{11} c_3[-8] \xrightarrow{11} c_4[-19] \xrightarrow{9} c_5[-28] \xrightarrow{9} c_6[5, -6] \xrightarrow{9} c_7[-14] \xrightarrow{9} c_8[-23].$$

Приказане путање добијене су уз одговарајућа проширења и сажимања (нпр. $a_2[26] \rightarrow a_3[-29, 30] \rightarrow a_4[32]$). Следећа табела илуструје кретање битова уз постављање услова у циљу добијања жељене путање. Колона *Разлика регистара* указује на почетну разлику регистара, док колона *Излаз поруке* M' упућује на промене у излазном регистру у зависности од затечених стања битова. На пример, ако у колони "Разлика регистара" у деветом кораку стоји 2^{28} при излазу поруке M' једнаком $a'_3 = a_3[-29, 30]$, следи да важи $a'_3 - a_3 = 2^{28}$. Слично, ако у поменутој колони у 23. кораку стоји -2^4 при излазу поруке M' једнаком $c'_6 = c_6[5, -6]$, следи да важи $c'_6 - c_6 = -2^4$.

Корак	Излаз поруке M	Реч	Померај	Δm_i	Разлика регистара	Излаз поруке M'	Довољни услови
1	a_1	m_0	3				
2	d_1	m_1	7				
3	c_1	m_2	11				
4	b_1	m_3	19				
5	a_2	m_4	3	2^{22}	2^{25}	$a_2[26]$	$a_{2,26} = 0$
6	d_2	m_5	7			d_2	$b_{1,26} = c_{1,26}$
7	c_2	m_6	11			c_2	$d_{2,26} = 0$
8	b_2	m_7	19			b_2	$c_{2,26} = 1$
9	a_3	m_8	3		2^{28}	$a_3[-29, 30]$	$a_{3,29} = 1$, $a_{3,30} = 0$

Корак	Излаз поруке M	Реч	Померај	Δm_i	Разлика регистара	Излаз поруке M'	Довољни услови
10	d_3	m_9	7			d_3	$b_{2,29} = c_{2,29}$, $b_{2,30} = c_{2,30}$
11	c_3	m_{10}	11		-2^7	$c_3[-8]$	$c_{3,8} = 1$, $d_{3,29} = 1$, $d_{3,30} = 0$
12	b_3	m_{11}	19			b_3	$c_{3,29} = 1$, $c_{3,30} = 1$, $d_{3,8} = a_{3,8}$
13	a_4	m_{12}	3		2^{31}	$a_4[32]$	$a_{4,32} = 0$, $b_{3,8} = 0$
14	d_4	m_{13}	7			d_4	$b_{3,32} = c_{3,32}$, $a_{4,8} = 1$
15	c_4	m_{14}	11		-2^{18}	$c_4[-19]$	$c_{4,19} = 1$, $d_{4,32} = 0$
16	b_4	m_{15}	19			b_4	$d_{4,19} = a_{4,19}$, $c_{4,32} = 1$
17	a_5	m_0	3		2^2	$a_5[3]$	$a_{5,3} = 0$, $b_{4,19} = d_{4,19}$
18	d_5	m_4	5	2^{22}	$2^7 + 2^{27}$	$d_5[8, 28]$	$d_{5,8} = 0$, $d_{5,28} = 0$, $b_{4,3} = c_{4,3} + 1$, $a_{5,19} = b_{4,19}$
19	c_5	m_8	9		-2^{27}	$c_5[-28]$	$c_{5,28} = 1$, $a_{5,8} = b_{4,8}$, $d_{5,3} = b_{4,3}$, $a_{5,28} = b_{4,28}$
20	b_5	m_{12}	13			b_5	$c_{5,3} = d_{5,3}$, $c_{5,8} = a_{5,8}$
21	a_6	m_1	3		2^5	$a_6[6]$	$a_{6,6} = 0$, $b_{5,8} = c_{5,8}$
22	d_6	m_5	5		2^{12}	$d_6[13]$	$d_{6,13} = 0, b_{5,6} = c_{5,6}$ $a_{6,8} = b_{5,8} + 1$
23	c_6	m_9	9		-2^4	$c_6[5, -6]$	$c_{6,5} = 0$, $c_{6,6} = 1$, $d_{6,6} = b_{5,6}$, $a_{6,13} = b_{5,13}$
24	b_6	m_{13}	13			b_6	$d_{6,5} = a_{6,5}$, $c_{6,13} = a_{6,13}$
25	a_7	m_2	3			a_7	$b_{6,5} = d_{6,5}$, $b_{6,13} = c_{6,13}$, $b_{6,6} = d_{6,6} + 1$
26	d_7	m_6	5		2^{17}	$d_7[18]$	$a_{7,5} = b_{6,5}$, $a_{7,6} = b_{6,6}$, $d_{7,18} = 0$
27	c_7	m_{10}	9		-2^{13}	$c_7[-14]$	$c_{7,14} = 1$, $a_{7,18} = b_{6,18}$
28	b_7	m_{14}	13			b_7	$d_{7,14} = a_{7,14}$, $c_{7,18} = a_{7,18}$
29	a_8	m_3	3			a_8	$b_{7,14} = d_{7,14}$, $b_{7,18} = c_{7,18}$
30	d_8	m_7	5		2^{22}	$d_8[23]$	$d_{8,23} = 0$, $a_{8,14} = b_{7,14}$
31	c_8	m_{11}	9		-2^{22}	$c_8[-23]$	$c_{8,23} = 1$, $a_{8,23} = b_{7,23}$
32	b_8	m_{15}	13			b_8	
33	a_9	m_0	3			a_9	
34	d_9	m_8	9			d_9	$a_{9,23} = b_{8,23}$
35	c_9	m_4	11	2^{22}		c_9	
36	b_9	m_{12}	15			b_9	

Табела 3.10. Колизиона путања - Ју

3.5.3 Вероватноћа налажења колизије

Уколико порука M задовољава свих 62 услова из претходне табеле, онда заједно са поруком $M' = M + \Delta M$ даје исти хеш $h(M)$, што значи да алгоритам MD4 није слабо отпоран на колизију. Тачније, за сваку почетну разлику поруке $\Delta m_4 = \pm 2^i, 0 \leq i \leq 31$, могуће је наћи, са 62 потребна услова, путању промена битова која води до колизије. Другим речима, ако у четвртој речи поруке M променимо појединачни бит на сва 32 места, било на нулу, било на јединицу, добијамо 64 поруке, тако да је за сваку од њих потребно генерисати 2^{62} порука да би са поруком M дошло до поклапања хеш вредности; вероватноћа да нека од њих задовољава потребна 62 услова једнака је $64/2^{62}$. Односно, за сваку поруку M, могуће је наћи $2 \cdot 2^5$ порука M' тако да ако су задовољена 62 услова, долази до поклапања њиховог хеша. Дакле, свака порука M је *слаба* са вероватноћом $2^{-62} \cdot 2^5 \cdot 2 = 2^{-56}$.

3.5.4 Модификација поруке

У циљу добијања колизије, неопходно је почетну поруку M_0 променити у слабу поруку M и то се постиже следећим техникама:

1. Проста модификација поруке

   Циљ основне модификације поруке је модификација задатих услова у првој рунди, што за сваки услов захтева корекцију једног бита поруке. На пример, корекција услова $a_{2,26} = 1$ на жељени услов $a_{2,26} = 0$ постиже се променом 23. бита речи m_4:

   $m_4 \leftarrow m_4 \oplus 2^{22}$, а следи из $a_2 = (a_1 + f(b_1, c_1, d_1) + m_4) \ll 3$.

   Уколико је 23. бит речи m_4 промењен са јединице на нулу, очигледно је да ће услов $a_{2,26} = 1$ прећи у $a_{2,26} = 0$. Али, ако је 23. бит речи m_4 промењен са нуле на јединицу, тада ће прелазак са $a_{2,26} = 1$ на $a_{2,26} = 0$ изазвати пренос јединице, што ће променити следећи 24. бит $a_{2,24}$. Ова се промена толерише, будући да је од интереса задовољити само назначених 62 услова у које не спада бит $a_{2,24}$. Треба међутим нагласити, да уколико се у истом регистру коригује два или више услова, онда је неопходно кориговати прво онај са битовима мање тежине. На пример, у кораку 9, прво се коригује услов $b_{2,29} = c_{2,29}$, па тек онда услов $b_{2,30} = c_{2,30}$. У супротном, корекција услова $b_{2,29} = c_{2,29}$ могла би због преноса јединице пореметити претходну корекцију услова $b_{2,30} = c_{2,30}$. Пошто се у првој рунди налази 24 услова, њихова корекција из полазне поруке M_0, ствара поруку M са Хеминговом удаљеношћу (у односу на M_0) у просеку једнаком 12 (очекује се

да је половина услова већ задовољена). Како је остало још 38 потенцијално незадовољених услова, M је слаба порука са вероватноћом 2^{-38}.

2. Сложена модификација поруке

   Сложена модификација поруке коригује услове друге рунде, при чему се не смеју пореметити услови прве рунде.

   (a) Корекција услова који се односе на $a_{5,3}, a_{5,8}, a_{5,19}, a_{5,28}$
   Из $a_5 = (a_4 + G(b_4, c_4, d_4) + m_0 + 5A827999) \lll 3$ следи да се наведени услови могу кориговати узастопном променом речи m_0, m_1, m_2, m_3, m_4. На пример, да се испуни услов $a_{5,3} = 0$, из $a_{5,3} = (a_{4,32} + G(b_{4,32}, c_{4,32}, d_{4,32}) + m_{0,32} + (5A827999)_{32}) \lll 3$ следи да је довољно променити бит $m_{0,32}$. Међутим, то утиче на промену регистра a_1 у првој рунди што може узроковати промене на осталим регистрима. Промена регистра a_1, уколико за тим има потребе, лако се коригује простом модификацијом у првој рунди, али је важно спречити евентуалне промене следећих регистара. Промена регистра a_1, делује у четири узастопна корака, али се компензује њен утицај, тако да регистар a_2 остаје непромењен. Уз одговарајућу промену речи m_1, m_2, m_3, m_4, регистри d_1, c_1, b_1, a_2 остају непромењени.

Корак	Померај	Промењена реч m_i	Стање регистара
1	3	$m_0 \leftarrow m_0 \pm 2^{31}$	$a_1^{novo} = a_1[\mp 3]$, b_0, c_0, d_0
2	7	$m_1 \leftarrow (d_1 \ggg 7) - d_0 - F(a_1^{novo}, b_0, c_0)$	$d_1, a_1^{novo}, b_0, c_0$
3	11	$m_2 \leftarrow (c_1 \ggg 11) - c_0 - F(d_1, a_1^{novo}, b_0)$	$c_1, d_1, a_1^{novo}, b_0$
4	19	$m_3 \leftarrow (b_1 \ggg 19) - b_0 - F(c_1, d_1, a_1^{novo})$	$b_1, c_1, d_1, a_1^{novo}$
5	3	$m_4 \leftarrow (a_2 \ggg 3) - a_1^{novo} - F(b_1, c_1, d_1)$	a_2, b_1, c_1, d_1

Табела 3.11. Модификација бита $a_{5,3}$

У описаној модификацији поруке, испуњавање једног услова зависи од најмање пет битова поруке, па Хемингово растојање промењене поруке M у односу на полазну поруку M_0 расте за око три (јер је нека од промењених речи m_0, m_1, m_2, m_3, m_4 можда већ промењена простом модификацијом у првој рунди). Исти ефекат уз мање повећање Хемингового растојања, може се постићи применом алтернативног начина за испуњавање услова $a_{5,3} = 0$:
$b'_4 = b_4 \oplus 2^{31}$
$m_{15} = b'_4 \ggg 19 - b_3 - f(c_4, d_4, a_4)$
$a_5 = (a_4 + G(b'_4, c_4, d_4) + m_0 + 5A827999) \lll 3$
Ако је испуњен услов $c_{4,32} \neq d_{4,32}$, онда промена над битом $b_{4,32}$ мења излаз функције G, а самим тим и регистар

a_5. С друге стране, ако променимо регистар b_4, потребно је променити и одговарајућу реч m_{15}, да би се очувала једнакост у датом кораку.

(б) Корекција услова $d_{5,3}, d_{5,8}, d_{5,28}, c_{5,3}, c_{5,8}, b_{5,6}, b_{5,8}$

Корекција ових услова обавља се на сличан начин. На пример, уколико је потребно, постављање услова $c_{5,3} \neq d_{5,3}$ подразумева промену бита $c_{4,26}$:

$c_{5,3} \leftarrow ((c_{4,26} + G(d_{5,26}, a_{5,26}, b_{4,26}) + m_{8,26} + (5A827999)_{26}) \ll 9)$

Такође, промена регистра c_4 не сме утицати на наредне регистре b_4, a_5, d_5. Табела која следи показује како се мењањем речи m_{14} и m_{15} постиже жељено стање регистара c_4 и b_4, док се непроменљивост регистара a_5 и d_5 постиже одговарајућим условима $b_{4,26} = d_{4,26}$ и $a_{5,26} = b_{4,26}$. На овај начин, постиже се мања модификација поруке и самим тим мања Хемингова удаљеност између почетне и модификоване поруке.

Корак	Померај	Корекција промене	Стање регистара
15	11	$m_{14} \leftarrow c'_4 \gg 11 - c_3 - F(d_4, a_4, b_3)$	$c'_4 = c_4 \oplus 2^{25}, d_4, a_4, b_3$
16	19	$m_{15} \leftarrow b_4 \gg 19 - b_3 - F(c'_4, d_4, a_4)$	b_4, c'_4, d_4, a_4
17	3	$b_{4,26} = d_{4,26}$	a_5, b_4, c'_4, d_4
18	5	$a_{5,26} = b_{4,26}$	d_5, a_5, b_4, c'_4
19	9		c'_5, d_5, a_5, b_4

Табела 3.12. Модификација бита $c_{5,3}$

Потребно је напоменути да последње описаним начином није могуће задовољити услов $c_{5,28} = 1$. Из једнакости
$c_{5,28} \leftarrow ((c_{4,19} + G(d_{5,19}, a_{5,19}, b_{4,19}) + m_{8,19} + (5A827999)_{19} \ll 9)$
следила би следећа табела:

Корак	Померај	Корекција промене	Стање регистара
15	11	$m_{14} \leftarrow c'_4 \gg 11 - c_3 - F(d_4, a_4, b_3)$	$c'_4 = c_4 \oplus 2^{25}, d_4, a_4, b_3$
16	19	$m_{15} \leftarrow b_4 \gg 19 - b_3 - F(c'_4, d_4, a_4)$	b_4, c'_4, d_4, a_4
17	3	$b_{4,19} = d_{4,19}$	a_5, b_4, c'_4, d_4
18	5	$a_{5,19} = b_{4,19}$	d_5, a_5, b_4, c'_4
19	9		c'_5, d_5, a_5, b_4

Табела 3.13. Неуспели покушај модификације бита $c_{5,28}$

Пошто услови $c_{4,19} = 1, b_{4,19} = d_{4,19}, a_{5,19} = b_{4,19}$ спадају у фиксирана 62 услова која воде до колизије, немогуће је претходним поступком променити бит $c_{4,19}$ (ако је простом модификацијом задовољен услов $c_{4,19} = 1$, промена бита $c_{4,19}$ изазива његово нарушавање).

3. Претраживање битова

   Могуће је применити следећу технику: после корекције 24 услова у првој (простом модификацијом) и 11 услова у другој рунди (сложеном модификацијом), од почетна 62 услова, остаје потенцијално неиспуњено још 27 услова. У претходном одељку је показано, да у циљу смањења Хемингове удаљености између порука, сложену модификацију можемо унапредити променом само речи m_{14} и m_{15}. Парови ових двеју речи бирају се случајно. За сваки пар се онда обавља описана модификација, тако да сви услови друге рунде (осим $c_{5,28} = 1$, који се задовољава стандардном сложеном модификацијом) буду задовољени. Онда се проверава да ли је преосталих 27 услова задовољено. Ако јесте, добијена порука M је слаба.

3.6 Рачунарско тражење једне класе колизија

Алгоритам MD4 састоји се од 3 рунде, а свака рунда од 16 корака. У сваком кораку рачуна се један од регистара a, b, c, d. За представљање регистара користе се низови (нпр. x_i означава неки од регистара) при чему је индекс низа истовремено индекс (редни број) регистра. Како се сва четири регистра наизменично појављују, сваки од њих у свакој рунди доживљава четири промене, односно дванаест у целом алгоритму. То значи да се индекс сваког регистра налази у скупу $1, .., 12$. У случају да је индекс једнак нули, регистри типа x_0 сматрају се почетним константама. *Следбеником* неког регистра сматра се истоимени регистар који се рачуна у кораку увећаном за четири. *Претходником* неког регистра сматра се истоимени регистар који се рачуна у кораку умањеном за четири. За представљање појединих битова тридесетдвобитних регистара користе се матрице $x_{i,j}$, где је прва координата истовремено индекс регистра, а друга редни број бита. Дакле i припада скупу $1, ..., 12$, а j скупу $1, ..., 32$. Промена j-тог бита у i-тој речи поруке означава се са $m_{i,j}$, а сама промењена реч са m_i. Без смањења општости може се претпоставити да је бит $m_{i,j}$ промењен са нуле на јединицу. Алгоритам који следи, налази путању промене битова која води до колизије, под условом да се две поруке разликују у једном биту. Ово уједно представља оригинални допринос у овом раду.

3.6.1 Проширења промена над битовима

У складу са ознакама из радова ([1,5]), $x_{i,j}$ означава да је j-ти бит регистра са редним бројем i (x_i), промењен са нуле на јединицу. Под условом да је бит $x_{i+1,j+pomeraj}$ регистра следбеника x_{i+1} претходно имао вредност нула, под утицајем промене регистра претходника, промениће се на јединицу. Али, ако је његова претходна вредност била јединица, онда ће се променити на нулу, уз пренос јединице која опет

може изазвати даље промене. На тај начин, промена једног бита, у зависности од стања бита на који утиче, може изазвати промене једног или више битова. Важно је напоменути да се сва проширења, опет уз одговарајућа стања битова, могу вратити на промену једног бита. То значи да се проширења могу вршити по потреби, у тренутку када су потребна и потом сузити на промену само једног бита. Ово може повећати број почетних услова стања битова које треба задовољити, али то не утиче на добијање колизије. На пример, у кораку 9 путање коју је пронашла Ју [5], услед постављања услова $a_{3,29} = 1$ и $a_{3,30} = 0$ долази до проширења $a_2[26] \rightarrow a_3[-29, 30]$, а већ у кораку 13, услед услова $a_{4,32} = 0$, долази до сужавања $a_3[-29, 30] \rightarrow a_4[32]$.

3.6.2 Основна идеја

Како је сваки регистар функција од регистара из четири претходна корака, добијање колизије у алгоритму MD4 своди се на добијање низа од четири узастопна непромењена регистра a, b, c, d у трећој рунди. Још је неопходно да се после појаве поменута четири регистра у будућим корацима треће рунде не појављује промењена реч m_i. Следи да би природно било да у кораку у коме добијамо последњи од четири узастопна непромењена регистра истовремено анулирамо и утицај промењене речи m_i. Другим речима, ако је y_i поменути регистар, онда би његов претходник y_{i-1} могао да анулира утицај промењене речи m_i. Како се регистар y_{i-1} појављује и у претходна три корака, где такође може изазвати промене, у циљу корекције његовог утицаја потребно је наћи регистар z_{i-1} који се такође појављује у тим корацима. На тај начин, уз одговарајућу поставку битова у регистрима y_{i-1} и z_{i-1}, долазимо до жељене појаве четири узастопна непромењена регистра a, b, c, d (нулте разлике) у трећој рунди. Сада се основни задатак своди на проналажење путање од почетне промене регистра x_k у првој рунди до регистара y_{i-1} и z_{i-1}. На пример, у анализи путање коју је пронашла Ју [5], издвајамо четири кључна корака:

32. $b_8 \leftarrow ((b_7 + G(c_8, d_8, a_8) + m_{15} + 5A827999) \lll 13)$
33. $a_9 \leftarrow ((a_8 + H(b_8, c_8, d_8) + m_0 + 6ED9EBA1) \lll 3)$
34. $d_9 \leftarrow ((d_8 + H(a_9, b_8, c_8) + m_8 + 6ED9EBA1) \lll 9)$
35. $c_9 \leftarrow ((c_8 + H(d_9, a_9, b_8) + m_4 + 6ED9EBA1) \lll 11)$

Пожељан сценарио своди се на истовремено анулирање утицаја речи m_4 и добијање једнакости $\Delta b_8 = 0, \Delta a_9 = 0, \Delta d_9 = 0, \Delta c_9 = 0$. То се може постићи уколико регистар c_8 анулира утицај речи m_4 у кораку 35, док је истовремено потребно да неки други регистар анулира утицај c_8 у корацима 32,33 и 34. Једини регистар који се поред регистра c_8 појављује у тим корацима је регистар d_8 и његов бит промене треба бити такав да излаз функције H буде супротног знака од бита промене регистра c_8. Дакле, сада је извесно да бит $c_8[-23]$ анулира

утицај $\Delta m_4 = 2^{22}$, а да бит $d_8[23]$ (уз услов $a_{9,23} = b_{8,23}$) анулира утицај бита $c_8[-23]$. Остаје да се пронађе путања (низ промењених регистара) од почетне промене у првој рунди ($a_2[26]$) до битова $c_8[-23]$ и $d_8[23]$.

3.6.3 Алгоритам

Алгоритам који следи, за две поруке које се разликују у једном биту, аутоматски ствара путању промена битова од почетне промене у првој рунди до завршних промена у трећој рунди, после којих су сви наредни регистри две поруке исти, па следи колизија.
У циљу што јаснијег описа алгоритма, регистри ће бити представљени преко једне променљиве t. У складу с тим, функција компресије $f(T, M)$, где је $T = (t_{-3}, t_0, t_{-1}, t_{-2}) \in B^{128}$ и $M = (M_0, M_1, ..., M_{15}) \in B^{512}$ даје излаз $(t_{-3} + t_{45}, t_0 + t_{48}, t_{-1} + t_{47}, t_2 + t_{46})$ где је $t_i = (t_{i-4} + f_i(t_{i-1}, t_{i-2}, t_{i-3}) + M_{p(i)} + c_i) \ll s_i,\ i = 1, 2, ..., 48$. Сада је следбеник, односно претходник регистра t_i регистар t_{i+4}, односно регистар t_{i-4}.
Нека је промењена реч (m_i) и нека је у овиру ње промењен j-ти бит $(m_{i,j})$.

1. Улаз: пар (i, j).

2. Пронаћи корак n у трећој рунди у коме се налази промењена реч m_i. Пошто се у том кораку рачуна регистар t_n, из регистра претходника t_{n-4} издвојити бит $t_{n-4}[-j]$.

3. Забележити индекс $n - 4$ из корака 2 и то прогласити индексом крајњег регистра.

4. Пронаћи регистар који није t_{n-4}, а такође се јавља у сва три корака који претходе кораку n који је нађен у кораку 2. То је регистар t_{n-5}. Регистри t_{n-4} и t_{n-5} проглашавају се крајњим регистрима.

5. Пронаћи корак у првој рунди у коме се налази промењена реч m_i. То је корак $i + 1$ и у њему се рачуна регистар t_{i+1}. Издвојити бит $t_{i+1}[l]$ где је $l = (j + s_{i+1})\ mod\ 32$. Ако је $l\ mod\ 32 = 0$ нека $l \leftarrow 32$.

6. Забележити индекс $i + 1$ из корака 5 и то прогласити индексом почетног регистра.

7. Пронаћи све следбенике почетног регистра чиме се прави путања битова од индекса почетног до индекса крајњег регистра. Ова путања назива се *путања са почетка*.

8. Пронаћи све претходнике крајњих регистара чиме се праве две путање битова од индекса крајњих до индекса почетног регистра. Ове путање називају се *путањама са краја*.

9. Уколико за неко k из скупа $\{i + 1, ..., n - 4\}$ и за неки бит r важи $(t_k[r] = t_{k+1}[r+$ померај$])$ путања постоји и састоји се од дела путање

из корака 7 до индекса k и делова путања из корака 8 од индекса k. Мањи неискоришћен део путање са почетка назива се *слепа грана прве рунде*.

10. Уколико нису задовољени услови под 9, за неко k из корака 9 и за неки бит r врше се проширења типа $t_k[r] \to t_k[-r, r+1], t_k[r] \to t_k[-r, -(r+1), r+2]$....

11. Извршити корекцију слепе гране прве рунде (слепе гране путање са почетка) у кораку 9 коришћењем корака 10.

12. Извршити корекцију слепе гране настале под утицајем промењене речи m_i у другој рунди коришћењем корака 10.

13. Излаз: излаз из корака 9.

Следи псеудокôд описаног поступка (оператори $++$, односно $--$, представљају инкрементирање односно декрементирање):

$function\ prosiri(k, r, i)$
  $for(jj = 0; jj < i; jj++)\ dodaj\_u\_putanju\_sa\_pocetka(t_k[-(r+jj)])$
  $izbaci\_iz\_putanje\_sa\_pocetka(t_k[r+i-1])$
  $dodaj\_u\_putanju\_sa\_pocetka(t_k[r+i])$
$function\ suzi(k, r, i)$
  $for(jj = 0; jj < i; jj++)\ izbaci\_iz\_putanje\_sa\_pocetka(t_k[-(r+jj)])$
  $dodaj\_u\_putanju\_sa\_pocetka(t_k[r+i-1])$
  $izbaci\_iz\_putanje\_sa\_pocetka(t_k[r+i])$

улаз (i, j)
$if(i > 32)\ n \leftarrow w^{-1}(m_i)$
$y \leftarrow n - 4$
$z \leftarrow n - 5$
$x \leftarrow i + 1$

$for(pom = x, k = j, p = 0; pom \le y; pom = pom + 4)$
  $k \leftarrow k + s_{pom}$
  $putanja\_sa\_pocetka[p++] \leftarrow t_{pom}[k]$

$for(pom = y, k = j, k_1 = 0; pom \ge x; pom = pom - 4)$
  $k \leftarrow k - s_{pom}$
  $putanja\_sa\_kraja1[k_1++] \leftarrow t_{pom}[k]$

$for(pom = z, k = -j, k_2 = 0; pom \ge x; pom = pom - 4)$
  $k \leftarrow k - s_{pom}$
  $putanja\_sa\_kraja2[k_2++] \leftarrow t_{pom}[k]$

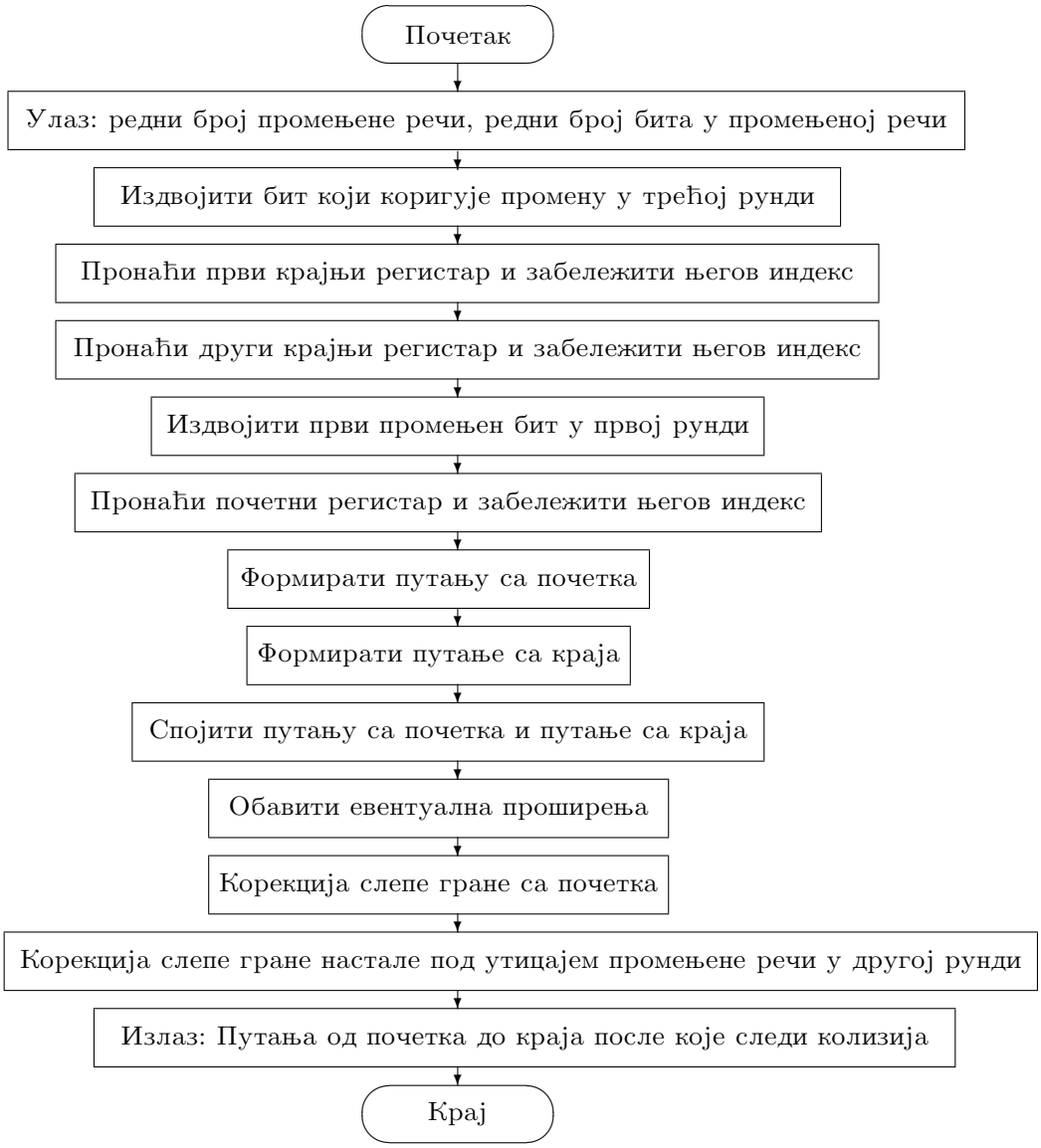
$pom \leftarrow x, m \leftarrow 0, q \leftarrow 0, dub \leftarrow 0, ind \leftarrow false$
$while(t\ in\ putanja\_sa\_kraja1\ and\ ind = false)$

$while(t\ in\ putanja\_sa\_pocetka)$
$for(r = 1; r \leq 32; r++)$
$if(t_k[r] = t_{k+1}[r + s_k])$
$ind \leftarrow true$
$while(pom \leq k)$
$spoj\_putanje1[m++] \leftarrow putanja\_sa\_pocetka[q++]$
$pom \leftarrow pom + 4$
$pom \leftarrow pom - 3$
$q \leftarrow q - 1$
$while(pom \leq y)$
$spoj\_putanje1[m++] \leftarrow putanja\_sa\_kraja1[q++]$
$pom \leftarrow pom + 4$
$if(ind = true)\ return\ spoj\_putanje1$
$else\ prosiri(k, r, dub++)$
$suzi(k + 4, r, --dub)$

$pom \leftarrow x, m \leftarrow 0, q \leftarrow 0, dub \leftarrow 0, ind \leftarrow false$
$while(t\ in\ putanja\_sa\_kraja2\ and\ ind = false)$
$while(t\ in\ putanja\_sa\_pocetka)$
$for(r = 1; r \leq 32; r++)$
$if(t_k[r] = t_{k+1}[r + s_k])$
$ind \leftarrow true$
$while(pom \leq k)$
$spoj\_putanje2[m++] \leftarrow putanja\_sa\_pocetka[q++]$
$pom \leftarrow pom + 4$
$pom \leftarrow pom - 3$
$q \leftarrow q - 1$
$while(pom \leq z)$
$spoj\_putanje2[m++] \leftarrow putanja\_sa\_kraja2[q++]$
$pom \leftarrow pom + 4$
$if(ind = true)\ return\ spoj\_putanje2$
$else\ prosiri(k, r, dub++)$
$suzi(k + 4, r, --dub)$

$korekcija\ slepe\ grane\ prve\ runde$
$korekcija\ slepe\ grane\ druge\ runde$

$izlaz : spoj\_putanje1, spoj\_putanje2$



Слика 3.3. Алгоритам налажења путање која води до колизије

Приликом тражења путање од почетног промењеног регистра t_{i+1} до крајњих регистара t_{n-4} и t_{n-5}, неопходно је наћи тренутак када путања са почетка прелази у путање са краја. На овај начин, почетна промена бита, уз одговарајуће услове, доводи до жељене промене битова у крајњим регистрима. То је тренутак када за неки бит r важи $t_k[r] = t_{k+1}[r+pomeraj]$. Уколико тражени тренутак преласка не постоји,

односно уколико ни за једно k не важе поменуте једнакости, неопходно је приступити проширењу промењених битова. Проширење може ићи у произвољну дубину, док се не добије бит који омогућава прелазак путање са почетка у путању са краја.

Да би се пронашла путања од почетног бита промене до битова који успостављају четири узастопна непромењена регистра у трећој рунди, путања с почетка у неком тренутку треба прећи у обе путање с краја. То значи да остатак путање с почетка после тренутка преласка у путање с краја није више битан, али се мора анулирати његов будући утицај. Краћи од два остатка приликом преласка путање са почетка у две путање са краја, назива се *слепа грана прве рунде* и неопходно је анулирати неки од чланова слепе гране, тако што ће неки њен члан постати део путање или ће доћи до потирања са супротним битом другог регистра. Анулирање се врши преко неких од чланова путање која је добијена спајањем путање с почетка и путања с краја. У циљу тога, чланови ове путање се могу проширивати, чиме се добија коначна путања. На пример, у анализи путање коју је дала Ју, слепу грану чине битови $a_6[6], a_7[9], a_8[12]$. Корекција ове слепе гране догодила се под утицајем бита $c_6[-6]$ који је анулирао утицај бита $a_6[6]$, а самим тим и утицај целе слепе гране.

3.6.4 Пример нове путање која води до колизије

Као улаз, алгоритам захтева два податка: индекс речи m_i која је промењена и редни број бита промене. Уколико изаберемо четврту реч и трећи бит, алгоритам избацује следеће информације:

Путања са почетка: $a_2[6] \xrightarrow{3} a_3[9] \xrightarrow{3} a_4[12] \xrightarrow{3} a_5[15] \xrightarrow{3} a_6[18] \xrightarrow{3} a_7[21] \xrightarrow{3} a_8[24]$.

Прва путања са краја: $d_8[3] \xrightarrow{7} d_7[30] \xrightarrow{7} d_6[25] \xrightarrow{5} d_5[20] \xrightarrow{5} d_4[15] \xrightarrow{5} d_3[8] \xrightarrow{5} d_2[1]$.

Слепа грана прве рунде: $a_6[18] \xrightarrow{3} a_7[21] \xrightarrow{3} a_8[24]$.

Друга путања са краја: $c_8[-3] \xrightarrow{9} c_7[-26] \xrightarrow{9} c_6[-17] \xrightarrow{9} c_5[-8] \xrightarrow{9} c_4[-31] \xrightarrow{11} c_3[-20] \xrightarrow{11} c_2[-9]$.

Спој путање са почетка и прве путање са краја: $a_2[6] \xrightarrow{3} a_3[9] \xrightarrow{3} a_4[12] \xrightarrow{3} a_5[15] \xrightarrow{5} d_5[20] \xrightarrow{5} d_6[25] \xrightarrow{5} d_7[30] \xrightarrow{5} d_8[3]$.

Да би се спојила путања са почетка и друга путања са краја потребно је извршити следеће проширење и сажимање: $a_2[6] \xrightarrow{3} a_3[-9, 10] \xrightarrow{3} a_4[12]$. На тај начин добијамо: $a_2[6] \xrightarrow{3} a_3[-9] \xrightarrow{11} c_3[-20] \xrightarrow{11} c_4[-31] \xrightarrow{9}$

$c_5[-8] \xrightarrow{9} c_6[-17] \xrightarrow{9} c_7[-26] \xrightarrow{9} c_8[-3]$.

Слепа грана друге рунде: $d_5[8] \xrightarrow{5} d_6[13] \xrightarrow{5} d_7[18] \xrightarrow{5} d_8[23]$.

Слепа грана друге рунде, односно њена почетна промена $d_5[8]$, анулира се променом $c_5[-8]$ која је део друге путање са краја.

Да би се анулирала слепа грана прве рунде, потребно је извршити проширење и сажимање $c_5[-8] \xrightarrow{9} c_6[17,-18] \xrightarrow{9} c_7[-26]$, чиме се промена $a_6[18]$ (почетак слепе гране) и промена $c_6[-18]$ међусобно анулирају. На тај начин добија се коначни спој путање са почетка са првом и другом путањом са краја:

$a_2[6] \xrightarrow{3} a_3[-9,10] \xrightarrow{3} a_4[12] \xrightarrow{3} a_5[15] \xrightarrow{5} d_5[20] \xrightarrow{5} d_6[25] \xrightarrow{5} d_7[30] \xrightarrow{5} d_8[3]$.

$a_2[6] \xrightarrow{3} a_3[-9] \xrightarrow{11} c_3[-20] \xrightarrow{11} c_4[-31] \xrightarrow{9} c_5[-8] \xrightarrow{9} c_6[17,-18] \xrightarrow{9} c_7[-26] \xrightarrow{9} c_8[-3]$.

Дакле, почетна разлика $m_{4,3}$ у првој рунди изазива промену $a_2[6]$. Та промена, уз одговарајуће услове, изазива промене $c_8[-3]$ и $d_8[3]$. Промена $c_8[-3]$ анулираће утицај промене $m_{4,3}$ у трећој рунди, док ће промена $d_8[3]$ анулирати утицај промене $c_8[-3]$ у корацима који претходе кораку треће рунде у коме се појављује реч $m_{4,3}$. Тиме тај корак представља последњи од четри узастопна, у којима регистри a, b, c, d остају непромењени, што доводи до колизије.

Следи оригинални излаз програма:

$---------- Putanja ----------$
$a[2][6]$ $a[3][9]$ $a[4][12]$ $a[5][15]$ $a[6][18]$ $a[7][21]$ $a[8][24]$
$---skraja-----$
$d[8][3]$ $d[7][30]$ $d[6][25]$ $d[5][20]$ $d[4][15]$ $d[3][8]$ $d[2][1]$

$a[2][6]$ $a[3][9]$ $a[4][12]$ $a[5][15]$ $d[5][20]$ $d[6][25]$ $d[7][30]$ $d[8][3]$

$---skraja-----$
$c[8][-3]$ $c[7][-26]$ $c[6][-17]$ $c[5][-8]$ $c[4][-31]$ $c[3][-20]$ $c[2][-9]$

$a[2][6]$ $a[3][9]$ *Potrebno je prosirenje*! $a[3][-9]$ $a[3][10]$ $c[3][-20]$
$c[4][-31]$ $c[5][-8]$ $c[6][-17]$ $c[7][-26]$ $c[8][-3]$

$----------slepaGrana----------$
$a[6][18]$ $a[7][21]$ $a[8][24]$

Promena u drugoj rundi : $d[5][8]$

$---$*slepa grana druge runde*$-----$
$d[5][8]$ $d[6][13]$ $d[7][18]$ $d[8][23]$

Korekcija slepe grane druge runde nije potrebna! $c[5][-8]$ $d[5][8]$

? $c[6][-17]$ *Korekcija slepe grane prve runde je potrebna*!

Putanja : $c[6][17]$ $c[8][-3]$ $c[7][-26]$ $c[6][-18]$ $c[5][-8]$ $c[4][-31]$ $c[3][-20]$ $a[3][-9]$ $d[8][3]$ $d[7][30]$ $d[6][25]$ $d[5][20]$ $a[5][15]$ $a[4][12]$ $a[3][10]$ $a[2][6]$

3.6.5 Анализа кретања битова кроз кораке алгоритма

За дату почетну разлику: $\Delta M = M^{'} - M$

где је $M = (m_0, m_1, ..., m_{15})$, а $\Delta M = (0, 0, 0, 0, 2^2, 0, ..., 0)$,

односно $\Delta m_4 = 2^2, \Delta m_i = 0, \forall i \neq 4$.

Сада се могу поставити довољни услови који воде до колизије:

1. $a_1 \leftarrow ((a_0 + F(b_0, c_0, d_0) + m_0) \lll 3)$

   Пошто је $\Delta m_0 = 0$, није дошло ни до какве промене, па је $a_1^{'} = a_1$

2. $d_1 \leftarrow ((d_0 + F(a_1, b_0, c_0) + m_1) \lll 7)$

   Пошто је $\Delta m_1 = 0$, није дошло ни до какве промене, па је $d_1^{'} = d_1$

3. $c_1 \leftarrow ((c_0 + F(d_1, a_1, b_0) + m_2) \lll 11)$

   Пошто је $\Delta m_2 = 0$, није дошло ни до какве промене, па је $c_1^{'} = c_1$

4. $b_1 \leftarrow ((b_0 + F(c_1, d_1, a_1) + m_3) \lll 19)$

   Пошто је $\Delta m_3 = 0$, није дошло ни до какве промене, па је $b_1^{'} = b_1$

5. $a_2 \leftarrow ((a_1 + F(b_1, c_1, d_1) + m_4) \lll 3)$

   Под утицајем промене 3. бита у речи m_4, због помераја за 3 дошло је до промене $a_2[6]$. Да би се ова промена могла догодити, неопходно је да важи услов $a_{2,6} = 0$.

6. $d_2 \leftarrow ((d_1 + F(a_2, b_1, c_1) + m_5) \lll 7)$

   Из претходног корака преноси се промена $a_2[6]$. Да би се анулирао њен утицај на регистар d_2, неопходно је да важи услов $b_{1,6} = c_{1,6}$.

7. $c_2 \leftarrow ((c_1 + F(d_2, a_2, b_1) + m_6) \lll 11)$

   Из претходних корака преноси се промена $a_2[6]$. Да би се анулирао њен утицај на регистар c_2, неопходно је да важи услов $d_{2,6} = 0$.

8. $b_2 \leftarrow ((b_1 + F(c_2, d_2, a_2) + m_7) \ll 19)$

   Из претходних корака преноси се промена $a_2[6]$. Да би се анулирао њен утицај на регистар b_2, неопходно је да важи услов $c_{2,6} = 1$.

9. $a_3 \leftarrow ((a_2 + F(b_2, c_2, d_2) + m_8) \ll 3)$

   Из претходних корака преноси се промена $a_2[6]$. Ако би у овом кораку важио услов $a_{3,9} = 0$ следила би (због $a_2[6]$) промена $a_3[9]$. Међутим, ако су испуњени услови $a_{3,9} = 1$ и $a_{3,10} = 0$, тада важи следеће: $a_{3,9} = 1 \Rightarrow a_{2,6} + f(b_{2,6}, c_{2,6}, d_{2,6}) + m_{8,6} = 1$. Пошто је пре било каквих промена важило $a_{2,6} = 0$, следи да важи $f(b_{2,6}, c_{2,6}, d_{2,6}) + m_{8,6} = 1$. Пошто је дошло до промене $a_2[6]$ (тј. $a_{2,6} = 1$), следи (због $f(b_{2,6}, c_{2,6}, d_{2,6}) + m_{8,6} = 1$) $a_{3,9} = 0$ (тј. $a_3[-9]$) уз пренос јединице. Пошто је полазна претпоставка да је испуњен услов $a_{3,10} = 0$, следи да ће због преноса јединице важити $a_{3,10} = 1$ тј. $a_3[10]$. Дакле, догодила се промена $a_3[-9, 10]$.

10. $d_3 \leftarrow ((d_2 + F(a_3, b_2, c_2) + m_9) \ll 7)$

    Из претходног корака преноси се промена $a_3[-9, 10]$. Да би се анулирао њен утицај на регистар d_3, неопходно је да важе услови $b_{2,9} = c_{2,9}$ и $b_{2,10} = c_{2,10}$.

11. $c_3 \leftarrow ((c_2 + F(d_3, a_3, b_2) + m_{10}) \ll 11)$

    Из претходних корака преноси се промена $a_3[-9, 10]$. У овом кораку се уз услов $d_{3,9} = 1$ пушта утицај промене $a_3[-9]$ на регистар c_3, док се условом $d_{3,10} = 0$ анулира утицај промене $a_3[10]$. Дакле, промена $a_3[-9]$, због помераја за 11, изазива промену $c_3[-20]$ при чему мора важити услов $c_{3,20} = 1$.

12. $b_3 \leftarrow ((b_2 + F(c_3, d_3, a_3) + m_{11}) \ll 19)$

    Из претходних корака преносе се промене $a_3[-9, 10], c_3[-20]$. Уз услове $c_{3,9} = 1, c_{3,10} = 1, d_{3,20} = a_{3,20}$ респективно се анулира утицај наведених промена на регистар b_3.

13. $a_4 \leftarrow ((a_3 + F(b_3, c_3, d_3) + m_{12}) \ll 3)$

    Из претходних корака преносе се промене $a_3[-9, 10]$ и $c_3[-20]$.

    Услов $b_{3,20} = 0$ анулира промену $c_3[-20]$.

    Ако се постави услов $a_{4,12} = 0$, он сугерише да је пре промене $a_3[-9]$, односно уз стање $a_{3,9} = 1$ важило $F(b_{3,9}, c_{3,9}, d_{3,9}) + m_{12,9} = 1$ уз пренос јединице. Ово и даље важи, јер сви битови у претходном изразу нису промењени. Међутим, пошто се десила промена $a_3[-9]$, тј. када вредност бита $a_{3,9}$ падне на нулу, следи да се због $F(b_{3,9}, c_{3,9}, d_{3,9}) + m_{12,9} = 1$ дешава промена $a_4[12]$ и то без преноса једнице. То значи да уз промену $a_3[-9]$, нема преноса јединице који би се без те промене догодио, па промена $a_3[10]$ замењује утицај преноса јединице из претходног бита, тако да

вредност бита $a_{4,13}$ остаје непромењена. Дакле, уз услове $b_{3,20} = 0$ и $a_{4,12} = 0$, излазна промена у овом кораку је $a_4[12]$. Важно је приметити да је погодно изабран услов, $(a_{4,12} = 0)$, две промене битова $(a_3[-9, 10])$ сузио у једну $(a_4[12])$.

14. $d_4 \leftarrow ((d_3 + F(a_4, b_3, c_3) + m_{13}) \lll 7)$

Из претходних корака преносе се промене $a_4[12]$ и $c_3[-20]$ које се респективно анулирају условима $b_{3,12} = c_{3,12}$ и $a_{4,20} = 1$.

15. $c_4 \leftarrow ((c_3 + F(d_4, a_4, b_3) + m_{14}) \lll 11)$

Из претходних корака преносе се промене $a_4[12]$ и $c_3[-20]$.

Уз услов $d_{4,12} = 0$ анулира се промена $a_4[12]$.

Промена $c_3[-20]$, уз померај 11, изазива промену $c_4[-31]$ ако пре тога важи услов $c_{4,31} = 1$.

16. $b_4 \leftarrow ((b_3 + F(c_4, d_4, a_4) + m_{15}) \lll 19)$

Из претходних корака преносе се промене $a_4[12]$ и $c_4[-31]$ које се респективно анулирају условима $c_{4,12} = 1$ и $d_{4,31} = a_{4,31}$.

17. $a_5 \leftarrow ((a_4 + G(b_4, c_4, d_4) + m_0 + 5A827999) \lll 3)$

Из претходних корака преносе се промене $a_4[12]$ и $c_4[-31]$. Ово је први корак у другој рунди тако да се функција F замењује функцијом G у којој промена једног бита не мења излаз уколико су преостала два бита истог знака. Промена $c_4[-31]$ анулира се условом $b_{4,31} = d_{4,31}$, док промена $a_4[12]$ даље изазива промену $a_5[15]$ уз услов $a_{5,15} = 0$.

18. $d_5 \leftarrow ((d_4 + G(a_5, b_4, c_4) + m_4 + 5A827999) \lll 5)$

Из претходних корака преносе се промене $a_5[15]$ и $c_4[-31]$.

Промена $c_4[-31]$ анулира се условом $a_{5,31} = b_{4,31}$.

Промена $a_5[15]$ уз померај 5, изазива промену $d_5[20]$ уз услов $d_{5,20} = 0$. Овде је важно нагласити свосство функције G у којој промена једног бита мења излаз, уколико су преостала два бита супротног знака. У овом случају, промена $a_5[15]$ мења излаз функције G под условом да је задовољено $b_{4,15} \neq c_{4,15}$ односно $b_{4,15} = c_{4,15} + 1$.

Такође, у овом кораку је присутна разлика у полазној поруци $\Delta m_4 = 2^2$, што уз померај 5 изазива промену $d_5[8]$ уз услов $d_{5,8} = 0$.

19. $c_5 \leftarrow ((c_4 + G(d_5, a_5, b_4) + m_8 + 5A827999) \lll 9)$

Из претходних корака преносе се промене $a_5[15], c_4[-31]$ и $d_5[8, 20]$.

Промена $c_4[-31]$, уз померај 9, изазива промену $c_5[-8]$ уз услов $c_{5,8} = 1$.

Промена $a_5[15]$ анулира се условом $d_{5,15} = b_{4,15}$.

Промене $d_5[8, 20]$ анулирају се редом условима $a_{5,8} = b_{4,8}$ и $a_{5,20} = b_{4,20}$.

20. $b_5 \leftarrow ((b_4 + G(c_5, d_5, a_5) + m_{12} + 5A827999) \ll 13)$

Из претходних корака преносе се промене $a_5[15], c_5[-8]$ и $d_5[8, 20]$.

Промене $c_5[-8]$ и $d_5[8]$ међусобно се анулирају.

Промена $a_5[15]$ анулира се условом $c_{5,15} = d_{5,15}$.

Промена $d_5[20]$ анулира се условом $c_{5,20} = a_{5,20}$.

21. $a_6 \leftarrow ((a_5 + G(b_5, c_5, d_5) + m_1 + 5A827999) \ll 3)$.

Из претходних корака преносе се промене $a_5[15], c_5[-8]$ и $d_5[8, 20]$.

Промене $c_5[-8]$ и $d_5[8]$ међусобно се анулирају.

Промена $a_5[15]$ изазива промену $a_6[18]$ уз услов $a_{6,18} = 0$.

Промена $d_5[20]$ анулира се условом $b_{5,20} = c_{5,20}$.

22. $d_6 \leftarrow ((d_5 + G(a_6, b_5, c_5) + m_5 + 5A827999) \ll 5)$

Из претходних корака преносе се промене $a_6[18], c_5[-8]$ и $d_5[8, 20]$.

Промене $c_5[-8]$ и $d_5[8]$ међусобно се анулирају ако промена $c_5[-8]$ мења излаз функције G, а то се дешава ако важи $a_{6,8} \neq b_{5,8}$, односно $a_{6,8} = b_{5,8} + 1$.

Промена $a_6[18]$ анулира се условом $b_{5,18} = c_{5,18}$.

Промена $d_5[20]$ уз померај 5 изазива промену $d_6[25]$ уз услов $d_{6,25} = 0$.

23. $c_6 \leftarrow ((c_5 + G(d_6, a_6, b_5) + m_9 + 5A827999) \ll 9)$

Из претходних корака преносе се промене $a_6[18], c_5[-8]$ и $d_6[25]$.

Промена $d_6[25]$ анулира се условом $a_{6,25} = b_{5,25}$.

Промена $a_6[18]$ анулира се условом $d_{6,18} = b_{5,18}$.

Уколико би важио услов $c_{6,17} = 1$, промена $c_5[-8]$ изазвала би промену $c_6[-17]$, односно промена једног бита изазвала би такође промену једног бита. Међутим, претпоставља се да у поруци почетно важе услови $c_{6,17} = 0$ и $c_{6,18} = 1$. Тада, да није дошло до промене $c_5[-8]$ важило би $c_{5,8} = 1$, па би из једнакости $c_{6,17} \leftarrow ((c_{5,8} + G(d_{6,8}, a_{6,8}, b_{5,8}) + m_{9,8} + (5A827999)_8) \ll 9)$ следило да је $G(d_{6,8}, a_{6,8}b_{5,8}) + m_{9,8} + (5A827999)_8 = 1$ уз пренос јединице. Али, пошто је дошло до промене $c_5[-8]$ тј. пошто важи $c_{5,8} = 0$ (и $G(d_{6,8}, a_{6,8}b_{5,8}) + m_{9,8} + (5A827999)_8 = 1$), следи промена $c_6[17]$, при чему нема преноса јединице. Пошто нема преноса јединице, $c_{6,18}$ пада на нулу ($c_6[-18]$), ако је пре тога важио услов $c_{6,18} = 1$. Дакле, уз услове $c_{6,17} = 0$ и $c_{6,18} = 1$, промена $c_5[-8]$ изазива промену $c_6[17, -18]$.

24. $b_6 \leftarrow ((b_5 + G(c_6, d_6, a_6) + m_{13} + 5A827999) \lll 13)$

Из претходних корака преносе се промене $a_6[18], c_6[17, -18]$ и $d_6[25]$.

Промене $a_6[18]$ и $c_6[-18]$ међусобно се анулирају.

Промене $c_6[17]$ и $d_6[25]$ анулирају се редом условома $d_{6,17} = a_{6,17}$ и $b_{6,25} = c_{6,25}$.

25. $a_7 \leftarrow ((a_6 + G(b_6, c_6, d_6) + m_2 + 5A827999) \lll 3)$.

Из претходних корака преносе се промене $a_6[18], c_6[17, -18]$ и $d_6[25]$.

Промене $a_6[18]$ и $c_6[-18]$ међусобно се анулирају уз услов $b_{6,18} \neq d_{6,18}$ тј. $b_{6,18} = d_{6,18} + 1$.

Промена $c_6[17]$ анулира се условом $b_{6,17} = d_{6,17}$.

Промена $d_6[25]$ анулира се условом $b_{6,25} = c_{6,25}$.

26. $d_7 \leftarrow ((d_6 + G(a_7, b_6, c_6) + m_6 + 5A827999) \lll 5)$

Из претходних корака преносе се промене $c_6[17, -18]$ и $d_6[25]$.

Промене $c_6[17, -18]$ анулирају се редом условима $a_{7,17} = b_{6,17}$ и $a_{7,18} = b_{6,18}$.

Промена $d_6[25]$ уз померај 5 изазива промену $d_7[30]$ уз услов $d_{7,30} = 0$.

27. $c_7 \leftarrow ((c_6 + G(d_7, a_7, b_6) + m_{10} + 5A827999) \lll 9)$

Из претходних корака преносе се промене $c_6[17, -18]$ и $d_7[30]$.

Промена $d_7[30]$ анулира се условом $a_{7,30} = b_{6,30}$.

Ако се постави услов $c_{7,26} = 1$ и ако се претпостави да се није десила промена $c_6[17, -18]$ тј. да важи $c_{6,17} = 0$ и $c_{6,18} = 1$, из $c_{7,26} = ((c_{6,17} + G(d_{7,17}, a_{7,17}, b_{6,17}) + m_{14,17} + (5A827999)_{17}) \lll 9)$ следи $G(d_{7,17}, a_{7,17}, b_{6,17}) + m_{14,17} + (5A827999)_{17} = 1$. Ако се ипак узме у обзир промена $c_6[17, -18]$, бит $c_{7,26}$ пада на нулу (тј. $c_7[-26]$) уз пренос јединице чији се ефекат анулира јер се догодила промена $c_6[-18]$.

28. $b_7 \leftarrow ((b_6 + G(c_7, d_7, a_7) + m_{14} + 5A827999) \lll 13)$

Из претходних корака преносе се промене $c_7[-26]$ и $d_7[30]$. Промене $c_7[-26]$ и $d_7[30]$ анулирају се редом условима $d_{7,26} = a_{7,26}$ и $c_{7,30} = a_{7,30}$.

29. $a_8 \leftarrow ((a_7 + G(b_7, c_7, d_7) + m_3 + 5A827999) \lll 3)$.

Из претходних корака преносе се промене $c_7[-26]$ и $d_7[30]$. Промене $c_7[-26]$ и $d_7[30]$ анулирају се редом условима $b_{7,26} = d_{7,26}$ и $b_{7,30} = c_{7,30}$.

30. $d_8 \leftarrow ((d_7 + G(a_8, b_7, c_7) + m_7 + 5A827999) \ll 5)$

Из претходних корака преносе се промене $c_7[-26]$ и $d_7[30]$.

Промена $c_7[-26]$ анулира се условом $a_{8,26} = b_{7,26}$.

Промена $d_7[30]$ уз померај 5 изазива промену $d_8[3]$ уз услов $d_{8,3} = 0$.

31. $c_8 \leftarrow ((c_7 + G(d_8, a_8, b_7) + m_{11} + 5A827999) \ll 9)$

Из претходних корака преносе се промене $c_7[-26]$ и $d_8[3]$.

Промена $d_8[3]$ анулира се условом $a_{8,3} = b_{7,3}$.

Промена $c_7[-26]$ уз услов $c_{8,3} = 1$ изазива промену $c_8[-3]$.

32. $b_8 \leftarrow ((b_7 + G(c_8, d_8, a_8) + m_{15} + 5A827999) \ll 13)$

Из претходних корака преносе се промене $c_8[-3]$ и $d_8[3]$ које се међусобно анулирају.

33. $a_9 \leftarrow ((a_8 + H(b_8, c_8, d_8) + m_0 + 6ED9EBA1) \ll 3)$.

Из претходних корака преносе се промене $c_8[-3]$ и $d_8[3]$ које се међусобно анулирају.

34. $d_9 \leftarrow ((d_8 + H(a_9, b_8, c_8) + m_8 + 6ED9EBA1) \ll 9)$

Из претходних корака преносе се промене $c_8[-3]$ и $d_8[3]$ које се међусобно анулирају уколико излаз из функције H са јединице пада на нулу, а то ће се десити ако важи $a_{9,3} = b_{8,3}$.

35. $c_9 \leftarrow ((c_8 + H(d_9, a_9, b_8) + m_4 + 6ED9EBA1) \ll 11)$

Из претходног корака преноси се промена $c_8[-3]$. Како је у овом кораку присутна реч m_4 и како полазна промена поруке гласи $\Delta m_4 = 2^2$, следи да се промене 3. бита у речи m_4 и у регистру c_8 међусобно анулирају. Тиме су добијена 4 узастопна непромењена регистра b_8, a_9, d_9, c_9 (тј. $\Delta b_8 = 0, \Delta a_9 = 0, \Delta d_9 = 0, \Delta c_9 = 0$) после којих се не јавља промењена реч m_4, што све скупа води до колизије.

На основу претходне анализе добија се следећа таблица услова који воде до колизије:

Корак	Излаз поруке M	Реч	Померај	Δm_i	Разлика регистара	Излаз поруке M'	Довољни услови
1	a_1	m_0	3			a_1	
2	d_1	m_1	7			d_1	
3	c_1	m_2	11			c_1	
4	b_1	m_3	19			b_1	
5	a_2	m_4	3	2^2	2^5	$a_2[6]$	$a_{2,6} = 0$
6	d_2	m_5	7			d_2	$b_{1,6} = c_{1,6}$
7	c_2	m_6	11			c_2	$d_{2,6} = 0$
8	b_2	m_7	19			b_2	$c_{2,6} = 1$
9	a_3	m_8	3		2^8	$a_3[-9, 10]$	$a_{3,9} = 1$, $a_{3,10} = 0$
10	d_3	m_9	7			d_3	$b_{2,9} = c_{2,9}$, $b_{2,10} = c_{2,10}$
11	c_3	m_{10}	11		-2^{19}	$c_3[-20]$	$c_{3,20} = 1$, $d_{3,9} = 1$, $d_{3,10} = 0$
12	b_3	m_{11}	19			b_3	$c_{3,9} = 1$, $c_{3,10} = 1$, $d_{3,20} = a_{3,20}$
13	a_4	m_{12}	3		2^{11}	$a_4[12]$	$a_{4,12} = 0$, $b_{3,20} = 0$
14	d_4	m_{13}	7			d_4	$b_{3,12} = c_{3,12}$, $a_{4,20} = 1$
15	c_4	m_{14}	11		-2^{30}	$c_4[-31]$	$c_{4,31} = 1$, $d_{4,12} = 0$
16	b_4	m_{15}	19			b_4	$d_{4,31} = a_{4,31}$, $c_{4,12} = 1$
17	a_5	m_0	3		2^{14}	$a_5[15]$	$a_{5,15} = 0$, $b_{4,31} = d_{4,31}$
18	d_5	m_4	5	2^2	$2^7 + 2^{19}$	$d_5[8, 20]$	$d_{5,8} = 0$, $b_{4,15} = c_{4,15} + 1$, $d_{5,20} = 0$, $a_{5,31} = b_{4,31}$
19	c_5	m_8	9		-2^7	$c_5[-8]$	$c_{5,8} = 1$, $a_{5,8} = b_{4,8}$, $d_{5,15} = b_{4,15}$, $a_{5,20} = b_{4,20}$
20	b_5	m_{12}	13			b_5	$c_{5,15} = d_{5,15}$, $c_{5,20} = a_{5,20}$
21	a_6	m_1	3		2^{17}	$a_6[18]$	$a_{6,18} = 0$, $b_{5,20} = c_{5,20}$
22	d_6	m_5	5		2^{24}	$d_6[25]$	$d_{6,25} = 0$, $b_{5,18} = c_{5,18}$, $a_{6,8} = b_{5,8} + 1$
23	c_6	m_9	9		-2^{16}	$c_6[17, -18]$	$c_{6,17} = 0$, $c_{6,18} = 1$, $d_{6,18} = b_{5,18}$, $a_{6,25} = b_{5,25}$
24	b_6	m_{13}	13			b_6	$d_{6,17} = a_{6,17}$, $c_{6,25} = a_{6,25}$
25	a_7	m_2	3			a_7	$b_{6,17} = d_{6,17}$, $b_{6,25} = c_{6,25}$, $b_{6,18} = d_{6,18} + 1$
26	d_7	m_6	5		2^{29}	$d_7[30]$	$a_{7,17} = b_{6,17}$, $a_{7,18} = b_{6,18}$, $d_{7,30} = 0$

Корак	Излаз поруке M	Реч	Померај	Δm_i	Разлика регистара	Излаз поруке M'	Довољни услови
27	c_7	m_{10}	9		-2^{25}	$c_7[-26]$	$c_{7,26}=1$, $a_{7,30}=b_{6,30}$
28	b_7	m_{14}	13			b_7	$d_{7,26}=a_{7,26}$, $c_{7,30}=a_{7,30}$
29	a_8	m_3	3			a_8	$b_{7,26}=d_{7,26}$, $b_{7,30}=c_{7,30}$
30	d_8	m_7	5		2^2	$d_8[3]$	$d_{8,3}=0$, $a_{8,26}=b_{7,26}$
31	c_8	m_{11}	9		-2^2	$c_8[-3]$	$c_{8,3}=1$, $a_{8,3}=b_{7,3}$
32	b_8	m_{15}	13			b_8	
33	a_9	m_0	3			a_9	
34	d_9	m_8	9			d_9	$a_{9,3}=b_{8,3}$
35	c_9	m_4	11	2^2		c_9	

Табела 3.14. Диференцијални пут као излаз алгоритма

Корак	Регистар	Довољни услови
1	a_1	
2	d_1	
3	c_1	
4	b_1	$b_{1,6}=c_{1,6}$
5	a_2	$a_{2,6}=0$
6	d_2	$d_{2,6}=0$
7	c_2	$c_{2,6}=1$
8	b_2	$b_{2,9}=c_{2,9},\ b_{2,10}=c_{2,10}$
9	a_3	$a_{3,9}=1,\ a_{3,10}=0$
10	d_3	$d_{3,20}=a_{3,20},\ d_{3,9}=1,\ d_{3,10}=0$
11	c_3	$c_{3,9}=1,\ c_{3,10}=1,\ c_{3,20}=1$
12	b_3	$b_{3,20}=0,\ b_{3,12}=c_{3,12}$
13	a_4	$a_{4,12}=0,\ a_{4,20}=1$
14	d_4	$d_{4,12}=0,\ d_{4,31}=a_{4,31}$
15	c_4	$c_{4,12}=1,\ c_{4,31}=1$
16	b_4	$b_{4,31}=d_{4,31},\ b_{4,15}=c_{4,15}+1$
17	a_5	$a_{5,15}=0,\ a_{5,31}=b_{4,31},\ a_{5,8}=b_{4,8},\ a_{5,20}=b_{4,20}$
18	d_5	$d_{5,8}=0,\ d_{5,20}=0,\ d_{5,15}=b_{4,15}$
19	c_5	$c_{5,8}=1,\ c_{5,15}=d_{5,15},\ c_{5,20}=a_{5,20}$
20	b_5	$b_{5,20}=c_{5,20},\ b_{5,18}=c_{5,18}$

Корак	Регистар	Довољни услови
21	a_6	$a_{6,18} = 0,\ a_{6,8} = b_{5,8} + 1,\ a_{6,25} = b_{5,25}$
22	d_6	$d_{6,25} = 0,\ d_{6,18} = b_{5,18},\ d_{6,17} = a_{6,17}$
23	c_6	$c_{6,17} = 0,\ c_{6,18} = 1,\ c_{6,25} = a_{6,25}$
24	b_6	$b_{6,17} = d_{6,17},\ b_{6,25} = c_{6,25},\ b_{6,18} = d_{6,18} + 1$
25	a_7	$a_{7,17} = b_{6,17},\ a_{7,18} = b_{6,18},\ a_{7,30} = b_{6,30}$
26	d_7	$d_{7,30} = 0,\ d_{7,26} = a_{7,26}$
27	c_7	$c_{7,26} = 1,\ c_{7,30} = a_{7,30}$
28	b_7	$b_{7,26} = d_{7,26},\ b_{7,30} = c_{7,30}$
29	a_8	$a_{8,26} = b_{7,26},\ a_{8,3} = b_{7,3}$
30	d_8	$d_{8,3} = 0$
31	c_8	$c_{8,3} = 1$
32	b_8	
33	a_9	$a_{9,3} = b_{8,3}$
34	d_9	
35	c_9	

Табела 3.15. Скуп довољних услова који воде до колизије, гледано по регистрима алгоритма

3.6.6 Модификација поруке

Слично путањи коју је пронашла Jy, новодобијена путања даје 62 услова које је у циљу налажења поруке, потребно задовољити. Пожељно је да се услови који дотичу и прву и други рунду могу задовољтити простом модификацијом. Уколико то није могуће (рецимо у условима $d_{5,15} = b_{4,15}$ и $a_{5,31} = b_{4,31}$ није могуће променити битове $b_{4,15}$ и $b_{4,31}$ јер су они већ фиксирани претходним условима), тада је неопходно применити сложену модификацију. То значи да се простом модификацијом коригује 26, а сложеном 36 услова. После избора почетне поруке, потребно је установити који су услови већ задовољени и обавити модификацију поруке на остатку услова. На услове прве рунде примењује се проста модификација мењањем одговарајућег бита одговарајуће речи, док је на услове друге рунде неопходно применити сложену и специјално, напредну сложену модификацију.

3.6.7 Проста модификација поруке

Корекција бита $x_{i,j}$ прве рунде постиже се следећим корацима:

1. Наћи $x_i = (x_{i-4} + f_i(x_{i-1}, x_{i-2}, x_{i-3}) + m_i + \text{константа}) \ll \text{померај}$

2. За сваки услов на регистру x_i променити j-ти бит променом речи m_i (односно бита $m_{i,j}$). Новодобијено стање регистра означава се са a_i^{novo}.

3. Кориговати нову вредност речи m_i (тако да задовољава једнакост у тренутном кораку) преко:
$m_i^{novo} = (a_i^{novo} \ll \text{померај}) - x_{i-4} - f_i(x_{i-1}, x_{i-2}, x_{i-3})-$ константа

Простом модификацијом добија се промењена порука $M^{novo} = (m_0^{novo}, ..., m_{15}^{novo})$ тако да пар $(M^{novo}, M^{novo} + \Delta M)$ чини колизију са вероватноћом 2^{-40}.

3.6.8 Сложена модификација

Проста модификација се не може применити у другој рунди, јер промена речи у другој рунди, такође значи промену те исте речи у првој рунди, што може довести до нарушавања услова прве рунде. У том циљу, промена у другој рунди треба да изазове интерну колизију у првој рунди која ће се неутралисати у пет корака, без утицаја на услове прве рунде. На пример, постављање услова $a_{5,15} = 0$, проистиче из једнакости $a_{5,15} \leftarrow ((a_{4,12} + G(b_{4,12}, c_{4,12}, d_{4,12}) + m_{0,12} + (5A827999)_{12}) \ll 3)$. Очигледно је да треба променити 12. бит речи m_0, што ће променити бит $a_{5,15}$. Ово даље може изазвати и промену рецимо бита $a_{5,16}$, али је то без значаја, пошто он не спада у услове које треба задовољити. Такође, све услове друге рунде који су можда већ задовољени, а могу бити нарушени под утицајем бита $a_{5,16}$, немају значаја у процесу модификације који тек предстоји.

Корак	Померај	Промењена реч m_i	Стање регистара
1	3	$m_0 \leftarrow m_0 \pm 2^{11}$	$a_1^{novo} = a_1[\mp 15], b_0, c_0, d_0$
2	7	$m_1 \leftarrow (d_1 \gg 7) - d_0 - F(a_1^{novo}, b_0, c_0)$	$d_1, a_1^{novo}, b_0, c_0$
3	11	$m_2 \leftarrow (c_1 \gg 11) - c_0 - F(d_1, a_1^{novo}, b_0)$	$c_1, d_1, a_1^{novo}, b_0$
4	19	$m_3 \leftarrow (b_1 \gg 19) - b_0 - F(c_1, d_1, a_1^{novo})$	$b_1, c_1, d_1, a_1^{novo}$
5	3	$m_4 \leftarrow (a_2 \gg 3) - a_1^{novo} - F(b_1, c_1, d_1)$	a_2, b_1, c_1, d_1

Табела 3.16. Модификација бита $a_{5,15}$ (постављење услова $a_{5,15} = 0$)

Овде је значајно приметити да је у циљу модификације бита $a_{5,15}$ промена 12. бита речи m_0 у првој рунди изазвала промену бита $a_{1,15}$. Управо вредност тог бита одређује додавање или одузимање јединице на 12. месту речи m_0. Уколико важи $a_{1,15} = 0$, следи $m_0 \leftarrow m_0 + 2^{11}$, а уколико важи $a_{1,15} = 1$, следи $m_0 \leftarrow m_0 - 2^{11}$. На овај начин, избегава се ефекат преноса јединице и коригује се вредност само једног бита $a_{1,15}$ који не спада у услове прве рунде.

Корак	Померај	Промењена реч m_i	Стање регистара
5	3	$m_4 \leftarrow m_4 + 2^{14}$	$a_2^{novo} = a_2[15], b_2, c_2, d_2$
6	7	$m_5 \leftarrow (d_2 \gg 7) - d_1 - F(a_2^{novo}, b_1, c_1)$	$d_1, a_2^{novo}, b_1, c_1$
7	11	$m_6 \leftarrow (c_2 \gg 11) - c_1 - F(d_1, a_2^{novo}, b_1)$	$c_2, d_2, a_2^{novo}, b_1$
8	19	$m_7 \leftarrow (b_2 \gg 19) - b_1 - F(c_2, d_2, a_2^{novo})$	$b_2, c_2, d_2, a_2^{novo}$
9	3	$m_8 \leftarrow (a_3 \gg 3) - a_2^{novo} - F(b_2, c_2, d_2)$	a_3, b_2, c_2, d_2

Табела 3.17. Модификација бита $d_{5,20}$ (постављење услова $d_{5,20} = 0$)

Пошто је затечено стање бита $a_{2,15} = 0$ следило је $m_4 \leftarrow m_4 + 2^{14}$. Међутим, претходним начином није могуће модификовати бит $d_{5,8}$.

Корак	Померај	Промењена реч m_i	Стање регистара
5	3	$m_4 \leftarrow m_4 + 2^2$	$a_2^{novo} = a_2[6], b_2, c_2, d_2$
6	7	$m_5 \leftarrow (d_2 \gg 7) - d_1 - F(a_2^{novo}, b_1, c_1)$	$d_1, a_2^{novo}, b_1, c_1$
7	11	$m_6 \leftarrow (c_2 \gg 11) - c_1 - F(d_1, a_2^{novo}, b_1)$	$c_2, d_2, a_2^{novo}, b_1$
8	19	$m_7 \leftarrow (b_2 \gg 19) - b_1 - F(c_2, d_2, a_2^{novo})$	$b_2, c_2, d_2, a_2^{novo}$
9	3	$m_8 \leftarrow (a_3 \gg 3) - a_2^{novo} - F(b_2, c_2, d_2)$	a_3, b_2, c_2, d_2

Табела 3.18. Модификација бита $d_{5,8}$ (постављење услова $d_{5,8} = 0$)

Пошто је затечено стање бита $a_{2,6} = 0$ следило је $m_4 \leftarrow m_4 + 2^2$. Међутим, поменути услов мора остати фиксиран, будући да спада у услове прве рунде. Зато се мора применити другачија форма сложене модификација, што се такође назива напредном сложеном модификацијом.

Корак	Померај	Додатни услов	Промењена реч m_i	Стање регистара
14	7	$d_{4,3} = 0$	$m_{13} \leftarrow m_{13} + 2^{27}$	$d_4^{novo} = d_4[3], a_4, b_3, c_3$
15	11	$a_{4,3} = b_{3,3}$		$c_4, d_4^{novo}, a_4, b_3$
16	19	$c_{4,3} = 0$		$b_4, c_4, d_4^{novo}, a_4$
17	3	$b_{4,3} = c_{4,3}$		$a_5, b_4, c_4, d_4^{novo}$
18	5			$d_5^{novo}, a_5, b_4, c_4$

Табела 3.19. Модификација бита $d_{5,8}$ (постављење услова $d_{5,8} = 0$)

3.6.9 Проналажење парова порука који чине колизију

Због једноставности коришћења просте модификације, пожељно је да већина услова које треба задовољити буду у првој рунди. Како се услови приближавају крају друге рунде, то их је теже задовољити, будући да се не смеју реметити већ задовољени услови друге рунде. Уз примену сложене модификације остало је незадовољено шест услова друге рунде. Алтернативни начин за њихово решавање је коришћење минималне грубе силе проналажењем $2^6 = 64$ порука које задовољавају све услове сем поменутих шест ($62 - 6 = 56$). До ових порука најједноставније је доћи променом једног, два или три бита у поруци која задовољава 56 услова и провером да ли тако промењена порука и даље испуњава претходно задовољене услове. Групе битова, чија истовремена промена не ремети претходно задовољене услове називају се *неутралним* битовима. Узастопном применом неутралних битова долазимо до порука које задовољавају неке или све од последњих шест

услова, чиме се добијају *слабе* поруке. Остаје још да се на сваку од њих примени почетна разлика на биту $m_{4,3}$. Нека је M_0 заједничка полазна порука, M добијена слаба порука, а M' слаба порука M на коју је додата почетна разлика. У даљем тексту следе неки примери колизија (у циљу лакшег уочавања разлика између порука, црвена боја означава промену у односу на претходну поруку, док плава боја указује на почетну разлику две поруке):

M_0	ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff ffffffff

M	ffffffff ffdfdfff 320fffff 8ffbffff ffffbffb 500fefee 0001feef ffa5ffbf 0000ffff 0007efbb 9fc07eff 0000fffe ffff7eff 00003fef fffffffe b7f5ffff
M'	ffffffff ffdfdfff 320fffff 8ffbffff ffffbfff 500fefee 0001feef ffa5ffbf 0000ffff 0007efbb 9fc07eff 0000fffe ffff7eff 00003fef fffffffe b7f5ffff
Хеш	3de338c2 b3348272 6947146e ff30ea46

M	ffffffff ffdfdfff 320fffff 8ffbfff7 ffffbffb 500fefee 0001feef ffe5ffbf 0000ffff 0007efbb 9fc07eff 0000fffe ffff7eff 00003fef fffffffe b7f5ffff
M'	ffffffff ffdfdfff 320fffff 8ffbfff7 ffffbfff 500fefee 0001feef ffe5ffbf 0000ffff 0007efbb 9fc07eff 0000fffe ffff7eff 00003fef fffffffe b7f5ffff
Хеш	63b150ed c1130c41 41622afe 7de5d07a

M	ffffffff ffdfdfff 320fffff 8ffbffff ffffbffb 500fefee 0001feef ffa5ffbd 0000ffff 0007efbb 9fc07eff 0010fffe ffff7eff 00003fef fffffffe b7f5ffff
M'	ffffffff ffdfdfff 320fffff 8ffbffff ffffbfff 500fefee 0001feef ffa5ffbd 0000ffff 0007efbb 9fc07eff 0010fffe ffff7eff 00003fef fffffffe b7f5ffff
Хеш	9e7eb99c 685e3b6b 19bfe669 e4a9b104

M	ffffffff ffdfdfff 320fffff 8ffbffff ffffbffb 500fefee 0001feef ffa5ffbf 0000ffff 0007efbb 9fc07eff 0001fffe ffff7eff 00003fef fffffffe b7f5fff7
M'	ffffffff ffdfdfff 320fffff 8ffbffff ffffbfff 500fefee 0001feef ffa5ffbf 0000ffff 0007efbb 9fc07eff 0001fffe ffff7eff 00003fef fffffffe b7f5fff7
Хеш	acd84ef1 0f841657 9e3b8ee5 d2a2df96

M	ffffffff ffdfdfff 320fffff 8ffbffff ffffbffb 500fefee 0001feef ffa5ffbf 0000ffff 0007efbb 9fc07eff 0080fffe ffff7eff 00003fef fffffffe b7f5fbff
M'	ffffffff ffdfdfff 320fffff 8ffbffff ffffbfff 500fefee 0001feef ffa5ffbf 0000ffff 0007efbb 9fc07eff 0080fffe ffff7eff 00003fef fffffffe b7f5fbff
Хеш	61dfad81 3443700f e229fb9a 06e4fc5c

M	ffffffff ffdfdfff 320fffff 9ffbfff7 ffffbffb 500fefee 00017eef ffe57fbf 0000ffff 0007efbb 9fc07eff 0000fffe ffff7eff 00003fef fffffffe b7f5ffff
M'	ffffffff ffdfdfff 320fffff 9ffbfff7 ffffbfff 500fefee 00017eef ffe57fbf 0000ffff 0007efbb 9fc07eff 0000fffe ffff7eff 00003fef fffffffe b7f5ffff
Хеш	b7a9fb9b 7df3cd83 b01b9fe3 3bfbd53c

M	$fffffffff$ $ffdfdfff$ $320ffffff$ $9ffbfff7$ $ffffbffb$ $500fefee$ $00017eef$ $ffe57fbd$ $0000ffff$ $0007efbb$ $9fc07eff$ $0010fffe$ $ffff7eff$ $00003fef$ $ffffffff$ $b7f5ffff$
M'	$fffffffff$ $ffdfdfff$ $320ffffff$ $9ffbfff7$ $ffffbfff$ $500fefee$ $00017eef$ $ffe57fbd$ $0000ffff$ $0007efbb$ $9fc07eff$ $0010fffe$ $ffff7eff$ $00003fef$ $ffffffff$ $b7f5ffff$
Хеш	$b205a235$ $e6becea3$ $c9486174$ $eba09f4c$

M	$fffffffff$ $ffdfdfff$ $320ffffff$ $9ffbfff7$ $ffffbffb$ $500fefee$ $00017eef$ $ffe57fbf$ $0000ffff$ $0007efbb$ $9fc07eff$ $0001fffe$ $ffff7eff$ $00003fef$ $ffffffff$ $b7f5fff7$
M'	$fffffffff$ $ffdfdfff$ $320ffffff$ $9ffbfff7$ $ffffbfff$ $500fefee$ $00017eef$ $ffe57fbf$ $0000ffff$ $0007efbb$ $9fc07eff$ $0001fffe$ $ffff7eff$ $00003fef$ $ffffffff$ $b7f5fff7$
Хеш	$3b500f94$ $61517e1f$ $758b85de$ $1102b04a$

M	$fffffffff$ $ffdfdfff$ $320ffffff$ $9ffbfff7$ $ffffbffb$ $500fefee$ $00017eef$ $ffe57fbf$ $0000ffff$ $0007efbb$ $9fc07eff$ $0080fffe$ $ffff7eff$ $00003fef$ $ffffffff$ $b7f5fbff$
M'	$fffffffff$ $ffdfdfff$ $320ffffff$ $9ffbfff7$ $ffffbfff$ $500fefee$ $00017eef$ $ffe57fbf$ $0000ffff$ $0007efbb$ $9fc07eff$ $0080fffe$ $ffff7eff$ $00003fef$ $ffffffff$ $b7f5fbff$
Хеш	$1ba2c88e$ $b1fc97f2$ $fa850b05$ $aa20d95f$

M	$fffffffff$ $ffdfdfff$ $320ffffff$ $8ffbfff7$ $ffffbffb$ $500fefee$ $0001feef$ $ffe5ffbf$ $0000ffff$ $0007efbb$ $9fc07eff$ $0080fffe$ $ffff7eff$ $00003fef$ $ffffffff$ $b7f5fbff$
M'	$fffffffff$ $ffdfdfff$ $320ffffff$ $8ffbfff7$ $ffffbfff$ $500fefee$ $0001feef$ $ffe5ffbf$ $0000ffff$ $0007efbb$ $9fc07eff$ $0080fffe$ $ffff7eff$ $00003fef$ $ffffffff$ $b7f5fbff$
Хеш	$f5f8bcba$ $b5afbdc0$ $9b0593e4$ $43086ba3$

M	$fffffffff$ $ffdfdfff$ $320ffffff$ $8ffbffff$ $ffffbffb$ $500fefee$ $0001feef$ $ffa5ffbd$ $0000ffff$ $0007efbb$ $9fc07eff$ $0090fffe$ $ffff7eff$ $00003fef$ $ffffffff$ $b7f5fbff$
M'	$fffffffff$ $ffdfdfff$ $320ffffff$ $8ffbffff$ $ffffbfff$ $500fefee$ $0001feef$ $ffa5ffbd$ $0000ffff$ $0007efbb$ $9fc07eff$ $0090fffe$ $ffff7eff$ $00003fef$ $ffffffff$ $b7f5fbff$
Хеш	$eb1df0ba$ $19f9bf88$ $b4416914$ $63786ee0$

M	$fffffffff$ $ffdfdfff$ $320ffffff$ $8ffbffff$ $ffffbffb$ $500fefee$ $0001feef$ $ffa5ffbf$ $0000ffff$ $0007efbb$ $9fc07eff$ $0081fffe$ $ffff7eff$ $00003fef$ $ffffffff$ $b7f5fbf7$
M'	$fffffffff$ $ffdfdfff$ $320ffffff$ $8ffbffff$ $ffffbfff$ $500fefee$ $0001feef$ $ffa5ffbf$ $0000ffff$ $0007efbb$ $9fc07eff$ $0081fffe$ $ffff7eff$ $00003fef$ $ffffffff$ $b7f5fbf7$
Хеш	$d2e284e2$ $853ee2e1$ $f5933f0f$ $f0102ab5$

M	$fffffffff$ $ffdfdfff$ $320ffffff$ $9ffbffff$ $ffffbffb$ $500fefee$ $00017eef$ $ffa57fbd$ $0000ffff$ $0007efbb$ $9fc07eff$ $0011fffe$ $ffff7eff$ $00003fef$ $ffffffff$ $b7f5fff7$
M'	$fffffffff$ $ffdfdfff$ $320ffffff$ $9ffbffff$ $ffffbfff$ $500fefee$ $00017eef$ $ffa57fbd$ $0000ffff$ $0007efbb$ $9fc07eff$ $0011fffe$ $ffff7eff$ $00003fef$ $ffffffff$ $b7f5fff7$
Хеш	$e2d92913$ $14c3e9c9$ $c05ae4c4$ $d71b818e$

M	$fffffffff$ $ffdfdfff$ $320ffffff$ $9ffbffff$ $ffffbffb$ $500fefee$ $00017eef$ $ffa57fbd$ $0000ffff$ $0007efbb$ $9fc07eff$ $0090fffe$ $ffff7eff$ $00003fef$ $ffffffff$ $b7f5fbff$
M'	$fffffffff$ $ffdfdfff$ $320ffffff$ $9ffbffff$ $ffffbfff$ $500fefee$ $00017eef$ $ffa57fbd$ $0000ffff$ $0007efbb$ $9fc07eff$ $0090fffe$ $ffff7eff$ $00003fef$ $ffffffff$ $b7f5fbff$
Хеш	$9f8a53bb$ $bc5d22e1$ $bea932f7$ $68fdfa3f$

M	$ffffffff$ $ffdfdfff$ $320fffff$ $8ffbfff7$ $ffffbffb$ $500fefee$ $0001feef$ $ffe5ffbd$ $0000ffff$ $0007efbb$ $9fc07eff$ $0090fffe$ $ffff7eff$ $00003fef$ $ffffffffe$ $b7f5fbff$
M'	$ffffffff$ $ffdfdfff$ $320fffff$ $8ffbfff7$ $ffffbfff$ $500fefee$ $0001feef$ $ffe5ffbd$ $0000ffff$ $0007efbb$ $9fc07eff$ $0090fffe$ $ffff7eff$ $00003fef$ $ffffffffe$ $b7f5fbff$
Хеш	$3308ee34$ $6fd31df7$ $d2b60488$ $30b791b4$

M	$ffffffff$ $ffdfdfff$ $320fffff$ $8ffbffff$ $ffffbffb$ $500fefee$ $0001feef$ $ffa5ffbd$ $0000ffff$ $0007efbb$ $9fc07eff$ $0091fffe$ $ffff7eff$ $00003fef$ $ffffffffe$ $b7f5fbf7$
M'	$ffffffff$ $ffdfdfff$ $320fffff$ $8ffbffff$ $ffffbfff$ $500fefee$ $0001feef$ $ffa5ffbd$ $0000ffff$ $0007efbb$ $9fc07eff$ $0091fffe$ $ffff7eff$ $00003fef$ $ffffffffe$ $b7f5fbf7$
Хеш	$95f8bcc1$ $c2e2fc2a7$ $08d4279$ 12494874

Табела 3.20. Примери колизија

Потребно је приметити да се сви парови порука разликују у трећем биту четврте речи (рачуна се и нулта реч), односно $ffffbffb$ је замењено са $ffffbfff$. Даљом применом парова и тројки неутралних битова, могуће је добити још колизија.

3.7 Вишеструка колизија

Под појмом вишеструке колизије (енгл. multicollision) подразумева се појава када више од две поруке дају исти хеш. Вишеструка k-колизија краће се назива k-колизија. Претпоставља се да пар порука $(M, M + \Delta M_1)$ преко путање P_1, почетне разлике ΔM_1 и уз услове C_1 производи колизију. Нека други пар порука $(M, M + \Delta M_2)$ преко путање P_2, почетне разлике ΔM_2 и уз услове C_2 такође производи колизију. Уколико не постоји контрадикција између услова C_1 и C_2, можемо посматрати скуп $C = C_1 \cup C_2$ као полазни скуп услова за разматрање које следи. Уколико порука M задовољава скуп услова C, она задовољава и скуп услова C_1, па пар порука $(M, M + \Delta M_1)$ чини колизију. Слично, уколико порука M задовољава скуп услова C, она задовољава и скуп услова C_2, па пар порука $(M, M + \Delta M_2)$ чини колизију. Дакле, добија се 3-колизија од порука $(M, M + \Delta M_1, M + \Delta M_2)$.
Уколико су скупови C_1 и C_2 дисјунктни, тада пар порука $(M, M + \Delta M_1)$ задовољава скуп услова C_2. Пошто порука $M + \Delta M_1$ задовољава скуп услова C_2, следи да пар порука $(M + \Delta M_1, M + \Delta M_1 + \Delta M_2)$ даје колизију. Слично, уколико су скупови C_1 и C_2 дисјунктни, тада пар порука $(M, M + \Delta M_2)$ задовољава скуп услова C_1. Пошто порука $M + \Delta M_2$ задовољава скуп услова C_1, следи да пар порука $(M + \Delta M_2, M + \Delta M_2 + \Delta M_1)$ даје колизију. Дакле, поруке $M, M + \Delta M_1, M + \Delta M_2, M + \Delta M_1 + \Delta M_2$ чине 4-колизију. Важно је нагласити, да добијена 4-колизија захтева не само да услови скупова C_1 и C_2 буду непротивуречни, већ да они буду и дисјунктни.
Претходним разматрањем може се генерализовати појам k-колизије.

Нека важи претпоставка да се тражећи k-колизију, полази од t путања које заједно стварају t-колизију ($k \leq 2^t$). Потребно је прво изабрати t разлика порука $\Delta M_i, i = 0, 1, ..., t-1$. За сваку разлику ΔM_i, потребно је пронаћи скуп довољних услова C_i. Ако постоји било која контрадикција међу условима $C_i, i = 0, 1, ..., t-1$, неопходно је променити путање тако да поменуте контрадикције буду отклоњене. Затим се налази порука M која задовољава услове $C = \cup C_i$. Тада поруке $(M, M+\Delta M_0, M+\Delta M_1, ..., M+\Delta M_{t-1})$ чине t-колизију. Тачније, ако постоји t међусобно независних путања (не постоји пресек битова између скупова $C_i, i = 0, 1, ..., t-1$), све поруке из скупа $\{M+\Delta M \mid \Delta M$ је линеарна комбинација од $M_i, i = 0, 1, ..., t-1\}$ чине 2^t колизију заједно са поруком M. Ово је последица чињенице да t коефицијената у поменутој линеарној комбинацији могу узети вредности нула или један, што укупно даје 2^t могућности.

3.8 Хеш функције и проблем SAT

Наставак поступка проналажења колизија после одређивања диференцијалног пута може се свести на решавање проблема задовољивости (SAT, скраћеница од satisfiability). У овој тачки даје се приказ поступка свођења и преглед резултата из литературе.

Нека је дат Булов израз у конјуктивној нормалној форми (КНФ), односно конјукција неколико *клауза* (дисјункција група *литерала* - симбола променљивих или њихових негација). Познато је да се свака Булова функција може представити изразом у КНФ. За Булов израз се каже да је задовољив, ако постоји такво додељивање нула и јединица променљивим, да израз има вредност 1. Проблем SAT састоји се у утврђивању да ли је задати израз задовољив (при чему није неопходно пронаћи одговарајуће вредности променљивих). Проблем SAT је NP-комплетан проблем.

Општи поступак свођења проблема проналажења колизије код хеш функције на проблем SAT описан је у [34]. Нека је $p_1p_2...p_M$ низ битова улазне поруке која даје N-битни хеш $h_1h_2...h_N, (M \leq N)$. Битови хеш вредности h_i, $(i = 1, ..., N)$ могу се представити изразима $h_i = H_i(p_1, p_2, ..., p_M)$ полазећи од описа алгоритма израчунавања вредности хеш функције.

Да би се показало да хеш функција није једносмерна, потребно је на основу низа $h_1h_2...h_N$ одредити низ $p_1p_2...p_M$. Другим речима, потребно је пронаћи валуацију v, тако да за њену интерпретацију I_v важи

$$I_v(H_i(p_1, p_2, ..., p_M)) = h_i \ (i = 1, 2, ..., N) \tag{3.1}$$

односно

$$I_v(H_i(p_1, p_2, ..., p_M)) = \begin{cases} 1, & \text{ако је } h_i = 1 \\ 0, & \text{ако је } h_i = 0 \end{cases}$$

Следи да сви изрази $\overline{H}_i$,

$$\overline{H}_i(p_1,p_2,...,p_M)=\begin{cases} H_i(p_1,p_2,...,p_M), & \text{ако је } h_i=1\\ \neg(H_i(p_1,p_2,...,p_M)), & \text{ако је } h_i=0\end{cases}$$

имају вредност 1, ако важи (3.1).

$$\overline{\overline{H}}_i(p_1,p_2,...,p_M)=\bigwedge_{j=1,2,...,N}\overline{H}_i(p_1,p_2,...,p_M)$$

Према томе, нејединосмерност хеш функције еквивалентна је задовољивости израза $\overline{\overline{H}}_i(p_1,p_2,...,p_M)$, чиме је проблем једносмерности сведен на проблем SAT.

Провера слабе отпорности на колизију такође се може свести на проблем SAT. Да би се показало да хеш функција није слабо отпорна на колизију, за дати улазни низ $p_1p_2...p_M$ треба пронаћи други улазни низ $q_1q_2...q_M$ (на пример исте дужине), тако да оба улазна низа дају исту хеш вредност $h_1h_2...h_N$. Израз

$$H'(q_1,q_2,...,q_M)=\overline{\overline{H}}(q_1,q_2,...,q_M)\wedge(q_1^{p_1}\vee q_2^{p_2}\vee...\vee q_M^{p_M})$$

где

$$q_i^{p_i}=\begin{cases}\neg q_i, & \text{ако је } p_i=1\\ q_i, & \text{ако је } p_i=0\end{cases}$$

узима у обзир захтев да се друга улазна порука разликује од прве у барем једном биту. Тежина тестирања задовољивости израза $\overline{\overline{H}}_i$, брзо расте са M. С друге стране, за хеш функције са равномерном расподелом хеш вредности је мало вероватно да за $M<N$ две поруке дају исту хеш вредност, односно мало је вероватно да ће израз H' бити задовољив. Тиме се добија тежак и незадовољив пример улаза за SAT проблем.

На сличан начин, тражење колизије за хеш функцију, тј. проналажење било које две различите поруке $p_1p_2...p_M$ и $p'_1p'_2...p'_M$ које дају исту хеш вредност своди се на проверу задовољивости формуле

$$\bigwedge_{j=1,2,...,N}(H_i(p_1,p_2,...,p_M)\Leftrightarrow H_i(p'_1,p'_2,...,p'_M))\wedge\ \neg\bigwedge_{j=1,2,...,N}(p_i\Leftrightarrow p'_i).$$

Криптоанализа алгоритма MD4 може се поделити у четири фазе:

1. Избор почетне колизионе разлике
2. Проналажење диференцијалног пута
3. Извођење довољних услова који дају колизију
4. Модификација поруке

Мала ефикасност алгоритама за аутоматизацију добијања колизија у алгоритму MD4, карактеристика је корака 3 и 4. Међутим, како је показано [33], коришћењем SAT решавача извршавање управо корака 3 и 4 може значајно убрзати. Трансформишући диференцијални пут у конјуктивну нормалну форму са 53228 променљивих и 221440 клауза, Миронов (Mironov) и Ченг (Chang) [33] успели су да за 10 минута одреде колизију у алгоритму MD4. За нешто дуже време (100 часова) они су успели да пронађу колизију у алгоритму MD5.

Глава 4

Проналажење колизије код SHA – 0

У овој глави су описани основни напади на алгоритам SHA-0 .

4.1 Поступак Шобон - Жу

Користећи принцип диференцијалне криптоанализе блоковских шифара, Шобон и Жу [11] проналазе напад на алгоритам SHA-0 са тежином 2^{61} што је значајно боље од рођенданског напада (2^{80}). У случају SHA-1 агоритма, овај напад не доноси поменуто побољшање, али су накнадна побољшања примењива и на SHA-1.

4.1.1 Основна идеја

Као што је већ поменуто, појам "диференцијалног напада" подразумева одабир две најчешће сличне поруке и анализу њиховог кретања кроз алгоритам. Основна идеја сваког напада на алгоритам SHA-0 заснива се на појму "интерне колизије"(енгл. local, internal collision). Слободно говорећи, интерна колизија представља промену у поруци, чије се ширење кроз одређен број корака може анулирати. На тај начин, промене у алгоритму изазване променом поруке су "краткотрајне" , па не утичу на крајњи излаз и самим тим не ремете потенцијалну колизију. За промену у поруци приликом овог напада узима се први бит (ако рачунамо да постоји и нулти), па се због ротације за 5 и 30 места, потребне корекције праве на битовима на местима 1,6 и 31 у тридесетдвобитној речи. Међутим, због рекурентног начина на који се формира 80 тридесетдвобитних речи, свака промена у некој речи изазваће промене и у следећим речима, па је и за њих потребно разрешити интерне колизије. Како разрешење сваке интерне колизије траје шест корака, могуће је да ће доћи до њиховог преклапања и све морају

бити завршене закључно са последњом, осамдесетом рундом. Значи, суштину напада чини погодан избор корака у којима ће се догодити интерна колизија. Важно је запазити да процес експанзије поруке може се изразити само преко регистра A тј. важи једнакост:

$$A_i = A_{i-1} \lll 5 + f_i(A_{i-2}, A_{i-3} \lll 30, A_{i-4} \lll 30) + A_{i-5} \lll 30 + W_i + K_i.$$

Одавде се види да промена узрокована у неком кораку, у наредних пет корака утиче на регистар A. Промењен регистар A означава се са A'.

	ΔW	Δa	Δb	Δc	Δd	Δe	$\Delta f()$
i	0000 0001	0000 0001	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
$i+1$	0000 0020	0000 0000	0000 0001	0000 0000	0000 0000	0000 0000	0000 0000
$i+2$	0000 0001	0000 0000	0000 0000	4000 0000	0000 0000	0000 0000	0000 0001
$i+3$	4000 0000	0000 0000	0000 0000	0000 0000	4000 0000	0000 0000	4000 0000
$i+4$	4000 0000	0000 0000	0000 0000	0000 0000	0000 0000	4000 0000	4000 0000
$i+5$	4000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000

Табела 5.1. SHA-0 интерна колизија

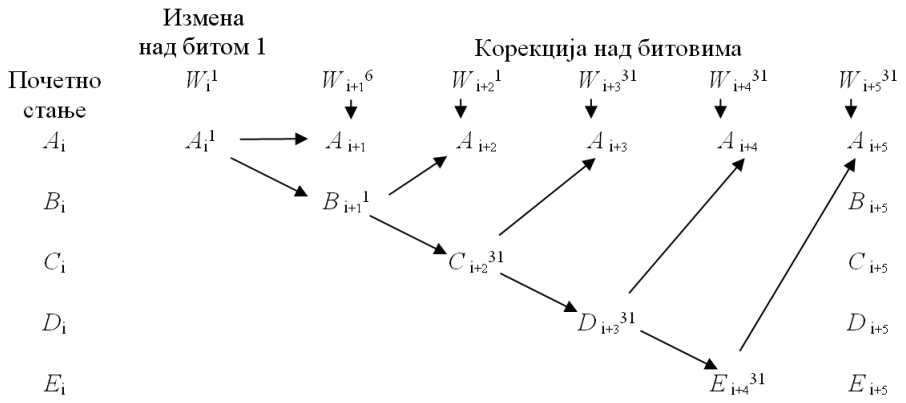
Све корекције на поруци (тј. битовима 32-битних речи $W_i^0, ..., W_i^{31}$, $0 \leq i < 80$) чине се у циљу анулирања почетних измена бита на месту 1. Другим речима, код сваке измене у речи W_i важи $A_{i+1} \oplus A'_{i+1} = 00000002$. Остали регистри тренутно нису промењени, али ће под утицајем A'_{i+1} сукцесивно бити промењени у следећим корацима. Потребно је спречити промену регистра у корацима који следе. Тако, на A'_{i+2} утиче A'_{i+1} померен за 5 места у лево, па би се могла десити промена A'^6_{i+2}. Зато је потребно променити W'^6_{i+2}. Уколико је између њих функција $\oplus$ тј. ако важи $A'^6_{i+2} \oplus W'^6_{i+2}$, тада је корекција регуларна, за разлику од функције сабирања која би могла произвести пренос и тиме онемогућити корекцију. То значи да ће даља анализа морати да има у виду вероватноћу када неће доћи до преноса. Слично претходном, и остали промењени регистри могу променити регистар A па је зато неопходно за сваки од њих кориговати одговарајућу реч W'_i. Када се затвори круг од шест промењених речи, иницијална промена првог бита у првој речи неће се наставити даље, али, још једном треба нагласити да речи које у даљем рекурентном развоју зависе од промењених речи, отварају нове интерне колизије које такође треба разрешити. Већ у овом тренутку згодно је приметити да регистри B', C', D' изазивају промену у тридесетпрвом, последњем биту у речи, где не може доћи до преноса, па није битно да ли се промена са нуле на јединицу коригује истом или супротном променом.

Рунда		i	$i+1$	$i+2$	$i+3$	$i+4$	$i+5$
Улазна реч	W^1	$0 \to 1$		$1 \to 0$			
	W^6		$1 \to 0$				
	W^{31}				$0 \leftrightarrow 1$	$0 \leftrightarrow 1$	$0 \leftrightarrow 1$
Жељени резултат	A^6		$0 \to 1$				
	f^1			$0 \to 1$			
	f^{31}				$0 \leftrightarrow 1$	$0 \leftrightarrow 1$	
	E^{31}						$0 \leftrightarrow 1$
Разлика регистара		A^1	B^1	C^1	D^{31}	E^{31}	
		$0 \to 1$	$0 \to 1$	$0 \to 1$	$0 \to 1$	$0 \to 1$	

Табела 5.2. Појединачне разлике и корекције

4.1.2 Анализа основног напада

Постоје два извора нелинеарности у алгоритму SHA-0: функција f_i и функција сабирања (ADD). Да би се извукли потребни закључци, тренутно ће се ослабити алгоритам заменом функције ADD функцијом ексклузивно-или (у даљем тексту ће се користити ознака XOR, без опасности да се помеша са функцијом $f_i(B,C,D) = B \oplus C \oplus D$ коју се такође зове XOR). Такође, све функције $f_i(B,C,D)$ поставиће се да буду $f_i(B,C,D) = B \oplus C \oplus D$. Другим речима, функције IF и MAJ замениће се функцијом XOR, па једнакост
$A_i = A_{i-1} \lll 5 + f_i(A_{i-2}, A_{i-3} \lll 30, A_{i-4} \lll 30) + A_{i-5} \lll 30 + W_i + K_i$
постаје $A_i = (A_{i-1} \lll 5) \oplus (A_{i-2} \oplus (A_{i-3} \lll 30) \oplus (A_{i-4} \lll 30)) \oplus (A_{i-5} \lll 30) \oplus W_i \oplus K_i$.
Нека се ова ослабљена варијанта алгоритма SHA-0 назива - SHI-1. Нека се посматра вектор $W = (W_0, ..., W_{79})$ тренутно заборављајући на процес експанзије речи. Ако се промени вредност W_i^1, изазивају се промене у битовима $A_{i+1}^1, B_{i+2}^1, C_{i+3}^{31}, D_{i+4}^{31}, E_{i+5}^{31}$. Да би се поништио утицај тих промена на битове $A_{i+2}^6, A_{i+3}^1, A_{i+4}^{31}, A_{i+5}^{31}, A_{i+6}^{31}$, потребно је инвертовати битове $W_{i+1}^6, W_{i+2}^1, W_{i+3}^{31}, W_{i+4}^{31}, W_{i+5}^{31}$. Дакле, уз промену и без промене битова $W_i^1, W_{i+1}^6, W_{i+2}^1, W_{i+3}^{31}, W_{i+4}^{31}, W_{i+5}^{31}$ добијају се две различите путање од A_i, B_i, C_i, D_i, E_i до $A_{i+6}, B_{i+6}, C_{i+6}, D_{i+6}, E_{i+6}$, што нема утицаја на наредне регистре. Тиме је створена интерна колизија. Треба запазити да је у иницијалној промени речи W_i намерно промењен први бит. Следећа слика описује интерну колизију у шест корака:



Слика 5.1. Интерна колизија алгоритма SHA-0

Пошто је све линеарно, у функцији SHI-1 може се направити различит број интерних колизија у циљу добијања две различите путање од A_0, B_0, C_0, D_0, E_0 до $A_{80}, B_{80}, C_{80}, D_{80}, E_{80}$. Прва путања користи оригиналну поруку W, док друга промењену поруку W'. Пошто се промене у поруци W' простиру даље у процесу експанзије (измена у W_i, такође мења $W_{i+3}, W_{i+8}, ...$), природно се поставља питање како изабрати интерне колизије, тако да кроз процес експанзије све сем иницијалних промена буду анулиране? Било би погодно направити маску која када се примени на полазну поруку даје нову са којом чини колизију. Пошто је потребно шест узастопних рунди за анулирање промене бита у датој рунди, јасно је да у последњих пет рунди не може доћи до почетне промене првог бита у речи ("измена"). Изабрати рунду у којој ће се десити интерна колизија, значи изабрати бинарни "вектор измена"(енгл. disturbance vector) $m_0 = (m_0^{(0)}, ..., m_0^{(79)})$ са јединицом на месту i ако је негирано W_i^1. Како процес експанзије представља примену операције $\oplus$ над битовима на истим позицијама, то значи да за вектор измена важи:

$$m_0^{(i)} = m_0^{(i-3)} \oplus m_0^{(i-8)} \oplus m_0^{(i-14)} \oplus m_0^{(i-16)},\ \forall i,\ 16 \leq i \leq 79$$

Из овог вектора извешће се маска пертурбације (енгл. perturbative mask) за поруку W на следећи начин: (елементи вектора M_0 су 32-битне речи)

$$M_0 = (M_0^{(-5)}, ..., M_0^{(79)})$$
$$\forall i,\ -5 \leq i \leq -1,\ M_0^{(i)} = 0$$
$$\forall i,\ 0 \leq i \leq 79,\ M_{0,k}^{(i)} = 0,\ k \neq i$$
$$\forall i,\ 0 \leq i \leq 79,\ M_{0,1}^{(i)} = m_0^{(i)}$$

Маска M_0 почиње са пет нула-речи, јер се остале маске M_1, M_2, M_3, M_4, M_5 добијају из M_0 транслацијом и ротацијом. Маска M_0 примењена на поруку, мења први бит у свим речима чији индекс одговара индексу у вектору измена где се налази јединица. Како су све остале промене узроковане овом променом првог бита непожељне, остале маске $(M_1, M_2, M_3, M_4, M_5)$ служе за њихову корекцију. Маска M_1 изводи се из маске M_0 транслацијом за једну рунду и ротацијом за пет битова у лево. Овим се анулира свака промена у шестом биту, настала услед ротације у алгоритму за пет места у лево:

$$\forall i,\ -4 \leq i \leq 79,\ M_1^{(i)} = ROL_5(M_0^{(i-1)})$$

Слично, у циљу кориговања осталих нежељених промена, из M_0 изводе се и остале маске:

$$\begin{gathered}\forall i,\ -3 \leq i \leq 79,\ M_2^{(i)} = M_0^{(i-2)} \\ \forall i,\ -2 \leq i \leq 79,\ M_3^{(i)} = ROL_{30}(M_0^{(i-3)}) \\ \forall i,\ -1 \leq i \leq 79,\ M_4^{(i)} = ROL_{30}(M_0^{(i-4)}) \\ \forall i,\ 0 \leq i \leq 79,\ M_5^{(i)} = ROL_{30}(M_0^{(i-5)})\end{gathered}$$

У табели 5.2. дат је могући приказ маске за првих шест речи $(W^{(0)}, ..., W^{(5)})$:

	$M^{(-5)}$	$M^{(-4)}$	$M^{(-3)}$	$M^{(-2)}$	$M^{(-1)}$	$M^{(0)}$	$M^{(1)}$	$M^{(2)}$	$M^{(3)}$	$M^{(4)}$	$M^{(5)}$
M_0						$M_0^{(0)}$					
M_1							$M_1^{(1)}$				
M_2								$M_2^{(2)}$			
M_3									$M_3^{(3)}$		
M_4										$M_4^{(4)}$	
M_5											$M_5^{(5)}$

Табела 5.3. Пример маске пертурбације

У случају да је прва промена била на биту W_0^1, приказане маске мењају почетну реч у првом биту (то ради $M_0^{(0)}$) и анулирају остале нежељене промене (то раде маске $M_1^{(1)}$, $M_2^{(2)}$, $M_3^{(3)}$, $M_4^{(4)}$, $M_5^{(5)}$). Маска $M^{(i)}$ задужена је за све промене над речју W_i. Из табеле 5.2 види се да је потребно да глобална маска дефинисана са:

$$M^{(i)} = M_0^{(i)} \oplus M_1^{(i)} \oplus M_2^{(i)} \oplus M_3^{(i)} \oplus M_4^{(i)} \oplus M_5^{(i)},\ \forall i,\ -5 \leq i \leq 79 \qquad (1)$$

буде излаз из процеса експанзије речи, односно генерално да важи:

$$W_i \oplus M^{(i)} = W_{i-3} \oplus M^{(i-3)} \oplus W_{i-8} \oplus M^{(i-8)} \oplus W_{i-14} \oplus M^{(i-14)} \oplus W_{i-16} \oplus M^{(i-16)}$$

одакле због начина експанзије поруке следи:

$$M^{(i)} = M^{(i-3)} \oplus M^{(i-8)} \oplus M^{(i-14)} \oplus M^{(i-16)},\ \forall i,\ 11 \leq i \leq 79$$

што је због (1) еквивалентно са:

$$M_k^{(i)} = M_k^{(i-3)} \oplus M_k^{(i-8)} \oplus M_k^{(i-14)} \oplus M_k^{(i-16)},\ \forall k \in \{1,2,3,4,5\},\ \forall i,\ 11 \leq i \leq 79.$$

Како M_1, M_2, M_3, M_4, M_5 настају транслацијом и ротацијом из M_0, претходна једнакост ће бити задовољена када важи:

$$M_0^{(i)} = M_0^{(i-3)} \oplus M_0^{(i-8)} \oplus M_0^{(i-14)} \oplus M_0^{(i-16)},\ \forall i,\ 11 \leq i \leq 79$$

Претходна једнакост формирана је из вектора измена и аналогна је односу битова у речима на месту један тј. $M_{0,1}^{(i)}$, а ово је аналогно стању битова у вектору измена. Дакле, глобална маска $M^{(i)},\ i = 0,1,2,3,4,5$ уклапаће се у процес експанзије ако важи:

$$m_0^{(i)} = m_0^{(i-3)} \oplus m_0^{(i-8)} \oplus m_0^{(i-14)} \oplus m_0^{(i-16)},\ \forall i,\ 11 \leq i \leq 79 \qquad (2)$$

где су због начина формирања маске (пошто важи $\forall i,\ -5 \leq i \leq -1,\ M_0^{(i)} = 0$) и због неопходности да се свака промена мора анулирати закључно са последњом рундом постављена ограничења:

$$m_0^{(-5)} = m_0^{(-4)} = m_0^{(-3)} = m_0^{(-2)} = m_0^{(-1)} = 0 \qquad (3)$$

$$m_0^{(75)} = m_0^{(76)} = m_0^{(77)} = m_0^{(78)} = m_0^{(79)} = 0 \qquad (4)$$

Из (2) и (3) следи:

$$m_0^{(i)} = m_0^{(i-3)} \oplus m_0^{(i-8)} \oplus m_0^{(i-14)} \oplus m_0^{(i-16)},\ \forall i,\ 16 \leq i \leq 79.$$

Сада је свих 80 променљивих $m_0^{(i)}$ одређено са било којих 16 узастопних, што значи да имамо 16 слободних променљивих, док се остале изражавају преко њих, па би потпуна претрага износила 2^{16} покушаја. Међутим, због рекурентног развоја и (4) добијају се два система од по пет једначина:

$$\begin{gathered}
m_0^{(6)} = m_0^{(0)} + m_0^{(1)} + m_0^{(2)} + m_0^{(4)} \\
m_0^{(7)} = m_0^{(0)} + m_0^{(4)} \\
m_0^{(8)} = m_0^{(0)} + m_0^{(1)} + m_0^{(5)} \\
m_0^{(9)} = m_0^{(4)} \\
m_0^{(10)} = m_0^{(0)} + m_0^{(5)}
\end{gathered}$$

$$\begin{gathered}
m_0^{(11)} = m_0^{(3)} + m_0^{(8)} \\
m_0^{(12)} = m_0^{(4)} + m_0^{(9)} \\
m_0^{(13)} = m_0^{(5)} + m_0^{(10)} \\
m_0^{(14)} = m_0^{(0)} + m_0^{(3)} + m_0^{(6)} + m_0^{(8)} \\
m_0^{(15)} = m_0^{(1)} + m_0^{(4)} + m_0^{(7)} + m_0^{(9)}
\end{gathered}$$

Ових 10 једначина са 16 непознатих даје 6 слободних променљивих $(m_0^{(0)}, m_0^{(1)}, m_0^{(2)}, m_0^{(3)}, m_0^{(4)}, m_0^{(5)})$, па се од њих може формирати $2^6 = 64$ различитих вектора измена који задовољавају услов да се маска M_μ (која покрива свих 80 речи) уклапа у процес експанзије E. Дакле, ако је дата 512-битна улазна маска $\mu = (M^{(0)}, ..., M^{(15)})$, за коју важи $M_\mu = E(\mu)$, следи да поруке $W = (W_0, ..., W_{15})$ и $W' = W \oplus \mu$ дају исти излаз у функцији SHI-1.

4.1.3 Увођење нелинеарних функција

Сада ће се анализирати утицај нелинеарних функција f_i које ће у одговарајућим рундама SHI-1 алгоритма заменити функцију $f_i = XOR$. Другим речима, ово ће бити случај реалног алгоритма SHA-0 ослабљеног тако што је сабирање (ADD) замењено $\oplus$ операцијом. Нека се овај алгоритам назива SHI-2. У неким случајевима, f_i ће се понашати као XOR функција, па важи претходни напад. Како се f_i разликује од XOR због функција IF и MAJ, остаје да се испита са којом се вероватноћом те функције понашају као XOR. Пошто ове функције раде паралелно са 32 бита, довољно је посматрати један бит. Посматраће се разлика у понашању функције $f(B, C, D)$ у случају да је састављена од промењених регистара, тј. када важи $f(B', C', D')$. Имајући у виду да се једино регистри C и D баве истим, тридесетпрвим битом, могу се десити следећи случајеви:

1. Не постоји разлика у улазу тј. $B_i = B'_i, C_i = C'_i, D_i = D'_i$. У овом случају функција XOR не мења излаз ($B^k \oplus C^k \oplus D^k = B'^k \oplus C'^k \oplus D'^k$), па се функција f_i понаша као XOR ако и само ако и она не мења излаз ($f_i(B_i, C_i, D_i) = f_i(B'_i, C'_i, D'_i)$). Очигледно је да је у случају функција IF и MAJ ово задовољено.

2. Постоји промена једино у биту првог регистра B_i тј. $B'_i = B_i \oplus 2^1$. Тада се f_i понаша као XOR ако и само ако важи $f_i(B_i, C_i, D_i) = f_i(B'_i, C'_i, D'_i) \oplus 2^1$.

3. Постоји промена у биту 31 или само регистра $C_i (C'_i = C_i \oplus 2^{31})$ или само регистра $D_i (D'_i = D_i \oplus 2^{31})$ Тада се f_i понаша као XOR ако и само ако важи $f_i(B_i, C_i, D_i) = f_i(B'_i, C'_i, D'_i) \oplus 2^{31}$.

4. Постоје две промене у биту 31 и то истовремено у регистрима C_i и D_i (као последица промене две речи у поруци у две узастопне рунде) тј. $C'_i = C_i \oplus 2^{31}$ и $D'_i = D_i \oplus 2^{31}$. Тада се f_i понаша као XOR ако и само ако важи $f_i(B_i, C_i, D_i) = f_i(B'_i, C'_i, D'_i)$.

Функција $MAJ = f_i(B, C, D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$ у случајевима 2 и 3 понаша се исто: промена једног бита изазива промену у излазу ако и само ако су остала два бита супротна. Ово се дешава са вероватноћом 1/2. У случају 4, функција MAJ не мења излаз (и тада се понаша као функција XOR) ако и само ако се битови C_i^{31} и D_i^{31}

мењају у супротном смеру, а то се дешава са вероватноћом 1/2 (од четири могуће вредности за пар битова у коме се оба мењају, одговарају случајеви када су битови супротни, па се мењају у супротном правцу - таквих случајева има два од четири).
Функција $IF = f_i(B, C, D) = (B \wedge C) \vee (\neg B \wedge D)$ у случају 2 не мења излаз ако и само ако су битови C_i^1 и D_i^1 супротног знака, што се дешава са вероватноћом 1/2. У случају 3, функција IF мења излаз ако и само ако важи:

$$\begin{cases} B_i^{31} = 1 & \text{, ако се } C_i^{31} \text{ променило } (C_i' = C_i \oplus 2^{31}) \\ B_i^{31} = 0 & \text{, ако се } D_i^{31} \text{ променило } (D_i' = D_i \oplus 2^{31}) \end{cases}$$

Ово се дешава са вероватноћом 1/2 $(1/2 \cdot 1/2 + 1/2 \cdot 1/2)$. У случају 4, функција IF увек мења излаз, док функција XOR никад, тако да је вероватноћа да се те две функције исто понашају једнака нули. Како се случај 4 може десити једино у случају промене у две узастопне рунде, следи да управо то треба избегавати. Функција IF делује у алгоритму у првих 20 рунди (0,...,19), па узастопних промена не сме бити пре рунде 16, док је у рундама 16 и 17 то могуће, будући да ће тек у рунди 20 (а ту више нема функције IF) бити истовремено промењена оба регистра C_i и D_i (C_i из рунде 17 и D_i из рунде 16). Ово последње ограничење додатно смањује број могућих вектора измена из функције SHI-1. Вектор измена који су пронашли Шобон и Жу изгледа:

```
00000 00100010000000010111
      01100011100000010100
      01000100100100111011
      00110000111110000000
```

Првих пет нула је присутно да би био задовољен начин формирања маске из почетног вектора измена. Позиције у вектору на којима се налазе јединице одговарају рундама у којима се променио први бит речи W_i. Најједноставнији начин добијања колизије у алгоритму SHI-2 свео би се на избор случајне поруке $W = (W_0, ..., W_{15})$ на коју се затим примењује добијена маска. Сада ће се у новонасталој поруци функције IF и MAJ са одређеном вероватноћом понашати као функција XOR, па је потребно формирати довољан број порука, да би у некој од њих важило поменуто понашање са вероватноћом 1. Суптилнији начин био би избор случајне речи W_0, примена маске на њу док се не деси да се у истој рунди функција f_i (IF) понаша као XOR. Затим се исто понавља и са речима $W_1, ..., W_{14}$ (с обзиром на дати вектор измена, потребно је поменуто применити само на речи W_2, W_6, W_{14}). Сада, када имамо фиксирано $W_0, ..., W_{14}$, изабраће се довољно различитих речи W_{15} (потребно је мање од од 2^{32} могућих комбинација), тако да када се примени маска на њих, добија се једна која доводи до колизије у 80 рунди. Пошто се први део претраге речи $W_0, ..., W_{14}$ обавља само једном за многе W_{15} случајеве, сложеност израчунавања одговара сложености тражења праве вредности за W_{15}. Претходно резоновање могуће је

применити на SHI-2 алгоритам, пошто је од могућих 2^{32} вредности за W_{15}, у просеку потребно 2^{24}.

Узнемирење у рунди i	f_{i+2}	случај	f_{i+3}	случај	f_{i+4}	случај	вероватноћа	log вероварноће
2	IF	2	IF	3	IF	3	1/8	3
6	IF	2	IF	3	IF	3	1/8	3
14	IF	2	IF	3	IF	3	1/8	$3 = 2 + 1$
16	IF	2	IF	3	XOR	-	1/4	2
17	IF	2	XOR	-	XOR	-	1/2	1
18,19,21								
22,26,27	XOR	-	XOR	-	XOR	-	1	0
28,35								
37	XOR	-	MAJ	3	MAJ	3	1/4	2
41	MAJ	2	MAJ	3	MAJ	3	1/8	3
45	MAJ	2	MAJ	3	MAJ	3	1/8	3
48	MAJ	2	MAJ	3	MAJ	3	1/8	3
51	MAJ	2	MAJ	3	MAJ	3	1/8	3
54	MAJ	2	MAJ	3	MAJ	4	$1/4\sqrt{2}$	2.5
55	MAJ	2	MAJ	4	MAJ	4	1/4	2
56	MAJ	2	MAJ	4	XOR	-	$1/2\sqrt{2}$	1.5
58,59,62								
63,68,69	XOR	-	XOR	-	XOR	-	1	0
70,71,72								

Табела 5.4. Вероватноћа успеха у SHI-2 алгоритму

Ако се промена извршила у двема узастопним рундама, дешава се случај 4. У случају функције IF, он је непожељан, па у првих 16 рунди мора бити избегнут. У случају функције MAJ, он се појављује у рундама 54,55 и 56 као последица истовремених промена битова $(C_{58}^{31}, D_{58}^{31})$ и $(C_{59}^{31}, D_{59}^{31})$. Функција MAJ тада се понаша као XOR са вероватноћом 1/2, па пошто се у табели 5.4. исти случај јавља у две рунде (58,59), вероватноћа која ће се на крају помножити дели се на $\sqrt{2}$ у обе рунде. Из табеле 5.4. такође се види да је вероватноћа налажења жељене поруке у којој се у рундама експанзије (после рунде 15) функције IF и MAJ понашају као XOR износи 2^{-26}. Уз додатне корекције, то се може свести на 2^{-24}. Као последица промене у рунди 14, у функцији IF бит B_{16}^{1} може изазвати промену у рунди 16. Функција IF се понаша као XOR ако су битови C_{16}^{1} и D_{16}^{1} супротни. Међутим, ови битови су познати у рунди 14, пошто су копије битова A_{14}^{3} и A_{13}^{3}, па су се још у рунди 14 могли одговарајуће подесити. Тиме се вероватноћа њихове супротности (1/2) преноси у првих 15 рунди (тачније у рунду 14), па се вероватноћа добијања жељене поруке у процесу експанзије спушта са 2^{-26} на 2^{-25}. Такође, измена у рунди 14 изазива промену бита C_{17}^{31} у рунди 17. Функција IF се тада понаша као XOR ако важи $B_{17}^{31} = 1$.

Пошто је B_{17}^{31} копија бита A_{16}^{31}, његова коректност се може проверити по избору речи W_{15} и по потреби променити бит W_{15}^{31}. Тиме се вероватноћа добијања жељене поруке у процесу експанзије спушта на 2^{-24}.
Сада се у алгоритам SHI-1 уводи сабирање (ADD) и нека се такав алгоритам назива SHI-3. Овај алгоритам се може схватити као и као потпун SHA-0 ослабљен заменом функција IF и MAJ функцијом XOR.
Као и за алгоритам SHI-2 , потребно је испитати када се алгоритам SHI-3 понаша као SHI-1. Тешкоћа може настати због могућег преноса јединице при сабирању. За анулирање промене, потребно је шест рунди, па пошто се у свакој рунди може десити пренос (са вероватноћом 1/2), вероватноћа да ни у једној неће доћи до преноса износи 2^{-6}.
Управо зато је за бит измена изабран први бит који у случају ротације за 30 места постаје последњи, тридесетпрви бит који при сабирању не може довести до преноса (сабирање је по модулу 2^{32}). То значи да у случају промене бита W_i^1, над битовима $W_{i+3}^{31}, W_{i+4}^{31}, W_{i+5}^{31}$ нису потребне корекције услед преноса, па се удвостручује вероватноћа да ни у једној рунди неће доћи до преноса. У намери да се избегне пренос, потребно је сваку промену бита кориговати супротном променом битова који у наредним рундама коригују промењени бит. На пример, ако је $W_i^1 = 0$, промена у првом биту значи $W_i'^1 = 1$. Да и после промене не би дошло до преноса, неопходно је да важи $ROL_5(A_i) + f_i(B_i, C_i, D_i) + E_i + K_i = 0$ (ово се дешава са вероватноћом 1/2), одакле следи да је

$$A_{i+1} = 0 \implies A'_{i+1} = 1. \qquad (5)$$

У следећој рунди потребно је избећи пренос, а да се истовремено обави корекција утицаја бита из претходне рунде. То значи да из $A_{i+2} = W_{i+1} + ROL_5(A_{i+1}) + f_{i+1}(B_{i+1}, C_{i+1}, D_{i+1}) + E_{i+1} + K_{i+1}$ следи да би се анулирао утицај $ROL_5(A'_{i+1})$ на A_{i+2}, мора важити $W'^6_{i+1} = 0$.
У следећој рунди, бит B_{i+2}^1 може променити регистар $A_{i+3} = W_{i+2} + ROL_5(A_{i+2}) + f_{i+2}(B_{i+2}, C_{i+2}, D_{i+2}) + E_{i+2} + K_{i+2}$. Уколико важи $C_{i+2}^1 = D_{i+2}^1$ (са вероватноћом 1/2), пошто су у алгоритму SHI-3 све функције $f_i = XOR = B \oplus C \oplus D$, следи да промена бита B_{i+2}^1 мења вредност функције f_{i+2}. Из (5) следи $B'_{i+2} = A'_{i+2} = 1$, па да не би дошло до преноса мора важити $W'^1_{i+2} = 0$ тј. $W^1_{i+2} = 1$. Другим речима, ако се у основној поруци фиксира бит $W^1_{i+2} = 1$, када се инвертује, са вероватноћом 1/2 неће доћи до преноса јединице ако важи услов $C^1_{i+2} = D^1_{i+2}$.
Када је испуњен поменути услов у функцији XOR промена у регистру B изазива исту промену у излазу, па се каже да тада функција XOR задржава "смер промене". Овакво понашање се очекује и од функција IF и MAJ, јер је од интереса у којим случајевима се оне понашају идетично као и функција XOR.
Супротно претходној претпоставци, ако је $W_i^1 = 1$ следи $W_i'^1 = 0$, па ако у основној поруци није било преноса, онда важи из

$$A_{i+1} = W_i + ROL_5(A_i) + f_i(B_i, C_i, D_i) + E_i + K_i \qquad (6)$$

следи $ROL_5(A_i) + f_i(B_i, C_i, D_i) + E_i + K_i = 0$ тј. $A^1_{i+1} = 1$ и $A'^1_{i+1} = 0$. И даље се жели избећи пренос, а да не промени регистар A_{i+2}, па пошто је $A'^1_{i+1} = 0$, следи да мора бити $W'^6_{i+1} = 1$. Аналогно претходном случају, уз услов $C^1_{i+2} = D^1_{i+2}$ добија се да је $W'^1_{i+2} = 1$ тј. $W^1_{i+2} = 0$.
Дакле, приликом избора поруке потребно је у одговарајућем распореду поставити битове $W^1_i, W^6_{i+1}, W^1_{i+2}$, па ће се са вероватноћом 1/4 алгоритам SHI-3 понашати као SHI-1 (1/2 да се обезбеди одсуство преноса у i-том кораку и 1/2 да се обезбеди услов $C^1_{i+2} = D^1_{i+2}$). Може се наћи вектор измена који проналази колизију у SHI-3 алгоритму са вероватноћом 2^{-44}.
Најзад се може посматрати потпун SHA-0 алгоритам. Сада се анализира понашање функције IF у односу на алгоритам SHI-2. У алгоритму SHI-2, случај 4 (истовремена промена битова C^{31} и D^{31}) је непожељан, јер се функција IF тада не понаша као функција XOR, па то важи и за SHA-0. Случај 3 (промена бита C^{31} или D^{31}) је идентичан, јер промена битова C^{31} или D^{31} не може произвести пренос (узгред, промена битова C^{31} или D^{31} у функцијама IF и MAJ не мења смер промене), па се функција IF понаша као XOR као у алгоритму SHI-2 (са вероватноћом 1/2). Случај 2 (промена бита B^1) у алгоритму SHA-0 за разлику од SHI-2) може довести до преноса, тако да мора бити задовољен услов из алгоритма SHI-3 који са вероватноћом 1/2 спречава пренос и услов када се као у алгоритму SHI-2 , функција IF понаша као XOR са вероватноћом 1/2. Ова два услова су испуњена са вероватноћом 1/4. У SHI-3 алгоритму је закључено да не би дошло до преноса, функција f_i не сме мењати смер промене (под условом да су фиксирани неки битови поруке). Ако се и у алгоритму SHA-0 фиксирају одговарајући битови, пошто функција MAJ никада не мења смер промене, значи да никада не изазива пренос, па се понаша као у алгоритму SHI-2 ("добро" у случајевима 2,3 и 4 са вероватноћом 1/2). Специјално, случај 4 у алгоритму SHA-0 (као и у SHI-3) разликује се од истог случаја у алгоритму SHI-2 због неопходности да се фиксирају одговарајући битови у циљу избегавања преноса. Као што је објашњено у SHI-3 алгоритму, важи $C^{31}_{i+3} = A^1_{i+1} = W^1_i$ и $D^{31}_{i+4} = A^1_{i+2} = W^1_{i+1}$. Истовремена промена (у истој рунди) битова C_{31} и D_{31} могућа је једино ако се иницијална промена првог бита (измена) десила у две узастопне рунде, па се тада функција MAJ понаша као XOR ако и само ако не мења вредност, тј. ако се битови C^{31}_{i+3} и D^{31}_{i+4} мењају у супротном смеру. Дакле, неопходно је да важи $C^{31}_{i+3} \neq D^{31}_{i+4}$, односно $W^1_i \neq W^1_{i+1}$. Међутим, фиксирањем одговарајућих битова у алгоритму SHI-3 , постиже се да се у случају 4, функција MAJ у алгоритму SHA-0 понаша са вероватноћом 1 као функција XOR у алгоритму SHI-1. Значи, када постоји измена бита у две узастопне рунде i и $i+1$ за $36 \leq i \leq 55$, ако се постави услов на биту W^1_i тј. $(W^1_i \neq W^1_{i+1})$, функција MAJ ће се понашати исто као XOR.
Пре него што се прикаже табела вероватноћа успеха у алгоритму SHA-0 , треба закључити следеће: када се жели да се избегне пренос, од шест рунди колико траје циклус анулирања почетне промене, битне

су прве три, јер се последње три баве променом последњег, тридесетпрвог бита речи где не може доћи до преноса. Ако се постави почетни услов на биту W_i^1, онда са вероватноћом 1/2 неће доћи до преноса. Ова вероватноћа се мора анализирати у свакој рунди алгоритма SHA-0 (као и у SHI-3) у којој се догодила иницијална измена првог бита. Да у следећој рунди не би дошло до преноса, довољно је поставити услов $W_{i+1}^6 = W_i^1$. Додатно, ако се и у следећој рунди постави услов $W_{i+2}^1 = W_{i+1}^6$, да би се избегао пренос неопходно је функција f_i задржи смер промене (у случају IF и XOR то се дешава са вероватноћом 1/2, до MAJ увек задржава смер промене, тј. са вероватноћом 1). За исти вектор измена као у алгоритму SHI-2 добија се следећа табела:

Узнемирење у рунди i	$f^{(i+2)}$	случај	$f^{(i+3)}$	случај	$f^{(i+4)}$	случај	вероватноћа	log вероват.
2	IF	2	IF	3	IF	3	1/32	5
6	IF	2	IF	3	IF	3	1/32	5
14	IF	2	IF	3	IF	3	1/32	5 = 4+1
16	IF	2	IF	3	XOR	-	1/16	4
17	IF	2	XOR	-	XOR	-	1/8	3
18,19,21								
22,26,27	XOR	-	XOR	-	XOR	-	1/4	2
28,35								
37	XOR	-	MAJ	3	MAJ	3	1/16	4
41	MAJ	2	MAJ	3	MAJ	3	1/16	4
45	MAJ	2	MAJ	3	MAJ	3	1/16	4
48	MAJ	2	MAJ	3	MAJ	3	1/16	4
51	MAJ	2	MAJ	3	MAJ	3	1/16	4
54	MAJ	2	MAJ	3	MAJ	4	1/8	3
55	MAJ	2	MAJ	4	MAJ	4	1/4	2
56	MAJ	2	MAJ	4	XOR	-	1/4	2
58,59,62								
63,68,69	XOR	-	XOR	-	XOR	-	1/4	2
70,71,72								

Табела 5.5. Вероватноћа успеха у алгоритму SHA-0

На пример, када се у другој рунди догоди измена, функција IF се понаша као XOR као и у алгоритму SHI-2 (у рундама 3,4 и 5 са вероватноћом 1/8), с тим што се мора имати у виду вероватноћа да због промене W_i^1 (уз почетне услове) не дође до преноса (1/2) и вероватноћа да функција IF задржава правац промене у четвртој рунди (1/2). Укупно, добија се вероватноћа 1/32. Треба приметити да се функција IF истовремено понаша као XOR и задржава смер промене са вероватноћом 1/4 (када је $C_1 \neq D_1$, а C_1 је истог знака као и нова вредност B_1). На пример, ако се измена десила у рунди 37, поново се узима у

обзир почетна вероватноћа да не дође до преноса (1/2), као и понашање функције XOR две рунде касније, где функција XOR задржава смер промене са вероватноћом 1/2. У случају 3, функција MAJ понаша се као XOR исто као у SHI-2 алгоритму, тј. са вероватноћом 1/2. Све скупа, добија се вероватноћа 1/16. Табела 5.5 у случају алгоритма SHA-0 , може се посматрати као табела 5.4 SHI-2 алгоритма, с тим што су вероватноће да се променом изазваном у одређеној рунди алгоритам понаша као SHI-1 , у случају IF и XOR функција помножене са 1/4, а у случају MAJ функције са 1/2. Такође, функција MAJ у случају 4, у SHA-0 алгоритму, понаша се као XOR са вероватноћом 1, док се у SHI-2 алгоритму понаша као XOR са вероватноћом 1/2.
Од посебног значаја је рунда 14, у којој се слично рунди 2, у процесу анулирања иницијалне измене алгоритам SHA-0 понаша као SHI-1 са вероватноћом 1/32. Рунда 14 може се посматрати као гранична рунда која одваја првих 15 (индиректно и првих 16) рунди од рунди насталих у процесу експанзије. Она, сама за себе, доприноси укупној вероватноћи само за 1/2 (да не дође до преноса), будући да не анулира претходне рунде.
Поменути вектор измена даје вероватноћу налажења колизије 2^{-69}. Као последица промене у рунди 14, у функцији IF, бит B_i^1 у рунди 16 може изазвати промену. Функција IF понаша се као XOR уколико су битови C_i и D_i супротни. Али, ови битови су познати у рунди 14, будући да су копије битова A_{14}^3 и A_{13}^3, па су се још у рунди 14 могли одговарајуће подесити. Тиме се вероватноћа њихове супротности (1/2) преноси у првих 15 рунди (тачније у рунду 14), па се вероватноћа добијања жељене поруке у процесу експанзије спушта на 2^{-68}. Даље, као последица измене у рунди 16, у функцији IF бит B_{18}^1 у рунди 18 може изазвати промену. Функција IF понаша се као XOR ако су битови C_{18}^1 и D_{18}^1 супротни. Али, ови битови су познати у рунди 16, пошто су копије битова A_{16}^3 и A_{15}^3, односно познати су већ у рунди 15, јер важи: $A_{16}^3 = A_{15}^3 \lll 5 + f_{15}(A_{14}^3, A_{13}^3 \lll 30, A_{12}^3 \lll 30) + A_{11}^3 \lll 30 + W_{15}^3 + K_{15}^3$. Слично, као последица измена у рунди 17, у рунди 19 бит B_{19}^1 функцији IF може изазвати промену. Функција IF се понаша као XOR ако су битови C_{19}^1 и D_{19}^1 супротни. Ови битови су познати још у рунди 17, пошто су копије битова A_{17}^3 и A_{16}^3. Имајући у виду да се A_{16}^3 може израчунати из рунде 15 и да важи:

$$A_{17}^3 = A_{16}^3 \lll 5 + f_{16}(A_{15}^3, A_{14}^3 \lll 30, A_{13}^3 \lll 30) + A_{12}^3 \lll 30 + W_{16}^3 + K_{16}^3$$

тј.

$$A_{17}^3 = A_{16}^3 \lll 5 + f_{16}(A_{15}^3, A_{14}^3 \lll 30, A_{13}^3 \lll 30) + A_{12}^3 \lll 30 + W_{13}^3 \oplus W_8^3 \oplus W_2^3 \oplus W_0^3 + K_{16}^3,$$

следи да су битови C_{19}^1 и D_{19}^1 познати још у рунди 15. Дакле, погодним избором речи W_{15} могуће је изазвати жељено понашање функције IF у рундама 18 и 19 настало као последица измена у рундама 16 и 17.

Тиме нестаје потреба за вероватноћом 1/16 и 1/8 жељеног понашања алгоритма услед измена у рундама 16 и 17, па вероватноћа налажења колизије пада са 2^{68} на 2^{61} што је и максимално достигнуће описаног напада.

4.2 Поступак Бихам - Чен

Израелски криптографи, Ели Бихам и Рафи Чен, дали су унапређење [12] претходног напада. Под појмом структура разлика (енгл. difference pattern), подразумева се низ разлика између две поруке, при чему је свака вредност у разлици створена у посебној рунди као разлика вредности порука M и M'. Пожељно је да две поруке имају разлике које су локализоване, па се у наредним рундама понашају исто, што доводи до колизије. Другим речима, пожељно је да се што касније одступи од оваквог понашања. Основу овог напада, чине појам "скоро колизије"(енгл. near collision), што значи да се хеш вредности две поруке могу разликовати за мали број битова и појам "неутралног бита" што значи да промена тих битова не мења структуру разлика у првих r рунди. Ако је $r \geq 16$ не могу се произвољно мењати битови а да се не промени структура разлика до r-те рунде. Али, тада на сцену ступају неутрални битови који се могу мењати тако да структура разлика остане иста. Тиме се добија одређен број порука које не мењају структуру разлика у првих r рунди, чиме се повећава вероватноћа да ће нека од њих задовољити услове тако да се очува жељена структура разлика у свих 80 или барем првих 75 рунди. Испоставиће се да је број тих порука једнак 1/8 од свих пробаних порука. Ово потиче из чињенице да истовремена промена групе неутралних битова може и не мора довести до промене у структури разлика две поруке.

4.2.1 Анализа напада

До сада се "Вектор измена" посматрао као осамдесетобитни вектор у коме се налази јединица на i-том месту уколико се у i-тој речи десила промена која узрокује промене у наредних пет речи. Нека је $D = (D_0, D_1, ..., D_{79})$ вектор од 80 тридесетдвобитних речи које одговарају рундама у алгоритму којих такође има 80. Свака реч у вектору поставља се на јединицу (мисли се да бит најмање тежине постаје јединица) уколико се у њој одговарајућој рунди десила измена. У супротном, реч се поставља на нулу (тј. сви битови су јој нуле). Сада ће се такав вектор третирати као вектор измена. Пошто анулирање интерне колизије захтева додатних пет рунди, она се не може десити у последњих пет рунди, па су последњих пет речи у вектору измена такође нуле. У случају "скоро колизије" претходни услов није неопходан, јер хеш вредности порука на крају не морају бити идентични.
Нека је $SR^l(D)$ вектор од 80 речи добијен додавањем l нула-речи на

првих $80 - l$ речи вектора D. Ово одговара нецикличком померају у вектору D за l места. Сада се за анулирање промена изазваних интерном колизијом коригује променом бита 6 у одговарајућој ненултој речи у $SR^1(D)$, бита 1 у одговарајућој ненултој речи у $SR^2(D)$ и бита 31 у одговарајућим ненултим речима у $SR^3(D)$,$SR^4(D)$ и $SR^5(D)$. Ако је $\Delta = M' \oplus M$ почетна разлика порука, онда се процес експанзије E (добијање 80 речи од почетних 16) може представити у форми:

$$E(\Delta) = ((D \oplus SR^2(D)) \ll 1) \oplus (SR^1(D) \ll 6) \oplus ((SR^3(D) \oplus SR^4(D) \oplus SR^5(D)) \ll 31)$$

Пример 4.2.1. Нека хипотетички постоји једина интерна колизија у нултој рунди. Тада за вектор узнемирења $D = (D_0, D_1, ..., D_{79})$ важи $D_0 = (0, ..., 0, 1)$ и $D_i = (0, ..., 0),\ i = 1, ..., 79$.

$$\begin{aligned}
D &= (D_0, D_1, D_2, D_3, D_4, D_5, D_6, ..., D_{79}) \\
SR^1(D) &= (0\ \ , D_0, D_1, D_2, D_3, D_4, D_5, ..., D_{78}) \\
SR^2(D) &= (0\ \ , 0\ \ , D_0, D_1, D_2, D_3, D_4, ..., D_{77}) \\
SR^3(D) &= (0\ \ , 0\ \ , 0\ \ , D_0, D_1, D_2, D_3, ..., D_{76}) \\
SR^4(D) &= (0\ \ , 0\ \ , 0\ \ , 0\ \ , D_0, D_1, D_2, ..., D_{75}) \\
SR^5(D) &= (0\ \ , 0\ \ , 0\ \ , 0\ \ , 0\ \ , D_0, D_1, ..., D_{74})
\end{aligned}$$

За анулирање утицаја промене бита $W_1^{(0)}$, потребно је додатно променити још пет речи у рундама које следе: $W_6^{(1)}, W_1^{(2)}, W_{31}^{(3)}, W_{31}^{(4)}, W_{31}^{(5)}$. Непосредно се види да ако су у анулиране промене настале из почетне, очекује се да вредност $\delta = (\delta_1, \delta_2, ..., \delta_{80}) = (A_{i+1} \oplus A'_{i+1})_{i=0,...,79}$ буде једнака вредности $D_i \ll 1,\ i = 0, ..., 79$. Другим речима, у случају да су све почетне промене у интерним колизијама остале, а оне које су следиле из њих кориговане, очекивана вредност за δ је $\delta = D \ll 1$.
У претходном нападу (Шобон, Жу), није свака порука била погодна за колизију, већ само ако су били задовољени одређени услови, а то се дешавало са одређеном вероватноћом. Основна идеја овог напада састоји се у увођењу интерне колизије у некој средишњој рунди да би се избегао утицај вероватноћа насталих под утицајем претходних рунди. У намери да се почне тражење колизија у r-тој рунди, почиње се са паром порука M и M', њиховом разликом $\Delta = M' \oplus M$ и са два услова која ће се додатно увести.

Дефиниција 4.2.1. *За дату разлику поруке Δ, ознака δ представља очекивану разлику у регистру A у свакој рунди. Каже се да тај пар порука одговара вредности δ_r ако важи $A_i \oplus A'_i = \delta_i,\ \forall i \in \{1, ..., r\}$.*

У претходној дефиницији мисли се да пар порука одговара вредности δ_r ако је разлика регистра A у првих r $(0, ..., r-1)$ рунди очекивана. Треба уочити да у случају колизије, поруке M и M' одговарају очекиваној вредности δ_{80} (јер је у том случају у свих 80 рунди (0,...,79) дошло до очекиваних стања регистра A).

Дефиниција 4.2.2. *Нека су M и M' две поруке које одговарају вредности δ_r за неко $r \geq 16$. Каже се да је i-ти бит порука ($i \in \{0, ..., 511\}$) неутралан бит ако после промене (инвертовања) тог бита у порукама M и M', оне и даље одговарају вредности δ_r. Каже се да је пар битова (i-ти и j-ти бит) порука неутралан пар битова ако сви парови порука добијени истовременим инвертовањем у обе поруке било ког подскупа (у смислу - мора важити за све подскупове) из скупа ($\{i\}, \{j\}, \{i, j\}$), такође одговарају вредности δ_r. Другим речима, пар битова је неутралан, уколико су оба бита појединачно неутрална и ако њиховом истовременом променом у обе поруке, оне и даље одговарају вредности δ_r. За скуп битова $S \subseteq \{0, 1, ..., 511\}$ каже се да је неутралан скуп битова за поруке M и M', ако истовременом променом било ког његовог подскупа битова у обе поруке, оне и даље одговарају вредности δ_r. За скуп битова $S \subseteq \{0, 1, ..., 511\}$ каже се да је 2-неутралан скуп битова за поруке M и M', уколико је сваки бит из скупа S неутралан и сваки пар битова из скупа S такође неутралан.*

Величина максималног 2-неутралног скупа (његов број елемената), за дате поруке M и M' и дато r, означава се са $k(r)$. За пар порука M и M' уводе се две особине за које ће се у даљем тексту подразумевати да важе:

1. Пар порука M и M' одговара вредности δ_r.

2. Пар порука M и M' имају довољно велики 2-неутралан скуп битова. За очекивати је да велики број поскупова овог скупа битова буду неутрални скупови.

Како се у порукама M и M' управо мења 2-нутрални скуп битова (промена једног или два бита не утиче на промену жељених стања регистара до r-те рунде, тј. одговара вредности δ_r, док промена групе од три или више битова може не одговарати вредности δ_r, укупно се може формирати $2^{k(r)}$ парова порука. Пошто се очекује да велики број подскупова поменутог 2-неутралног скупа битова одговара вредности δ_r, логично је да ће се управо из овог скупа бирати парови порука за које се очекује да са великом вероватноћом доводе до колизије.
Како изабрати вредности r и $k(r)$? Пожељно је да те вредности буду што веће. Ако је p_i вероватноћа успешне корекције у i-тој рунди (тако да важи $A_i \oplus A'_i = \delta_i$), нека је $p(r) = \prod_{i=r}^{79} p_i$ вероватноћа успешних корекција у рундама које почињу са r. Како у рундама до r није било потребе за корекцијама тј. важи $p(r) = \prod_{i=0}^{r-1} p_i = 1$, $p(r)$ уједно представља вероватноћу успешне корекције у свим рундама. Када вектор измена у последњих пет рунди садржи нуле, $p(r)$ представља вероватноћу налажења колизије; у супротном, очекује се скоро колизија. Пошто је

$p(r)$ вероватноћа долажења до колизије или скоро колизије, потребно је тестирати око $1/p(r)$ порука које одговарају вредности δ_r. Пошто се може тестирати сваки скуп од $k(r)$ неутралних битова, од тих битова се може формирати $2^{k(r)}$ порука, па следи да r треба изабрати тако да важи $2^{k(r)} \geq 1/p(r)$. Пожељно је да што веће r задовољава претходну једнакост.
Како наћи 2-неутралан скуп за дати пар порука? Улаз у алгоритам је пар порука M и M' са разликом $\Delta = M \oplus M'$ која одговара вредности δ_r. Алгоритам генерише 512 парова-кандидата истовременим инвертовањем једног бита у порукама M и M' (тиме се неће пореметити разлика Δ). Нека је $e_i, i \in \{0, ..., 511\}$ порука у којој се на i-тој позицији налази бит један, а на осталим позицијама су нуле. Кандидате чине парови $(M \oplus e^i, M' \oplus e^i), i \in \{0, ..., 511\}$. Сваки кандидат се тестира да би се установило да ли одговара вредности δ_r и ако је то задовољено i је неутрални бит. Када се пронађу неутрални битови, може се формирати граф тако да неутрални битови чине чворове, а ивице повезују оне неутралне битове, чијим се инвертовањем не мења структура разлика која одговара вредности δ_r. Сада треба пронаћи максималан скуп чворова у коме су сви чворови међусобно повезани. Иако је генерално ово NP-комплетан проблем, у нашем случају лако се добија довољно велики скуп чворова.

Алгоритам налажења 2-неутралног скупа

Улаз: парови порука M и M' који одговарају вредности δ_r.
Излаз: 2-неутрални скуп S

1. Нека је $e_i,\ i \in \{0, ..., 511\}$ скуп порука код којих се на i-тој позицији налази јединица, а на осталим позицијама нуле. Нека је T скуп чворова потенцијалног графа и нека је $S = T = \emptyset$

2. За сваки $i \in \{0, ..511\}$ тестирати да ли пар $(M \oplus e^i, M' \oplus e^i)$ одговара вредности δ_r. Ако одговара следи $T = T \cup e_i$.

3. Нека је $G = (T, E)$ граф са чворовима T и ивицама $E = 0$.

4. За сваки пар $(e_i, e_j) \subseteq T,\ i \neq j$, тестирај да ли пар $(M \oplus e^i \oplus e^j, M' \oplus e^i \oplus e^j)$ одговара $(M \oplus e^i, M' \oplus e^i)$. Ако одговара, онда важи $E = E \cup \{(e_i, e_j)\}$.

5. Посматрајући граф G, практично није тешко наћи довољно велики део S графа G.

Напомена 4.2.1. Ако је $k(r)$ величина 2-неутралног скупа, може се претражити $2^{k(r)}$ порука од којих ће $2^{k(r)-3}$ $(2^{k(r)}/8)$ и даље одговарати вредности δ_r.
Нека је $p(r \to r') = \prod_{i=r}^{r'-1} p_i$ вероватноћа да пар који одговара вредности

δ_r такође одговара вредности δ'_r, при чему треба приметити да важи $p(r) = p(r \to 80)$.

Напомена 4.2.2. Нека су r и r' рунде тако да важи $p(r \to r') \approx 2^{-k(r)}$. После генерисања $2^{k(r)}$ парова порука, не само да се добија очекивани број парова порука који одговарају вредности δ'_r, већ и део тих парова порука одговара вредности $\delta_{r'+l}$, што би се очекивало да ће се десити тек са већим скупом од $2^{k(r)+\alpha}$ парова порука где су $2 \leq l \leq 4$ и $3 \leq \alpha \leq 8$. Ово значи да је вероватноћа добијања (скоро) колизије већа него што би се очекивало.

Алгоритам се даље може унапредити претрагом парова ненеутралних битова чије истовремено инвертовање ствара парове порука који такође одговарају вредности δ_r. Слично се може применити и на триплете и квартете битова итд. Свака оваква група битова може се схватити као придодати неутрални бит, чиме се такође увећава вредност $k(r)$.

Дефиниција 4.2.3. *Скуп битова назива се симултано-неутралним, ако појединачно гледано нису неутрални битови, али се њиховим истовременим инвертовањем не ремети структура разлика до рунде r.*

У многим случајевима парови битова који су симултано-неутрални представљени су у форми W_i^j и $W_{i-l}^{j-5l \ mod \ 32}$ за мале вредности l, што је у даљем тексту појашњено.
Од посебног интереса за реализацију напада јесте налажење пара порука са максималним 2-неутралним скупом битова. Претпоставимо да већ имамо неки пар порука који одговара вредности δ_r. Потребно је модификовати поменути пар порука у намери да се пронађе следећи пар порука који такође одговара вредности δ_r, али има већи 2-неутрални скуп битова. Алгоритам који решава овај проблем, за улаз узима пар поменутих порука, мења их на одређен начин и позива алгоритам за налажење 2-неутралног скупа. Ако је величина добијеног скупа већа од величине скупа почетног пара порука, основни пар се замењује новим паром порука и алгоритам се покреће поново.
Променом пара порука, ствара се нови пар порука за у нади да ће одговарати вредности δ_r. Промена се обавља над свим могућим битовима тако да се максимизује вероватноћа налажења 2-неутралног скупа за тај пар порука, а то значи да се сем неутралних битова, мењају и симултано неутрални битови. Мењајући групе битова, могуће је да неки од добијених парова неће одговарати вредности δ_r. Зато је после сваке промене потребно проверити да ли добијени пар порука одговара вредности δ_r.
Поставља се питање избора најбољег начина на који треба променити поруку, тако да заједно са добијеном поруком чини пар порука који ће са вероватноћом приближно 1/8 одговарати вредности δ_r и можда имати већи 2-неутрални скуп битова? Већ је напоменуто да би се у многим случајевима погодан облик налажења симултано неутралног

скупа састојао од парова узастопних порука померених за пет места (W_i^j и $W_{i-l}^{j-5l \ mod \ 32}$).
Додатно, могуће је погодном променом поруке умањити утицај преноса. Жељена, односно очекивана разлика регистра A ($A_{i+1} \oplus A'_{i+1}$) је $00000002h$, ако је настала под утицајем промене бита на месту један у речи W_i, у супротном је $00000000h$. У рунди у којој се десила промена речи W_i ($W_i^1 \neq W_i'^1$), услед утицаја бита преноса може се изазвати промена у следећем биту, па важи $A_{i+1} \oplus A'_{i+1} \neq 00000002h$ и пар порука неће одговарати вредности δ_r. То се може десити са вероватноћом 1/2. Пошто се жели да се први бит у разлици порука (речи) мења само када се деси иницијална промена у вектору измена, промена првог бита која би настала под утицајем преноса из претходног, нултог бита је непожељна. Та промена се може спречити инвертовањем W_i^0 и $W_i'^0$ или инвертовањем других битова који индиректно утичу на A_{i+1}^0. Такви битови су W_{i-1}^{27} и $W_{i-1}'^{27}$ (који утичу на A_i^{27}, а после ротације и на A_{i+1}^0), или битови W_{i-1}^{22} и $W_{i-1}'^{22}$ или уопште $W_{i-l}^{(32-5l) \ mod \ 32}$ и $W_{i-l}'^{(32-5l) \ mod \ 32}$.
Инвертовање ових парова битова отклања утицај преноса са вероватноћом 1/2. Овим се дакле, повећава вероватноћа да ће стање регистра A бити очекивано, тј. да ће пар порука одговарати вредности δ_r.
Из претходног се види да би алгоритам за налажење пара порука са највећим 2-неутралним скупом битова, улазну поруку требао да модификује мењањем поднизова битова ротираних за пет места у узастопним речима. На пример, битови који се могу мењати припадају скупу $S = \{W_0^0, ..., W_0^3\} \cup \{W_1^5, ..., W_1^8\} \cup ... \cup \{W_5^{25}, ..., W_5^{28}\}$(запажа се да је овде $k(r) = 24$) и инвертовањем ових битова покриће се 2^{24-1} парова порука (ако се изузме скуп од 24 нуле). За сваку промену, позива се алгоритам за тражење максималног 2-неутралног скупа. Затим се ротира цео скуп битова за по 31 могућу ротацију. На крају се иста анализа спроводи почевши од речи $W_1, W_2, ..., W_{10}$. Пошто се порука састоји од првих 16 речи W_i и дотиче наредних пет, процес се завршава када битови у скупу почињу са речју W_{10}.
Како наћи највеће могуће r, односно највећу могућу рунду за коју важи да пар порука одговара вредности δ_r? Ако две поруке одговарају вредности δ_r, у намери да се нађе пар порука који ће одговарати вредности за неко δ'_r користи се максимални неутрални скуп битова за већ постојећи пар порука. Промени порука се приступа као што је већ описано док не добије пар порука који одговарају вредности δ'_r. Сада се позива алгоритам за налажење 2-неутралног скупа битова и алгоритам за налажење пара порука са максималним 2-неутралним скупом битова.
Имајући у виду претходно поменуто, тражење (скоро)колизије подразумева да постоји пар порука M и M' који одговарају вредности за највећу могућу вредност r и највећим могућим 2-неутралним скупом S. Онда се генерише $2^{k(r)} - 1$ модификација порука и инвертовањем сваког подскупа битова у S. Пошто велики број тих подскупова одго-

вара вредности δ_r, то претраживање пара који води до (скоро)колизије може ефикасно почети од рунде r. Ако важи $2^{k(r)} \geq 1/p(r)$ онда се очекује скоро(колизија) у зависности како се понашају разлике $A_i \oplus A'_i$ после r-те рунде. Ако за неко r' важи $2^{k(r)} \geq 1/p(r \leftarrow r')$, тада се са великом вероватноћом очекује скоро(колизија) у редукованом, са r' рунди алгоритму SHA-0.

Бихам и Чен су пронашли колизију у алгоритму SHA-0 редукованом на 65 рунди. Тежина овог напада износи 2^{35}. Пар $r = 22$ и $k(r) = 40$ са истом сложеношћу проналази скоро колизију у алгоритму SHA-0 редукованом на 76 рунди. У року од једног дана израчунавања на просечном рачунару, добијена је скоро колизија на пуном алгоритму SHA-0 продуженом на 82 рунде. Тежина напада пуног алгоритма SHA-0 спуштена је са 2^{61} (Шобон-Жу) на 2^{56}.

Глава 5

Закључци и даљи рад

Као и код других криптолошких проблема, тренутна граница примене грубе силе није тачно позната, па је нејасна сигурност 128-битних, па чак и 160-битних хеш вредности. Под таквим околностима, разумно се поставља питање о смислу усавршавања суптилних начина за тражење колизије у алгоритмима са потенцијално недовољном дужином хеш вредности. Међутим, пошто је дужина хеш вредности обично мања од дужине поруке, тражење колизија код криптографских хеш алгоритама је типичан пример проблема за које се зна да решење постоји, али га није могуће одредити у разумном временском интервалу.

Предмет рада у наставку истраживања може да буде уопштење на шири скуп почетних разлика улазних порука, како у алгоритму MD4, тако и у алгоритмима SHA-0 и SHA-1. Поред тога, процес проналажења колизија у хеш функцијама требало би у што већој мери аутоматизовати.

Глава 6

Литература

[1] X.Y.Wang, X.J.Lai etc, Cryptanalysis for Hash Functions MD4 and RIPEMD, Eurocrypt05, LNCS 3494, pp.1-18.

[2] X.Y.Wang, H.B.Yu, How to Break MD5 and Other Hash Functions, Eurocrypt05, LNCS 3494, pp.19-35.

[3] X.Y.Wang, H.B.Yu, Y.Lisa, Efficient Collision Search Attacks on SHA-0, Crypto05, LNCS 3621, pp.1-16, 2005.

[4] X.Y.Wang, Y.lisa, H.B.Yu, Finding collisions on the Full SHA-1, Crypto05, LNCS 3621, pp.17-36.

[5] H.B.Yu, G.L.Wang, G.Y.Zhang and X.Y.Wang, The Second-Preimage Attack on MD4, CANS 2005, LNCS 3810, pp.1-12.

[6] M. Daum. Cryptanalysis of Hash Functions of the MD4-Family. PhD thesis, Ruhr-University of Bochum, 2005.

[7] Yu Sasaki, Lei Wang, Kazuo Ohta, and Noboru Kunihiro. New Message Difference for MD4, 2007. To appear in FSE 07 proceedings.

[8] Yusuke Naito, Yu Sasaki, Noboru Kunihiro, and Kazuo Ohta. Improved Collision Attack on MD4 with Probability Almost 1. In Dongho Won and Seungjoo Kim, editors, ICISC, volume 3935 of Lecture Notes in Computer Science, pages 129145. Springer, 2005.

[9] Ronald L. Rivest: The MD4 Message Digest Algorithm, CRYPTO90 Proceedings, 1991, http://theory.lcs.mit.edu/ rivest/Rivest-MD4.txt

[10] R.L.Rivest, The MD5 message-digest algorithm, Request for Comments(RFC 1320), Internet Activities Board, Internet Privacy Task Force, 1992.

[11] Chabaud, F., and Joux, A. Differential collisions in SHA-0. In CRYPTO (1998), H. Krawczyk, Ed., vol. 1462 of Lecture Notes in Computer Science, Springer, pp. 5671.

[12] E. Biham and R. Chen, Near-collisions of SHA-0, Cryptology ePrint Archive, Report 2004/146, 2004. http://eprint.iacr.org/2004/146

[13] Biham, E., Chen, R., Joux, A., Carribault, P., Lemuet, C., and Jalby, W. Collisions of SHA-0 and reduced SHA-1. In Cramer [9], pp. 3657.

[14] Joux, A. Multicollisions in iterated hash functions. application to cascaded constructions. In Franklin [11], pp. 306316.

[15] Joux, A., and Peyrin, T. Hash functions and the (amplified) boomerang attack. In CRYPTO (2007), A. Menezes, Ed., vol. 4622 of Lecture Notes in Computer Science, Springer, pp. 244263.

[16] M. Nandi and D. R. Stinson, Multicollision Attacks on a Class of Hash Functions, IACR preprint archive, 2005.

[17] Martin Cochran. Notes on the wang et al.sha-1 differential path. Cryptology ePrint Archive, Report 2007/474, 2007. http://eprint.iacr.org/.

[18] National Institute of Standards and Technology. FIPS 180: Secure Hash Standard, May 1993. Available from: http://csrc.nist.gov.

[19] National Institute of Standards and Technology, Secure hash standard, FIPS Publication, no.180-1, 1995.

[20] Wagner, D. The boomerang attack. In Fast Software Encryption (1999), L. R. Knudsen, Ed., vol. 1636 of Lecture Notes in Computer Science, Springer, pp. 156170.

[21] Y. Naito, Y. Sasaki, T. Shimoyama, J. Yajima, N. Kunihiro and K. Ohta. Improved Collision Search for SHA-0. In X. Lai and K. Chen, editors, Advances in Cryptology ASIACRYPT06, volume 4284 of Lecture Notes in Computer Science, pages 2136. Springer-Verlag, 2006.

[22] Complexity of the Collision and Near-Collision Attack on SHA-0 with Different Message Schedules, Mitsuhiro HATTORI, Shoichi HIROSE, and Susumu YOSHIDA

[23] W.W. Peterson, Error-correcting codes, M.I.T. Press, 1961.

[24] A.J. Menezes, S.A. Vanstone, and P.C. Van Oorschot. Handbook of Applied Cryptography. CRC Press, Inc., Boca Raton, FL, USA, 1996.

[25] D. R. Stinson. Cryptography: Theory and Practice, Second Edition, CRC Press, 2002.

[26] R.C. Merkle. Secrecy, authentication, and public key systems. Stanford Ph.D. thesis 1979, pages 13-15.

[27] Martin Schläffer. Cryptanalysis of MD4. Masters thesis, Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria, February 2006.

[28] Martin Schläffer and Elisabeth Oswald. Searching for Differential Paths in MD4. In Robshaw , pages 242261.

[29] Миодраг Живковић. Криптографија. Скрипта, Математички факултет

[30] Multi-collision Attack on the Compression Functions of MD4 and 3-Pass HAVAL - 2007 Hongbo Yu, Xiaoyun Wang. Information Security and Cryptology- ICISC

[31] H. Dobbertin. Cryptanalysis of MD4. Journal of Cryptology, 11:253274, 1998.

[32] Stephane Manuel, Thomas Peyrin: Collisions on SHA-0 in One Hour. FSE 2008: 16-35

[33] Ilya Mironov, Lintao Zhang: Applications of SAT Solvers to Cryptanalysis of Hash Functions. SAT 2006: 102-115

[34] Dejan Jovanović and Predrag Janičić. Logical analysis of hash functions. In Bernhard Gramlich, editor, Frontiers of Combining Systems (FroCoS), vol-

ume 3717 of Lecture Notes in Artificial Intelligence, pages 200215. Springer Verlag, 2005.

Глава 7

Додатак - Објашњење изворног кôда

Програм који у неким случајевима реализује алгоритам написан је у програмском језику Јава и подељен је у два пакета. Пакет kolizijeMD4 тражи путању промена битова који воде до колизије, док пакет modifikacijaPorukeMD4 обавља модификацију поруке задовољавајући све услове прве и скоро све услове друге рунде. За преостале неиспуњене услове друге рунде, коришћењем неутралних битова, добијају се поруке које задовољавају све услове и самим тим воде до колизије.

7.1 Пакет **kolizijeMD4**

Класа *Elem* описује елемент листе која се користи у поступку стварања различитих путања битова.

Класа *Main* представља главну класу са *main* методом у којој се уносе улазни подаци: редни број промењене речи и у оквиру ње редни број промењеног бита. Ова класа обавља корак 1 у алгоритму налажења путање која води до колизије.

У класи *Obrada* у методама *inicijalizuj* вредностима помоћних регистара *aa,bb,cc,dd* сукцесивно се додељују бројеви од 1 до 52, при чему се потом у методи *dodela* исти бројеви стављају у низ тако да редослед бројева одговара поретку регистара у MD4 алгоритму, тј. *a,d,c,b*. Тиме су у алгоритму обрађени кораци 2,3 и 4. У оквиру класе *Obrada* даје се појединачно значење метода:

Метода *korak* враћа корак треће рунде у коме се десила промена речи.

Метода *prviPromenjen* рачуна вредност првог промењеног регистра у првој рунди.

Метода *pretvori* претвара бројчану вредност регистра у словну ознаку.

Метода *ObradaVaznihReci* враћа пар "важних" регистара: први који анулира промену речи у трећој рунди и други који анулира први регистар у претходне три рунде.

Метода *ispisiVazne* у циљу провере, исписује важне регистре.

Метода *toString* такође у циљу провере, исписује све регистре кроз алгоритам.

Класа *Putanja* изведена је из класе *Obrada*, обрађује путање у оба смера и у алгоритму покрива кораке 5,6,7,8 и 9. За ту сврху, листа представља погодан облик груписања битова. Атрибути класе *Putanja* су четири матрице *a,b,c,d* код којих прва координата представља редни број регистра, а друга број његовог промењеног бита. Даље, показивач *prvi* служи као глава листе у којој се налази тражена путања (од почетка до краја). Показивач *prviZaSlepuGranu* служи као глава листе за слепу грану насталу од путање са почетка. Показивач *slepaGranaDrugeRunde* служи као глава листе за слепу грану насталу променом речи у другој рунди.

У конструктору класе *Putanja* издваја се меморијски простор за новоуведене матрице *a,b,c,d*.

Метода *korakZaVrednost* враћа корак у коме се рачуна регистар са задатом вредношћу.

Метода *indexKrajnjegRegistra* проналази индекс регистра који коригује промену коју изазива промењена реч у трећој рунди.

Метода *indexPocetnogRegistra* проналази индекс првог промењеног регистра у првој рунди.

Метода *pretvori* претвара вредност неког елемента матрице (бит) у његову словну ознаку.

Метода *putanjaSkraja* уназад одређује низ промењених битова почев од регистра који коригује регистар који коригује промену речи у трећој рунди, па до регистра у првој рунди са истим индексом као и почетни регистар промене.

Метода *putanjaSpocetka* одређује низ промењених битова од првог промењеног бита у првој рунди до регистра који има исти индекс као и крајњи регистри.

Метода *sastaviPutanju* саставља путању с почетка у претходну путању с краја.

Метода *staviUputanju* у листу са главом *prvi* ставља елементе који чине тражену путању.

Метода *staviUslepuGranu* ставља елементе матрице (битове) у листе које представљају слепе гране. Метода *ispisPutanje*, у циљу провере исписује тражену путању.

Метода *ispisSlepeGrane* исписује листу која представља слепу грану насталу под утицајем почетне промене у првој рунди.

Метода *ispisSlepeGraneDrugeRunde* исписује листу која представља слепу грану насталу под утицајем промене речи у другој рунди.

Класа *Prosiri* изведена је из класе *Putanja* и обрађује кораке алгоритма 10,11,12 и 13.

Метода *putanjaSkraja2* одређује низ уназад промењених битова почев од регистра који коригује промену речи у трећој рунди до регистра у првој рунди са истим индексом као и почетни регистар.

Метода *sastaviPutanju2* саставља путању са почетка и претходну путању са краја.

Метода *pronalazenjeSlepihGrana* упоређује путању са почетка и тражену путању и издваја остатак путање са почетка који не припада траженој путањи. Тај остатак назива се "слепа грана са почетка".

Метода *slepaGranaDrugeRunde* упоређује путању насталу променом у другој рунди са траженом путањом. Уколико се ни један од битова те слепе гране не налази у траженој путањи, потребно је анулирати неки од чланова те путање. Боље је анулирати бит ближе почетку гране, да би се смањио број неопходних услова приликом модификације поруке.

Метода *korekcijaSlepeGrane* проширује неки од битова тражене путање, тако да се тим проширењем потире неки од битова слепе гране путање са почетка. Ова метода може да прихвати проширења дубине један (из једног бита проширењем добијамо још један).

Метода *korekcijaSlepeGraneDrugeRunde* само установљава да у нашем специјалном случају нема потребе за проширењем тражене путање.

Метода *toString* исписује тражену путању од почетка до краја. При томе, елементи матрица *a,b,c,d* који су негативни упућују да се промена бита догодила са јединице на нулу, док позитивни упућују да се промена догодила са нуле на јединицу.

7.2 Пакет modifikacijaPorukeMD4

Независно од претходних класа које траже путању промена битова која води до колизије, класа *MD4Mod* на основу добијене путање и услова, модификује произвољну полазну поруку и на добијену слабу поруку додаје почетну разлику чиме се добија колизија.

Конструктор иницијализује почетну вредност регистара a, d, c, b, претвара поруку у низ битова и одваја простор за помоћне матрице *aaa*, *ddd*, *ccc*, *bbb* које ће при свакој промени поруке наново испитивати стање регистара, односно задовољеност потребна 62 услова. Конструктор такође прихвата поруку унету у главном програму, док је метода *vratiPoruku* враћа као вредност приватног атрибута.

Методе *add,sub,lshift,rshift,ROL,ROR,not,or,and,xor* обављају одговарајуће операције над речима или паровима речи.

Метода *divideK* распоређује одговарајуће константе по рундама.

Метода *f* реализује по рундама функције F, G и H.

Метода *porukaUByte* распоређује поруку по речима.

Метода *byteUString* претвара низ битова у знаковни запис.

Метода *bitUHex* претвара низ битова у хексадецимални запис.

Метода *hexUBit* претвара хексадецималну поруку у низ битова.

Метода *toString* исписује поруку са размацима између речи.

Метода *konacnaPromena* додаје колизиону разлику на слабу поруку.

Метода *vratiPromenjeneUslove* враћа преостале незадовољене услове.

Метода *modifikacijaPrveRunde* модификује услове прве рунде.

Метода *proveraUslova* проверава тренутну задовољеност довољних услова.

Метода *runde1* пуни помоћне регистре тренутним вредностима правих регистара, као и помоћну матрицу (x) у којој се та стања чувају.

Методе *moda6_17, modc6_17, moda6_25, modd5_20, modd6_18, modd6_5, modb5_18,modb5_20* редом модификују одговарајуће битове друге рунде.

Методе *promeniBit, vratiBit, promeniPar, vratiPar, promeniTriplet, vratiTriplet* редом мењају или анулирају неутралне битове, парове или тројке неутралних битова.

Метода *grubaSila* задовољава преосталих шест услова, тако што по потреби мења парове односно тројке неутралних битова.

Метода *runde* прати промене регистара кроз рунде, исписује стања појединих битова и даје излазну хеш вредност.