VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Magistro baigiamasis darbas

Statistinė SHA-3 konkurso maišos funkcijų analizė (Statistical Analysis of Hash Functions from SHA-3 Competition)

Atliko: 2 kurso, 2 grupės studentė
Halina Orvidaitė

Darbo vadovas:
doc. dr. Vilius Stakėnas

Vilnius
2012

Turinys

Sutartinių terminų sąrašas

Kriptologija	Mokslas, nagrinėjantis informacijos privatumo apsaugą.
Kriptografija	Mokslas, tiriantis metodus kaip informaciją užšifruoti ir iššifruoti.
Paprastas tekstas (*angl. plaintext*), **pradinis pranešimas**	Pradiniai duomenys, kuriuos šifruojame arba norime perskaityti.
Šifruotas tekstas (*angl. ciphertext*), **kriptograma** (*angl. cryptogram*)	Pranešimas, gautas pritaikius kriptografinį algoritmą.
Atsitiktinis kintamasis	Kintamasis, kuris įgyja reikšmę „1“ su tikimybę p, ir reikšmę „0“ su tikimybe $1 - p$.
Dvejetainė seka	Nulių ir vienetų seka.
Blokas	Bitų srauto dali, turinti nustatytą ilgį.
Kritinė reikšmė	Vertė, mažesnė už testo statistiką labai mažu procentu.
Alternatyvioji hipotezė H_a	Teiginys, kuris būna teisingas, jeigu analizuojama bitų seka nėra atsitiktinė.
Nulinė hipotezė H_a	Teiginys, kuris būna teisingas, jeigu analizuojama bitų seka yra atsitiktinė.
P-reikšmė	tikimybė, kad analizuojama sekos rezultatai nebus prastesni, nei tikėtini atsitiktinės sekos rezultatai.
Bitų eiga (*angl. run*)	Nenutrūkstanti panašių bitų seka
Pradinė bitų seka (sėkla) (*angl. seed*)	PRNG įvesties duomuo.
NIST (*angl. National Institute of Standards and Technologies*)	JAV Nacionalinis standartų ir technologijų institutas.
RNG (*angl. Random Number Generator*)	Atsitiktinių skaičių generatorius.
NSA (*angl. National Security Agency*)	JAV Nacionalinė saugumo agentūra.
SHA (*angl. Secure Hash Algorithm*)	Saugus maišos algoritmas, nurodantis kriptografinių maišos funkcijų, sukurtų MD4 pagrindu, šeimą.
Maišos funkcijos (*angl. hash function*)	Vienakryptės funkcijos, kurios bet kokio ilgio pranešimui suformuoja fiksuoto ilgio maišos reikšmę (santrauką).
$\oplus$ arba **XOR**	Sudėtis moduliu 2^w, kur w – žodžio ilgis bitais.
$\wedge$ arba **AND**	Loginė daugyba.
$\vee$ arba **OR**	Loginė sudėtis.
$<<< s$	32 bitų žodžio postūmis į kairę per s bitų.
$\leftarrow$	Reikšmės kintamajam priskyrimas.

Anotacija

Pagrindinis magistro baigiamojo darbo tikslas buvo, pasinaudojant NIST SHA-3 maišos algoritmų kompresijos funkcijomis, sukurti pseudo-atsitiktinių skaičių generatorių ir atliktų juo sugeneruotų sekų statistinius testus. Darbo metu surinkau pagrindinę teorinę bazę, reikalingą, norint susipažinti su naujosiomis SHA-3 maišos funkcijomis bei NIST pateikiamu statistinių testų paketu. Detaliai išanalizavau algoritmus, kurie šiuo metu yra maišos funkcijų standartai, ir kurių savybių tenkinimas yra minimalus reikalavimas SHA-3 algoritmų kandidatams. Detaliai pristačiau kiekvieną iš penkių finalinių SHA-3 algoritmų, testavimo algoritmus, kurie yra pateikti statistinių testų pakete: aptariau jų idėją ir tikslą, pateikiamus įvesties kintamuosius, atliekamus algoritmų žingsnius, reikalavimus funkcijoms paduodamiems kintamiesiems bei gautų rezultatų interpretavimo aspektus. Taip pat pristačiau sugalvotą pseudo-atsitiktinių skaičių generatoriaus algoritmą ir jo Java realizaciją. Sugeneravus testinių duomenų paketą, jį įvertinau NIST statistinių testų pagalba.

Summary

The main aim of my final master paper work was to gather theoretical basis, which provides description of cryptology and it‘s elements, valid hash function standards and NIST competition for SHA-3. During my studies I’ve gathered needed information to understand hash algorithms which are represented by five finalists of NIST SHA-3 competition. I’ve analyzed algorithms of current hash function standards and main requirements participants must fulfil in order to become a winner of a competition in detail. I’ve represented each SHA-3 finalist’s function with deep analysis. Also I’ve gathered theoretical basis, which provides description of US National Institute of Standards and Technology created Statistical Test Suite. This statistical test suite is testing binary streams generated by random or pseudorandom number generators. I have given a detailed description of algorithms in given statistical suite: I have provided the main idea and aim of those tests, variables used for input, steps of those algorithms, requirements for input data and possible interpretation of results. Also I’ve introduced an algorithm of pseudorandom numbers generator and have given its’ realization in Java. Finally I’ve created a test data suite and have assessed it with NIST provided statistical test suite.

Įvadas

Kriptografija, kaip priemonė perduoti informaciją, nesudarant galimybės bet kam jos perskaityti, minima jau antikos laikais. Laikui bėgant kriptografiniai šifrai buvo tobulinami ir automatizuojami. Šiais laikais kriptografiniais protokolais remiasi dauguma kompiuterinių saugumo standartų, sertifikatų. Vienas kriptografinių algoritmų – skaitmeninis parašas – vis dažniau naudojamas patvirtinant dokumento autentiškumą.

Kriptografinės funkcijos ir algoritmai yra nuolat testuojami, ieškant galimų trūkumų, optimizuojami, bandant padaryti juos spartesniais ir galinčiais operuoti didesniais informacijos blokais ir t.t. Dar viena kriptografijos dalis yra maišos funkcijos, šiuo metu dažnai patenkančios į diskusijų centrą dėl NIST vykdomo SHA-3 maišos funkcijų konkurso, kurio nugalėtojas taps nauju maišos funkcijų standartu.

Maišos funkcijose tiek duomenys, tiek kriptografinis raktas dažniausiai būna bitų srautas. Yra sukurtą be galo daug įvairiausių testų, leidžiančių nustatyti naudojamo bitų srauto tinkamumą kriptografijai, tačiau šiuo metu žymiausias yra JAV NIST pateiktas statistinių testų rinkinys, leidžiantis bitų srautą išanalizuoti įvairiausiais aspektais.

Pagrindinis magistrinio darbo tikslas buvo susipažinti su pagrindiniais kriptografijos elementais ir šiuo metu galiojančiais maišos funkcijų standartais, taip pat su NIST pateiktais statistiniais testais bei pristatyti finalininkų, pretenduojančių tapti nauju SHA-3 standartu, algoritmus. Realizuoti pseudo-atsitiktinių skaičių generatorių, jo realizaciją Java kalba bei sugeneruoti testinių duomenų paketą ir jį įvertinti, pasinaudojant NIST statistinių testų paketu.

Literatūros analizę bei pristatytus darbus vykdžiau tokiais etapais.

- Surinkau informaciją apie bendrąsias kriptografijos sąvokas, jos skaidymą į atskirus metodus.
- Trumpai pristačiau kriptografijos elementus, kurie būtini susipažįstant su kriptografija, tačiau mano darbe nėra pagrindiniai.
- Detaliai išanalizavau simetrinio rakto kriptografiją, pristačiau kiekvieną iš trijų pagrindinių jos metodų.
- Detaliai išanalizavau MD4 ir MD5 algoritmų veikimo schemas, MD5 principą pristačiau savo darbe, nurodžiau šio algoritmo sąryšį su mane dominančia SHA algoritmų šeima.
- Detaliai išanalizavau kiekvieną SHA-1 ir SHA-2 grupių algoritmą, išsamiai pristačiau SHA-1 ir SHA-256 algoritmų veikimo schemas, nurodžiau kitų likusių SHA algoritmų panašumus ir skirtumus su jais.
- Susipažinau su NIST paskelbto SHA-3 maišos funkcijų konkurso sąlygomis, reikalavimais finalininkams ir ateities planais.
- Išsiaiškinau penkių konkurso finalininkų maišos funkcijų veikimo principus ir detaliai juos pristačiau savo darbe.
- Surinkau informaciją apie statistinius testus ir jų veikimo principus.
- Detaliai išanalizavau NIST pateiktą statistinių testų paketą, pristačiau kiekvieno rinkinyje esančio algoritmo:
    - idėją ir tikslą;
    - pateikiamus įvesties kintamuosius;
    - atliekamus algoritmų žingsnius;
    - reikalavimus funkcijoms paduodamiems kintamiesiems;
    - gautų rezultatų interpretavimo aspektus.
- Pasinaudodama virtualios mašinos VirtualBox pagalba, sukompiliavau NIST statistinių testų paketą Linux aplinkoje.
- Sugalvojau pseudo-atsitiktinių skaičių generatoriaus, kuris sekas generuoja pasinaudodamas SHA-3 maišos algoritmuose naudojama kompresijos funkcija, algoritmą bei realizavau jį Java programavimo kalba.
- Sugeneravau testinių duomenų paketą ir jį išanalizavau pasinaudodama NIST statistinių testų paketu.

I. Teorinė magistro darbo dalis

1. Kriptografija

Kriptografija [Orv11] (*graikiškai kryptós – paslėptas ir gráphein – rašyti*) – tai mokslas, tiriantis metodus kaip informaciją užšifruoti ir iššifruoti.

Kriptografijos negalima priskirti nei vienai platesnei mokslų kategorijai – šią sritį galima priskirti informacijos teorijai, matematikai ar net inžinerijai.

Kriptologiją sudaro dvi sritys: kriptografija bei kriptoanalizė.

Šių dienų kriptografiją galima suskirstyti į tokius metodus:

1) simetrinis (slaptojo rakto);
2) asimetrinis (viešojo rakto);
3) maišos funkcijos.

Toliau trumpai priminsiu apie maišos funkcijas NIST vykdomą SHA-3 maišos funkcijų konkursą.

1.1. Maišos funkcijos

Maišos funkcijos yra vienakryptės funkcijos, kurios bet kokio ilgio pranešimui suformuoja fiksuoto ilgio maišos reikšmę (santrauką). Maišos funkcijos naudojamos informacijos vientisumo patikrinimui, slaptažodžių saugojimui, paieškos raktų formavimui duomenų bazėse ir panašiai.

Pagal [MOV96], [Wu11] pagrindinės maišos funkcijos savybės:

1) apibrėžimo sritis didelė palyginti su fiksuoto dydžio reikšmių sritimi;
2) funkciją nesunku apskaičiuoti bet kokiam argumentui;
3) daug kartų kviečiant funkciją su įvairiais argumentais, jos rezultatai pasiskirsto tolygiai;
4) pagal rezultatą negalima vienareikšmiškai nustatyti argumento;
5) labai sunku surasti du argumentus m_1 ir m_1 tokius, kad $h(m_1) = h(m_2)$ – itin svarbi savybė kriptografijoje.

Kriptografinės maišos funkcijos atlieka esminį vaidmenį šiuolaikinėje kriptografijoje. Maišos funkcijos paima pranešimą ir suformuoja rezultatą, vadinamą maišos kodu, maišos rezultatu, maišos verte arba tiesiog maiša. Pagrindinė kriptografinių maišos funkcijų idėja yra ta, kad maišos reikšmės yra lyg kompaktiškas pranešimo vaizdas ir gali būti naudojamas su tuo pranešimu.

Maišos funkcijos naudojamos duomenų vientisumui užtikrinti, jungiant kartu su skaitmeniniais parašais. Tokiu atveju pirmiausia, pasinaudojant maišos funkcija, gaunama pranešimo santrauka, ir po to ji pasirašoma skaitmeniniu parašu.

Aukščiausiame lygyje maišos funkcijos dalinamos į dvi klases:

1) maišos funkcijos neturinčios rakto – tai tokios funkcijos, kurios turi tik vieną įvesties parametrą – patį pranešimą;
2) maišos funkcijos turinčios raktą – tai tokios funkcijos, kurios turi du įvesties parametrus – pranešimą ir slaptą raktą.

Formalus maišos funkcijos apibrėžimas reikalauja, kad būtų tenkinamos tokios dvi savybės:

1) maišos funkcijos kompresija – h sujungia baigtinio bitų ilgio įvesties x, su fiksuoto bitų ilgio n rezultatu h(x);
2) maišos funkcijos skaičiavimų paprastumas – duota funkcija h ir įvesties parametras x, o rezultatą h(x) yra lengva apskaičiuoti.

Šios savybės taikomos maišos funkcijoms be rakto. Būtent šios maišos funkcijos mus ir domina, nes yra skiriamos 3 šių maišos funkcijų klasės:

1) maišos funkcijos paremtos blokiniais šifrais;
2) maišos funkcijų variacijos;
3) maišos funkcijos paremtos moduline aritmetika.

1.2. SHA-3 maišos funkcijos konkursas

NIST maišos funkcijos konkursas yra atviras visiem norintiems dalyvauti. Šį konkursą organizuoja Nacionalinis standartų ir technologijų institutas, norint sukurti naują SHA-3 maišos funkciją, kuri turėtų pakeisti senas SHA-1 ir SHA-2. Konkursas oficialiai pradėtas 2007 metais lapkričio 2 dieną. NIST suteikia

galimybę sukurti vieną ar kelias maišos funkcijas, pasinaudojant viešu konkursu. Panašiu principu buvo sukurtas pažangus šifravimo standartas (AES).

Dalyviai sukurtas maišos funkcijas turėjo pateikti iki 2008 metų spalio 31 dienos, o tų pačių metų gruodžio 9 dieną buvo paskelbtas visų į pirmą turą patekusių dalyvių sąrašas. 2009 metų vasarį NIST surengė konferenciją, kuriose autoriai pristatė sukurtus algoritmus, bei NIST teisėjai nustatė kriterijus, pagal kuriuos turi būti atrinkti kandidatai į antrąjį turą, į kurį pateko 14 kandidatų. Jų sąrašas buvo paskelbtas 2009 metų liepos 24 dienos. 2010 metų rugpjūčio 23-24, Kalifornijos universitete, Santa Barbaroje buvo surengta antroji konferencija, kurioje buvo aptarti į antrąjį etapą patekę kandidatai. Finalininkai buvo paskelbti 2010 metų gruodžio 10 dieną. Tikimasi konkurso nugalėtoją paskelbti 2012 metais.

Į pirmąjį turą buvo atrinktas 51 kandidatas, į antrąjį – 14, o finalininkai 5.

NIST paskelbė kelis kriterijus, pagal kuriuos buvo atrinkti finalistai.

1. Vykdymas: dalis algoritmų buvo atmesta dėl jų per didelio techninių galimybių reikalavimo.
2. Saugumas: dalis algoritmų buvo atmesta dėl klaidų vykdant algoritmą, nors ir nebuvo rasta akivaizdi galimybė įvykdyti prieš juos atakas.
3. Analizė: dalis algoritmų atmesta, nes trūko algoritmo kriptoanalizės ir susidarė įspūdis, kad realizacija nebuvo pilnai ištestuota.
4. Įvairovė: dalis algoritmų buvo atmesta dėl algoritmų konstrukcijos įvairovės.

NIST išleido dokumentą, kuriame detaliai išdėstė kiekvieno algoritmo trūkumus, dėl kurių jis nepateko tarp finalininkų.

Dabar detaliai supažindinsiu su vieno iš penkių finalininkų paskelbtu algoritmu. Platesnė likusių keturių algoritmų analizė bus pristatyta baigiamajame magistriniame darbe.

2. SHA-3 maišos funkcijų konkurso finalininkai

2.1. BLAKE algoritmas

BLAKE [AHM+10] algoritmas yra sukurtas remiantis anksčiau sukurtų algoritmų komponentais.

- BLAKE algoritmo iteracijų modelis yra HAIFA – patobulinta Merkle-Damgard algoritmo versija. Taip pat naudoja druską.
- BLAKE vidinė struktūra yra vidinis platus vamzdis, kuris buvo naudojamas ir su LAKE maišos funkcija. Jis neleidžia įvykti vidinėms kolizijoms.
- BLAKE kompresijos algoritmas yra modifikuota Bernsteino srautinio šifro ChaCha versija, kurios saugumas buvo intensyviai analizuotas ir rezultatai puikūs.

BLAKE algoritmas tekina visus NIST iškeltus kriterijus.

- Pranešimo santraukos 224, 256, 384 ir 512 bitų ilgio.
- Naudojami tokie patys parametrų dydžiai kaip ir SHA-2 algoritme.
- Vieno praėjimo srauto modelis.
- Maksimalus pranešimo ilgis bent $2^{64} - 1$ bitai.

BLAKE įgyvendino papildomai tokias savybes:

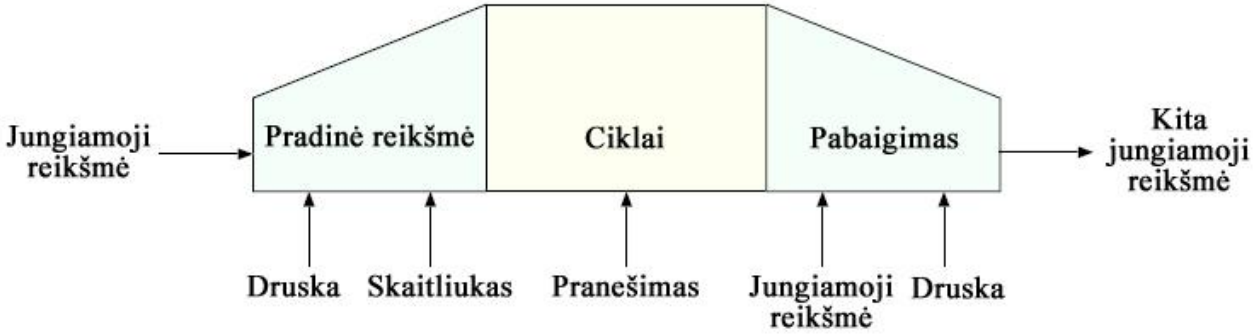
1) maišos funkcija dirba su druska;
2) galimas paralelizavimas;
3) leidžiamas vykdymo efektyvumo savybių pasirinkimas;
4) tinkamas paprastoms aplinkoms.

2.1.1. Bendroji BLAKE algoritmo informacija

BLAKE algoritmų šeimą sudaro 4 funkcijos: BLAKE-224, BLAKE-256, BLAKE-384 ir BLAKE-512. Kaip ir SHA-2 atveju BLAKE-256 yra 32 bitų versijos, o BLAKE-512 yra 64 bitų versijos. Kiti parametrai yra išvesti naudojant skirtingas pradines reikšmes, praplėtimą ir skaidytą išvedimą.

Algoritmas	Žodis	Pranešimas	Blokas	Rezultatas	Druska
BLAKE-224	32	$< 2^{64}$	512	224	128
BLAKE-256	32	$< 2^{64}$	512	256	128
BLAKE-384	64	$< 2^{128}$	1024	384	256
BLAKE-512	64	$< 2^{128}$	1024	512	256

2.1.1.1 Lentelė. BLAKE algoritmo parametrų reikšmės [AHM+10].



Pav. 2.1.1.1. BLAKE algoritmo veikimo schema [AHM+10].

BLAKE maišos funkcija naudoja HAIFA iteracijos metodą: kompresijos funkcija priklauso nuo druskos ir bitų, apdorotų maišos funkcijos, kiekio (skaitliuko), kad suspaustų kiekvieną pranešimo bloką su atitinkama funkcija. BLAKE kompresijos funkcija yra panaudota iš LAKE algoritmo (žr. 2.1.1.1 pav.): didelė pradinė būsena yra inicializuojama pradine reikšme, druska ir skaitliuku. Tada reikšmė yra atnaujinama visuose cikluose ir galutinis suspaustas rezultatas gražinamas į seką. Tokia strategija yra pavadinta vidiniu plačiu vamzdžiu (*angl. local wide-pipe*).

Vidinė kompresijos funkcijos dalis yra pristatoma kaip 4×4 žodžių matrica. BLAKE-256 ciklai yra modifikuoti srautinio šifro ChaCha dvigubi ciklai: pirmiausia, nepriklausomai vienas nuo kito yra atnaujinami visi keturi stulpeliai, o po to keturios įstrižainės. Atnaujinant kiekvieną stulpelį ar įstrižainę, du pranešimo žodžiai yra įvesties duomenys priklausomai nuo ciklo keitinio. Kiekvienas ciklas parametrizuojamas skirtingomis konstantomis, taip sumažinant panašumo galimybę. Po ciklų sekos, būsena yra sumažinama per pusę, priskiriant pradinę reikšmę ir druską.

BLAKE algoritmo realizacija nereikalauja galingos techninės įrangos ir yra greita tiek techninėje, tiek programinėje dalyje.

Tikimasi, kad visoms BLAKE maišos funkcijoms neturėtų būti sėkmingesnių atakų nei standartiniai brutaliosios jėgos metodai:

- ieškant kolizijų su ta pačia ar skirtinga druska;
- ieškant atvaizdžių su sutartine druska.

Blake algoritmas turėtų būti saugus naudojant atsitiktinę maišą. Turėtų būti neįmanoma išskirti BLAKE pavyzdžių taikant nežinomą druską.

Galima išskirti pagrindinius algoritmo privalumus.

1. Algoritmo yra paprastas struktūra, maiša naudojama kartu su druska.
2. Algoritmas yra greitas tiek programinės, tiek techninės įrangų lygiuose.
3. Paremtas ilgai testuotu ChaCha algoritmo komponentu.
4. Atsparus įprastam antro atvaizdžio atakai (*angl. second preimage attack*).
5. Atsparus paslėpto kanalo atakai (*angl. side-channel attack*).
6. Atsparus pranešimo ilgio praplėtimui.

Deja nėra algoritmų, neturinčių trūkumų. BLAKE algoritmui priskiriami 3 trūkumai.

1. Pranešimo ilgis yra apribotas iki 2^{64} ir 2^{128} atitinkamai BLAKE-256 ir BLAKE-512 algoritmuose.
2. Atsparumas Joux‘s daugiakolizijoms yra panašus į SHA-2.
3. Fiksuoti taškai surasti per trumpesnį laiką, nei idealios funkcijos atveju.

Toliau pristatysiu BLAKE algoritmo veikimo schemą.

2.1.2. BLAKE-256

BLAKE-256 maišos funkcija dirba su 32-bitų žodžiais ir grąžina 32 baitų maišos reikšmę. Toliau nuosekliai pristatysiu šios funkcijos konstantas, kompresijos funkciją ir iteracijas.

Konstantos

BLAKE-256 naudoja tokias pat pradines reikšmes kaip ir SHA-256:

$$IV_0 = 6A09E667 \qquad IV_1 = BB67AE85$$
$$IV_2 = 3C6EF372 \qquad IV_3 = A54FF53A$$
$$IV_4 = 510E527F \qquad IV_5 = 9B05688C$$
$$IV_6 = 1F83D9AB \qquad IV_7 = 5BE0CD19$$

BLAKE-256 naudojasi 16 konstantų:

$$c_0 = 243F6A88 \qquad c_1 = 85A308D3$$
$$c_2 = 13198A2E \qquad c_3 = 03707344$$
$$c_4 = A4093822 \qquad c_5 = 299F31D0$$
$$c_6 = 082EFA98 \qquad c_7 = EC4E6C89$$
$$c_8 = 452821E6 \qquad c_9 = 38D01377$$
$$c_{10} = BE5466CF \qquad c_{11} = 34E90C6C$$
$$c_{12} = C0AC29B7 \qquad c_{13} = C97C50DD$$
$$c_{14} = 3F84D5B5 \qquad c_{15} = B5470917$$

BLAKE funkcijų naudojami keitiniai $\{0, \dots, 15\}$ pateikti žemiau esančioje lentelėje 2.1.2.1.

σ_0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
σ_1	14	10	4	8	9	15	13	6	1	12	0	2	11	7	5	3
σ_2	11	8	12	0	5	2	15	13	10	14	3	6	7	1	9	4
σ_3	7	9	3	1	13	12	11	14	2	6	5	10	4	0	15	8
σ_4	9	0	5	7	2	4	10	15	14	1	11	12	6	8	3	13
σ_5	2	12	6	10	0	11	8	3	4	13	7	5	15	14	1	9
σ_6	12	5	1	15	14	13	4	10	0	7	6	3	9	2	8	11
σ_7	13	11	7	14	12	1	3	9	5	0	15	4	8	6	2	10
σ_8	6	15	14	9	11	3	0	8	12	2	13	7	1	4	10	5
σ_9	10	2	8	4	7	6	1	5	15	11	9	14	3	12	13	0

2.1.2.1 Lentelė. BLAKE funkcijų naudojami keitiniai $\{0, \dots, 15\}$

Kompresijos funkcija

BLAKE-256 kompresijos funkcijai yra paduodami keturi kintamieji:

- tarpinė maišos funkcijų grandinės reikšmė $\mathrm{h} = \mathrm{h}_0, \dots, \mathrm{h}_7$;
- pranešimo blokas $\mathrm{m} = \mathrm{m}_0, \dots, \mathrm{m}_{15}$;
- druska $\mathrm{s} = \mathrm{s}_0, \dots, \mathrm{s}_3$;
- skaitliukas $\mathrm{t} = \mathrm{t}_0, \mathrm{t}_1$.

Šie keturi įvesties duomenys iš viso atvaizduoja 30 žodžių (t.y. 120 baitų = 960 bitų). Galime aprašyti kompresijos funkciją h'

$$h' = \boldsymbol{compress}(h, m, s, t).$$

Pradinių reikšmių suformavimas

Yra suformuoja 16-os žodžių būsena $v_0, \dots, v_{15}$ taip, kad skirtingi įvesties duomenys pagamina skirtingas pradines būsenas. Būsena yra apibrėžiama kaip 4×4 matrica, kuri yra užpildoma taip:

$$\begin{pmatrix} v_0 & v_1 & v_2 & v_3 \\ v_4 & v_5 & v_6 & v_7 \\ v_8 & v_9 & v_{10} & v_{11} \\ v_{12} & v_{13} & v_{14} & v_{15} \end{pmatrix} \leftarrow \begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 & h_7 \\ s_0 \oplus c_0 & s_1 \oplus c_1 & s_2 \oplus c_2 & s_3 \oplus c_3 \\ t_0 \oplus c_4 & t_0 \oplus c_5 & t_1 \oplus c_6 & t_1 \oplus c_7 \end{pmatrix}$$

Ciklo funkcija

Kai sukuriama būsena v, kompresijos funkcija yra iteruojama 14 kartų. Vienas ciklas yra būsenos v transformacija, kuri apskaičiuoja

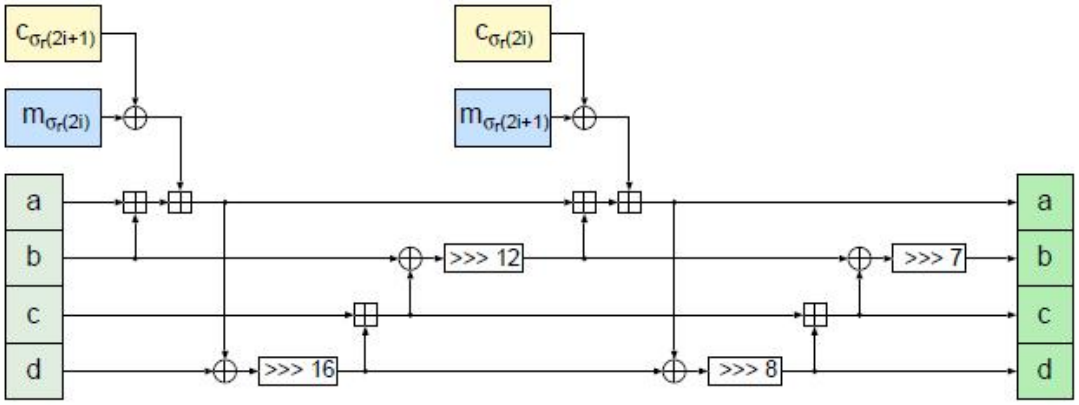
$$G_0(v_0, v_4, v_8, v_{12}) \quad G_1(v_1, v_5, v_9, v_{13}) \quad G_2(v_2, v_6, v_{10}, v_{14}) \quad G_3(v_3, v_7, v_{11}, v_{15})$$
$$G_4(v_0, v_5, v_{10}, v_{15}) \quad G_5(v_1, v_6, v_{11}, v_{12}) \quad G_6(v_2, v_7, v_8, v_{13}) \quad G_7(v_3, v_4, v_9, v_{14})$$

kur, cikle r, funkcija $G_i(a, b, c, d)$ įgauna reikšmes

$$a \leftarrow a + b + \left(m_{\sigma_r(2i)} \oplus c_{\sigma_r(2i+1)}\right)$$
$$d \leftarrow (d \oplus a) \ggg 16$$
$$c \leftarrow c + d$$
$$b \leftarrow (b \oplus c) \ggg 12$$
$$a \leftarrow a + b + \left(m_{\sigma_r(2i+1)} \oplus c_{\sigma_r(2i)}\right)$$
$$d \leftarrow (d \oplus a) \ggg 8$$
$$c \leftarrow c + d$$
$$b \leftarrow (b \oplus c) \ggg 7$$

Pirmieji keturi funkcijos G kvietimai $G_0, \dots, G_3$ gali būti vykdomi paraleliai, nes kiekvienas jų redaguoja atskirą matricos stulpelį. Procedūra, apskaičiuojanti $G_0, \dots, G_3$, vadinama *stulpelio žingsniu*. Taip pat paskutinai keturi funkcijos G kvietimai $G_4, \dots, G_7$ gali būti vykdomi paraleliai, nes kiekvienas jų redaguoja atskirą matricos įstrižainę, todėl tokia procedūra yra vadinama *įstrižainės žingsniu*. Kai ciklas $r > 9$, naudojamo keitinio indeksas apskaičiuojamas taip: $\sigma_{r \bmod 10}$ (pavyzdžiui, paskutiniame cikle $r = 13$ naudosime keitinį, kurio indeksas yra $\sigma_{13 \bmod 10} = \sigma_3$).

Piešinukai 2.1.2.1 ir 2.1.2.2 grafiškai pavaizduoja funkcijos G_i, stulpelio ir įstrižainės žingsnių veikimą.



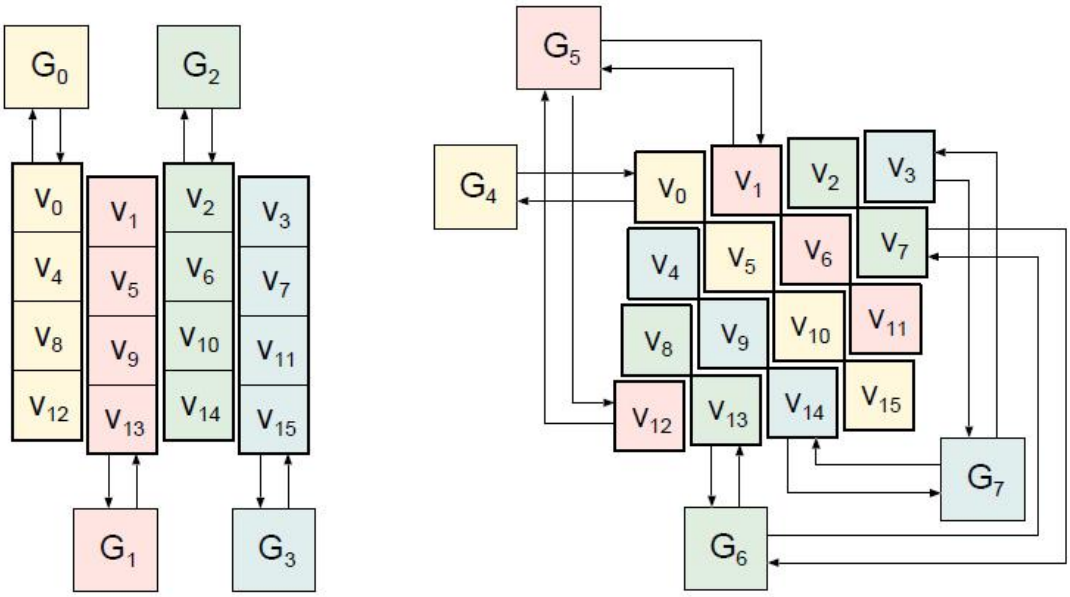
Pav. 2.1.2.1. G_i funkcija

Algoritmo pabaigimas

Atlikus ciklų seką, iš pradinių reikšmių $h = h_0, \dots, h_7$, būsenų $v_0, \dots, v_{15}$ ir druskos $s = s_0, \dots, s_3$ gaunama nauja maišos funkcijų grandinės reikšmių dalis $h' = h_0', \dots, h_7'$:

$$h_0' \leftarrow h_0 \oplus s_0 \oplus v_0 \oplus v_8$$
$$h_1' \leftarrow h_1 \oplus s_1 \oplus v_1 \oplus v_9$$
$$h_2' \leftarrow h_2 \oplus s_2 \oplus v_2 \oplus v_{10}$$
$$h_3' \leftarrow h_3 \oplus s_3 \oplus v_3 \oplus v_{11}$$
$$h_4' \leftarrow h_4 \oplus s_0 \oplus v_4 \oplus v_{12}$$

$$h_5' \leftarrow h_5 \oplus s_1 \oplus v_5 \oplus v_{13}$$
$$h_6' \leftarrow h_6 \oplus s_2 \oplus v_6 \oplus v_{14}$$
$$h_7' \leftarrow h_7 \oplus s_3 \oplus v_7 \oplus v_{15}$$



Pav. 2.1.2.2. Stulpelio ir įstrižainės žingsniai

Maišos taikymas pranešimui

Dabar aprašysiu, kaip maišos funkcija yra taikoma koduojant turimą pranešimą m, kurio ilgis bitais yra $l < 2^{64}$. Kaip ir visuose iteruojamose maišos funkcijose, pirmiausias pranešimas yra praplečiamas (BLAKE naudoja labai panašią į HAIFA algoritmo praplėtimo taisyklę) ir tada suskaidytas į blokus yra apdorojamas kompresijos funkcijos.

Pranešimo praplėtimas

Pirmiausia pranešimas yra praplečiamas taip, kad jo ilgis atitiktų 447 moduliu 512. Pranešimo praplėtimas yra vykdomas taip: yra pridedamas bitas „1“ ir likusi dalis po jo užpildoma „0“. Visada yra pridedamas bent vienas bitas. Tada sekos pabaigoje vėl pridedamas bitas „1“ ir 64 bitų pranešimo ilgis (be ženklo, labiausiai reikšmingas baitas eina pirmas):

$$m \leftarrow m \parallel 1000 \ldots 0001 \langle l \rangle_{64}.$$

Ši procedūra garantuoja, kad praplėsto pranešimo ilgis bus 512 kartotinis.

Maišos funkcijos iteracijos

Prieš pradedant maišos funkcijos iteracijas, pranešimas yra suskaidomas į 16 žodžių blokus $m^0, \ldots, m^{N-1}$. Tarkime l^i žymi pranešimo $m^0, \ldots, m^i$ bitų kiekį, t.y. pranešimo dalies, atmetus praplėtimą. Pavyzdžiui, jeigu originalaus pranešimo ilgis yra 600 bitų, tai praplėstas pranešimas turės du blokus ir $l^0 = 512$ ir $l^1 = 600$. Kai paskutinis blokas neturi nei vieno originalaus pranešimo bito, turime specifinę situaciją (pvz., 1020 bitų pranešimas, praplėstas turės tris blokus – atitinkamai 512, 508 ir 0 originalaus pranešimo bitų, tad nustatome $l^0 = 512$, $l^1 = 1020$ ir $l^2 = 0$). Pagrindinė taisyklė: jeigu paskutinis blokas neturi nei vieno originalaus pranešimo bito, tuomet skaitliukas prilyginamas 0. Tai garantuoja, kad jeigu $i \neq j$, tai ir $l_i \neq l_j$.

Druskos reikšmę s pasirenka naudotojas bei jai priskiriama nulinė reikšmė, jeigu druskos naudoti nereikia (t.y. $s_0 = s_1 = s_2 = s_3 = 0$). Tuomet praplėsto pranešimo maiša yra:

$h^0 \leftarrow IV$
$\boldsymbol{for}\ i = 0, \dots, N-1$
  $h^{i+1} \leftarrow \boldsymbol{compress}(h^i, m^i, s, l^i)$
$\boldsymbol{return}\ h^N$

Procedūra, atliekanti BLAKE-256 maišos funkcijos taikymą pranešimui m, yra žymima BLAKE-256$(m, s) = h^N$, kur m yra nepraplėstas pranešimas ir s yra druska. Kai nenaudojama druska, maišos funkciją pranešimui galima žymėti ir taip: BLAKE-256$(m) = h^N$.

BLAKE-512, BLAKE-224 ir BLAKE-384 veikia panašiai, kaip BLAKE-256 (žr. 1 priedą).

2.2. Groestl algoritmas

Groestl [GKM+11] algoritmas yra maišos funkcija iteruota su kompresijos funkcija iš dviejų fiksuoto dydžio, atskirų keitinių. Algoritmo projektas yra paremtas principais, kurie labai skirtingi nei naudoti SHA funkcijų šeimoje. Du keitiniai yra sukonstruoti naudojantis plataus tako strategija, kuri leidžia tvirtai teigti, kad Groestl algoritmas yra atsparus didžiajai klasei kriptoanalitinių atakų.

2.2.1. Bendrieji maišos algoritmo tikslai

Groestl maišos algoritmo autoriai iškėlė šiuos bendruosius tikslus.

- Analizės paprastumas, nes algoritmą sudaro nedidelis kiekis keitinių, o ne blokinis šifras.
- Saugi konstrukcija (darant prielaidą, kad keitiniai yra idealūs).
- Nėra išskiriama tam tikra platforma ar žodžio dydis.

Klaidoms tolerantiška struktūra

Neatsitiktinis keitinių elgesys nebūtinai lemia neidealias kompresijos funkcijos savybes. Atakos, nukreiptos prieš kompresijos funkciją, nebūtinai paveiks pačią maišos funkciją.

- Vidinė algoritmo būsena yra žymiai didesnė nei gaunamas rezultatas, todėl visos žinomos atakos yra atremiamos.
- Žinomi metodai, kuriuos taikant gaunamas ne idealus keitinių veikimas, realizuoti tik supaprastintuose algoritmo variantuose.
- Atakos, nukreiptos prieš kompresijos funkciją, nebūtinai paveikia pačią maišos funkciją.
- Nėra žinoma tokių kompresijos funkcijų atakų, kurios pasiektų vadinamąją „apatinę“ atsparumo ribą.

Algoritmo struktūros svarba kompresijos funkcijoje

Tradicinių maišos algoritmų struktūra yra paremta blokinių šifrų savybėmis, kurios yra gerai suprantamos. Tačiau rakto perdavimas blokiniame šifravime tampa galima saugumo spraga. Dėl šios priežasties Groestl algoritmas yra paremtas kelių individualių keitinių apskaičiavimu, o ne keitinių, numeruojamu nurodytu raktu, grupe. Galima išskirti šiuos tokios algoritmo struktūros privalumus:

- Nėra silpno rakto atakos galimybės.
- Spartesnis algoritmo veikimas.
- Paprastumas.

2.2.2. Groestl maišos algoritmo struktūra

Groestl yra maišos funkcijų rinkinys, gražinantis nuo 1 iki 64 baitų rezultato reikšmę. Toliau detaliai pristatysiu Groestl maišos funkcijas.

Maišos funkcijos konstrukcija

Groestl maišos funkcijos iteracijos būdu taiko kompresijos funkciją f. Pranešimas M yra praplečiamas ir padalinamas į l-bitų pranešimo blokus $m_1, \dots, m_t$, kurių kiekvienas yra apdorojamas eilės tvarka. Pradinė l-bitų reikšmė $h_0 = iv$ yra apibrėžta, o pranešimo blokas m_i yra panaudojamas taip

$$h_i \leftarrow f(h_{i-1}, m_i), \text{ kur } i = 1, \dots, t.$$

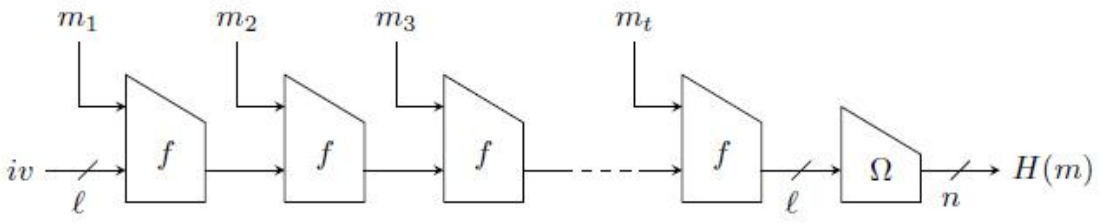
Taigi f pažymi dvi l-bitų ilgio įvestis bei gauna l-bitų ilgio išvesties duomenis. Pirmoji įvestis yra vadinama sekos įvestimi, o antrasis įvesties duomuo – pranešimo bloku. Kai Groestl algoritmas gražina iki

256 bitų rezultatą, l yra 512 bitai. Didesniems rezultatams l yra 1024 bitai.

Kai apdorojamas paskutinis pranešimo blokas, apskaičiuojamas maišos funkcijos rezultatas $H(M)$

$$H(M) = \Omega(h_t),$$

kur Ω yra išvesties transformacija. Ω dydis yra n bitų, bei $n \leq 2 \cdot l$ (žr. 2.2.2.1 pav.).



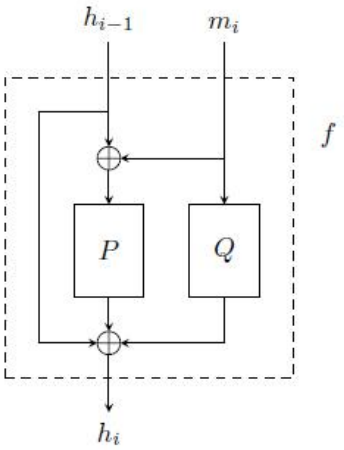
Pav. 2.2.2.1. Groestl maišos funkcija [GKM+11]

Kompresijos funkcijos struktūra

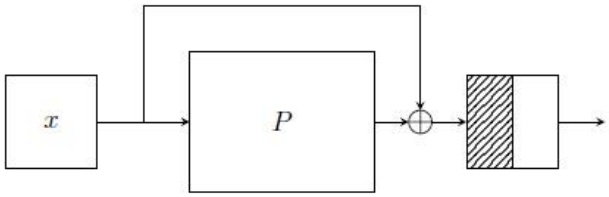
Kompresijos funkcija *f* yra paremta dviem *l* bitų ilgio keitiniais *P* ir *Q*. Ši funkcija yra aprašoma taip:

$$f(h, m) = P(h \oplus m) \oplus Q(m) \oplus h.$$

Funkcijos *f* struktūra yra pavaizduota 2.2.2.2 pav.



Pav. 2.2.2.2. Kompresijos funkcija *f* [GKM+11]



Pav. 2.2.2.3. Rezultato transformacija Ω apskaičiuoja $P(x) \oplus x$, sutrumpina rezultatą ir grąžina n bitų [GMK+11]

Maišos funkcijos rezultato transformacija

Tarkime, kad $trunk_n(x)$ yra operacija atmetanti visus bitus išskyrus n paskutiniųjų. Išvesties transformacija Ω yra pavaizduota 2.2.2.3.pav. ir aprašoma taip

$$\Omega(\mathrm{x}) = \mathrm{trunk_n}(\mathrm{P}(\mathrm{x}) \oplus \mathrm{x}).$$

Keitinių P ir Q struktūra

Kaip buvo paminėta anksčiau kompresijos funkcija f gali būti dviejų tipų – trumpo ir ilgo pranešimų rezultatai. Kiekvienas funkcijos tipas naudoja jam paskirtą ketinių *P* ir *Q* porą. Taigi iš viso yra apibrėžti keturi keitiniai. Reikiami keitiniai taikomi pagal naudojamą 512 arba 1024 bitų algoritmą.

P ir *Q* keitinių struktūra buvo sukurta pagal Rijndael blokinį šifrą. Tai reiškia, kad struktūra sudaryta iš ciklų skaičiaus *R*, kuriuose yra tam tikras ciklo transformacijų skaičius. Kadangi keitiniai *P* ir *Q* yra didesni nei 128 bitai, kurie buvo naudojami Rijndael algoritme, ciklo transformacijos yra apibrėžtos iš naujo. Groestl naudoja keturias ciklo transformacijas kiekvienam keitiniui. Jos yra tokios:

- *AddRoundConstant* (pridėti ciklo konstantą);
- *SubBytes* (pakeisti baitus);
- *ShiftBytes* (pastumti baitus);

- *MixBytes* (sumaišyti baitus).

Dideliems keitiniams P_{1024} ir Q_{1024} yra naudojama *ShiftBytesWide* transformacija. Kiekvienam keitiniui *AddRoundConstant* ir *ShiftBytes* transformacijos yra skirtingos, o *SubBytes* ir *MixBytes* visada išlieka vienodos.

Vienas ciklas yra sudarytas iš šių keturių transformacijų, atliekamų 2.2.2.4 paveikslėlyje nurodyta tvarka. Taigi,

$$R = MixBytes \circ ShiftBytes \circ SubBytes \circ AddRoundConstant.$$

Visi ciklai yra atliekami šiuo eiliškumu. Ciklų kiekį pažymime *r*.

Pav. 2.2.2.4. Vieno Groestl ciklo keitiniai P ir Q yra keturių transformacijų kompozicija [GKM+11].

Transformacijos atliekamos su baitų matrica *A*. Trumpiems pranešimams matrica yra sudaryta iš 8 eilučių ir 8 stulpelių (žymime *v*), dideliems – 8 eilučių ir 16 stulpelių. Toliau trumpai aprašysiu, kaip baitų seką pažymėti matrica ir vėl konvertuoti į seką bei kiekvieną iš atliekamų ciklo transformacijų.

Baitų sekos užrašymas matrica ir atvirkščiai

Kadangi Groestl algoritmas dirba su baitais, reikia aprašyti, kaip ją atvaizduoti matricoje *A*. Atvaizdavimas vykdomas panašiai, kaip Rijndael algoritme. Taigi 64-baitų seka $00\ 01\ 02 \dots 3f$ į 8×8 matricą yra atvaizduojama taip:

$$\begin{bmatrix} 00 & 08 & 10 & 18 & 20 & 28 & 30 & 38 \\ 01 & 09 & 11 & 19 & 21 & 29 & 31 & 39 \\ 02 & 0a & 12 & 1a & 22 & 2a & 32 & 3a \\ 03 & 0b & 13 & 1b & 23 & 2b & 33 & 3b \\ 04 & 0c & 14 & 1c & 24 & 2c & 34 & 3c \\ 05 & 0d & 15 & 1d & 25 & 2d & 35 & 3d \\ 06 & 0e & 16 & 1e & 26 & 2e & 36 & 3e \\ 07 & 0f & 17 & 1f & 27 & 2f & 37 & 3f \end{bmatrix}.$$

8×16 dydžio matricai jis metodas yra praplečiamas kol užpildomi visi 16 stulpelių. Norėdami iš matricos *A* gauti baitų seką, vykdome atvirškinę operaciją.

AddRoundConstant

AddRoundConstant transformacija prie matricos A prideda (atlieka operaciją XOR) nuo ciklo priklausančios konstantos reikšmę. Tiksliau, cikle *i AddRoundConstant* transformacija atnaujina matricą *A*

$$A \leftarrow A \oplus C[i],$$

kur $C[i]$ yra ciklo konstanta cikle i. Keitiniai *P* ir *Q* turi skirtingas ciklo konstantas.

Ciklo konstantos keitiniams P_{512} ir Q_{512} yra

$$P_{512}\ \colon \mathrm{C[i]} = \begin{bmatrix} 00 \oplus \mathrm{i} & 10 \oplus \mathrm{i} & 20 \oplus \mathrm{i} & 30 \oplus \mathrm{i} & 40 \oplus \mathrm{i} & 50 \oplus \mathrm{i} & 60 \oplus \mathrm{i} & 70 \oplus \mathrm{i} \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \end{bmatrix}$$

ir

$$Q_{512}\ \colon \mathrm{C[i]} = \begin{bmatrix} \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} \\ \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} \\ \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} \\ \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} \\ \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} \\ \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} \\ \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} \\ \mathrm{ff} \oplus \mathrm{i} & \mathrm{ef} \oplus \mathrm{i} & \mathrm{df} \oplus \mathrm{i} & \mathrm{cf} \oplus \mathrm{i} & \mathrm{bf} \oplus \mathrm{i} & \mathrm{af} \oplus \mathrm{i} & 9\mathrm{f} \oplus \mathrm{i} & 8\mathrm{f} \oplus \mathrm{i} \end{bmatrix}$$

kur *i* yra ciklo numeris, užrašytas 8 bitų reikšme, o visos kitos reikšmės pateiktos šešioliktainėje sistemoje.

Panašiai ciklo konstantos keitiniams P_{1024} ir Q_{1024} yra

$$P_{1024}\ \colon \mathrm{C[i]} = \begin{bmatrix} 00 \oplus \mathrm{i} & 10 \oplus \mathrm{i} & 20 \oplus \mathrm{i} & 30 \oplus \mathrm{i} & 40 \oplus \mathrm{i} & 50 \oplus \mathrm{i} & 60 \oplus \mathrm{i} & 70 \oplus \mathrm{i} & \cdots & \mathrm{f}0 \oplus \mathrm{i} \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & \cdots & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & \cdots & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & \cdots & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & \cdots & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & \cdots & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & \cdots & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & \cdots & 00 \end{bmatrix}$$

ir

$$Q_{1024}\ \colon \mathrm{C[i]} = \begin{bmatrix} \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \cdots & \mathrm{ff} \\ \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \cdots & \mathrm{ff} \\ \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \cdots & \mathrm{ff} \\ \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \cdots & \mathrm{ff} \\ \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \cdots & \mathrm{ff} \\ \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \cdots & \mathrm{ff} \\ \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \mathrm{ff} & \cdots & \mathrm{ff} \\ \mathrm{ff} \oplus \mathrm{i} & \mathrm{ef} \oplus \mathrm{i} & \mathrm{df} \oplus \mathrm{i} & \mathrm{cf} \oplus \mathrm{i} & \mathrm{bf} \oplus \mathrm{i} & \mathrm{af} \oplus \mathrm{i} & 9\mathrm{f} \oplus \mathrm{i} & 8\mathrm{f} \oplus \mathrm{i} & \cdots & 0\mathrm{f} \oplus \mathrm{i} \end{bmatrix}$$

kur *i* taip pat yra ciklo numeris, užrašytas 8 bitų reikšme.

SubBytes

SubBytes transformacija kiekvieną matricos *A* baitą pakeičia Sbox *S* reikšme. Sbox yra naudojama tokia pati, kokia ir Rijndael algoritme, ir jos aprašymas pateiktas priede Nr. 2. Taigi, jei $a_{i,j}$ yra elementas, esantis *i* eilutėje ir *j* matricos *A* stulpelyje, *SubBytes* atlieka tokią transformaciją:

$$a_{i,j} = S(a_{i,j}),\ 0 \le i \le 8, 0 \le j \le v.$$

ShiftBytes ir ShiftBytesWide

ShiftBytes ir *ShiftBytesWide* periodiškai baitus pastumia eilute į kairę per tam tikrą pozicijų skaičių. Tarkime, kad $\sigma = [\sigma_0, \sigma_1, \dots, \sigma_7]$ yra skirtingi sveikieji skaičiai iš intervalo nuo 0 iki *v-1*. Tada *ShiftBytes* eilutėje *i* pastumia visus baitus matricos pozicijose σ_i į kairę. Vektorius σ keitiniams *P* ir *Q* yra skirtingas. Atlikdami *ShiftBytes* transformaciją keitiniui *P* naudojame $\sigma = [0,1,2,3,4,5,6,7]$, keitiniui $Q - \sigma = [1,3,5,7,0,2,4,6]$. Panašiai atlikdami *ShiftBytesWide* transformaciją, keitiniams *P* ir *Q* atitinkamai naudojame

$\sigma = [0,1,2,3,4,5,6,11]$ ir $\sigma = [1,3,5,11,0,2,4,6]$.

MixBytes

Atliekant *MixBytes* transformaciją, kiekvienas matricos stulpelis yra transformuojamas nepriklausomai nuo kitų. Norėdami aprašyti šią transformaciją pirmiausia turime apibrėžti baigtinį lauką F_{256}. Jis yra apibrėžiamas taip pat kaip ir Rijndael algoritme pasinaudojant kraštutiniu daugianariu $x^8 \oplus x^4 \oplus x^3 \oplus x \oplus 1$. Matricos *A* elementai gali būti traktuojami kaip F_{256} nariai, t.y. kaip 7 laipsnio daugianariai su koeficientais iš $\{0,1\}$. Mažiausiai reikšminis bitas kiekviename baite žymi x^0 koeficientą ir t.t.

MixBytes padaugina kiekvieną matricos *A* stulpelį iš konstantos matricos *B*. Taigi visos matricos *A* transformacija gali būti užrašyta taip:

$$A \leftarrow B \times A.$$

Matrica *B* yra užrašoma taip

$$B = \begin{bmatrix} 02 & 02 & 03 & 04 & 05 & 03 & 05 & 07 \\ 07 & 02 & 02 & 03 & 04 & 05 & 03 & 05 \\ 05 & 07 & 02 & 02 & 03 & 04 & 05 & 03 \\ 03 & 05 & 07 & 02 & 02 & 03 & 04 & 05 \\ 05 & 03 & 05 & 07 & 02 & 02 & 03 & 04 \\ 04 & 05 & 03 & 05 & 07 & 02 & 02 & 03 \\ 03 & 04 & 05 & 03 & 05 & 07 & 02 & 02 \\ 02 & 03 & 04 & 05 & 03 & 05 & 07 & 02 \end{bmatrix}.$$

Ši matrica yra žiedinė, t.y. kiekviena nauja eilutė yra lygi prieš tai ėjusiai pasuktai į dešinę per vieną poziciją. Trumpai būtų galima užrašyti $B = circ(02, 02, 03, 04, 05, 03, 05, 07)$.

Ciklų skaičius

Ciklų skaičius *r* yra suderintas saugumo parametras. Rekomenduojamos tokios *r* ciklų reikšmės keturiems keitiniams.

Keitiniai	Rezultato dydis	Rekomenduojamas ciklų skaičius r
P_{512} ir Q_{512}	8-256	10
P_{1024} ir Q_{1024}	264-512	14

Pradinės reikšmės

Žemiau esanti lentelė rodo pradines reikšmes reikalaujamiems išvestiems dydžiams.

N	iv_n				
224	00	...	00	00	$e0$
256	00	...	00	10	00
384	00	...	00	10	80
512	00	...	00	00	00

Pranešimo praplėtimas

Kaip buvo minėta, kiekvieno bloko ilgis yra *l*. Kad būtų galima apdoroti kintančio ilgio pranešimus, reikia naudoti praplėtimo funkciją *pad*. Praplėtimo funkcija ima *N* bitų seką *x* ir grąžina praplėstą eilutę $x^* = pad(x)$, kurios ilgis yra *l* kartotinis.

Praplėtimo funkcija atlieka tokius veiksmus:

1. prie eilutės x galo prideda vieną bitą ‘1‘;
2. prideda $w = -N - 65 \bmod l$ ‘0‘;
3. prijungia 64 bitų $(N + w + 65)/l$ išraišką. Šis skaičius nurodo pranešimo blokų skaičių praplėstoje jo išraiškoje.

2.3. JH algoritmas

JH [Wu11] algoritmų šeimą sudaro keturi maišos algoritmai: JH-224, JH-256, JH- ir JH-512. Šie algoritmai yra produktyvūs techniniame lygmenyje, nes yra sukurti remiantis paprastais komponentais. Taip

pat jie yra produktyvūs ir programinės įrangos lygmenyje drauge su SIMD instrukcijomis, kadangi paprasti komponentai gali būti apskaičiuojami ir paraleliai bei būti tinkami bitų skaidymo realizacijoje.

JH algoritme siūloma nauja kompresijos funkcijos struktūra, skirta sukonstruoti kompresijos funkciją iš didelio blokinio šifro su pastoviu raktu. Taip pat apibendrinama AES struktūros metodologija didelėse dimensijose taip, kad iš mažų komponentų gali būti lengvai sukonstruojami dideli blokiniai šifrai. Atsižvelgiant į naują kompresijos funkcijos struktūrą ir apibendrintą AES dizaino metodologiją, JH algoritmo kompresijos funkcijos saugumas gali būti analizuojamas labai lengvai.

JH algoritmas yra stiprus saugumo atžvilgiu. Kiekvienas pranešimo blokas sudarytas iš 64 baitų. Pranešimo blokas leidžiamas per 42 ciklų kompresijos funkciją, kuri apima 10752 4×4 bitų dydžio Sbox. Buvo pastebėta, kad skirtumo takas kompresinėsse funkcijose apima daugiau nei 700 aktyvių Sbox. Didelis aktyvių Sbox skaičius užtikrina, kad JH yra stiprus prieš skirtumines atakas.

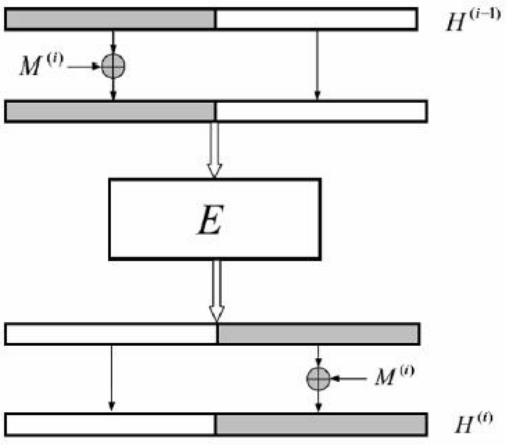
JH algoritme naudojamos dvi technikos. Buvo pasiūlyta nauja kompresijos funkcijos struktūra, skirta sukonstruoti kompresijos funkciją iš blokinio šifro naudojantis pastoviu raktu, bei panaudota apibendrinta AES dizaino metodologija, kad iš mažų komponentų sukonstruoti didelį blokinį šifrą.

Aukščiau aprašyta kompresijos funkcijos struktūra yra paprasta ir produktyvi. Kai blokinio šifro raktas yra pastovus, viduryje kompresijos funkcijoje nereikia jokių papildomų kintamųjų, todėl daug lengviau analizuoti šios funkcijos saugumą atsižvelgiant į skirtuminę ataką. Neišvalant rezultato šifro bloko, ši struktūra yra gana veiksminga.

Atsižvelgiant į skirtuminę kriptoanalizę buvo pastebėta, kad saugumo įvertinimo kaina yra mažesnė; kai tuo tarpu Davies-Meyer struktūros saugumo įvertinimo kaina yra labai didelė.

Nauja kompresijos funkcijos struktūra

JH algoritmo kompresijos funkcija yra bijekcija (blokinis šifras, turintis pastovų raktą). 2.3.1 paveikslėlyje parodyta šios funkcijos struktūra. Blokinio šifro bloko dydis yra 2m bitų. Vykdant kompresijos funkciją, 2m-bitų ilgio maišos reikšmė $H^{(i-1)}$ ir m-bitų pranešimo blokas $M^{(i)}$ yra suspaudžiami į 2m-bitų ilgio $H^{(i)}$ maišą. Pranešimo kompresijos funkcijos rezultatas yra m bitų ilgio.



Pav. 2.3.1. JH kompresijos funkcijos struktūra [Wu11].

JH kompresijos funkcijos struktūra yra paprasta ir efektyvi. Kadangi naudojamas pastovus (gautas pritaikius keitinį) blokinio šifro raktas, algoritmo vykdymo metu nereikia naudoti papildomų kintamųjų. Dėl šios priežasties yra lengva analizuoti algoritmo saugumą atitinkamų atakų metu. Netrumpinant blokinio šifro rezultato, ši algoritmo struktūra taip pat yra ir gana efektyvi.

Neapibrėžtos AES struktūros metodika

AES naudoja keitinių-derinių tinklą (*angl. substitution-permutation network (SPN)*), kurio įvesties duomenys yra dvimatis masyvas. Lyginių ciklų metu (laikoma, kad pirmasis ciklas yra nulinis) dvimačio masyvo stulpeliams yra taikoma MDS (maksimalaus atstumo atskirimo) matrica, o esant nelyginiams ciklams, ji taikoma eilutėms. Dėl eilučių posūkių, ciklų funkcijos AES algoritme yra vienodos.

JH autoriai AES struktūrą apibendrino taip, kad iš mažų komponentų būtų galima sukonstruoti didelį blokinį šifrą. Naudojant šią metodiką, įvesties bitai yra suskaidomi į $\prod_{i=0}^{d-1} \alpha_i, (\alpha_i \geq 2)$ elementus, kurie yra

d-dimensinis masyvas. Tiesiniame *r* ciklo lygyje (*r mod d*) dimensijai yra taikoma MDS. Algoritmo autoriai yra įsitikinę, kad neapibrėžtos AES struktūros metodologija yra paprasčiausias būdas iš mažų komponentų sugeneruoti didelius blokinius šifrus.

JH blokiniams šiframs konstruoti naudoja 8 dimensijų neapibrėžtos AES struktūrą. 1024 bitų įvesties duomenys yra suskaidomi į 256 4 bitų elementus, kurie suformuoja 8 dimensijų masyvą. Pastovūs ciklų raktai yra suformuojami naudojantis 6 dimensijų blokiniu šifru.

2.3.1. JH funkcijos

JH naudoja visas žemiau aprašytas funkcijas

Sbox

S_0 ir S_1 yra 4×4 bitų ilgio JH algoritme naudojami Sbox. Vietoj to, kad paprasčiausiai būtų atliekama bitų sudėties (XOR) operacija, kiekvieno ciklo konstanta parenka, kuri iš Sbox yra naudojama. Taip yra padidinamas algebrinis algoritmo sudėtingumas.

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$S_0(x)$	9	0	4	11	13	12	3	15	1	10	2	6	7	5	8	14
$S_1(x)$	3	12	6	13	5	7	1	9	15	2	0	4	11	10	14	8

Lentelė 2.3.1.1. Sbox indeksai [Wu11]

Tiesinė transformacija

Tarkime *A*, *B*, *C* ir *D* yra 4 bitų ilgio žodžiai, t.y. $A = A^0 \parallel A^1 \parallel A^2 \parallel A^3$, $B = B^0 \parallel B^1 \parallel B^2 \parallel B^3$, $C = C^0 \parallel C^1 \parallel C^2 \parallel C^3$ ir $D = D^0 \parallel D^1 \parallel D^2 \parallel D^3$. Daugianario forma A galima užrašyti taip $A^0x^3 + A^1x^2 + A^2x + A^3$; $2 \cdot A$ yra $A^1x^3 + A^2x^2 + (A^0 + A^3)x + A^0$, kur $2 \cdot A$ yra dvejetainio daugianario daugyba moduliu $x^4 + x + 1$. Tada funkcija $(C, D) = L(A, B)$ yra apskaičiuojama taip:

$$
\begin{aligned}
D^0 &= B^0 \oplus A^1; & D^1 &= B^1 \oplus A^2;\\
D^2 &= B^2 \oplus A^3 \oplus A^0; & D^3 &= B^3 \oplus A^0;\\
C^0 &= A^0 \oplus D^1; & C^1 &= A^1 \oplus D^2;\\
C^2 &= A^2 \oplus D^3 \oplus D^0; & C^3 &= A^3 \oplus D^0.
\end{aligned}
$$

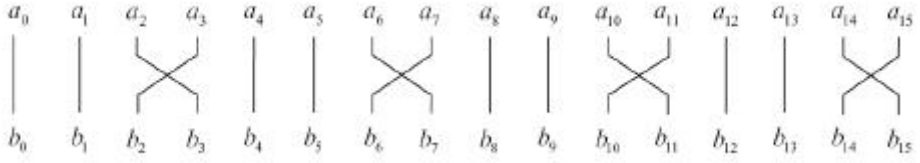
Keitinys P_d

P_d yra paprastas 2^d elementų keitinys, panašus į AES eilučių posūkius. Jį sudaro π_d, P_d' ir ϕ_d. 2^d įvesties duomenų pažymime $A = (a_0, a_1, \dots, a_{2^d-1})$, o išvesties duomenų – $B = (b_0, b_1, \dots, b_{2^d-1})$.

Keitinys π_d

π_d dirba su 2^d elementų. $B = \pi_d(A)$ yra apskaičiuojamas taip:

$$
\begin{aligned}
b_{4i+0} &= a_{4i+0} \quad i \in [0, 2^{d-2} - 1];\\
b_{4i+1} &= a_{4i+1} \quad i \in [0, 2^{d-2} - 1];\\
b_{4i+2} &= a_{4i+2} \quad i \in [0, 2^{d-2} - 1];\\
b_{4i+3} &= a_{4i+3} \quad i \in [0, 2^{d-2} - 1].
\end{aligned}
$$



Pav. 2.3.1.1. Keitinys π_4 [Wu11].

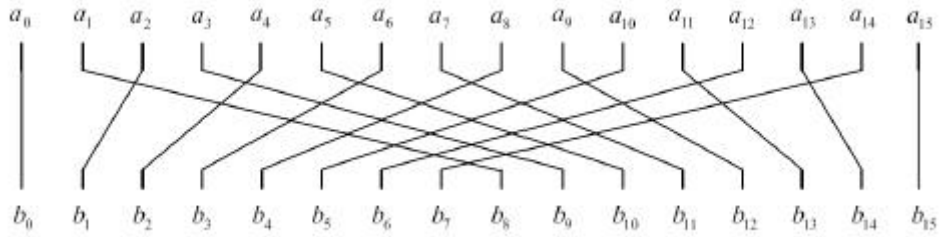
Keitinys π_4 yra pavaizduotas 2.3.1.1 pav.

Keitinys $P_d^{'}$

$P_d^{'}$ yra 2^d elementų keitinys. $B = P_d^{'}(A)$ yra apskaičiuojamas taip:

$$b_i = a_{2i} \qquad i \in [0, 2^{d-1} - 1];$$
$$b_{i+2^{d-1}} = a_{2i+1} \qquad i \in [0, 2^{d-1} - 1].$$

Keitinys $P_4^{'}$ yra pavaizduotas 2.3.1.2 pav.



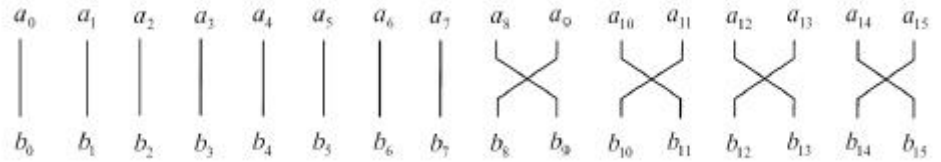
Pav. 2.3.1.2. Keitinys $P_4^{'}$ [Wu11].

Keitinys ϕ_d

ϕ_d yra 2^d elementų keitinys. $B = \phi_d(A)$ yra apskaičiuojamas taip:

$$b_i = a_i \qquad i \in [0, 2^{d-1} - 1];$$
$$b_{2i+0} = a_{2i+1} \quad i \in [2^{d-2}, 2^{d-1} - 1];$$
$$b_{2i+1} = a_{2i+0} \quad i \in [2^{d-2}, 2^{d-1} - 1].$$

Keitinys ϕ_4 yra pavaizduotas 2.3.1.3 pav.



Pav. 2.3.1.3. Keitinys ϕ_4 [Wu11].

Keitinys P_d

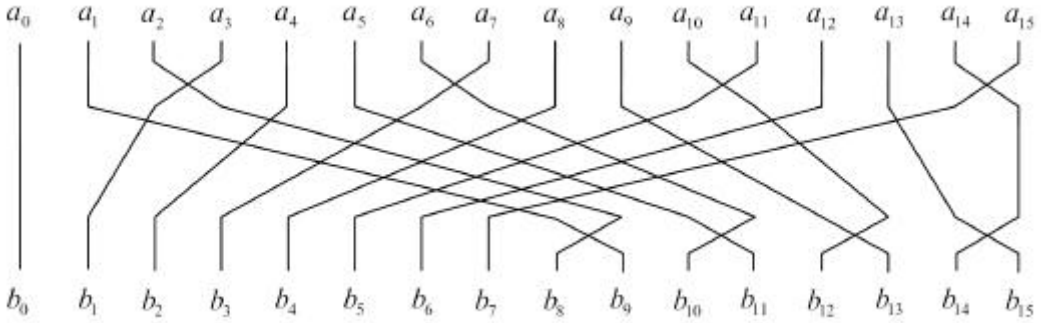
P_d yra keitinių π_d, $P_d^{'}$ ir ϕ_d kompozicija:

$$P_d = \phi_d \circ P_d^{'} \circ \pi_d$$

Keitinys P_4 yra pavaizduotas 2.3.1.4 pav.

Ciklo funkcija R_d

Ciklo funkcija R_d realizuoja neapibrėžtos AES struktūrą. Ji yra sudaryta iš trijų sluoksnių: Sbox, tiesinės transformacijos ir keitinio P_d. Ciklo funkcijos R_d tiek įvesties, tiek išvesties duomenų dydis yra 2^{d+2} bitai.

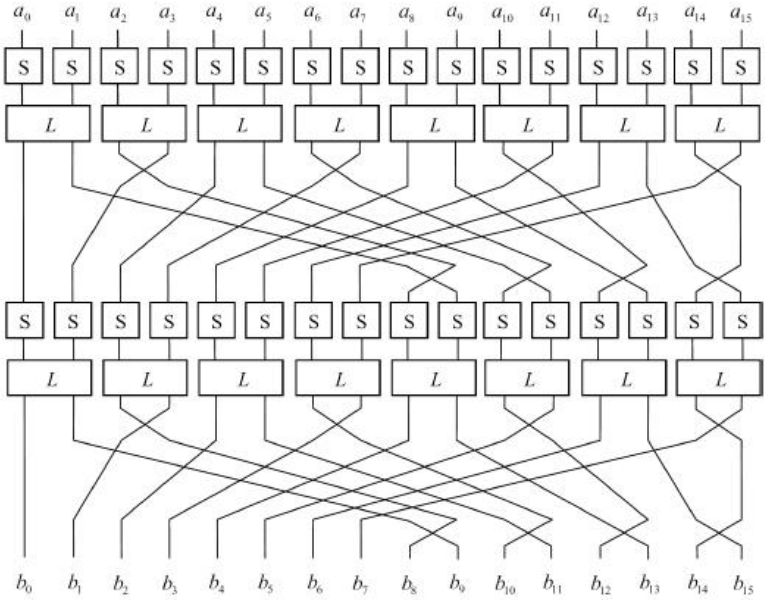


Pav. 2.3.1.4. Keitinys P_4 [Wu11].

Ciklo įvesties duomenys yra žymimi $A = \left(a_0 \parallel a_1 \parallel \cdots \parallel a_{2^d-1}\right)$, kur a_i žymi 4 bitų ilgio žodį, o rezultatas – $B = \left(b_0 \parallel b_1 \parallel \cdots \parallel b_{2^d-1}\right)$, kur b_i žymi 4 bitų ilgio žodį. 2^d bitų ilgio ciklo konstanta *r*-ajame cikle yra žymima $C_r^{(d)} = C_r^{(d),0} \parallel C_r^{(d),1} \cdots \parallel C_r^{(d),2^d-1}$. Tarkime kiekvienas v_i ir w_i $(0 \le i \le 2^d - 1)$ žymi 4 bitų ilgio žodį. Tuomet ciklo funkcijos rezultatą $B = R_d\left(A, C_r^{(d)}\right)$ galima apskaičiuoti taip:

1. **for** $i=0$ **to** $2^d - 1$
   {
   if $C_r^{(d),i} = 0$, **then** $v_i = S_0(a_i)$;
   if $C_r^{(d),i} = 1$, **then** $v_i = S_1(a_i)$;
   }
2. $(w_{2i}, w_{2i+1}) = L(v_{2i}, v_{2i+1})$, kai $0 \le i \le 2^d - 1$;
3. $\left(b_0, b_1, \cdots, b_{2^d-1}\right) = P\left(w_0, w_1, \cdots, w_{2^d-1}\right)$.

Du R_4 ciklai yra pavaizduoti 2.3.1.5 pav.



Pav. 2.3.1.5. Du R_4 ciklai (be ciklo konstantos) [Wu11].

Bijektyvi funkcija E_d

Funkcija E_d realizuoja neapibrėžtos AES struktūrą. Ji yra sukonstruota iš 6(d-1) R_d ciklų. 2^{d+2} bitų ilgio įvesties ir išvesties duomenys yra atitinkamai žymimi *A* ir *B*. Tarkime, kad Q_r yra 2^{d+2} bitų ilgio žodis, kai $0 \le r \le 6(d-1)$, ir $Q_r = \left(q_{r,0} \parallel q_{r,1} \parallel \cdots \parallel q_{r,2^d-1}\right)$, kur kiekvienas $q_{r,i}$ žymi 4 bitų ilgio žodį. Tuomet rezultatas $B = E_d(A)$ apskaičiuojamas taip:

1. *A* bitai sugrupuojami į 2^d 4 bitų ilgio žodžius, kad būtų galima gauti Q_0;
2. kai $r = 0$ iki $6(d-1) - 1$, $Q_{r+1} = R_d\left(Q_r, C_r^{(d)}\right)$;
3. 2^d 4 bitų ilgio žodžiai pergrupuojami į $Q_{6(d-1)}$ taip, kad būtų gaunamas rezultatas.

E_d ciklo konstantos

E_d ciklo konstantos $C_r^{(d)}$ yra generuojamos iš ciklo funkcijos R_{d-2} (visos R_{d-2} ciklo konstantos yra 0). Kiekviena ciklo konstanta $C_r^{(d)}$ yra 2^d bitų ilgio žodžiai. Jos yra gaunamos taip:

$C_0^{(d)}$ yra sveikoji $\left(\sqrt{2} - 1\right) \times 2^{2^d}$ dalis;
$C_r^{(d)} = R_{d-2}\left(C_{r-1}^{(d)}, 0\right)$, kai $1 \le r < 6(d-1)$.

Konstantos $C_r^{(8)}$ $(0 \le r \le 41)$ reikšmės yra duotos priede Nr. 3.

Kompresijos funkcija F_d

Kompresijos funkcija F_d yra konstruojama iš funkcijos E_d. Kompresijos funkcija suspaudžia 2^{d+1} bitų ilgio pranešimo bloką $M^{(i)}$ ir 2^{d+2} bitų ilgio maišą $H^{(i-1)}$ į 2^{d+2} bitų ilgio $H^{(i)}$:

$$H^{(i)} = F_d\left(H^{(i-1)}, M^{(i)}\right).$$

Pagal E_d apibrėžimą, įvestis į kiekvieno pirmojo sluoksnio Sbox yra paveiktas dviejų pranešimo bitų, o rezultatui iš kiekvieno paskutinio sluoksnio Sbox su dviem pranešimo bitais yra atliekama XOR operacija.

F_8

F_8 yra JH algoritme naudojama kompresijos funkcija. F_8 spaudžia 512 bitų ilgio pranešimo bloką $M^{(i)}$ ir 1024 bitų ilgio maišą $H^{(i-1)}$ į 1024 bitų ilgio $H^{(i)}$. F_8 yra sudaryta iš E_8. Tarkime A, B žymi du 1024 bitų ilgio žodžius. Tada maiša $H^{(i)} = F_8\left(H^{(i-1)}, M^{(i)}\right)$ yra apskaičiuojama taip:

1. $A^j = H^{(i-1),j} \oplus M^{(i),j}$ $\quad 0 \le j \le 511;$
   $A^j = H^{(i-1),j}$ $\quad 512 \le j \le 1023;$
2. $B = E_8(A);$
3. $H^{(i),j} = B^j$ $\quad 0 \le j \le 511;$
   $H^{(i),j} = B^j \oplus M^{(i),j-512}$ $\quad 512 \le j \le 1023.$

2.3.2. JH maišos algoritmai

JH maišos funkciją sudaro penki žingsniai: pranešimo *M* praplėtimas, praplėsto pranešimo suskaidymas į blokus, pradinės maišos $H^{(0)}$ apskaičiavimas, galutinės maišos reikšmės $H^{(N)}$ radimas ir rezultato reikšmės generavimas sutrumpinant $H^{(N)}$.

Pranešimo praplėtimas

Pranešimas M yra praplečiamas taip, kad būtų 512 bitų kartotinis. Tarkime, kad pradinio pranešimo ilgis yra *l* bitų. Prie pranešimo galo prijungiame bitą '1', paskui kurį seka $384 - 1 + (-l \bmod 512)$ nuliniai bitai. Tuomet prijungiamas 128 bitų ilgio blokas, kuris yra l dvejetainė išraiška (labiausiai reikšminis bitas eina priekyje). Taigi pradinis pranešimas *M* yra papildomas bent 512 bitų.

Praplėsto pranešimo skaidymas

Kai pranešimas yra praplečiamas, jis suskaidomas į *N* 512 bitų ilgio blokus $M^{(1)}, M^{(2)}, \dots, M^{(N)}$. 512 bitų pranešimo blokas yra sudarytas iš 4 128 bitų ilgio žodžių. Pirmieji 128 bitai *i* pranešimo bloke yra žymimi $M_0^{(i)}$, kiti 128 – $M_1^{(i)}$ ir taip toliau iki $M_3^{(i)}$.

Pradinės maišos $H^{(0)}$ radimas

Pradinės maišos $H^{(0)}$ reikšmė yra nustatoma pagal pranešimo ilgį. Pirmieji du $H^{(-1)}$ baitai yra nustatomi tokie, kaip rezultato dydis, o likusiems $H^{(-1)}$ baitams priskiriamas 0. Taip pat $M^{(0)}$ yra priskiriamas 0. Tuomet $H^{(0)} = F_8\left(H^{(-1)}, M^{(0)}\right)$.

Tiksliau $H_0^{(-1),0} \parallel H_0^{(-1),1} \parallel \cdots \parallel H_0^{(-1),15}$ reikšmė JH-224, JH-256, JH-384 ir JH-512 algoritmuose yra atitinkamai 0x00E0, 0x0100, 0x0180 ir 0x0200.

Tarkime $H^{(-1),j} = 0$, kai $16 \le j \le 1023$. 512 bitų ilgio pranešimo blokui $M^{(0)}$ priskiriame 0. Tuomet 1024 bitų ilgio pradinės maišos $H^{(0)}$ reikšmė randama taip:

$$H^{(0)} = F_8\left(H^{(-1)}, M^{(0)}\right).$$

Galutinės maišos $H^{(N)}$ radimas

Norint gauti $H^{(N)}$ reikšmę, funkcija F_8 yra nuosekliai taikoma kiekvienam blokui $M^{(1)}, M^{(2)}, \dots, M^{(N)}$. Tuomet galutinė 1024 bitų ilgio maišos reikšmė yra apskaičiuojama tokiu būdu:

for $i = 1$ **to** N
$H^{(i)} = F_8\left(H^{(i-1)}, M^{(i)}\right)$.

Rezultato reikšmės generavimas

Galutinė rezultato reikšmė yra randama sutrumpinant $H^{(N)}$.

JH-224

JH-224 rezultatas yra paskutiniai $\mathrm{H}^{(\mathrm{N})}$ 224 bitai:

$$H^{(N),800} \parallel H^{(N),801} \parallel \cdots \parallel H^{(N),1023}.$$

JH-256

JH-256 rezultatas yra paskutiniai $\mathrm{H}^{(\mathrm{N})}$ 256 bitai:

$$H^{(N),768} \parallel H^{(N),769} \parallel \cdots \parallel H^{(N),1023}.$$

JH-384

JH-384 rezultatas yra paskutiniai $\mathrm{H}^{(\mathrm{N})}$ 384 bitai:

$$H^{(N),640} \parallel H^{(N),641} \parallel \cdots \parallel H^{(N),1023}.$$

JH-512

JH-512 rezultatas yra paskutiniai $H^{(N)}$ 512 bitai:

$$H^{(N),512} \parallel H^{(N),513} \parallel \cdots \parallel H^{(N),1023}.$$

2.4. Keccak algoritmas

Keccak [BDP+11] algoritmas yra sukurtas Sponge konstrukcijos pagrindu, bei paveldi daug jo savybių, tad jis priklauso Sponge funkcijų šeimai. Šiame algoritme pranešimo blokai yra apdorojami į pradinę bitų būseną, naudojantis XOR operacija. Didžiausias pavyzdys apima 5×5 dimensijos 64 bitų ilgio žodžių masyvą, iš viso turintį 1600 bitų. Kol mažesnio laipsnio dydžiai gali būti naudojami testuoti kriptoanalitines atakas, vidutinio dydžio (nuo $w = 4$, 100 bitų, iki $w = 32$, 800 bitų) taip pat suteikia praktines alternatyvas.

Vienas pagrindinių algoritmo dalių yra Keccak blokų perstatos. Toks keitinys apibrėžtas bet kokio laipsnio dviejų žodžių dydžiui, $w = 2^l$ bitų. Pagrindinės SHA-3 konkurso dalyvės naudoja 64 bitų ilgio žodžius, kai $l = 6$.

Pagrindu gali būti laikomas $5 \times 5 \times w$ bitų masyvas. Tarkime $a[i, j, k]$ yra įvesties duomenų bitai, išdėstyti taip, kad mažiausiai reikšmingas bitas eina pirmas, o kiti išdėstyti didėjimo tvarka. Visos aritmetinės operacijos atliekamos moduliu 5 arba w.

Pagrindinė bloko perstatos funkcija turi $12 + 2l$ iteracijų penkiuose subcikluose, kurių kiekvienas individualiai yra labai paprastas.

Keccak algoritmo naudojama Sponge konstrukcija, „įsiurbia“ pranešimą į maišos funkciją duotu momentu ir po to grąžina rezultatą. Tam, kad paimtų r bitų duomenų, yra atlieka XOR operacija su pirmaisiais bloke esančiais bitais ir po to atliekama bloko perstata.

Vykstant NIST Hash funkcijos konkursui dalyviams buvo leista suskaidyti savo algoritmus, kad būtų pabrėžta, kokie klausimai iškilo. Keccak algoritmas buvo pakeistas taip:

- padidintas ciklų skaičius nuo $12 + l$ iki $12 + 2l$, kad būtų dar labiau užtikrinamas algoritmo saugumas;
- pranešimo išdėstymo schema iš itin sudėtingos buvo pakeista paprastesne;
- rodiklis *r* buvo pakeltas iki saugumo limito, nei apvalinamas iki artimiausio dvejeto laipsnio.

Keccak algoritmo autoriai teigia, kad jų algoritmas yra žymiai spartesnis techniniame lygmenyje nei kitų finalininkų algoritmai.

Toliau pristatysiu algoritmo dalis, Keccak algoritmo naudojamus 7 keitinius ir Sponge funkcijas.

Pranešimo praplėtimo taisyklės

Pranešimo *M* praplėtimas *x* bitų yra žymimas $M \parallel pad[x](|M|)$.

Pranešimo praplėtimas vykdomas taip: prie bloko prijungiamas bitas ‘1‘, paskui kurį seka minimalus bitų ‘0‘ skaičius, bei paskutinis bitas ‘1‘. Praplėsto bloko ilgis yra bloko ilgio kartotinis.

Pranešimo praplėtimo metu mažiausiai yra pridedami 2 bitai, o daugiausiai – bloke esančių bitų skaičius plius vienas bitas.

Keccak-*f* keitiniai

Keccak-f algoritmas naudoja 7 keitinius, žymimus Keccak-$f[b]$, kur $b = 25 \times 2^l$ ir $l \in [0, 6]$. Keccak-$f[b]$ yra keitinys virš $\mathbb{Z}_2^b$, kai bitai yra numeruojami nuo 0 iki $b - 1$. b yra vadinamas keitinio pločiu.

Keccak-$f[b]$ keitinys yra būsenos a, kuri trimatis masyvas $a[5][5][w]$, kur $w = 2^l$, operacijų seka. Keccak-$f[b]$ yra kartotinė keitinių seka, sudaryta iš sekos n_r ciklų R, numeruojamų i_r nuo 0 iki $n_r - 1$. Vienas ciklas yra sudarytas iš penkių žingsnių:

$$R = \iota \circ \chi \circ \pi \circ \rho \circ \theta, \text{ kai}$$

$$\theta: \quad a[x][y][z] \leftarrow a[x][y][z] + \sum_{y'=0}^{4} a[x-1][y'][z] + \sum_{y'=0}^{4} a[x+1][y'][z-1],$$

$$\rho: \quad a[x][y][z] \leftarrow a[x][y][z-(t+1)(t+2)/2],$$
$$kai\ 0 \le t \le 24\ ir\ \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^{'} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix},$$
$$arba\ t = -1, jei\ x = y = 0,$$

$$\pi: \quad a[x][y] \leftarrow a[x'][y'], kai\ \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}^{'} \begin{pmatrix} x' \\ y' \end{pmatrix},$$

$$\chi: \quad a[x] \leftarrow a[x] + (a[x+1] + 1)a[x+2],$$

$$\iota: \quad a \leftarrow a + RC[i_r].$$

Išskyrus ciklo konstantas $RC[i_r]$, šie ciklo žingsniai yra vienodi. Ciklo konstantos yra apskaičiuojamos taip:

$$RC[i_r][0][0]\left[2^j - 1\right] = rc[j + 7i_r] \text{ visiems } 0 \le j \le l,$$

ir visos kitos $RC[i_r][x][y][z]$ yra nuliai. $rc[t]$ reikšmės yra gaunamos taip:

$$rc[t] = (x^{'} mod\ x^8 + x^6 + x^5 + x^4 + 1)\ mod\ x.$$

Ciklų kiekis n_r yra gaunamas iš keitinio pločio taip:

$$n_r = 12 + 2l.$$

Sponge funkcijos struktūra

Sponge struktūra apibrėžia funkciją $sponge[f, pad, r]$, kuri, gavus kintamo dydžio įvesties duomenis, grąžina sutarto ilgio rezultatą. Sponge funkcija naudoja fiksuoto ilgio keitinį (arba transformaciją) f, praplėtimo taisyklę „*pad*“ ir bitų dažnio kintamąjį r. Keitinys f yra vykdomas su fiksuoti ilgio bitų eilute, kurios plotis yra b. Kintamasis $c = b - r$ yra vadinamas tūriu.

Reikšmė 0^b vadinama pradine (šaknine) būsena (*angl. root state*). Ši reikšmė yra fiksuota ir niekada nėra paduodama kempinės funkcijai kaip įvesties duomuo. Tai yra būtina saugumo sąlyga sponge funkcijoje.

Keccak algoritmo sponge funkcijos

Keccak naudoja Keccak$[r, c]$ kempinės funkciją, kuriai paduodamas Keccak-$f[r + c]$ keitinys, pranešimo praplėtimas ir bitų dažnis r.

$$Keccak[r, c] \triangleq sponge[Keccak - f[r, c], pad10 * 1, r].$$

Keccak$[r, c]$ apibrėžiamas bet kokiai $r > 0$ ir c, kai $r + c$ yra Keccak-f palaikomas keitinio plotis, kombinacijai.

Numatytoji r reikšmė yra $1600 - c$, o numatytoji c reikšmė yra 576:

$$Keccak[r] \triangleq Keccak - f[r = 1600 - c, c],$$
$$Keccak[] \triangleq Keccak - f[c = 576].$$

2.4.1. Keitiniai Keccak-*f*

Šioje dalyje trumpai pristatysiu algoritmo naudojamus keitinius Keccak-*f*, kurie yra labai svarbūs, kad būtų užtikrintas algoritmo saugumas.

Postūmio invariantas

Tarkime $b = \tau(a)$, kur τ žymi 1 bito postūmį z ašies kryptimi. Kai $0 \le z \le w$, turime $b[x][y][z] = a[x][y][z-1]$ ir, kai $z = 0$, tada $b[x][y][0] = a[x][y][w-1]$. Kai vykdomas postūmis per t bitų, tada $b[x][y][z] = a[x][y][(z-t)\ mod\ w]$. Apskritai, postūmis $\tau[t_x][t_y][t_z]$ gali būti pavaizduotas trimačiu vektoriumi (t_x, t_y, t_z). Tada naujas koordinates atlikus postūmį apskaičiuoti galima taip:

$$b[x][y][z] = a[(x - t_x)\ mod\ 5][(y - t_y)\ mod\ 5][(z - t_z)\ mod\ w].$$

Dabar galima apibrėžti postūmio invariantą:

$$\tau[t_x][t_y][t_z] \circ \alpha = \alpha \circ \tau[t_x][t_y][t_z].$$

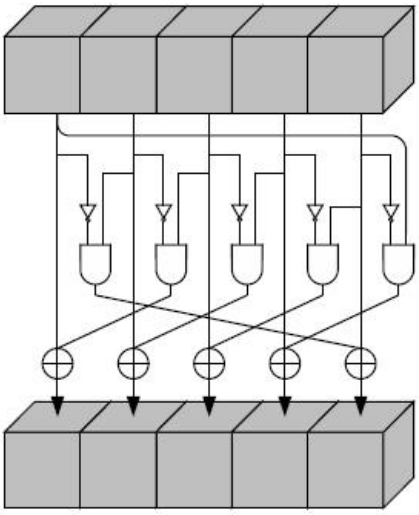
Keitinio Keccak-*f* žingsnio atvaizdžiai

Ciklas yra sudarytas iš nurodytų atvaizdžių, kurių kiekvienas atlieka tam tikrą užduotį, sekos.

Toliau pateiksiu kiekvieno keitinio žingsnio pseudo-kodą, kur kintamieji $a[x, y]$ žymi pradines vienmačio bitų, einančių z ašies kryptimi, reikšmes, o $A[x, y]$ – naujas. Atliekami veiksmai yra loginės dvejetainės operacijos ir posūkiai. Pseudo-kode $ROT(a, d)$ žymi a postūmį per d bitus, kai bitas, esantis z pozicijoje, atvaizduojamas $z + d\ mod\ w$ pozicija.

Atvaizdžio χ savybės

2.4.1.1 pav. grafiškai atvaizduoja, o 2.4.1.1 lentelėje esantis pseudo-kodas rodo atvaizdžio χ veikimo schemą.



Pav. 2.4.1.1. Atvaizdžio χ taikymas vienai eilutei [BDP+11]

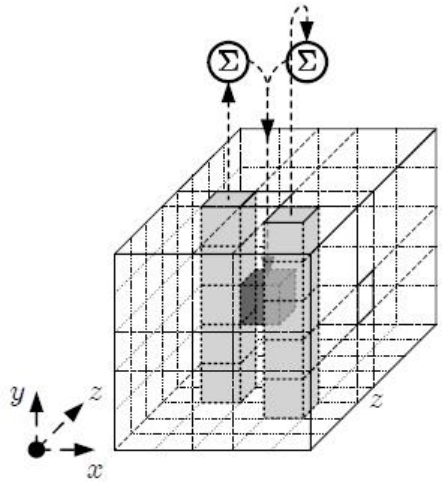
Lentelė 2.4.1.1. Atvaizdis χ

for $y = 0$ **to** 4 **do**
  for $x = 0$ **to** 4 **do**
    $A[x, y] = a[x, y] \oplus ((NOT\, a[x+1, y])\ AND\ a[x+2, y])$
  end for
end for

χ yra vienintelis netiesinis keitinio Keccak-*f* elementas, be kurio keitinys būtų tiesinis. Tai galima rodyti kaip lygiagretų $5w$ Sbox taikymą 5 bitų ilgio eilutėms. χ yra postūmio invariantas visomis kryptimis ir jo algebrinis laipsnis yra lygus dviem.

Atvaizdžio θ savybės

2.4.1.2 pav. grafiškai atvaizduoja, o 2.4.1.2 lentelėje esantis pseudo-kodas rodo atvaizdžio θ veikimo schemą.



Pav. 2.4.1.2. Atvaizdžio θ taikymas vienam bitui [BDP+11]

Lentelė 2.4.1.2. Atvaizdis θ

for $x = 0$ **to** 4 **do**
  $C[x] = a[x, 0]$
  for $y = 1$ **to** 4 **do**
    $C[x] = C[x] \oplus a[x, y]$
  end for
end for
for $x = 0$ **to** 4 **do**
  $D[x] = C[x - 1] \oplus ROT(C[x + 1], 1)$
  for $y = 0$ **to** 4 **do**
    $A[x, y] = a[x, y] \oplus D[x]$
  end for
end for

Atvaizdis θ yra tiesinis. Jo veikimą galima apibūdinti taip: prie kiekvieno bito $a[x][y][z]$ yra pridedami du stulpeliai $a[x-1][\cdot][z]$ ir $a[x+1][\cdot][z-1]$. Be atvaizdžio θ Keccak-f ciklo funkcija neduotų jokio reikšmių pasiskirstymo.

Atvaizdžio π savybės

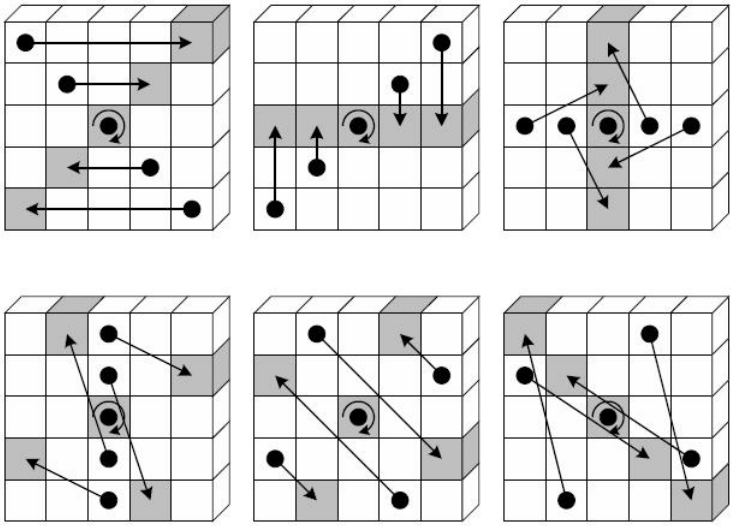
2.4.1.3 pav. grafiškai atvaizduoja, o 2.4.1.3 lentelėje esantis pseudo-kodas rodo atvaizdžio π veikimo schemą. Taip pat efektyvioje programoje π gali būti įgyvendinama nurodant netiesiogiai.

Lentelė 2.4.1.3. Atvaizdis π

for $x = 0$ **to** 4 **do**
  for $y = 0$ **to** 4 **do**
    $\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$
    $A[X, Y] = a[x, y]$
  end for
end for

Atvaizdis π yra perstata. Be jo Keccak-f būtų sudarytas iš pasikartojančių derinių. π vykdomas tiesiškai: bitų eilutė iš pozicijos (x,y) perkeliama į poziciją $(x, y)M^{\mathrm{T}}$, kur M yra dvimatė matrica $\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}$. Dėl matricos M bitų eilutė iš pradinės pozicijos $(0,0)$ nėra perstatoma. Kadangi π kiekvieną bitų eilutė apdoroja individualiai, atvaizdis π yra postūmio invariantas z ašies kryptimi. Atvirkštinis atvaizdžiui π apibrėžiamas

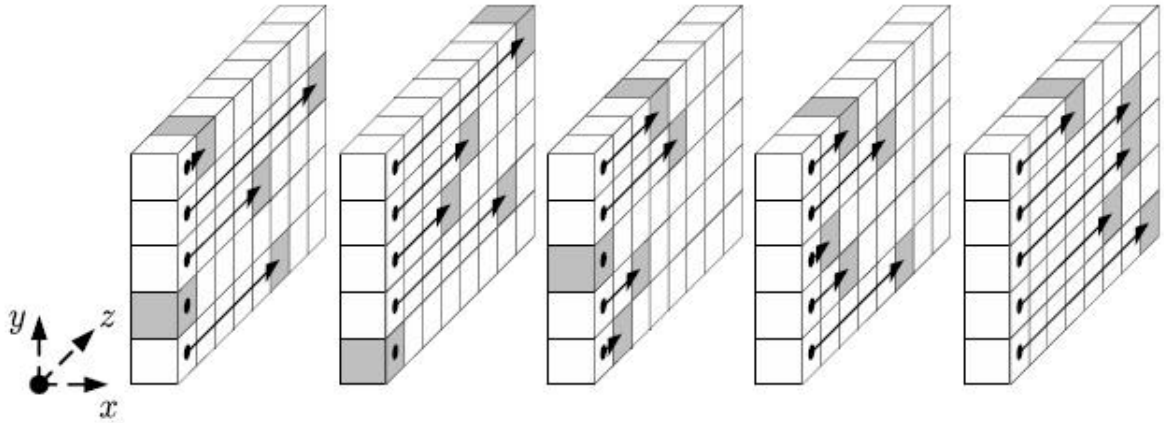
M^{-1}.



Pav. 2.4.1.3. Atvaizdžio π taikymas vienam pjūviui. Centro koordinatės visada $x = y = 0$ [BDP+11]

Atvaizdžio ρ savybės

2.4.1.4 pav. grafiškai atvaizduoja atvaizdžio ρ veikimo schemą. 2.4.1.5 lentelėje yra pateiktas postūmio poslinkių sąrašas. 2.4.1.4 lentelėje esantis pseudo-kodas rodo atvaizdžio ρ veikimo algoritmą.



Pav. 2.4.1.4. Atvaizdžio ρ taikymas vienai bitų eilutėms *z* ašimi. Centro koordinatės visada $x = y = 0$ [BDP+11]

Lentelė 2.4.1.4. Atvaizdis ρ

$A[0,0] = a[0,0]$

$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$

for $t = 0$ **to** 23 **do**

$A[X, Y] = ROT(a[x, y], (t + 1)(t + 2)/2)$

$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$

end for

	$x=3$	$x=4$	$x=0$	$x=1$	$x=2$
$y=2$	153	231	3	10	171
$y=1$	55	276	36	300	6
$y=0$	28	91	0	1	190
$y=4$	120	78	210	66	253
$y=3$	21	136	105	45	15

Lentelė 2.4.1.5. Atvaizdžio ρ postūmiai

25 postūmio konstantos yra apibrėžiamos $i(i+1)/2\ mod\ (bitų\ sekos\ ilgis)$. Galima įrodyti, kad bet kokiam l seka $i(i+1)/2\ mod\ 2^l$ turi periodą, kurio ilgis 2^{l+1}. Iš to seka, kad bitų sekos, kurios ilgis yra 64 ar 32 bitai, visos postūmio konstantos yra skirtingos. Kai bitų sekos ilgis yra 16, turime, kad 9 postūmio konstantos pasikartos dukart ir 7 – vienąkart. Kai bitų eilutės ilgis yra 8, 4 ir 2, visos postūmio konstantos pasikartos vienodai dažnai.

Atvaizdžio ι savybės

Atvaizdis ι yra sudarytas iš ciklo konstantų sudėties. Jis atsakingas už simetrijos suardymą. Be atvaizdžio ι ciklo funkcija būtų postūmio invariantas z ašies kryptimi ir visi ciklai būtų vienodi. Ciklo konstantų aktyvių bitų pozicijų kiekis, t.y. bitų pozicijos, kuriose ciklo konstantos nėra lygios nuliui, yra $l+1$. Jeigu l didėja, ciklo konstantos prideda vis daugiau asimetriškumo.

Žingsnių tvarka cikle

Dėl Keccak-f naudojimo sponge funkcijos konstrukcijoje, ciklo funkcija pradeda nuo atvaizdžio θ. Šis atvaizdis atlieka vidinės ir išorinės bloko dalių maišymą. Kitų atvaizdžių naudojimo seka yra sutartinė.

2.5. Skein algoritmas

Skein [FLS+08] yra kriptografinė funkcija, viena iš penkių NIST maišos funkcijų konkurso finalininkių bei pretendenčių į SHA-3 standarto vietą.

Skein yra maišos funkcijų šeima, kuriose galimi trys vidinių būsenų dydžiai: 256, 512 ir 1024 bitai.

Pirmasis pasiūlymas buvo Skein-512. Jis saugiai gali būti naudojamas su visomis šiuometinėmis maišos aplikacijomis. Tuo tarpu Skein-1024 maišos funkcija yra itin konservatyvus sprendimas. Kadangi jos vidinė dalis yra dvigubai didesnė nei Skein-512, tai ji yra atspari atakoms. Kitaip sakant, jei ateityje kam nors pavyktų sėkmingai atakuoti Skein-512 versiją, tai panašu, kad Skein-1024 algoritmas liktų saugus. Taip pat Skein-1024 algoritmas gali būti įvykdytas beveik dvigubai greičiau, nes Skein-512 yra sukurtas techninės įrangos realizacijai. Ir paskutinė Skein šeimos funkcija yra Skein-256. Tai yra mažai atminties reikalaujantis algoritmo variantas – realizacija gali būti leidžiama naudojant tik apie 100 RAM baitų.

Visi šie pradinės būsenos dydžiai palaiko bet kokį rezultato dydį. Skein algoritmo idėja yra sukonstruoti maišos funkciją remiantis skaidinio blokiniu šifru, kuris leidžia sumaišyti konfigūracijos duomenis su įvesties tekstu kiekviename bloke ir sukurti kiekvieną kompresijos funkcijos rezultatą unikalų. Šis atributas pagerina Skein algoritmo lankstumą.

Konkrečiau, Skein algoritmas yra sudarytas iš šių trijų komponentų.

- Threefish – yra skaidinio blokinis šifras, naudojamas Skein branduolyje, apibrėžtas 256, 512 ir1024 bitų blokais.
- Unikali bloko iteracija (*angl. Unique Block Iteration – UBI*) – sekos modelis, kuris naudojasi Threefish, kad sukurtų tokią kompresijos funkciją, kuri susieja sutartinio dydžio įvesties duomenis ir fiksuoto dydžio rezultatus.
- Papildomų argumentų sistema (*angl. Optional Argument System*) – leidžia Skeino algoritmui palaikyti daugybę papildomų parametrų be įspūdingų implementacijų ar papildomų aplikacijų, leidžiančių tomis savybėmis naudotis.

Analizuojant Skein algoritmą pagal šiuos tris žingsniu yra lengviau jį suprasti, išsianalizuoti ir įrodyti anksčiau paminėtas savybes. Threefish algoritmas pasirinktas ne atsitiktinai, jis pasirinktas remiantis metais patirties ir analizės, studijuojant blokinius šifrus. UBI yra įrodytai saugus ir gali būti naudojamas su bet kokiais skaidinio blokiniais šifrais. Papildomų parametrų sistema leidžia Skein algoritmą pritaikyti įvairiems tikslams. Visi šie trys elementai yra tarpusavyje nepriklausomi ir naudojami savarankiškai. Tačiau jų visų

kombinacija parodo tikrus algoritmo privalumus. Ir kiekvienas Skein algoritmo aspektas buvo skurtas taip, kad šiuos privalumus optimizuotų.

Threefish yra didelis skaidinio blokinis šifras. Jis yra apibrėžtas trims bloko dydžiams: 256, 512 ir 1024. Rakto dydis yra toks pats, koks ir bloko, o skaidinio reikšmė yra 128 bitai visiems blokams, nepriklausomai nuo jų dydžio. Threefish algoritmas naudoja tik tris matematines operacijas: modulinę sudėtį (XOR), sudėtį ir pastovųjį posūkį 64 bitų žodžiuose.

Threefish algoritmo branduolys yra funkcija MIX – paprasta, netiesinė maišymo funkcija, dirbanti su 64 bitų ilgio žodžiais. Kiekviena MIX funkcija sudaryta iš vieno sudėties, pastovaus postūmio ir XOR veiksmų.

Unikalios bloko iteracijos sekos modelis sujungia įvesties sekos reikšmes su nustatyto ilgio pranešimo tekstu ir sugeneruoja fiksuoto ilgio rezultatą. Paprasčiausias būdas paaiškinti veikimą yra pasitelkiant pavyzdį. Į Threefish funkciją kreipiamasi tris kartus. Kiekvienas pranešimo blokas M_0 ir M_1 sudaryti iš 64 baitų duomenų, o M_2 yra išskirtas blokas rezultatams, kurio dydis 38 baitai. Kiekvienam blokui skirstinio reikšmė užkoduoja, kiek baitų buvo panaudota, ar tai yra pirmasis/paskutinis UBI skaičiavimo blokas.

2.5.1. Pilnas Skein maišos algoritmo aprašas

Bitų ir baitų eiliškumas

Skein algoritmas visada naudoja tokį baitų eiliškumą, kai mažiausiai reikšminis baitas eina pirmas. Bitų eilutė yra saugoma kaip baitų eilutė. Kiekviena 8 bitų grupė yra koduojama viename baite. Pirmasis bitas saugomas septintajame baito bite, kitas – šeštame baito bite ir t.t. Jeigu bitų eilutės ilgis nėra 8 kartotinis, paskutinysis baitas yra panaudojamas tik dalinai (panaudojant tik žemesnes bitų pozicijas).

Norint baitų seką išreikšti sveikuoju skaičiumi, naudojama tvarka, kai mažiausiai reikšminis baitas eina pirmasis. Tarkime $b_0, \dots, b_n$ yra n baitų eilutė. Tada

$$ToInt(b_0, b_1, \dots, b_{n-1}) \coloneqq \sum_{t=0}^{n-1} b_i \cdot 256^t.$$

Atvirkštinis veiksmas atliekamas *ToBytes* funkcija:

$$ToBytes(v, n) \coloneqq b_0, b_1, \dots, b_{n-1}, \text{ kur } b_i \coloneqq \left\lfloor \frac{v}{256^t} \right\rfloor mod\ 256.$$

Ši funkcija taikoma tik tada, kai $0 \le v \le 256^n$, kad baitai pilnai užkoduotų v reikšmę. Dažniausiai konvertuojami 8*n* baitai ir *n* 64 bitų ilgio žodžių eilutės. Tarkime turime $b_0, \dots, b_{8n-1}$ baitų eilutę. Tada

$$BytesToWords(b_0, \dots, b_{8n-1}) \coloneqq w_0, \dots, w_{n-1}, \text{ kur } w_i \coloneqq ToInt(b_{8i}, b_{8i+1}, \dots, b_{8n+7}).$$

Atvirkštinis atvaizdavimas vykdomas taip:

$$WordToBytes(w_0, \dots, w_{n-1}) \coloneqq ToBytes(w_0, 8) \parallel ToBytes(w_1, 8) \parallel \cdots \parallel ToBytes(w_{n-1}, 8).$$

2.5.2. Funkcijos Threefish aprašymas

Threefish naudoja nežymiai patobulintą (*angl. tweakable*) blokinį šifrą, kai bloko dydis yra 256, 512 arba 1024 bitai. Papildinys (*angl. tweak*) visada yra 128 bitai.

Šifravimo funkcija $E(K, T, P)$ naudoja tokius kintamuosius:

K blokinio šifro raktas; eilutė, kurios dydis yra 32, 64 arba 128 baitai (256, 512 arba 1024 bitai).

T papildinys, eilutė, kurios ilgis 16 baitų (128 bitai).

P atviras tekstas, baitų eilutė, kurios ilgis toks pat kaip ir rakto.

Threefish dirba su 64 bitų ilgio (be ženklo) žodžiais. Visi įvesties duomenys yra konvertuojami į 64 bitų ilgio žodžių eilutes. Tarkime, N_w yra žodžių rakte skaičius. Raktas K yra sudarytas iš raktinių žodžių $(k_0, k_1, \dots, k_{N_w-1})$, papildinys T yra sudarytas iš žodžių (t_0, t_1) ir atviras tekstas iš $(p_0, p_1, \dots, p_{N_w-1})$.

$$\begin{aligned} k_0, \dots, k_{N_w-1} &\coloneqq BytesToWords(K) \\ t_0, t_1 &\coloneqq BytesToWords(T) \\ p_0, \dots, p_{N_w-1} &\coloneqq BytesToWords(P) \end{aligned}$$

Ciklų skaičius N_r yra bloko dydžio funkcija, kuri parodyta 2.5.2.1 lentelėje.

Bloko/Rakto dydis	# Žodžiai N_w	# Ciklai N_r
256	4	72
512	8	72
1024	16	80

Lentelė 2.5.2.1. Ciklų skaičius skirtingiems bloko dydžiams

Tarkime $v_{d,i}$ yra i-ojo žodžio reikšmė atlikus d ciklų. Pradinės reikšmės yra

$$v_{0,i} \coloneqq p_i, \text{ kai } i = 0, \dots, N_w - 1.$$

Tuomet atliekami N_r ciklai, kurie numeruojami$d = 0, \dots, N_r - 1$. Kiekviename cikle, jeigu $d \bmod 4 = 0$, yra pridedamas poraktis. Tada, kai $i = 0, \dots, N_w - 1$, turime:

$$e_{d,i} \coloneqq \begin{cases} \left(v_{d,i} + k_{d/4,i}\right) \bmod 2^{64}, & kai\ d \bmod 4 = 0 \\ v_{d,i}, & priešingu\ atveju \end{cases}.$$

Maišymas ir žodžių perstata yra apibrėžiama taip:

$$\begin{aligned} \left(f_{d,2j}, f_{d,2j+1}\right) &\coloneqq MIX_{d,j}\left(e_{d,2j}, e_{d,2j+1}\right), \quad kai\ j = 0, \dots, N_w/2 - 1, \\ v_{d+1,i} &\coloneqq f_{d,\pi(i)}, \quad kai\ i = 0, \dots, N_w - 1. \end{aligned}$$

$f_{d,i}$ yra MIX funkcijos reikšmės, o žodžio keitinių rezultatas yra ciklo išvesties duomenys. Žodžių keitinių $\pi(\,)$ reikšmės yra išvardytos 2.5.2.2 lentelėje.

i / N_w	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4	0	3	2	1												
8	2	1	4	7	6	5	0	3								
16	0	9	2	13	6	11	4	15	10	7	12	3	14	5	8	1

Lentelė 2.5.2.2. Žodžių keitinių $\pi(i)$ reikšmės

Šifruotas tekstas *C* gaunamas:

$$\begin{aligned} c_i &\coloneqq \left(v_{N_r,i} + k_{N_r/4,i}\right) \bmod 2^{64}, \quad kai\ i = 0, \dots, N_w - 1, \\ C &\coloneqq WordsToBytes\left(c_0, \dots, c_{N_w - 1}\right). \end{aligned}$$

Funkcijos MIX

Funkcijai $MIX_{d,j}$ yra paduodami du žodžiai (x_0, x_1) ir jis grąžina du žodžius (y_0, y_1), panaudodama šiuos sąryšius:

$$\begin{aligned} y_0 &\coloneqq (x_0 + x_1) \bmod 2^{64} \\ y_1 &\coloneqq \left(x_1 \lll R_{(d \bmod 8),j}\right) \oplus y_0 \end{aligned}$$

kur $\lll$ žymi posūkį į kairę. Konstantos $R_{d,j}$ reikšmės yra nurodytos 2.5.2.3 lentelėje.

N_w		4		8				16							
j		0	1	0	1	2	3	0	1	2	3	4	5	6	7
$d =$	0	5	56	38	30	50	53	55	43	37	40	16	22	38	12
	1	36	28	48	20	43	31	25	25	46	13	14	13	52	57
	2	13	46	34	14	15	27	33	8	18	57	21	12	32	54
	3	58	44	26	12	58	7	34	43	25	60	44	9	59	34
	4	26	20	33	49	8	42	28	7	47	48	51	9	35	41
	5	53	35	39	27	41	14	17	6	18	25	43	42	40	15
	6	11	42	29	26	11	9	58	7	32	45	19	18	2	56
	7	59	50	33	51	39	35	47	49	27	58	37	48	53	56

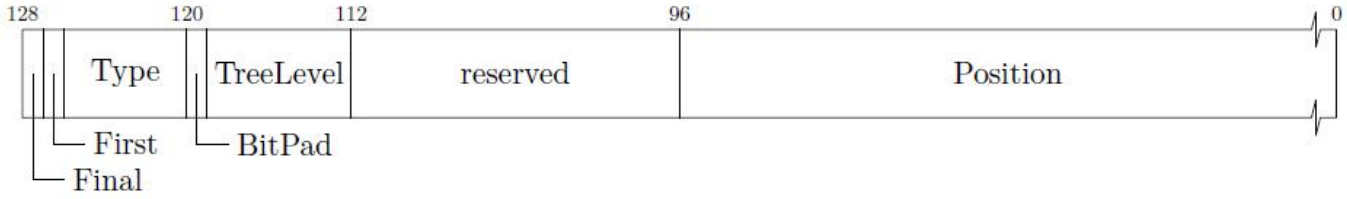
Lentelė 2.5.2.3. Posūkio konstantos $R_{d,j}$ kiekviename N_w

Rakto schema

Rakto schema pradedama apibrėžiant du papildomus žodžius k_{N_w} ir t_2:

$$k_{N_w} \coloneqq \lfloor 2^{64}/3 \rfloor \oplus \oplus_{i=0}^{N_w-1} k_i \text{ ir } t_2 \coloneqq t_0 \oplus t_1.$$

Konstanta $\lfloor 2^{64}/3 \rfloor$ užtikrina, kad praplėstas raktas nebūtų sudarytas vien iš nulių. Tokiu atveju rakto schema apibrėžiama taip:



Pav. 2.5.3.1. Papildinio reikšmės laukai

$$\begin{aligned}
k_{s,i} &\coloneqq k_{(s+i)\ mod\ (N_w+1)}, && kai\ i = 0, \dots, N_w - 4,\\
k_{s,i} &\coloneqq k_{(s+i)\ mod\ (N_w+1)} + t_{s\ mod\ 3}, && kai\ i = N_w - 3,\\
k_{s,i} &\coloneqq k_{(s+i)\ mod\ (N_w+1)} + t_{(s+1)\ mod\ 3}, && kai\ i = N_w - 2,\\
k_{s,i} &\coloneqq k_{(s+i)\ mod\ (N_w+1)} + s, && kai\ i = N_w - 1,
\end{aligned}$$

kur visos sudėties operacijos atliekamos moduliu 2^{64}.

Dešifravimas

Threefish funkcijos dešifravimas atvirkštinai veiksmai, kurie buvo atliekami šifruojant. Porakčiai yra naudojama atbuline tvarka ir kiekvieną ciklą sudaro atvirkštiniai žodžių keitiniai, po kurių eina atvirkštinės MIX funkcijos.

2.5.3. Detalus UBI aprašas

UBI jungimo metodas yra paremtas nežymiai patobulintu blokiniu šifru, kurio bloko ir rakto dydžiai yra N_b baitų bei Papildinys, kurio dydis yra 16 baitų. Funkcija $UBI(G, M, T_s)$ naudoja:

G pradinė reikšmė, kurios ilgis N_b baitų.

M sutartinio ilgio (iki $2^{99} - 8$ bitų) pranešimo eilutė, užkoduota baitų eilute.

T_s 128 bitų ilgio pradinė papildinio reikšmė.

UBI apdoroja pranešimo blokus, kiekvienam jų naudodama unikalią papildinio reikšmę. Papildinio laukai yra pavaizduoti 2.5.3.1 pav. ir 2.5.3.1 lentelėje.

Pavadinimas	Bitai	Aprašymas
Position	0-95	Apdorotų bitų skaičius.
reserved	96-111	Rezervuota vėlesniam naudojimui (privalo būti 0).
TreeLevel	112-118	Lygis maišos medyje.
BitPad	119	Nustatomas tik tada, kai bloke yra paskutinis pranešimo, kurio ilgis nėra integralus baitų skaičius, baitas. 0 priešingu atveju.
Type	120-125	Lauko tipas (konfigūracija, pranešimas, rezultatas ir t.t.)
First	126	Nustatomas pirmam UBI kompresijos blokui.
Final	127	Nustatomas paskutiniam UBI kompresijos blokui.

Lentelė 2.5.3.1. Papildinio reikšmės laukai

Jeigu bitų skaičius pranešime M yra 8 kartotinis, nustatome $B := 0$ ir $M' := M$. Jeigu bitų skaičius pranešime M nėra 8 kartotinis, vadinasi paskutinis baitas nėra pilnai panaudotas. Reikšmingiausios paskutinio baito pozicijos laiko pranešimo bitus. Tokiu atveju, paskutinis baitas yra praplečiamas, paskutiniam nepanaudotam reikšmingiausiam bitui priskiriant bitą ‘1‘ ir likusiems ‘0‘. Tokiu atveju $B := 1$ ir M' yra praplėstas blokas.

Tarkime N_M yra baitų skaičius bloke M'. M' yra praplečiamas su p ‘0‘ bitų tol, kol ilgis yra bloko dydžio kartotinis, užtikrinant, kad užpildomas bent vienas pilnas blokas.

$$p := \begin{cases} N_b, & jei\ N_M = 0, \\ (-N_M)\ mod\ N_b & priešingu\ atveju, \end{cases}$$
$$M'' := M' \parallel 0^p.$$

M'' yra padalinamas į pranešimo blokus $M_0, \dots, M_{k-1}$, kurių kiekvieno dydis yra N_b baitai. Funkcijos UBI rezultatas yra apskaičiuojamas taip:

$$H_0 := G$$
$$H_{i+1} := E(H_i, ToBytes(T_s + min(N_M, (i+1)N_b) + a_i 2^{126} + b_i(B2^{119} + 2^{127}), 16), M_i) \oplus M_i,$$

kur $a_0 = b_{k-1} = 1$, o visos likusios a_i ir b_i yra 0, E() yra blokinė šifravimo funkcija ir H_k yra UBI funkcijos rezultatas.

Papildinio reikšmė kiekvienam blokui yra konstruojama taip:

$$T_s + min(N_M, (i+1)N_b) + a_i 2^{126} + b_i(B2^{119} + 2^{127}).$$

2.5.4. Skein algoritmo struktūra

Tipo reikšmės

Skein algoritmas sudarytas iš daug galimų kintamųjų, kurių kiekvienas (nesvarbu privalomas ar pasirinktinis) turi unikalų tipą ir reikšmę. Tipo reikšmės priklauso intervalui [0..63]. Skein algoritmo parametrai pateikiami didėjančia tvarka (žr. 2.5.4.1 lentelę).

Simbolis	Reikšmė	Aprašymas
T_{key}	0	Raktas
T_{cfg}	4	Konfigūracijos blokas
T_{prs}	8	Personalizacijos eilutė
T_{PK}	12	Viešasis raktas
T_{kdf}	16	Rakto atpažinėjas
T_{non}	20	Sutartinis atsitiktinis skaičius (srautiniams šiframs arba atsitiktinei maišai)
T_{msg}	48	Pranešimas
T_{out}	63	Rezultatas

Lentelė 2.5.4.1. Tipo lauko reikšmės

Konfigūracijos eilutė

Konfigūracijos eilutė yra sudaryta iš tokių duomenų:

- Schemos atpažinėjas. Tai tiesioginė konstanta.

- Versijos numeris, ateities plėtiniams palaikyti.
- N_0: skaičiavimo rezultato ilgis bitai. Šis parametras užtikrina, kad du Skein algoritmo skaičiavimo rezultatai, kurie skiriasi tik bitų skaičiumi, nebūtų tarpusavyje susiję.
- Y_l: medžio lapo dydžio kodavimas. Nustatomas 0, jeigu medžio maiša nėra naudojama.
- Y_m: maksimalus medžio aukštis. Nustatomas 0, jeigu medžio maiša nėra naudojama.

Rezultato funkcija

Rezultato funkcija $Output(G, N_0)$ naudoja tokius kintamuosius:
G jungiamoji reikšmė.
N_0 reikalingas išvesties duomenų bitų skaičius.
Ši funkcija grąžina N_0 bitų ilgio rezultatą.
Rezultatą sudaro svarbiausi $\lceil N_0/8 \rceil$ baitai:

$$\begin{aligned} O \coloneqq\ & UBI(G, ToBytes(0,8), T_{out}\, 2^{120}) \parallel \\ & UBI(G, ToBytes(1,8), T_{out}\, 2^{120}) \parallel \\ & UBI(G, ToBytes(2,8), T_{out}\, 2^{120}) \parallel \\ & \dots \end{aligned}$$

Jeigu $N_0\ mod\ 8 = 0$, tai rezultatą sudaro integralus baitų skaičius. Jeigu $N_0\ mod\ 8 \neq 0$, tai paskutinis baitas panaudotas tik dalinai.

Paprastoji maiša

Paprastosios Skein maišos apskaičiavimas naudoja tokius kintamuosius:
N_b Vidinės būsenos dydis baitais. Privalo būti 32, 64 arba 128.
N_0 Rezultato dydis bitais.
M Pranešimas, kuriam bus skaičiuojama maišos reikšmė (eilutė, sudaryta iki $2^{99} - 8$ bitų).
Tarkime *C* yra konfigūracijos eilutė, kurios $Y_l = Y_m = 0$. Tada galime apibrėžti:

$$\begin{aligned} K' &\coloneqq 0^{N_b} && \text{eilutė iš } \mathrm{N_b} \text{ nulinių baitų} \\ G_0 &\coloneqq UBI\left(K', C, T_{cfg}\, 2^{120}\right) \\ G_1 &\coloneqq UBI\left(G_0, M, T_{msg}\, 2^{120}\right) \\ H &\coloneqq Output(G_1, N_0) \end{aligned}$$

kur *H* yra maišos rezultatas.

Pilnas Skein algoritmas

Pilna Skein algoritmo forma skaičiavimams naudoja tokius kintamuosius:
$\mathrm{N_b}$ Vidinės būsenos dydis baitais. Privalo būti 32, 64 arba 128.
N_0 Rezultato dydis bitais.
K $\mathrm{N_k}$ baitų dydžio raktas. Jeigu raktas nenaudojamas, paduodama tuščia eilutė.
Y_l Medžio lapo dydžio kodavimas.
Y_m Maksimalus medžio dydis.
L t porų (T_i, M_i) sąrašas, kur T_i yra tipo reikšmė ir M_i yra bitų eilutė, atvaizduota baitų eilutėje.
Turime:

$$L \coloneqq (T_0, M_0), \dots, (T_{t-1}, M_{t-1}).$$

Reikalaujame, kad $T_{cfg} < T_0$, $T_i < T_{i+1}$ visiems i ir $T_{t-1} < T_{out}$. Leidžiama, kad sąrašas L būtų tuščias. Kiekvienas M_i gali būti iki $2^{99} - 8$ bitų dydžio.
Pirmiausia yra apdorojamas raktas. Jeigu $N_k = 0$, pradinė reikšmė yra sudaryta iš 0.

$$K' \coloneqq 0^{N_b}.$$

Jeigu $N_k \neq 0$, tai raktas yra kompresuojamas naudojantis UBI funkcija ir gaunama pradinė reikšmė:

$$K' \coloneqq UBI\left(0^{N_b}, K, T_{key}\, 2^{120}\right).$$

Tarkime *C* yra konfigūracijos eilutė. Tada

$$G_0 := UBI\big(K', C, T_{cfg}\,2^{120}\big).$$

Kintamieji yra apdorojami šia tvarka:

$$G_{i+1} := UBI(G_i, M_i, T_i 2^{120}), \text{ kai } i = 0, \dots, t-1.$$

Tačiau kintamieji Y_l ir Y_m nėra lygūs nuliui.

Galutinis Skein algoritmo rezultatas apskaičiuojamas:

$$H := Output(G_t, N_0).$$

3. Statistiniai testai

Pasak [Sta06], [RSN+10], vertindami bitų seką, sprendžiame klausimą, ar ji tinka kriptografijos reikmėms, ar ne. Hipotezių tikrinimo terminologija ir metodai – geri įrankiai ieškant atsakymo.

Toliau trumpai pristatysiu dviejų tipų atsitiktinių sekų generatorius bei pagrindines statistinio testo savybes.

3.1. Atsitiktinių skaičių generatoriai

Yra du pagrindiniai generatorių tipai: atsitiktinių skaičių generatoriui (*angl. Random Number Generators – RNGs,* [Ken05], [RSN+10]) ir pseudo-atsitiktinių skaičių generatoriai (*angl. Pseudorandom Number Generators – PRNGs,* [Ken05], [RSN+10]). Kriptografinėms programoms abiejų tipų generatoriai sugeneruoja nulių ir vienetų seką, kuri gali būti padalinama į mažesnius srautus arba atsitiktinių skaičių blokus.

Pirmasis sekos generatoriaus tipas yra atsitiktinių skaičių generatorius (RNG). RNG naudojasi neapibrėžtu šaltiniu (t.y. entropiniu šaltiniu), kurio apdorojimo funkcija (t.y. entropijos distiliavimo procesas) realizuoja atsitiktinumą. Apdorojimo funkcijos panaudojimas svarbus ir reikalingas, norint įveikti bet kokius entropinio šaltinio trūkumus, kurie gali įtakoti ne atsitiktinių elementų sugeneravimą (pvz., ilgų nulių ar vienetų sekų). RNG išvesties duomenys gali būti naudojami atsitiktiniams skaičiams generuoti arba paduoti pseudo-atsitiktinių skaičių generatoriui (PRNG). Jeigu RNG išvesties duomenys naudojami tiesiogiai (t.y. be tolimesnio apdorojimo), jie turi tenkinti griežtus atsitiktinumo kriterijus, kurie yra apibrėžiami statistiniais testais, skirtais nustatyti, ar RNG įvesties šaltiniai yra atsitiktiniai.

Kriptografiniais tikslais, RNG išvesties duomenys turi būti nenuspėjami, tačiau kai kurie fiziniai šaltiniai (pvz., datos/laiko vektoriai) yra gana nuspėjami. Be to, aukštos kokybės atsitiktinių skaičių generavimo laiko sąnaudos gali būti pernelyg didelės, dėl kurių sugeneruoti didelį kiekį atsitiktinių skaičių tampa problematiška. Norint generuoti daug atsitiktinių skaičių geriau naudotis pseudo-atsitiktinių skaičių generatoriais (PRNGs).

PRNG naudojasi vienu ar keliais įvesties duomenimis, kad sugeneruotų kelis „pseudo-atsitiktinius“ skaičius. PRNG įvesties duomenys vadinami pradine bitų seka arba tiesiog sėkla. Kadangi nenuspėjamumas yra būtinas, pati pradinė bitų seka turi būti atsitiktinė ir nenuspėjama. Taigi, pagal nutylėjimą, pradinė bitų seka yra sugeneruojama RNG, t.y. kad veiktų PRNG, yra būtinas ir RNG. PRNG rezultatai paprastai yra apibrėžtos pradinių bitų sekos funkcijos, t.y. visas tikrasis atsitiktinumas yra apribotas šios sekos generavimu. Ironiška, tačiau pseudo-atsitiktiniai skaičiai yra labiau atsitiktiniai nei gaunami iš fizinių šaltinių. Jeigu pseudo-atsitiktinių skaičių seka yra sukonstruota tinkamai, kiekviena sekos reikšmė yra gaunama iš prieš ją ėjusios, pasinaudojant transformacijomis, kurios įtakoja atsitiktinumą. Tokių transformacijų rinkinys gali pašalinti statistinę autokoreliaciją tarp įvesties ir išvesties. Taigi PRNG gaunami rezultatai gali būti geresnių statistinių savybių ir generuojami greičiau nei RNG.

3.2. Statistinių testų savybės

Pasak [Ken05], [RSN+10], įvairiausi statistiniai testai gali būti taikomi sekai, kai norima palyginti ir įvertinti, ar seka tikrai yra atsitiktinė. Yra begalinis galimų statistinių testų, kurie įvertina, ar egzistuoja aprašytas „modelis“, nurodantis, kad seka nėra atsitiktinė, skaičius, todėl nėra sukurta jokio specifinio testų rinkinio, kuris būtų laikomas pilnu ir galutiniu.

Statistinis testas yra suformuluotas išbandyti specifinę *nulinę hipotezę* (H_0). Testuojamos sekos nulinė

hipotezė reiškia, kad seka yra atsitiktinė. Su nuline hipoteze taip pat susijusi yra *alternatyvioji hipotezė* (H_a), kuri reiškia, kad seka nėra atsitiktinė. Kiekvieno taikomo testo gauta išvada arba sprendimas tenkina arba atmeta nulinę hipotezę, t.y. nurodo, ar generatoriaus pagaminta seka yra (nėra) atsitiktinė.

Atliekant kiekvieną bandymą, turi būti pasirinkta tinkama atsitiktinė statistika, naudojama patvirtinti arba paneigti nulinę hipotezę. Tokios statistikos teorinis reikšmių pasiskirstymas pagal nulinę hipotezę yra apskaičiuojamas naudojantis matematiniais metodais. Yra apibrėžiama *kritinė* tokio pasiskirstymo *reikšmė* (paprastai, 99% kartų, ši reikšmė parenkama toli pasiskirstymo kraštuose). Atliekant bandymą, naudojantis turimais duomenimis yra skaičiuojama statistinė testo reikšmė (tikrinama seka). Gauta statistinė bandymo reikšmė yra lyginama su nustatyta kritine verte. Jeigu statistinė bandymo reikšmė yra didesnė už kritinę, tai nulinė hipotezė yra atmetama, priešingu atveju yra patvirtinama.

Statistinis hipotezės testavimas yra išvados generavimo procedūra, turinti tik du galimus rezultatus: arba priimti H_0 (duomenys yra atsitiktiniai), arba priimti H_a (duomenys nėra atsitiktiniai). Žemiau pateikta lentelė nurodo ryšį tarp turimų duomenų ir iš testavimo procedūros gautų rezultatų.

	Išvada	
	Priimta H_0	Priimta H_a
Duomenys yra atsitiktiniai (H_0 teigiama)	Nėra klaidos	I tipo klaida
Duomenys nėra atsitiktiniai (H_a teigiama)	II tipo klaida	Nėra klaidos

Lentelė 3.2.1. Hipotezės patvirtinimo tipai [Ken05], [RSN+10].

Jeigu duomenys yra atsitiktiniai, tada tikimybė, kad nulinė hipotezė bus atmesta yra maža. Tokia išvada vadinama I tipo klaida. Jeigu duomenys nėra atsitiktiniai, tada rezultatas, kai priimama nulinė hipotezė vadinamas II tipo klaida. Rezultatas, kuris priima H_0 hipotezę, kai duomenys yra tikrai atsitiktiniai bei priima H_a hipotezę, kai duomenys tikrai nėra atsitiktiniai, yra teisingas.

I tipo klaidos tikimybė dažnai yra vadinama testo reikšmingumo lygiu. Ši tikimybė gali būti nustatyta iki bandymo pradžios ir yra žymima α. Testavimo metu ši tikimybė reiškia, kad gaunamas rezultatas, jog seka nėra atsitiktinė, kai iš tikrųjų ji yra atsitiktinė. Tai reiškia, kad net „gero“ generatoriaus sugeneruota seka turi neatsitiktinių elementų savybių. Dažniausiai α reikšmės kriptografijoje yra apie 0,01.

II tipo klaidos atsiradimo tikimybė žymima β. Testavimo metu ši tikimybė reiškia, kad gautas rezultatas, nurodantis, kad seka yra atsitiktinė, kai iš tikrųjų ji tokia nėra, t.y. „blogas“ generatorius sugeneravo tokią seką, kuri turi atsitiktinumo savybes tenkinančių elementų. Skirtingai nuo α, β nėra fiksuotas dydis. β gali įgyti daug skirtingų reikšmių, nes yra begalinis skaičius skirtingų būdų, nurodančių, kad seka nėra atsitiktinė, ir kiekvienas būdas turi skirtingą β reikšmę. II tipo klaidos β reikšmės skaičiavimas yra sudėtingesnis nei α dėl galimų skirtingų neatsitiktinumo rūšių.

Vienas iš pagrindinių statistinių testų tikslų yra sumažinti II tipo klaidos tikimybę, t.y. sumažinti tikimybę, kad „blogo“ generatoriaus sugeneruota seka būtų priimta kaip gera. Tikimybės α ir β yra susijusios tarpusavyje ir su testuojamos sekos dydžiu n taip, kad jeigu dvi iš šių reikšmių yra nurodomos, trečioji paskaičiuojama automatiškai. Praktikai dažniausiai pasirenka sekos dydžio n ir α reikšmes (prisiminkime, kad I tipo klaidos tikimybė – testo reikšmingumo lygis). Tada kritinis duotos statistikos taškas parenkamas taip, kad būtų gaunama mažiausia β reikšmė (II tipo klaidos tikimybė). Tai reiškia, kad tinkamas sekos dydis pasirenkamas atsižvelgiant į tikimybę, kad blogas generatorius gali sukurti tikrai atsitiktinę seką. Tada rezultato priimtinumas pasitenkamas taip, kad tikimybė klaidingai priimti seką kaip atsitiktinę yra mažiausia.

Testo statistika yra naudojama apskaičiuoti P – reikšmę, kuri apibendrina visų, prieš nulinę hipotezę surinktų, įrodymų tvirtumą. Kiekviena P – reikšmė yra tikimybė, kad idealus atsitiktinių skaičių generatorius sukurs tokią seką, kuri bus mažiau atsitiktinė, jei anksčiau testuota, taip sukuriant neatsitiktinumo požymį, kurį testas nustatys. Jeigu testo P – reikšmė lygi 1, tai seka yra idealiai atsitiktinė. Nulinė P – reikšmė nurodo, kad seka yra visiškai neatsitiktinė. Jeigu P – reikšmė $\geq$ α, tada nulinė hipotezė yra patvirtinama, t.y. seka yra atsitiktinė, priešingu atveju, seka nėra atsitiktinė. Dažniausiai alfa parenkamas iš intervalo $[0{,}001, 0{,}01]$.

3.3. NIST statistinių testų rinkinys

NIST testų rinkinys ([RSN+10]) – tai statistinis paketas, kurį sudaro 15 testų, kurie buvo sukurti išbandyti dvejetainių sekų, sugeneruotų RNG ar PRNG, atsitiktinumą. Šiuose testuose dėmesys kreipiamas į

daugybę neatsitiktinumų, kurie egzistuoja sekose, požymių rūšis. Kai kurie testai patys yra skaidomi į mažesnius testus. Šie 15 testų yra:

1) dažnio (monobito) testas (*angl. The Frequency (Monobit) Test*);
2) dažnio testas bloke (*angl. The Frequency Test within a Block*);
3) eigos testas (*angl. The Runs Test*);
4) ilgiausios vienetų sekos bloke testas (*angl. Tests for the Longest-Run-of-Ones in a Block*);
5) dvejetainės matricos laipsnio testas (*angl. The Binary Matrix Rank Test*);
6) diskrečioji Furjė transformacijos (spektrinis) testas (*angl. The Discrete Fourier Transform (Spectral) Test*);
7) nepersidengiančių šablonų parinkimo testas (*angl. The Non-overlapping Template Matching Test*);
8) persidengiančių šablonų parinkimo testas (*angl. The Overlapping Template Matching Test*);
9) Maurer „Universalus statistinis“ testas (*angl. Maurer‘s „Universal Statistical Test*);
10) linijinio sudėtingumo testas (*angl. The Linear Complexity Test*);
11) serijos testas (*angl. The Serial Test*);
12) vidutinės entropijos testas (*angl. The Approximate Entropy Test*);
13) sukauptų sumų (Cusums) testas (*angl. The Cumulative Sums (Cusums) Test*);
14) atsitiktinių ekskursijų testas (*angl. The Random Excursions Test*);
15) atsitiktinių ekskursijų variantų testas (*angl. The Random Excursions Variant Test*).

Šių išvardintų testų taikymo seka yra sutartinė, tačiau rekomenduojama pirmiausiai taikyti dažnio testą, nes jis pateikia daugiausia pagrindinių įrodymų, kad analizuojama seka nėra atsitiktinė, konkrečiai – ieško sekoje vienodumų. Jeigu šis bandymas nepavyksta, tikimybė, kad nepavyks ir visi kiti testai yra itin didelė. Taip pat reikia paminėti, kad didžiausių laiko sąnaudų reikalauja linijinio sudėtingumo testas.

Dalis testų rinkinyje yra standartinio normaliojo, dalis chi-kvadrat (X^2) dėsnio. Jeigu bandoma seka iš tikrųjų nėra atsitiktinė, apskaičiuota testo statistika paklius į kraštutines pasiskirstymo dalis. Standartinio normaliojo pasiskirstymo testas yra $z = (x - \mu)/\sigma$, kur x yra testo statistikos reikšmė, o μ ir σ^2 yra atitinkamai tikėtina bandymo statistikos reikšmė ir nuokrypis. X^2 pasiskirstymas yra naudojamas palyginti testuojamų kriterijų ir hipotetiniame pasiskirstyme tikėtinų kriterijų suderinamumus. Testo statistika yra $X^2 = \sum((o_i - e_i)^2/e_i)$, kur atitinkamai o_i ir e_i yra testuojami ir tikėtini kriterijų dažnumai.

Daugeliui šiame rinkinyje esančių testų yra keliamas reikalavimas, kad sekos ilgis n turi būti didelis skaičius (nuo 10^3 iki 10^7). Būtent tokios ilgoms testuojamoms sekoms buvo išvesti ir pritaikyti testavime asimptotiniai reikšmių skirstiniai. Keliems bandymams gali naudoti ir mažesnius sekų ilgius, tačiau tokiu atveju asimptotiniai skirstiniai būtų netinkami ir juos reiktų pakeisti tiksliais pasiskirstymais, o tai būtų sudėtinga apskaičiuoti.

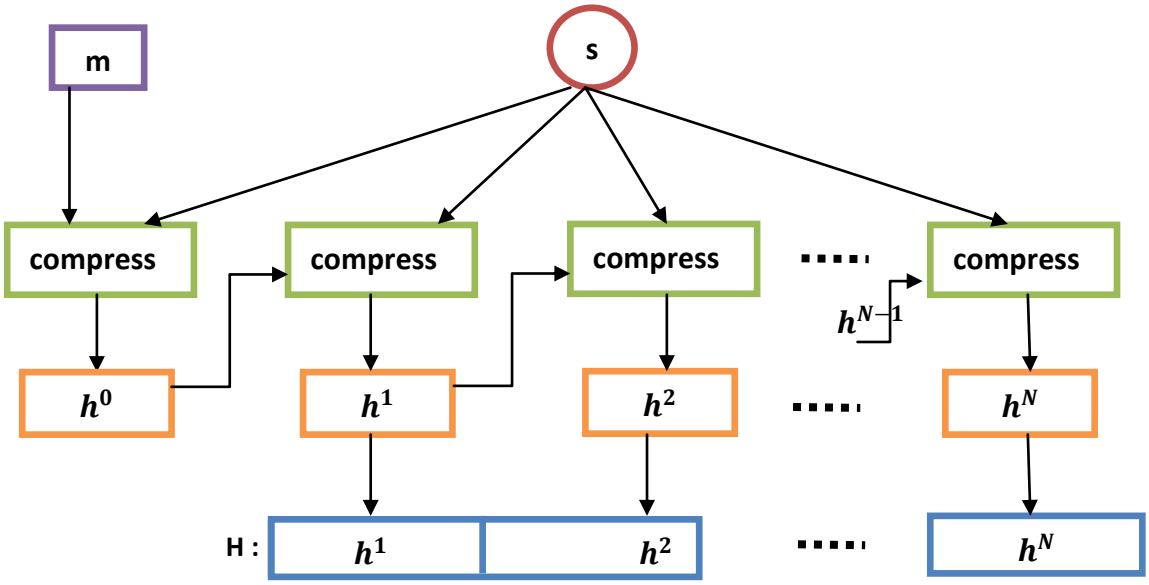
Visas detalus NIST statistinių testų paketo aprašas pateiktas priede Nr. 4.

II. Praktinė magistro darbo dalis

1. Atsitiktinių bitų sekos generavimas, pasinaudojant kompresijos algoritmu

Šio algoritmo [Orv12] tikslas yra, pasinaudojant NIST SHA-3 maišos funkcijų konkurso finalininkų algoritmais ir juose pateiktomis kompresijos funkcijomis, sugeneruoti pseudo-atsitiktinius bitų srautus.

Žemiau pateiktoje 1.1 schemoje matome, kokiu būdu siūlau šį pseudo-atsitiktinių skaičių generatorių realizuoti.



1.1 Schema. Bitų srauto generavimo, pasinaudojant kompresijos funkcijomis, algoritmo schema [Orv12].

Algoritmas sudarytas iš šių žingsnių.

1. Pasirenkame bet kokį pranešimą m bei atitinkamai nuo naudojamos kompresijos funkcijos tipo n bitų druską s.
2. Šiuos du kintamuosius paduodame kompresijos funkcijai ir gauname pirmąją maišą h^o. Pastaba. Ši reikšmė į atsitiktinių skaičių srautą neįtraukiama.
3. Dabar turėdami pirmąją maišos reikšmę, vėl kreipiamės į kompresijos algoritmą ir jam paduodame druską s bei gautą pirmąją maišos reikšmę h^o. Kompresijos funkcija mums grąžina naują maišą h^1. Ši seka bus pirmoji mūsų generuojame sraute $H = h^1$.
4. Vėl kartojame 3 žingsnį, tačiau šį kartą kaip pranešimą paduodame h^1. Gauname maišą h^2 ir ją prijungiame prie srauto $H = h^1 \parallel h^2$.
5. Šiuos žingsnius kartojame tol, kol kompresijos funkcija grąžina maišą h^N. Taip turime sugeneruotą srautą $H = h^1 \parallel h^2 \parallel \cdots \parallel h^N$, kuriam taikome statistinius testus.

Vykdant mano aprašytą algoritmą, druska naudojama tik BLAKE šeimos maišos funkcijose.

1.1. Druskos reikšmė

Pseudo-atsitiktinių skaičių generatorius gali formuoti sekas imdamas du įvesties duomenis: pranešimą ir druską. Tačiau tik BLAKE SHA-3 maišos algoritmas naudoja druskos reikšmę kompresijos funkcijai vykdyti.

Aš pasiūliau dviejų tipų atsitiktines sekas, kurios generuojamos pasinaudojant BLAKE maišos funkcijų šeima. Pirmojo tipo sekos yra generuojamos naudojant konstantinę druskos reikšmė. Ši druska yra sugeneruojama algoritmo vykdymo pradžioje ir nekinta iki pat pabaigos. Antrojo tipo sekos yra generuojamos kiekvienoje iteracijoje naudojant naują druskos reikšmę – druska generuojama kiekvienos iteracijos metu. Gautas sekas palyginau, išanalizavus NIST statistinių testų rinkinio pagalba.

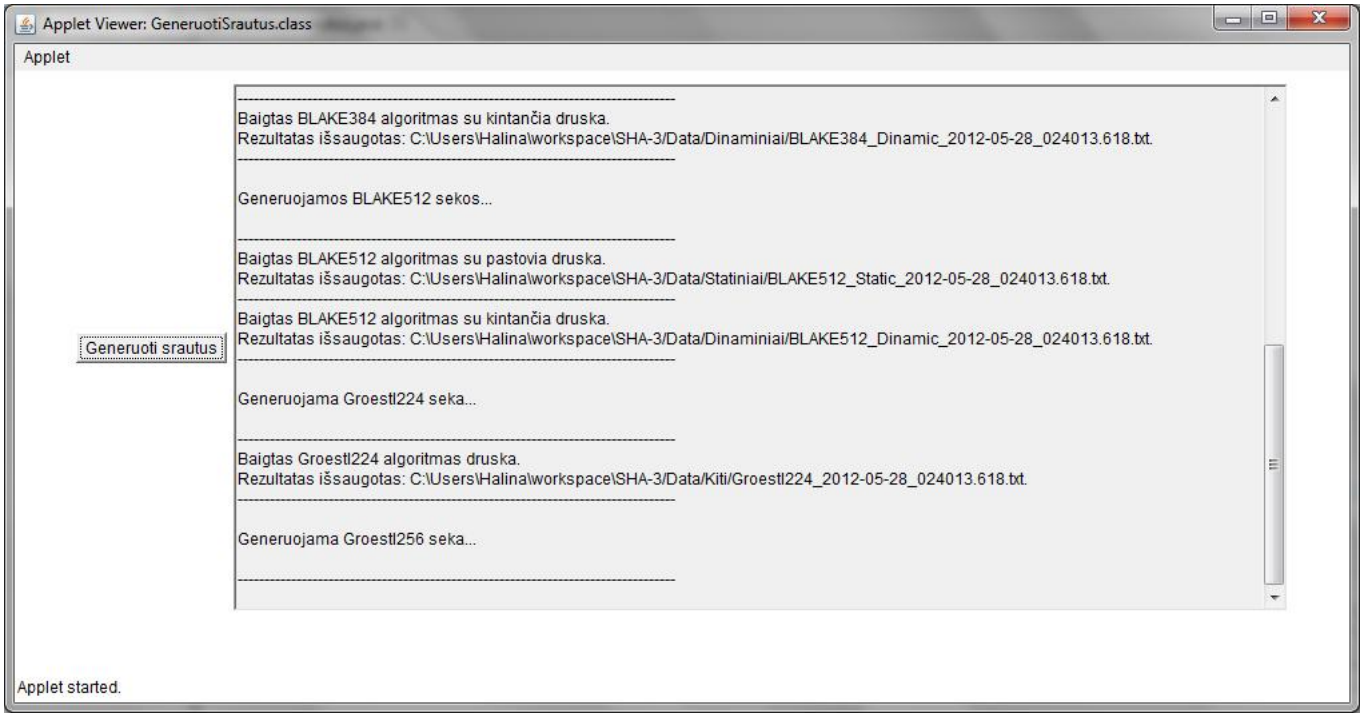
Likę SHA-3 maišos algoritmai kompresijos funkcijose nenaudoja druskos reikšmės, todėl turi tik vieną įvesties duomenį – pranešimą.

1.2. Iteracijos dydis *N*

Kompresijos funkcija aprašytame algoritme gali būti vykdoma neribotą kiekį kartų. Mano pagrindinis tikslas buvo gautas sekas išanalizuoti, naudojantis NIST statistinių testų paketu. Šiame pakete ne vienas testas reikalauja bent 10^6 bitų duomenų. Dėl šios priežasties pasirinkau, kad kompresijos algoritmas būtų kartojamas 5000 kartų. Tuomet tiek mažiausia 224 bitų ilgio grąžinama maiša, tiek 512 bitų ilgio sugeneruojama pakankamo ilgio bitų srautą, kuriam galima taikyti NIST statistinius testus.

2. Algoritmo JAVA realizacija

Sugalvotas pseudo-atsitiktinių skaičių generatorius buvo realizuotas JAVA Applet pagalba. 2.1 pav. matomas pagrindinis programos langas vykdymo metu.



Pav. 2.1. Pagrindinis programos langas vykdymo metu.

Algoritmą sudaro trys pagrindinės klasių grupės:

- SHA-3 maišos algoritmų kompresijos funkcijos;
- atsitiktinių sekų generatorius pranešimui ir druskai kurti;
- pagrindinė algoritmo dalis, jungianti visas klases tarpusavyje.

Kiekviena SHA-3 maišos algoritmo klasė nurodo, koks maišos algoritmas panaudotas ir kokio ilgio maiša yra grąžinama, pvz., BLAKE-224 klasė realizuoja BLAKE maišos funkciją, kuri gražina 224 bitų ilgio maišos rezultatą. Visi Java dokumentai yra saugomi „src“ kataloge, o jų sukompiliuotos klasės – kataloge „bin“.

Sugeneruotos atsitiktinės sekos yra saugomos tekstiniuose rinkmenose „Data“ kataloge, kuris suskaidytas į smulkesnes rinkmenas. Yra naudojama atitinkama saugojimo struktūra:

- katalogas „Dinaminiai“ – saugomi BLAKE algoritmo sugeneruoti srautai, kai buvo naudojama kintanti druskos reikšmė;
- katalogas „Statiniai“ – saugomi BLAKE algoritmo sugeneruoti srautai, kai buvo naudojama pastovi druskos reikšmė;
- katalogas „Kiti“ – saugomi Groestl, JH, Keccak ir Skein algoritmais sugeneruoti srautai (druska nebuvo naudojama).

2.1. Algoritmo veikimo schema

Paleidus Java programą, atsidariusiame lange yra matomas tik vienas mygtukas „Generuoti srautus“ ir laukas tekstiniams pranešimams generuoti. Aktyvavus srautų generavimą, suformuoti pseudo-atsitiktinių

skaičių srautai saugomi nurodytose direktorijose. Vienu programos vykdymu yra sugeneruojami 24 atsitiktinių bitų srautai.

Programos vykdymas yra pilnai automatizuotas – pradedant pradinių reikšmių generavimu, baigiant tekstinių bylų, kuriose saugomi srautai, vardų sukūrimu. Kiekvieno srauto bylos vardas yra sudarytas tokiu būdu:

$$SHA - 3\ algoritmo\ vardas + maišos\ dydis\ bitais + sisteminė\ data.$$

Dėl tokios rinkmenos vardo struktūros, apsisaugoma nuo dviejų ar daugiau identiškų vardų.

3. Gautų srautų analizė NIST statistiniais testais

NIST statistinių testų paketo testus taikiau kiekvienai sugeneruotai sekai. Testavau 10^6 bitų ilgio srautus, bei testus atlikau 30 kartų. Tokiu būdu surinkau pakankamai duomenų, kad galėčiau atlikti lyginamąsias sugeneruotų srautų atsitiktinumo analizes. Atlikau dažnio (monobito) ir linijinio sudėtingumo testus. Tuomet radau gautų testų tikimybių vidurkį ir atlikau lyginimą. Pirmiausia pagal gautus rezultatus pastebėjau, kad nesvarbu, kokio ilgio maišos rezultatas yra grąžinamas, tačiau vienos šeimos funkcijų vidutinė tikimybė yra beveik vienoda. 3.1 lentelėje pateiktos BLAKE šeimos, kai naudojama pastovi druskos reikšmė, tikimybių vidurkiai.

224	256	384	512
0,131177	0,134838	0,145067	0,142375

Lentelė 3.1. BLAKE šeimos algoritmais sugeneruotų srautų atsitiktinumo tikimybės (kai druska yra pastovi)

Dėl šios priežasties nebetyriau kiekvienos maišos funkcijos šeimos narių atskirai, bet ėmiau jų atsitiktinumo tikimybių vidurkį ir lyginau tarpusavyje pačias šeimas.

Pirmiausia palyginau BLAKE sugeneruotų srautų atsitiktinumo tikimybes, kai taikome pastovią, ir kai taikome kintančią druskos reikšmę. 3.1 diagramoje yra akivaizdu, kad kai BLAKE algoritmui paduodama kintanti druskos reikšmė, srauto atsitiktinumo tikimybė yra didesnė – vadinasi gautas srautas yra labiau atsitiktinis. Tačiau skirtumas tarp šių tikimybių yra nedidelis, todėl atsižvelgiant į tai, kad kiekvienai iteracijai kurdamas naują druskos reikšmę, algoritmas srautą generuoja ilgiau ir laiko sąnaudos yra didesnės, turime nuspręsti, ar algoritmas naudojantis kintančia druskos reikšmę yra tikrai efektyvesnis.

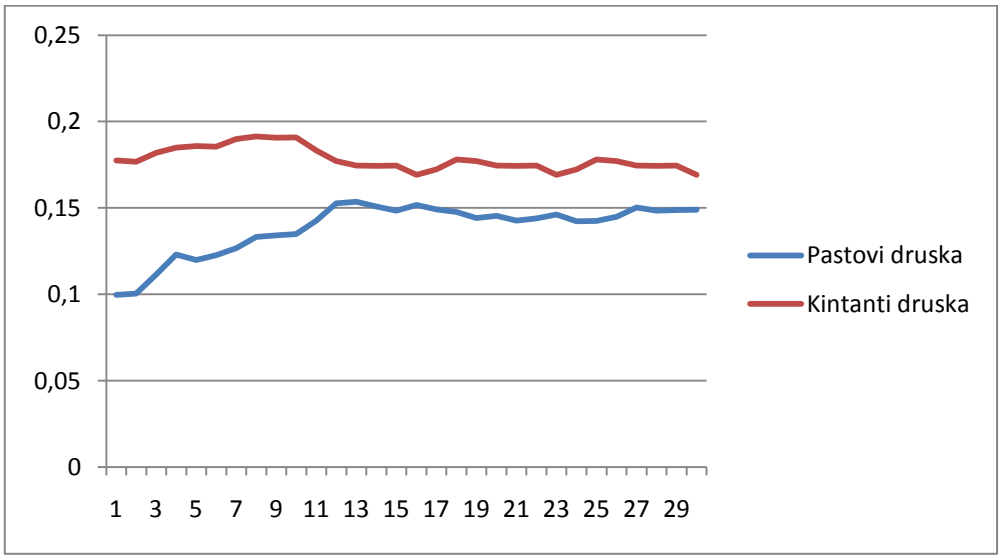


Diagrama 3.1. BLAKE algoritmų su kintančia ir pastovia druskos reikšme atsitiktinumo tikimybių palyginimas.

Jeigu yra itin svarbus srauto elementų atsitiktinumas ir net vos didesnė atsitiktinumo tikimybė būtina, tuomet BLAKE algoritmas, naudojantis kintančią druskos reikšmę yra geresnis pasirinkimas. Tačiau jeigu

svarbesnis yra srauto generavimo laikas ir sistemos apkrova, geresnis yra BLAKE algoritmas su pastovia druskos reikšme.

Toliau analogiškai palyginau Groestl, JH, Keccak ir Skein maišos algoritmais sugeneruotų srautų atsitiktinumo tikimybes.

3.2 diagramoje matome, kad iš šių keturių maišos algoritmų mažiausiai atsitiktinės sekas sugeneravo Groestl algoritmas, o likę – JH, Keccak ir Skein sugeneruoti srautai, pagal atsitiktinumą konkuruoja tarpusavyje.

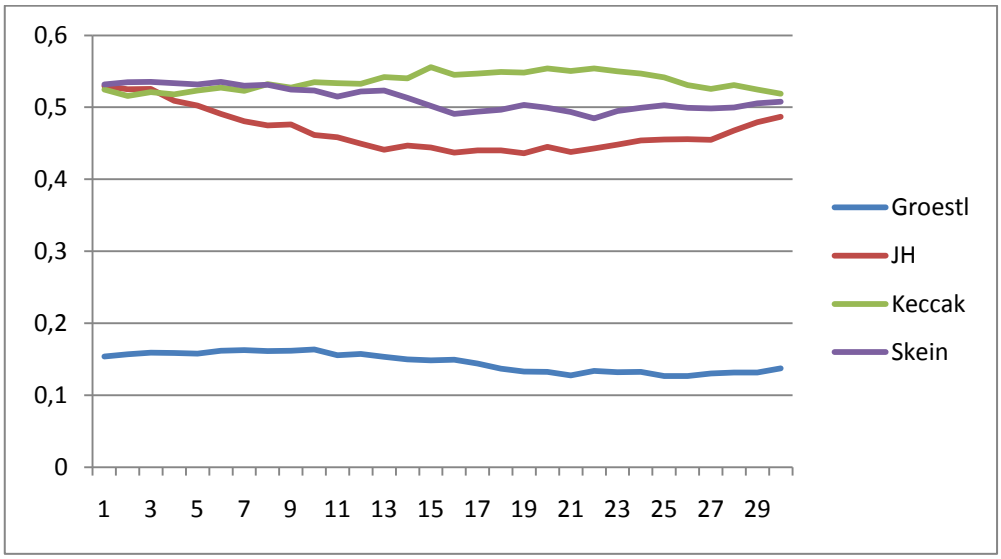


Diagrama 3.2. Groestl, JH, Keccak ir Skein maišos funkcijomis sugeneruotų sekų atsitiktinumo tikimybės palyginimas.

Galiausiai tarpusavyje palyginau visus srautus. 3.3 diagramoje matome, kad labiausiai atsitiktinius srautus generuoja Keccak ir Skein maišos funkcijomis paremtas algoritmas, prasčiausiai BLAKE algoritmas, naudojantis pastovią druskos reikšmę (visos šių srautų atsitiktinumo tikimybės pateiktos priede Nr. 5).

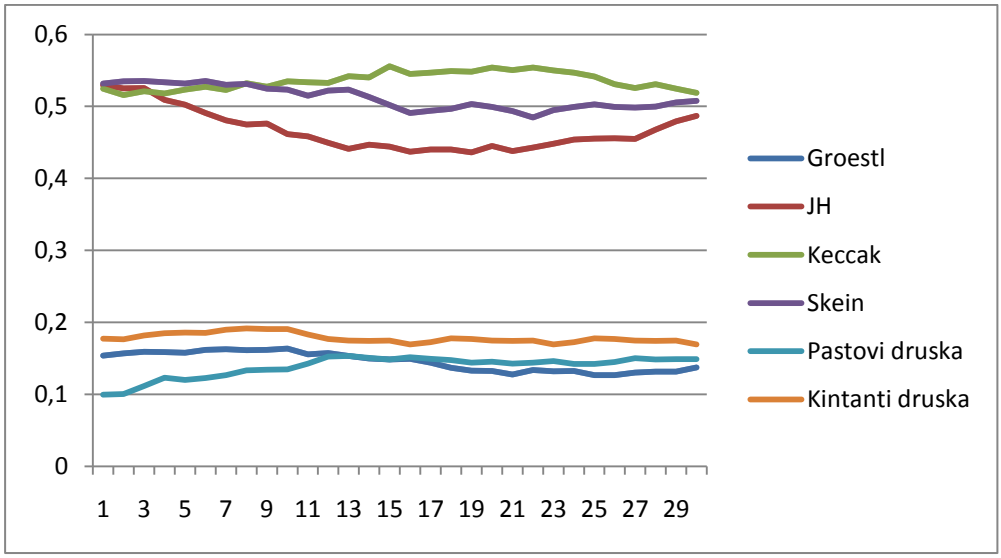


Diagrama 3.3. BLAKE, Groestl, JH, Keccak ir Skein maišos funkcijomis sugeneruotų sekų atsitiktinumo tikimybės palyginimas.

Tad sugeneruoti srautai patenka į dvi dalis:

- BLAKE ir Groestl atsitiktinumo tikimybės neviršija 0,2;
- JH, Keccak ir Skein generuojami srautai yra ne mažesnės nei 0,4 atsitiktinumo tikimybės.

3.2 lentelėje yra pateikti kiekvieno algoritmo sugeneruoto bitų srauto vidutinė atsitiktinumo tikimybė.

BLAKE su pastovia druska	BLAKE su kintančia druska	Groestl	JH	Keccak	Skein
0,138364	0,17826	0,145567	0,466538	0,535615	0,511943

Lentelė 3.2. Kiekvieno algoritmo sugeneruoto bitų srauto vidutinė atsitiktinumo tikimybė

Akivaizdu, kad nors maišos funkcijos JH, Keccak ir Skein yra sudėtingesnės, atlieka žymiai daugiau skaičiavimų, tačiau generuojami srautai yra labiau atsitiktiniai ir jų atsitiktinumo tikimybė yra didžiausia.

Išvados

Rašydama magistro baigiamąjį darbą atlikau išsamią literatūros, statistinių testų analizę, ieškojau naujausių straipsnių kriptografijos tema bei detalių naujų algoritmų aprašymų, aiškinausi NIST statistinio testo paketo kompiliavimo subtilybes Linux aplinkoje.

Rašydama šį darbą galiu pateikti tokias pagrindines išvadas.

- Susipažinau su pagrindinėmis kriptografijos sąvokomis ir dalimis. Išnagrinėjau kriptografinių algoritmų skirstymą ir nustačiau, kuriai daliai priklauso mano nagrinėjamos funkcijos.
- Susipažinau su kriptografinės maišos funkcijos sąvoka. Išnagrinėjau jos veikimo principą ir kelis pavyzdžius.
- Detaliai išsiaiškinau MD4 ir MD5 kriptografinius algoritmus, o MD5 veikimo principą išdėsčiau ir šiame darbe, nes šiais algoritmais yra paremta visa SHA funkcijų šeima.
- Susipažinau su visais SHA šeimos algoritmais ir pastebėjau, kad:
    - nors SHA-1 ir SHA-256 priklaus skirtingiems standartams, tačiau tarpusavyje mažai skiriasi;
    - pagrindiniai skirtumai tarp algoritmų SHA-2 klasėje yra operuojamo bloko ir jame esančio žodžio ilgiai, pagal šiuos parametrus kinta ir visi kiti.
- Susipažinau su NIST vykdomu SHA-3 maišos funkcijos konkursu, reikalavimais ir kriterijais, pagal kuriuos buvo atrinkti finalininkai.
- Nagrinėdama 5 finalininkų algoritmus, pastebėjau, kad nei vienas nėra tarpusavyje panašūs ir visi algoritmo pagrindu naudoja skirtingas funkcijas.
- Visi finalininkų algoritmai nurodomi saugesni daugeliu žinomų atakų atžvilgiu, taip pat greičiau veikia.
- Susipažinau su statistinio testo sąvoka, pagrindiniais jo veikimo principais.
- Susipažinau su atsitiktinės ir pseudoatsitiktinės generatorių apibrėžimu ir jų veikimo principu.
- Išsianalizavau visus NIST statistinio paketo testus, jiems keliamus reikalavimus ir galimas rezultatų interpretacijas.
- Sugalvojau ir realizavau algoritmą, kuris, pasinaudodamas SHA-3 maišos funkcijų finalininkų kompresijos funkcijomis, generuoja pseudo-atsitiktinių bitų seką.
- Pasinaudodama sugeneruotu testiniu srautų paketu, jam taikiau NIST statistinius testus bei pateikiau gautus rezultatus.

Literatūros sąrašas

[AHM+10] J.-P. Aumasson, L. Henzen, W. Meier, R. C.-W. Phan. SHA-3 proposal BLAKE. 2010, p. 76.

[BDP+11] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche. The Keccak Reference. 2011, p. 69

[FLS+08] N. Ferguson, S. Lucks, B. Schneier, D. Whiting, M. Bellare, T. Kohno, J. Callas, J. Walker. The Skein Hash Function Family. 2008, p. 71.

[GKM+11] P. Gauravaram, L. R. Knudsen, K. Matusiewicz, F. Mendel, Ch. Rechberger, M. Schläffer, S. S. Thomsen. Groestl – a SHA-3 Candidate. 2011, p. 41.

[Ken05] C. Kenny. Random Number Generators: An Evaluation and Comparison of Random.org and Some Commonly Used Generators. Trinity College Dublin. 2005, pp. 8-10, 15-25.

[MOV96] A. J. Menezes, P. C. Van Oorschot, S. A. Vanstone. Handbook of Applied Cryptography. CRC Press. 1996, pp. 191-216, 321-376.

[Orv11] H. Orvidaitė. Statistinė SHA-3 konkurso maišos funkcijų analizė. I dalis, 2011.

[Orv12] H. Orvidaitė. Statistinė SHA-3 konkurso maišos funkcijų analizė. II dalis, 2012.

[RSN+10] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. NIST. 2010, pp.1-70, 95-102.

[Sta06] doc. dr. V. Stakėnas. Kodai ir Šifrai. 2006, pp. 200-341

[Wu11] H. Wu. The Hash Function JH. 2011, p. 53.

[1] Wikipedia. Cryptography. http://en.wikipedia.org/wiki/Cryptography, 2012-05-28.

[2] RSA Laboratories. What is a stream cipher? http://www.rsa.com/rsalabs/node.asp?id=2174, 2012-05-28.

Priedas Nr. 1

BLAKE-512

BLAKE-512 maišos funkcija dirba su 64 bitų žodžiais ir grąžina 64 baitų maišos reikšmę. Visų kintamųjų ilgiai, lyginant su BLAKE-256, yra padvigubinti: tarpinė maišos funkcijų grandinės reikšmės yra 512, pranešimo blokai – 1024, druska – 256 ir skaitliukas – 128 bitų ilgio.

Konstantos

BLAKE-512 naudoja tokias pat pradines reikšmes kaip ir SHA-512:

$$IV_0 = 6A09E667F3BCC908 \qquad IV_1 = BB67AE8584CAA73B$$
$$IV_2 = 3C6EF372FE94F82B \qquad IV_3 = A54FF53A5F1D36F1$$
$$IV_4 = 510E527FADE682D1 \qquad IV_5 = 9B05688C2B3E6C1F$$
$$IV_6 = 1F83D9ABFB41BD6B \qquad IV_7 = 5BE0CD19137E2179$$

BLAKE-512 naudojasi 16 konstantų:

$$c_0 = 243F6A8885A308D3 \qquad c_1 = 13198A2E03707344$$
$$c_2 = A4093822299F31D0 \qquad c_3 = 082EFA98EC4E6C89$$
$$c_4 = 452821E638D01377 \qquad c_5 = BE5466CF34E90C6C$$
$$c_6 = C0AC29B7C97C50DD \qquad c_7 = 3F84D5B5B5470917$$
$$c_8 = 9216D5D98979FB1B \qquad c_9 = D1310BA698DFB5AC$$
$$c_{10} = 2FFD72DBD01ADFB7 \qquad c_{11} = B8E1AFED6A267E96$$
$$c_{12} = BA7C9045F12C7F99 \qquad c_{13} = 24A19947B3916CF7$$
$$c_{14} = 0801F2E2858EFC16 \qquad c_{15} = 636920D871574E69$$

BLAKE-512 funkcija naudoja tuos pačius keitiniai $\{0, \dots, 15\}$ (žr. 2.2.1.1 lentelę).

Kompresijos funkcija

BLAKE-512 kompresijos funkcija yra panaši į BLAKE-256, išskyrus ciklo funkciją, kuri vietoj 14 kartų yra vykdoma 16 ir gaunamos $G_i(a, b, c, d)$ reikšmės

$$a \leftarrow a + b + \left(m_{\sigma_r(2i)} \oplus c_{\sigma_r(2i+1)}\right)$$
$$d \leftarrow (d \oplus a) \ggg 32$$
$$c \leftarrow c + d$$
$$b \leftarrow (b \oplus c) \ggg 25$$
$$a \leftarrow a + b + \left(m_{\sigma_r(2i+1)} \oplus c_{\sigma_r(2i)}\right)$$
$$d \leftarrow (d \oplus a) \ggg 16$$
$$c \leftarrow c + d$$
$$b \leftarrow (b \oplus c) \ggg 11$$

Vienintelis funkcijų G skirtumas nuo BLAKE-256 yra žodžio ilgis (64 bitai vietoj 32) ir posūkio atstumai.

Maišos taikymas pranešimui

BLAKE-512 algoritmui pranešimas praplečiamas taip: pridedamas „1“ ir tiek „0“, kol pranešimo ilgis lygsta 895 moduliu 1024. Tada vėl pridedamas „1“ ir 128 bitų pranešimo ilgis (be ženklo, labiausiai reikšmingas baitas eina pirmas):

$$m \leftarrow m \parallel 1000 \dots 0001\langle l \rangle_{128}.$$

Ši procedūra garantuoja, kad praplėsto pranešimo ilgis bus 1024 kartotinis.

Iteracijos algoritmas yra identiškas BLAKE-256 aprašytam.

BLAKE-224

BLAKE-224 yra panašus į BLAKE-256, išskyrus:

- naudoja SHA-224 pradines reikšmes:

$$IV_0 = C1059ED8 \qquad IV_1 = 367CD507$$
$$IV_2 = 3070DD17 \qquad IV_3 = F70E5939$$
$$IV_4 = FFC00B31 \qquad IV_5 = 68581511$$
$$IV_6 = 64F98FA7 \qquad IV_7 = BEFA4FA4$$

- vykdant pranešimo praplėtimą bitas, einantis prie pranešimo ilgį, pakeičiamas „0“:

$$m \leftarrow m \parallel 1000 \ldots 0000\langle l \rangle_{64}$$

- rezultatas sutrumpinamas iki 224 pirmųjų bitų, t.y. maiša po iteracijų grąžina $\mathrm{h}_0^{\mathrm{N}}, \ldots, \mathrm{h}_6^{\mathrm{N}}$ vietoje $\mathrm{h}^{\mathrm{N}} = \mathrm{h}_0^{\mathrm{N}}, \ldots, \mathrm{h}_7^{\mathrm{N}}$.

BLAKE-384

BLAKE-384 yra panašus į BLAKE-512, išskyrus:

- naudoja SHA-384 pradines reikšmes:

$$IV_0 = CBBB9D5DC1059ED8 \qquad IV_1 = 629A292A367CD507$$
$$IV_2 = 9159015A3070DD17 \qquad IV_3 = 152FECD8F70E5939$$
$$IV_4 = 67332667FFC00B31 \qquad IV_5 = 8EB44A8768581511$$
$$IV_6 = DB0C2E0D64F98FA7 \qquad IV_7 = 47B5481DBEFA4FA4$$

- vykdant pranešimo praplėtimą bitas, einantis prie pranešimo ilgį, pakeičiamas „0“:

$$m \leftarrow m \parallel 1000 \ldots 0000\langle l \rangle_{128}$$

- rezultatas sutrumpinamas iki 384 pirmųjų bitų, t.y. maiša po iteracijų grąžina $\mathrm{h}_0^{\mathrm{N}}, \ldots, \mathrm{h}_5^{\mathrm{N}}$ vietoje $\mathrm{h}^{\mathrm{N}} = \mathrm{h}_0^{\mathrm{N}}, \ldots, \mathrm{h}_7^{\mathrm{N}}$.

Priedas Nr. 2

Žemiau esančioje lentelėje pateiktos Groestl algoritmo naudojamos Sbox reikšmės:

	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
10	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
20	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
30	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
40	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
50	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
60	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
70	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
80	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
90	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a0	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b0	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c0	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d0	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e0	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f0	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Priedas Nr. 3

E_8 ciklo konstantos

Ciklo konstantos yra apskaičiuojamas naudojantis pirmojo ciklo konstantomis ir ciklo funkcija *R6* (kai *R6* ciklo konstantos yra 0).

$$C00 = 6a09e667f3bcc908b2fb1366ea957d3e3adec17512775099da2f590b0667322a$$
$$C01 = bb896bf05955abcd5281828d66e7d99ac4203494f89bf12817deb43288712231$$
$$C02 = 1836e76b12d79c55118a1139d2417df52a2021225ff6350063d88e5f1f91631c$$
$$C03 = 263085a7000fa9c3317c6ca8ab65f7a7713cf4201060ce886af855a90d6a4eed$$
$$C04 = 1cebafd51a156aeb62a11fb3be2e14f60b7e48de85814270fd62e97614d7b441$$
$$C05 = e5564cb574f7e09c75e2e244929e9549279ab224a28e445d57185e7d7a09fdc1$$
$$C06 = 5820f0f0d764cff3a5552a5e41a82b9eff6ee0aa615773bb07e8603424c3cf8a$$
$$C07 = b126fb741733c5bfcef6f43a62e8e5706a26656028aa897ec1ea4616ce8fd510$$
$$C08 = dbf0de32bca77254bb4f562581a3bc991cf94f225652c27f14eae958ae6aa616$$
$$C09 = e6113be617f45f3de53cff03919a94c32c927b093ac8f23b47f7189aadb9bc67$$
$$C10 = 80d0d26052ca45d593ab5fb3102506390083afb5ffe107dacfcba7dbe601a12b$$
$$C11 = 43af1c76126714dfa950c368787c81ae3beecf956c85c962086ae16e40ebb0b4$$
$$C12 = 9aee8994d2d74a5cdb7b1ef294eed5c1520724dd8ed58c92d3f0e174b0c32045$$
$$C13 = 0b2aa58ceb3bdb9e1eef66b376e0c565d5d8fe7bacb8da866f859ac521f3d571$$
$$C14 = 7a1523ef3d970a3a9b0b4d610e02749d37b8d57c1885fe4206a7f338e8356866$$
$$C15 = 2c2db8f7876685f2cd9a2e0ddb64c9d5bf13905371fc39e0fa86e1477234a297$$
$$C16 = 9df085eb2544ebf62b50686a71e6e828dfed9dbe0b106c9452ceddff3d138990$$
$$C17 = e6e5c42cb2d460c9d6e4791a1681bb2e222e54558eb78d5244e217d1bfcf5058$$
$$C18 = 8f1f57e44e126210f00763ff57da208a5093b8ff7947534a4c260a17642f72b2$$
$$C19 = ae4ef4792ea148608cf116cb2bff66e8fc74811266cd641112cd17801ed38b59$$
$$C20 = 91a744efbf68b192d0549b608bdb3191fc12a0e83543cec5f882250b244f78e4$$
$$C21 = 4b5d27d3368f9c17d4b2a2b216c7e74e7714d2cc03e1e44588cd9936de74357c$$
$$C22 = 0ea17cafb8286131bda9e3757b3610aa3f77a6d0575053fc926eea7e237df289$$
$$C23 = 848af9f57eb1a616e2c342c8cea528b8a95a5d16d9d87be9bb3784d0c351c32b$$
$$C24 = c0435cc3654fb85dd9335ba91ac3dbde1f85d567d7ad16f9de6e009bca3f95b5$$
$$C25 = 927547fe5e5e45e2fe99f1651ea1cbf097dc3a3d40ddd21cee260543c288ec6b$$
$$C26 = c117a3770d3a34469d50dfa7db020300d306a365374fa828c8b780ee1b9d7a34$$
$$C27 = 8ff2178ae2dbe5e872fac789a34bc228debf54a882743caad14f3a550fdbe68f$$
$$C28 = abd06c52ed58ff091205d0f627574c8cbc1fe7cf79210f5a2286f6e23a27efa0$$
$$C29 = 631f4acb8d3ca4253e301849f157571d3211b6c1045347befb7c77df3c6ca7bd$$
$$C30 = ae88f2342c23344590be2014fab4f179fd4bf7c90db14fa4018fcce689d2127b$$
$$C31 = 93b89385546d71379fe41c39bc602e8b7c8b2f78ee914d1f0af0d437a189a8a4$$
$$C32 = 1d1e036abeef3f44848cd76ef6baa889fcec56cd7967eb909a464bfc23c72435$$
$$C33 = a8e4ede4c5fe5e88d4fb192e0a0821e935ba145bbfc59c2508282755a5df53a5$$
$$C34 = 8e4e37a3b970f079ae9d22a499a714c875760273f74a9398995d32c05027d810$$
$$C35 = 61cfa42792f93b9fde36eb163e978709fafa7616ec3c7dad0135806c3d91a21b$$
$$C36 = f037c5d91623288b7d0302c1b941b72676a943b372659dcd7d6ef408a11b40c0$$
$$C37 = 2a306354ca3ea90b0e97eaebcea0a6d7c6522399e885c613de824922c892c490$$
$$C38 = 3ca6cdd788a5bdc5ef2dceeb16bca31e0a0d2c7e9921b6f71d33e25dd2f3cf53$$
$$C39 = f72578721db56bf8f49538b0ae6ea470c2fb1339dd26333f135f7def45376ec0$$
$$C40 = e449a03eab359e34095f8b4b55cd7ac7c0ec6510f2c4cc79fa6b1fee6b18c59e$$
$$C41 = 73bd6978c59f2b219449b36770fb313fbe2da28f6b04275f071a1b193dde2072$$

Priedas Nr. 4

1. Dažnio (monobito) testas

Testo tikslas

Šiame teste dėmesys kreipiamas nulių ir vienetų santykiui sekoje. Šio bandymo tikslas yra nustatyti, ar nulių ir vienetų skaičius sekoje yra apytiksliai toks pat, kokio būtų tikimąsi tikrai atsitiktinėje sekoje. Testas įvertina vienetų dalies artumą ½, t.y. nulių ir vienetų kiekis sekoje turėtų būti beveik vienodas. Visi tolimesni bandymai priklauso nuo šio testo rezultato.

Kreipimasis į funkciją

Frequency(n), kur:

n bitų srauto ilgis

Dar vienas įvesties duomuo, kurį naudoja funkcija, bet pateikiamas paties testo:

ε testuojama bitų seka, kuri yra sugeneruota RNG arba PRNG; funkcijos kvietimo metu, pateikiama tokiu pavidalu: $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$.

Testo statistika ir reikšmių pasiskirstymas

S_{obs} absoliuti sumos X_i reikšmė (kur $X_i = 2\varepsilon_i - 1 = \pm 1$) sekoje, padalinta iš sekos ilgio kvadratinės šaknies.

Testo statistikos reikšmių pasiskirstymas dideliems n yra pusiau normalusis. (Pastaba. Jeigu z (kur $z = S_{obs}/\sqrt{2}$) yra normaliojo pasiskirstymo, tai $|z|$ yra pusiau normaliojo pasiskirstymo.) Jeigu seka yra atsitiktinė, tai ± 1 vienas kitą panaikins ir testo statistika bus artima 0. Jeigu yra per daug nulių arba vienetų, tai testo statistika bus didesnė už nulį.

Testo aprašas

1. Keitiniai į ± 1: testui duotos sekos (ε) nuliai ir vienetai yra konvertuojami į atitinkamai -1 ir $+1$ ir iš jų suformuojama suma $S_n = X_1 + X_2 + \cdots + X_n$, kur $X_i = 2\varepsilon_i - 1$.
2. Apskaičiuojama testo statistika $S_{obs} = \frac{|S_n|}{\sqrt{n}}$.
3. Apskaičiuojama $P - \text{reikšmė} = \text{erfc}\left(\frac{S_{obs}}{\sqrt{2}}\right)$, kur erfc yra papildanti klaidos funkcija.

Apsisprendimo taisyklė (1% lygmenyje)

Jeigu apskaičiuota $P - \text{reikšmė} < \alpha$, kur $\alpha \in [0{,}001, 0{,}01]$, tai darome išvadą, kad seka nėra atsitiktinė, priešingu atveju, kad seka yra atsitiktinė.

Išvados ir gautų rezultatų interpretavimas

Reikia paminėti, kad jeigu $P -$ reikšmė yra mažesnė už α, tai šis rezultatas buvo gautas, nes $|S_n|$ arba $|S_{obs}|$ buvo pakankamai didelis skaičius. Didelė teigiama S_n reikšmė nulemiama per didelio vienetų kiekio sekoje, o didelė neigiama reikšmė – per didelio nulių kiekio.

Reikalavimai įvesties duomenims

Rekomenduojama, kad kiekviena testuojama seka būtų bent 100 bitų ilgio (t.y. $n \geq 100$).

2. Dažnio testas bloke

Testo tikslas

Šiame teste dėmesys kreipiamas vienetų santykiui M-bitų bloke. Šio bandymo tikslas yra nustatyti, ar vienetų dalis M-bitų bloke yra apytiksliai $M/2$, kaip teigiama atsitiktinumo prielaidoje. Jeigu $M = 1$, tada gaunamas dažnio (monobito) testas.

Kreipimasis į funkciją

BlockFrequency(M, n), kur:

M kiekvieno bloko ilgis
n bitų srauto ilgis

Dar vienas įvesties duomuo, kurį naudoja funkcija, bet pateikiamas paties testo:

ε testuojama bitų seka, kuri yra sugeneruota RNG arba PRNG; funkcijos kvietimo metu, pateikiama tokiu pavidalu: $\varepsilon = \varepsilon_1, \varepsilon_2, \dots, \varepsilon_n$.

Testo statistika ir reikšmių pasiskirstymas

$X^2(obs)$ įvertinimas, kaip tiksliai vienetų dalis duotame M-bitų bloke tenkina santykį $(1/2)$.
Testo statistikos reikšmių pasiskirstymas yra X^2 skirstinys.

Testo aprašas

1. Seka padalinama į $\mathrm{N} = \left\lfloor \frac{\mathrm{n}}{\mathrm{M}} \right\rfloor$ nepersidengiančių blokų. Išmetami nenaudojami bitai.
2. Naudojantis formule $\pi_\mathrm{i} = \frac{\sum_{\mathrm{j}=1}^{\mathrm{M}} \varepsilon_{(\mathrm{i}-1)\mathrm{M}+\mathrm{j}}}{\mathrm{M}}$, kur $\mathrm{i} \in [1, \mathrm{N}]$, apskaičiuojamas vienetų kiekis π_i kiekviename M-bitų bloke.
3. Apskaičiuojama statistikos X^2 reikšmė: $\mathrm{X}^2(\mathrm{obs}) = 4\mathrm{M} \sum_{\mathrm{i}=1}^{\mathrm{N}} (\pi_\mathrm{i} - 1/2)^2$.
4. Apskaičiuojama $\mathrm{P} - \text{reikšmė} = \mathrm{igamc}(\mathrm{N}/2, \mathrm{X}^2(\mathrm{obs})/2)$, kur igamc yra nepilna gama funkcija $\mathrm{Q}(\mathrm{a}, \mathrm{x})$.

Apsisprendimo taisyklė (1% lygmenyje)

Jeigu apskaičiuota $\mathrm{P} - \text{reikšmė} < \alpha$, kur $\alpha \in [0{,}001, 0{,}01]$, tai darome išvadą, kad seka nėra atsitiktinė, priešingu atveju, kad seka yra atsitiktinė.

Išvados ir gautų rezultatų interpretavimas

Reikia paminėti, kad mažos $\mathrm{P} - \text{reikšmės}(< \alpha)$ būtų gautos esant dideliam nuokrypiui nuo reikiamo nulių ir vienetų santykio bent viename bloke.

Reikalavimai įvesties duomenims

Rekomenduojama, kad kiekviena testuojama seka būtų bent 100 bitų ilgio (t.y. $\mathrm{n} \geq 100$). Reikia pažymėti, kad $\mathrm{n} \geq \mathrm{MN}$. Vadinasi bloko dydis M turėtų būti pasirinktas toks, kad $\mathrm{M} \geq 20, \mathrm{M} > 0{,}01\mathrm{n}$ ir $\mathrm{N} < 100$.

3. Eigos testas

Testo tikslas

Šiame teste dėmesys kreipiamas į nenutrūkstamų identiškų bitų eilučių kiekį sekoje. Eilutė, kurios ilgis yra k, yra sudaryta iš k vienodų bitų bei prieš ir po kurios yra priešingos reikšmės bitas. Šio bandymo tikslas yra nustatyti, ar skirtingų ilgių nulių ir vienetų eilučių skaičius yra toks, kokio tikimasi atsitiktinėse sekose. Be to, šis testas nustato, ar svyravimas tarp tokių nulių ir vienetų eilučių yra per greitas ar per lėtas.

Kreipimasis į funkciją

$\mathrm{Runs}(n)$, kur:

n bitų srauto ilgis

Dar vienas įvesties duomuo, kurį naudoja funkcija, bet pateikiamas paties testo:

ε testuojama bitų seka, kuri yra sugeneruota RNG arba PRNG; funkcijos kvietimo metu, pateikiama tokiu pavidalu: $\varepsilon = \varepsilon_1, \varepsilon_2, \dots, \varepsilon_n$.

Testo statistika ir reikšmių pasiskirstymas

$V_n(obs)$ visų eilučių skaičius (t.y. eilučių iš nulių + eilučių iš vienetų) sekoje.
Testo statistikos reikšmių pasiskirstymas yra X^2 skirstinys.

Testo aprašas

Pastaba. Eigos testas atlieka ir dažnio testą.

1. Prieš atliekant testą, apskaičiuojama vienetų dalis π_i sekoje: $\pi_i = \frac{\sum_j \varepsilon_j}{n}$
2. Nustatoma, ar dažnio testas pavyko: jei galima tokia nelygybė $|\pi - 1/2| \geq \tau$, tada eigos testo vykdyti nebereikia (t.y. dažnio testas grąžino neigiamą atsakymą). Tokiu atveju $P-$ reiškmė grąžina 0. Reikia pabrėžti, kad šiam testui reikšmė $\tau = \frac{2}{\sqrt{n}}$ yra aprašyta testo kode.
3. Apskaičiuojama testo statistikos $V_n(obs) = \sum_{k=1}^{n-1} r(k) + 1$ reikšmė, kur $r(k) = 0$, jeigu $\varepsilon_k = \varepsilon_{k+1}$, priešingu atveju $r(k) = 1$.
4. Apskaičiuojama $P-$ reikšmė $= \text{erfc}\left(\frac{|V_n(obs) - 2n\pi(1-\pi)|}{2\sqrt{2n\pi}(1-\pi)}\right)$.

Apsisprendimo taisyklė (1% lygmenyje)

Jeigu apskaičiuota $P-$ reikšmė $< \alpha$, kur $\alpha \in [0{,}001, 0{,}01]$, tai darome išvadą, kad seka nėra atsitiktinė, priešingu atveju, kad seka yra atsitiktinė.

Išvados ir gautų rezultatų interpretavimas

Reikia paminėti, kad didelės $V_n(obs)$ reikšmės būtų per dažno bitų eilutės pasikartojimo dažnio, per mažos reikšmės – per reto bitų eilutės pasikartojimo dažnio požymis. Bitų pasikartojimo dažnis – tai bitų kitimas iš nulio į vienetą ir atvirkščiai). Seka su mažu bitų pasikartojimo dažniu yra laikoma turinti per mažai pasikartojimų nei tikimasi atsitiktinėje sekoje.

Reikalavimai įvesties duomenims

Rekomenduojama, kad kiekviena testuojama seka būtų bent 100 bitų ilgio (t.y. $n \geq 100$).

4. Ilgiausios vienetų sekos bloke testas

Testo tikslas

Šiame teste dėmesys kreipiamas į ilgiausią nenutrūkstamą vienetų eilutę M-bitų bloke. Šio bandymo tikslas yra nustatyti, ar ilgiausios vienetų eilutės ilgis bloke yra artimas ilgiausiai vienetų eilutei atsitiktinėje sekoje. Reikia pabrėžti, kad tikimosi ilgiausios vienetų eilutės ilgio netaisyklingumas nurodo, kad toks pat požymis bus ir su ilgiausia nulių eilute. Dėl šios priežasties užtenka tikrinti tik vienetų eilutes.

Kreipimasis į funkciją

LongestRunOfOnes(n), kur:

n bitų srauto ilgis

Papildomi įvesties duomenys, kuriuos naudoja funkcija, bet yra pateikiami paties testo:

ε testuojama bitų seka, kuri yra sugeneruota RNG arba PRNG; funkcijos kvietimo metu, pateikiama tokiu pavidalu: $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$.

M kiekvieno bitų bloko ilgis. Kodas palaiko tris galimus bloko ilgius: $M = 8$, $M = 128$ ir $M = 10^4$ atitinkamai nuo bitų srauto ilgio n:

Minimalus n	M
128	8
6272	128
750000	10^4

4.1 Lentelė. Srauto ir bloko dydžių santykis

N blokų skaičius, randamas atitinkamai iš bloko ilgio M.

Testo statistika ir reikšmių pasiskirstymas

$X^2(obs)$ skaičius, nurodantis, kaip tiksliai ilgiausios eilutės ilgis M-bitų bloke atitinka tikimasi ilgiausios eilutės ilgį atsitiktinėje sekoje.

Testo statistikos reikšmių pasiskirstymas yra X^2 skirstinys.

Testo aprašas

1. Seka padalinama į M-bitų blokus.
2. Ilgiausių vienetų bloke eilutės dažnumas v_i padalinamas į kategorijas, kur kiekvienas langelis saugo duoto ilgio ilgiausios vienetų eilutės skaičių.
   Pagal M palaikomas reikšmes, v_i langeliai turės tokias reikšmes:

v_i	$M = 8$	$M = 128$	$M = 10^4$
v_0	≤ 1	≤ 4	≤ 10
v_1	2	5	11
v_2	3	6	12
v_3	≥ 4	7	13
v_4		8	14
v_5		≥ 9	15
v_6			≥ 16

4.2 Lentelė. v_i reikšmės

3. Apskaičiuojama $X^2(\text{obs}) = \sum_{i=0}^{K} \frac{(v_i - N\pi_i)^2}{N\pi_i}$, kur N ir K reikšmės apskaičiuojamos pagal M reikšmės iš lentelės:

M	K	N
8	3	16
128	5	49
10^4	6	75

4.3 Lentelė. K ir N reikšmės

4. Apskaičiuojama $\text{P} - \text{reikšmė} = \text{igamc}\left(\frac{K}{2}, \frac{X^2(\text{obs})}{2}\right)$.

Apsisprendimo taisyklė (1% lygmenyje)

Jeigu apskaičiuota $\text{P} - \text{reikšmė} < \alpha$, kur $\alpha \in [0{,}001, 0{,}01]$, tai darome išvadą, kad seka nėra atsitiktinė, priešingu atveju, kad seka yra atsitiktinė.

Išvados ir gautų rezultatų interpretavimas

Reikia paminėti, kad didelės $X^2(\text{obs})$ reikšmės reiškia, testuojama seka turi vienetų grupes.

Reikalavimai įvesties duomenims

Rekomenduojama, kad kiekviena testuojama seka, būtų tokio ilgio, kuris nurodomas 3.3.4.1 lentelėje.

5. Dvejetainės matricos laipsnio testas

Testo tikslas

Šiame teste dėmesys kreipiamas į visos sekos matricų disjunkcijos laipsnį. Šio bandymo tikslas yra patikrinti tiesinę priklausomybę tarp fiksuoto ilgio eilučių analizuojamoje sekoje.

Kreipimasis į funkciją

Rank(n), kur:

n bitų srauto ilgis

Papildomi įvesties duomenys, kuriuos naudoja funkcija, bet yra pateikiami paties testo:

ε testuojama bitų seka, kuri yra sugeneruota RNG arba PRNG; funkcijos kvietimo metu, pateikiama tokiu pavidalu: $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$.

M eilučių kiekis kiekvienoje matricoje. Testavimo rinkinyje M yra lygus 32.

Q stulpelių skaičius matricoje. Testavimo rinkinyje Q yra lygus 32.

Testo statistika ir reikšmių pasiskirstymas

$X^2(obs)$ skaičius, nurodantis, kaip tiksliai matricos laipsnis atitinka tikimąsi matricos laipsnį atsitiktinėje sekoje.

Testo statistikos reikšmių pasiskirstymas yra X^2 skirstinys.

Testo aprašas

1. Seka nuosekliai padalijama į $M \cdot Q$ atskirus blokus. Tokių blokų bus $N = \left\lfloor \frac{n}{MQ} \right\rfloor$. Iš šių blokų suformuojamos MxQ matricos. Kiekviena eilutė yra nuosekliai išdėstyti originalios sekos Q-bitų blokai.
2. Apskaičiuojamas kiekvienos sekos dvejetainis laipsnis (R_l), kur $l = 1, \dots N$.
3. Apibrėžiamas F_M – matricų skaičius, kurių laipsnis yra $R_l = M$, F_{M-1} – matricų skaičius, kurių laipsnis yra $R_l = M - 1$ ir $N - F_M - F_{M-1}$ – likusių matricų skaičius.
4. Apskaičiuojama $X^2(\text{obs}) = \frac{(F_M - 0{,}2888N)^2}{0{,}2888N} + \frac{(F_{M-1} - 0{,}5776N)^2}{0{,}5776N} + \frac{(N - F_M - F_{M-1} - 0{,}1336N)^2}{0{,}1336N}$.
5. Apskaičiuojama $P - \text{reikšmė} = e^{-X^2(\text{obs})/2}$.

Apsisprendimo taisyklė (1% lygmenyje)

Jeigu apskaičiuota $P - \text{reikšmė} < \alpha$, kur $\alpha \in [0{,}001, 0{,}01]$, tai darome išvadą, kad seka nėra atsitiktinė, priešingu atveju, kad seka yra atsitiktinė.

Išvados ir gautų rezultatų interpretavimas

Reikia paminėti, kad didelės $X^2(\text{obs})$ bei atitinkamai mažos $P - \text{reikšmės}$ reiškia, laipsnių išsidėstymo nuokrypį nuo tikėtinos atsitiktinės sekos.

Reikalavimai įvesties duomenims

Rekomenduojama, kad minimalus bitų sekos ilgis būtų $n \geq 38MQ$, t.y. būtų sudarytos bent 38 matricos.

6. Diskrečioji Furjė transformacijos (spektrinis) testas

Testo tikslas

Šiame teste dėmesys kreipiamas į sekos diskrečiosios Furjė transformacijos maksimumus. Šio bandymo tikslas yra aptikti periodines analizuojamos sekos savybes, kurios lemtų nuokrypį nuo atsitiktinės sekos savybių. Tikslas pastebėti, ar elementas viršija 95% slenkstį yra reikšmingi 5%.

Kreipimasis į funkciją

DiscreteFourierTransform(n), kur:

n bitų srauto ilgis

Papildomas įvesties duomuo, kurį naudoja funkcija, bet yra pateikiamas paties testo:

ε testuojama bitų seka, kuri yra sugeneruota RNG arba PRNG; funkcijos kvietimo metu, pateikiama tokiu pavidalu: $\varepsilon = \varepsilon_1, \varepsilon_2, \dots, \varepsilon_n$.

Testo statistika ir reikšmių pasiskirstymas

d normalizuotas skirtumas tarp tiriamo ir tikėtino komponentų, kurie yra žemiau 95% slenksčio.

Testo statistikos reikšmių pasiskirstymas normalusis skirstinys.

Testo aprašas

1. Keitiniai į ± 1: testui duotos sekos (ε) nuliai ir vienetai yra konvertuojami į atitinkamai -1 ir $+1$ ir iš jų suformuojama seka $X = x_1, x_2, \dots, x_n$, kur $x_i = 2\varepsilon_i - 1$.
2. Sekai X taikoma diskrečioji Furjė transformacija $S = \text{DFT}(X)$. Gaunama kompleksinių kintamųjų seka, kuri atvaizduoja periodinius bitų sekos komponentų pasikartojimus.
3. Apskaičiuojamas $M = \text{modulus}(S') \equiv |S'|$, kur S' yra pusė pirmųjų elementų iš S, o modulio funkcija sukuria maksimumų seką.

4. Apskaičiuojamas $T = \sqrt{\left(\log\frac{1}{0,05}\right)n}$ – 95% maksimalių reikšmių slenkstis. Pagal atsitiktinumo prielaidą, 95% iš testo gautų reikšmių neturėtų viršyti T.
5. Apskaičiuojama teorinė maksimumų reikšmė $N_0 = 0,95n/2$.
6. Apskaičiuojama reali maksimumų reikšmė N_1.
7. Apskaičiuojama $d = \frac{(N_1 - N_0)}{\sqrt{n(0,95)(0,05)/4}}$.
8. Apskaičiuojama $P - \text{reikšmė} = \text{erfc}\left(\frac{|d|}{\sqrt{2}}\right)$.

Apsisprendimo taisyklė (1% lygmenyje)

Jeigu apskaičiuota $P - \text{reikšmė} < \alpha$, kur $\alpha \in [0,001, 0,01]$, tai darome išvadą, kad seka nėra atsitiktinė, priešingu atveju, kad seka yra atsitiktinė.

Išvados ir gautų rezultatų interpretavimas

Per maža d reikšmė, nurodo, kad buvo per mažai maksimumų žemiau reikšmės T ir per daug maksimumų virš jos.

Reikalavimai įvesties duomenims

Rekomenduojama, kad kiekviena testuojama seka būtų bent 1000 bitų ilgio (t.y. $n \geq 1000$).

7. Nepersidengiančių šablonų parinkimo testas

Testo tikslas

Šiame teste dėmesys kreipiamas į nurodytų eilučių radimą analizuojamoje sekoje. Šio bandymo tikslas yra nustatyti tokius generatorius, kurie sukuria per daug ne periodinių pasikartojimų. Tiek šiame teste, tiek persidengiančių šablonų parinkimo teste yra naudojamas m-bitų langas, kurio užduotis ieškoti konkrečių m-bitų šablonų. Jeigu šablonas nesurandamas, tai langas paslenkamas per vieną bitą, jei randamas – langas perkeliamas į kitą poziciją iš karto po rasto šablono pabaigos.

Kreipimasis į funkciją

NonOverlappingTemplateMatching(m, n), kur:

m bitų kiekis kiekviename šablone

n bitų srauto ilgis

Papildomi įvesties duomenys, kuriuos naudoja funkcija, bet yra pateikiami paties testo:

ε testuojama bitų seka, kuri yra sugeneruota RNG arba PRNG; funkcijos kvietimo metu, pateikiama tokiu pavidalu: $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$.

B m-bitų ilgio šablonas.

M testuojamos eilutės iš sekos ε ilgis bitais.

N nepriklausomų blokų kiekis. Teste šis kiekis yra 8.

Testo statistika ir reikšmių pasiskirstymas

$X^2(obs)$ skaičius, nurodantis, kaip tiksliai matricos surastų šablonų skaičius atitinka tikėtinų šablonų skaičių atsitiktinėje sekoje.

Testo statistikos reikšmių pasiskirstymas yra X^2 skirstinys.

Testo aprašas

1. Seka padalinama į N nepriklausomų blokų, kurių kiekvieno ilgis yra M.
2. Apibrėžiamas W_j, kai $j = 1, \ldots, N$. W_j – žymi, kiek kartų šablonas tikrino bloką j.
3. Remdamiesi atsitiktinumo prielaidą, paskaičiuojame teorines vidurkio μ ir nuokrypio σ^2 reikšmes:
4. $\mu = \frac{(M-m+1)}{2}$ $\qquad\qquad$ $\sigma^2 = M\left(\frac{1}{2^m} - \frac{2m-1}{2^{2m}}\right)$
5. Apskaičiuojama $X^2(obs) = \sum_{j=1}^{N} \frac{(W_j - \mu)^2}{\sigma^2}$.

6. Apskaičiuojama $\mathrm{P-reikšmė} = \mathrm{igamc}\left(\frac{\mathrm{N}}{2}, \frac{\mathrm{X}^2(\mathrm{obs})}{2}\right)$. Reikia atkreipti dėmesį, kad bus gautos kelios $\mathrm{P-reikšmės}$, t.y. kiekvienam šablonui apskaičiuota viena $\mathrm{P-reikšmė}$.

Apsisprendimo taisyklė (1% lygmenyje)

Jeigu apskaičiuota $\mathrm{P-reikšmė} < \alpha$, kur $\alpha \in [0{,}001, 0{,}01]$, tai darome išvadą, kad seka nėra atsitiktinė, priešingu atveju, kad seka yra atsitiktinė.

Išvados ir gautų rezultatų interpretavimas

Jeigu $\mathrm{P-reikšmė}$ yra labai maža ($< \alpha$), vadinasi sekoje yra nereguliarūs galimų šablonų atitikimai.

Reikalavimai įvesties duomenims

Rekomenduojama, kad $\mathrm{m} = 9$ arba $\mathrm{m} = 10$.

8. Persidengiančių šablonų parinkimo testas

Testo tikslas

Šiame teste dėmesys kreipiamas į nurodytų eilučių radimą analizuojamoje sekoje. Šio bandymo tikslas yra nustatyti tokius generatorius, kurie sukuria per daug ne periodinių pasikartojimų. Tiek šiame teste, tiek nepersidengiančių šablonų parinkimo teste yra naudojamas m-bitų langas, kurio užduotis ieškoti konkrečių m-bitų šablonų. Vienintelis skirtumas nuo nepersidengiančių šablonų testo yra tas, kad tiek neradus atitikimo šablonui, tiek jį suradus langas paslenkamas per vieną bito poziciją ir paieška vykdoma toliau.

Kreipimasis į funkciją

OverlappingTemplateMatching(m, n), kur:

- m bitų kiekis kiekviename šablone
- n bitų srauto ilgis

Papildomi įvesties duomenys, kuriuos naudoja funkcija, bet yra pateikiami paties testo:

- ε testuojama bitų seka, kuri yra sugeneruota RNG arba PRNG; funkcijos kvietimo metu, pateikiama tokiu pavidalu: $\varepsilon = \varepsilon_1, \varepsilon_2, \dots, \varepsilon_n$.
- B m-bitų ilgio šablonas.
- B laisvės laipsnių skaičius.
- M testuojamos eilutės iš sekos ε ilgis bitais. Šiame teste M yra nustatytas 1032.
- N nepriklausomų blokų kiekis. Teste šis kiekis yra 968.

Testo statistika ir reikšmių pasiskirstymas

- $X^2(obs)$ skaičius, nurodantis, kaip tiksliai matricos surastų šablonų skaičius atitinka tikėtinų šablonų skaičių atsitiktinėje sekoje.

Testo statistikos reikšmių pasiskirstymas yra X^2 skirstinys.

Testo aprašas

1. Seka padalinama į N nepriklausomų blokų, kurių kiekvieno ilgis yra M.
2. Apskaičiuojamas atitikimo šablonui B skaičius kiekviename bloke. Kiekvienas atitikimas šablonui yra pažymimas $\mathrm{v_i}$ reikšmes padidinant vienetu, kur $\mathrm{i} = 0, \dots, 5$. Jeigu seka neatitinka šablono padidinama $\mathrm{v_0}$ reikšmė, jei randamas vienas atitikimui – padidinama $\mathrm{v_1}$ reikšmė ir atitinkamai, jei randama 5 ir daugiau atitikimų, padidinama $\mathrm{v_5}$ reikšmė.
3. Apskaičiuojamos λ ir η reikšmės, kurios bus naudojamos apskaičiuoti teorines tikimybes π_i atitinkamai pagal $\mathrm{v_i}$ reikšmes:
4. $\lambda = ((\mathrm{M} - \mathrm{m} + 1))/2^\mathrm{m} \qquad \eta = \lambda/2$
5. Apskaičiuojama $\mathrm{X}^2(\mathrm{obs}) = \sum_{\mathrm{j}=0}^{5} \frac{(\mathrm{v_i} - \mathrm{N}\pi_\mathrm{i})^2}{\mathrm{N}\pi_\mathrm{i}}$, kur $\pi_0 = 0{,}364091$, $\pi_1 = 0{,}185659$, $\pi_2 = 0{,}139381$, $\pi_3 = 0{,}100571$, $\pi_4 = 0{,}070432$ ir $\pi_5 = 0{,}139865$.
6. Apskaičiuojama $\mathrm{P-reikšmė} = \mathrm{igamc}\left(\frac{5}{2}, \frac{\mathrm{X}^2(\mathrm{obs})}{2}\right)$.

Apsisprendimo taisyklė (1% lygmenyje)

Jeigu apskaičiuota P − reikšmė $< \alpha$, kur $\alpha \in [0{,}001, 0{,}01]$, tai darome išvadą, kad seka nėra atsitiktinė, priešingu atveju, kad seka yra atsitiktinė.

Išvados ir gautų rezultatų interpretavimas

Būtina atkreipti dėmesį, kad jeigu 2-bitų ilgio šablonui (B=11) visa seka turi per daug atitikimų, tada:

1) v_5 reikšmė būtų per didelė;
2) testo statistika būtų per didelė;
3) $P - reikšmė$ būtų per maža ($< \alpha$);
4) būtų gauta išvada, kad seka nėra atsitiktinė.

Reikalavimai įvesties duomenims

NIST rekomenduoja, kad $m = 9$ arba $m = 10$, o testuojamos bitų sekos ilgis būtų ne mažesnis nei 10^6 bitų (t.y. $n \geq 10^6$).

9. Maurer „universalus statistinis“ testas

Testo tikslas

Šiame teste dėmesys kreipiamas į bitų kiekį tarp rastų šablonų. Šio bandymo tikslas yra nustatyti, ar seka gali būti pastebimai suspausta neprarasdama turėtos informacijos. Pastebimai suspaudžiama seka yra laiko ne atsitiktine.

Kreipimasis į funkciją

Universal(L, Q, n), kur:

- L bitų kiekis kiekviename bloke.
- Q blokų kiekis sekoje.
- n bitų srauto ilgis

Papildomas įvesties duomuo, kurį naudoja funkcija, bet yra pateikiamas paties testo:

- ε testuojama bitų seka, kuri yra sugeneruota RNG arba PRNG; funkcijos kvietimo metu, pateikiama tokiu pavidalu: $\varepsilon = \varepsilon_1, \varepsilon_2, \dots, \varepsilon_n$.

Testo statistika ir reikšmių pasiskirstymas

f_n $\log_2$ atstumų tarp rastų atitikimų L-bitų šablonui suma, t.y. bitų tarp L-bitų šablonų suma.
Testo statistikos reikšmių pasiskirstymas yra pusiau normalusis skirstinys.

Testo aprašas

1. n-bitų seka (ε) yra padalinama į dvi dalis: pradinį segmentą, turintį Q L-bitų ilgio nepersidengiančių blokų ir testavimo segmentą, turintį K L-bitų ilgio nepersidengiančių blokų. Bitai nepatenkantys į blokus yra atmetami. Pirmieji Q blokai yra naudojami pradėti testavimą, likę K blokai naudojami vykdyti testavimą ($K = \lfloor n/L \rfloor - Q$).
2. Pradiniame segmente kiekvienai galimai L-bito reikšmei yra sudaroma lentelė (t.y. L-bito reikšmė lentelėje naudojama kaip titulinė dalis, indeksas).
3. Tikrinamas kiekvienas testavimo dalies blokas iš K galimų ir nustatomas blokų skaičius, atitikusių tą patį L-bitų bloką. Lentelėje yra pakeičiama naujoji bloko vieta. Paskaičiuotas atstumas yra pridedamas prie $\log_2$ sumos.
4. Apskaičiuojama testo statistika $f_n = \frac{1}{K}\sum_{i=Q+1}^{Q+K} \log_2(i - T_j)$, kur T_j yra atitinkamas L-bitų bloko dešimtainis atitikmuo sudarytoje lentelėje.
5. Apskaičiuojama $P - reikšmė = \mathrm{erfc}\left(\left|\frac{f_n - expectedValue\ (L)}{\sqrt{2}\sigma}\right|\right)$, kur $\sigma = c\sqrt{\frac{variance\ (L)}{K}}$, $c = 0{,}7 - \frac{0{,}8}{L} + \left(4 + \frac{32}{L}\right)\frac{K^{-3/L}}{15}$, o reikšmės $expectedValue(L)$ ir $variance(L)$ imamos iš žemiau pateiktos lentelės.

L	$expectedValue(L)$	$variance(L)$	L	$expectedValue(L)$	$variance(L)$
6	5,2177052	2,954	12	11,168765	3,401
7	6,1962507	3,125	13	12,168070	3,410
8	7,1836656	3,238	14	13,167693	3,416
9	8,1764248	3,311	15	14,167488	3,419
10	9,1723243	3,356	16	15,167379	3,421
11	10,170032	3,384			

9.1 Lentelė. $expectedValue(L)$ ir $variance(L)$ reikšmės

Apsisprendimo taisyklė (1% lygmenyje)

Jeigu apskaičiuota P – reikšmė $< \alpha$, kur $\alpha \in [0{,}001, 0{,}01]$, tai darome išvadą, kad seka nėra atsitiktinė, priešingu atveju, kad seka yra atsitiktinė.

Išvados ir gautų rezultatų interpretavimas

Jeigu f_n stipriai skiriasi nuo expectedValue(L), tada seka yra stipriai suspaudžiama.

Reikalavimai įvesties duomenims

Šis testas reikalauja ilgos duomenų sekos ($n \geq (Q + K)L$), kuri padalinama į dvi L-bitų blokų dalis. Geriausiai L rinktis iš intervalo $6 \leq L \leq 16$, o Q paskaičiuoti iš $10 \cdot 2^L$. Pasirenkamos L, Q ir n reikšmės parodytos žemiau esančioje lentelėje:

n	L	Q
≥ 387840	6	640
≥ 904960	7	1280
≥ 2068480	8	2560
≥ 4654080	9	5120
≥ 10342400	10	10240
≥ 22753280	11	20480
≥ 49643520	12	40960
≥ 107560960	13	81920
≥ 231669760	14	163840
≥ 496435200	15	327680
≥ 1059061760	16	655360

9.2 Lentelė. n, L ir Q reikšmės

10. Linijinio sudėtingumo testas

Testo tikslas

Šiame teste dėmesys kreipiamas į tiesinio grįžtamojo ryšio poslinkio registro (LFSR) ilgį. Šio bandymo tikslas yra nustatyti, ar seka yra pakankamai sudėtinga, kad būtų laikoma atsitiktine. Atsitiktinėmis sekomis laikomos tokios sekos, kurių LFSR yra ilgesnis. Per trumpas LFSR nurodo, kad seka nėra atsitiktinė.

Kreipimasis į funkciją

LinearComplexity(M, n), kur:

M bloko ilgis bitais.

n bitų srauto ilgis

Papildomi įvesties duomenys, kuriuos naudoja funkcija, bet yra pateikiami paties testo:

ε testuojama bitų seka, kuri yra sugeneruota RNG arba PRNG; funkcijos kvietimo metu, pateikiama tokiu pavidalu: $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$.

K laisvės laipsnių skaičius. Šiame rinkinyje jis yra nustatytas $K = 6$.

Testo statistika ir reikšmių pasiskirstymas

$X^2(obs)$ skaičius, nurodantis, kaip tiksliai LFSR ilgiai atitinka tikėtinų LFSR ilgius atsitiktinėje sekoje. Testo statistikos reikšmių pasiskirstymas yra X^2 skirstinys.

Testo aprašas

1. Seka padalinama į N nepriklausomų blokų, kurių kiekvieno ilgis yra M.
2. Naudojantis Berlekamp-Massey algoritmu, nustatomas kiekvieno bloko tiesinis sudėtingumas L_i
3. Remdamiesi atsitiktinumo prielaidą, paskaičiuojame teorinę vidurkio μ reikšmę:
4. $\mu = \frac{M}{2} + \frac{(9+(-1)^{M+1})}{36} - \frac{\left(\frac{M}{3}+2/9\right)}{2^M}$
5. Kiekvienai eilutei sekoje, apskaičiuojame reikšmę $T_i = (-1)^M \cdot (L_i - \mu) + 2/9$.
6. Atvaizduojame T_i reikšmes kintamuosiuose $v_0, \dots, v_6$ pagal tokią schemą:

$T_i \leq -2{,}5$	v_0 padidiname 1
$-2{,}5 < T_i \leq -1{,}5$	v_1 padidiname 1
$-1{,}5 < T_i \leq -0{,}5$	v_2 padidiname 1
$-0{,}5 < T_i \leq 0{,}5$	v_3 padidiname 1
$0{,}5 < T_i \leq 1{,}5$	v_4 padidiname 1
$1{,}5 < T_i \leq 2{,}5$	v_5 padidiname 1
$T_i > 2{,}5$	v_6 padidiname 1

10.1 Lentelė. T_i reikšmių atvaizdavimas kintamuosiuose $v_0, \dots, v_6$

7. Apskaičiuojama $X^2(\text{obs}) = \sum_{i=0}^{K} \frac{(v_i - N\pi_i)^2}{N\pi_i}$, kur $\pi_0 = 0{,}010417$, $\pi_1 = 0{,}03125$, $\pi_2 = 0{,}125$, $\pi_3 = 0{,}5$, $\pi_4 = 0{,}25$, $\pi_5 = 0{,}0{,}0625$ ir $\pi_6 = 0{,}020833$.
8. Apskaičiuojama $P - \text{reikšmė} = \text{igamc}\left(\frac{N}{2}, \frac{X^2(\text{obs})}{2}\right)$.

Apsisprendimo taisyklė (1% lygmenyje)

Jeigu apskaičiuota $P - \text{reikšmė} < \alpha$, kur $\alpha \in [0{,}001, 0{,}01]$, tai darome išvadą, kad seka nėra atsitiktinė, priešingu atveju, kad seka yra atsitiktinė.

Išvados ir gautų rezultatų interpretavimas

Jeigu $P - \text{reikšmė}$ yra labai maža $(< \alpha)$, vadinasi stebėti dažnio T_i kiekiai kintamuosiuose v_I pakito nuo tikėtųsi skaičių. Tikimąsi, kad dažnių T_i pasiskirstymas kintamuosiuose v_I būtų proporcingas apskaičiuotoms π_i reikšmėms.

Reikalavimai įvesties duomenims

Rekomenduojama, kad bitų srauto ilgis būtų $n \geq 10^6$, bloko ilgis M tenkintų lygybę $500 \leq M \leq 5000$. Tada galima tikėtis, kad testas pateiks teisingus rezultatus.

11. Serijos testas

Testo tikslas

Šiame teste dėmesys kreipiamas į visų galimų m-bitų šablonų analizuojamoje sekoje dažnį. Šio bandymo tikslas yra nustatyti, ar atrastų m-bitų šablonų skaičius yra apytiksliai toks pats, kokio tikimąsi atsitiktinėje sekoje. Atsitiktinės sekos turi vienodumo savybę, t.y. kiekvienas m-bitų šablonas turi tokią pačią galimybę atitikti dalį sekos kaip ir bet kuris kitas m-bitų šablonas. Jeigu $m = 1$, tai gaunamas dažnio testas.

Kreipimasis į funkciją

Serial(m, n), kur:

m bitų kiekis kiekviename šablone

n bitų srauto ilgis

Papildomas įvesties duomuo, kurį naudoja funkcija, bet yra pateikiamas paties testo:

ε testuojama bitų seka, kuri yra sugeneruota RNG arba PRNG; funkcijos kvietimo metu, pateikiama tokiu pavidalu: $\varepsilon = \varepsilon_1, \varepsilon_2, \dots, \varepsilon_n$.

Testo statistika ir reikšmių pasiskirstymas

$\nabla\psi_m^2(obs)$ ir $\nabla^2\psi_m^2(obs)$ skaičius, nurodantis, kaip tiksliai m-bitų šablonai atitinka analizuojamą seką palyginus su tikėtinais.

Testo statistikos reikšmių pasiskirstymas yra X^2 skirstinys.

Testo aprašas

1. Suformuojama papildyta seka ε': seka praplečiama prijungiant pirmus $m-1$ bitus prie jos pabaigos.
2. Nustatoma visų galimų persidengiančių m-bitų, $(m-1)$-bitų ir $(m-2)$-bitų ilgio šablonų atitikimo dažniai. m-bitų šablono dažnį pažymime $v_{i_1 \dots i_m}$, atitinkamai $(m-1)$ ir $(m-2)$ bitų ilgio šablonų dažnius $v_{i_1 \dots i_{m-1}}$ ir $v_{i_1 \dots i_{m-2}}$.
3. Apskaičiuojama: $\psi_m^2 = \frac{2^m}{n}\sum_{i_1 \dots i_m}\left(v_{i_1 \dots i_m} - \frac{n}{2^m}\right)^2 = \frac{2^m}{n}\sum_{i_1 \dots i_m} v_{i_1 \dots i_m}^2 - n$

$$\psi_{m-1}^2 = \frac{2^{m-1}}{n}\sum_{i_1 \dots i_{m-1}}\left(v_{i_1 \dots i_{m-1}} - \frac{n}{2^{m-1}}\right)^2 = \frac{2^{m-1}}{n}\sum_{i_1 \dots i_{m-1}} v_{i_1 \dots i_{m-1}}^2 - n$$

$$\psi_{m-2}^2 = \frac{2^{m-2}}{n}\sum_{i_1 \dots i_{m-2}}\left(v_{i_1 \dots i_{m-2}} - \frac{n}{2^{m-2}}\right)^2 = \frac{2^{m-2}}{n}\sum_{i_1 \dots i_{m-2}} v_{i_1 \dots i_{m-2}}^2 - n$$

4. Apskaičiuojama $\nabla\psi_m^2 = \psi_m^2 - \psi_{m-1}^2$ ir $\nabla^2\psi_m^2 = \psi_m^2 - 2\psi_{m-1}^2 + 2\psi_{m-2}^2$.
5. Apskaičiuojama $P - reikšmė1 = igamc(2^{m-2}, \nabla\psi_m^2)$ ir $P - reikšmė2 = igamc(2^{m-3}, \nabla^2\psi_m^2)$.

Apsisprendimo taisyklė (1% lygmenyje)

Jeigu apskaičiuota $P - reikšmė < \alpha$, kur $\alpha \in [0{,}001, 0{,}01]$, tai darome išvadą, kad seka nėra atsitiktinė, priešingu atveju, kad seka yra atsitiktinė.

Išvados ir gautų rezultatų interpretavimas

Jeigu $\nabla\psi_m^2$ arba $\nabla^2\psi_m^2$ reikšmė yra didelis skaičius, vadinasi m-bitų blokams taikoma nevienodumo savybė.

Reikalavimai įvesties duomenims

Rekomenduojama pasirinkti tokius m ir n, kad būtų tenkinama nelygybė $m < \lfloor \log_2 n \rfloor - 2$.

12. Vidutinės entropijos testas

Testo tikslas

Taip pat kaip ir serijos teste dėmesys kreipiamas į visų galimų m-bitų šablonų analizuojamoje sekoje dažnį. Šio bandymo tikslas yra palyginti dviejų gretimų persidengiančių blokų ilgius (m ir $m+1$) su tikėtinais atsitiktinės sekos rezultatais.

Kreipimasis į funkciją

ApproximateEntropy(m, n), kur:

m bitų kiekis kiekviename šablone – šiuo atveju pirmojo bloko ilgis yra m, antrojo $m+1$.

n bitų srauto ilgis

Papildomas įvesties duomuo, kurį naudoja funkcija, bet yra pateikiamas paties testo:

ε testuojama bitų seka, kuri yra sugeneruota RNG arba PRNG; funkcijos kvietimo metu, pateikiama tokiu pavidalu: $\varepsilon = \varepsilon_1, \varepsilon_2, \dots, \varepsilon_n$.

Testo statistika ir reikšmių pasiskirstymas

$X^2(obs)$ skaičius, nurodantis, kaip tiksliai tiriamos sekos $ApEn(m)$ reikšmė atsitiktinės sekos rezultatus.

Testo statistikos reikšmių pasiskirstymas yra X^2 skirstinys.

Testo aprašas

1. Suformuojama papildyta seka prijungiant pirmus $m - 1$ bitus prie jos pabaigos.
2. Suskaičiuojamas n persidengiančių blokų dažnių skaičius (pvz., jeigu blokas, sudarytas nuo ε_j iki ε_{j+m-1}, yra testuojamas j laiko mementu, tai blokas, sudarytas nuo ε_{j+1} iki ε_{j+m}, yra testuojamas $j + 1$ laiko momentu). Visų galimų m-bitų ($(m + 1)$-bitų) reikšmių kiekis žymimas C_i^m, kur i yra m-bitų bloko reikšmė.
3. Kiekvienai i reikšmei apskaičiuojama $C_i^m = \frac{\#i}{n}$.
4. Apskaičiuojama: $\varphi^{(m)} = \sum_{i=0}^{2^m-1} \pi_i \log \pi_i$, kur $\pi_i = C_j^3$ ir $j = \log_2 i$.
5. Pakartojami 1-4 algoritmo žingsniai m pakeičiant $m + 1$.
6. Apskaičiuojama testo statistika $X^2(obs) = 2n[\log 2 - ApEn(m)]$, kur $ApEn(m) = \varphi^{(m)} - \varphi^{(m+1)}$.
7. Apskaičiuojama $P - reikšmė = igamc\left(2^{m-1}, \frac{X^2}{2}\right)$.

Apsisprendimo taisyklė (1% lygmenyje)

Jeigu apskaičiuota $P - reikšmė < \alpha$, kur $\alpha \in [0{,}001, 0{,}01]$, tai darome išvadą, kad seka nėra atsitiktinė, priešingu atveju, kad seka yra atsitiktinė.

Išvados ir gautų rezultatų interpretavimas

Verta pažymėti, kad mažos funkcijos $ApEn(m)$ reikšmės nurodo stiprų taisyklingumą, o didelės reikšmės – duomenų kaitą ir asimetriškumą.

Reikalavimai įvesties duomenims

Rekomenduojama pasirinkti tokius m ir n, kad būtų tenkinama nelygybė $m < \lfloor \log_2 n \rfloor - 5$.

13. Sukauptų sumų (Cusums) testas

Testo tikslas

Šiame teste dėmesys kreipiamas į maksimalų kelią (nuo nulio), kuris įveikiamas atsitiktiniu keliu, apibrėžtu sukauptomis pritaikytų (-1, +1) skaitmenų sekoje sumomis. Šio bandymo tikslas yra nustatyti ar sukauptos sumos rezultatas yra per mažas/per didelis lyginant su atsitiktinės sekos rezultatais. Tad sukauptos sumos rezultatas gali būti vadinamas atsitiktiniu keliu. Atsitiktinėms sekoms nueitas atsitiktinis kelias turi būti artimas nuliui.

Kreipimasis į funkciją

CumulativeSums($mode, n$), kur:

n bitų srauto ilgis

Papildomi įvesties duomenys, kuriuos naudoja funkcija, bet yra pateikiami paties testo:

ε testuojama bitų seka, kuri yra sugeneruota RNG arba PRNG; funkcijos kvietimo metu, pateikiama tokiu pavidalu: $\varepsilon = \varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$.

$mode$ rodiklis nurodantis, kaip atlikti testą – į priekį per įvestą bitų srautą ($mode = 0$) ar dirbant su seka nuo galo ($mode = 1$).

Testo statistika ir reikšmių pasiskirstymas

z didžiausias kelias sekoje.

Testo statistikos reikšmių pasiskirstymas normalusis skirstinys.

Testo aprašas

1. Suformuojama normalizuota seka: testui duotos sekos (ε) nuliai ir vienetai yra konvertuojami į atitinkamai į X_i reikšmes, pagal $X_i = 2\varepsilon_i - 1$ priskiriant reikšmes -1 arba $+1$.
2. Apskaičiuojamos dalinės sumos S_i, kurių kiekvieną pradeda X_1, jeigu $mode = 0$, ir X_n, jeigu $mode = 1$: $S_k = S_{k-1} + X_k$, kai mode yra 0, ir $S_k = S_{k-1} + X_{n-k+1}$, kai mode lygus 1.

3. Apskaičiuojama testo statistikos reikšmė $z = \max_{i \le k \le n} |S_k|$, kur $z = \max_{i \le k \le n} |S_k|$ yra didžiausia dalinės sumos S_k reikšmė.
4. Apskaičiuojama:

$$P - reikšmė = 1 - \sum_{k=\frac{\left(\frac{-n}{x}-1\right)}{4}}^{\frac{\left(\frac{n}{x}-1\right)}{4}} \left[\Phi\left(\frac{(4k+1)z}{\sqrt{n}}\right) - \Phi\left(\frac{(4k-1)z}{\sqrt{n}}\right)\right] + \sum_{k=\frac{\left(\frac{-n}{x}-3\right)}{4}}^{\frac{\left(\frac{n}{x}-1\right)}{4}} \left[\Phi\left(\frac{(4k+3)z}{\sqrt{n}}\right) - \Phi\left(\frac{(4k+1)z}{\sqrt{n}}\right)\right],$$

kur Φ yra standartinė normalioji sudėtinių tikimybių pasiskirstymo funkcija (*angl. Standard Normal Cumulative Probability Distribution Function*).

Apsisprendimo taisyklė (1% lygmenyje)

Jeigu apskaičiuota $P - reikšmė < \alpha$, kur $\alpha \in [0{,}001, 0{,}01]$, tai darome išvadą, kad seka nėra atsitiktinė, priešingu atveju, kad seka yra atsitiktinė.

Išvados ir gautų rezultatų interpretavimas

Esant $mode = 0$, gautos didelės testo statistikos reikšmės rodo, kad sekoje yra arba per daug nulių, arba per daug vienetų sekos pradžioje. Esant $mode = 1$, gautos didelės testo statistikos reikšmės rodo, kad sekoje yra arba per daug nulių, arba per daug vienetų sekos pabaigoje. Mažos statistikos reikšmės nurodo, kad vienetai ir nuliai yra pasklidę per vienodai.

Reikalavimai įvesties duomenims

Rekomenduojama, kad kiekviena testuojama seka būtų bent 100 bitų ilgio (t.y. $n \ge 100$).

14. Atsitiktinių ekskursijų testas

Testo tikslas

Šiame teste dėmesys kreipiamas ciklų skaičius, gaunamas atliekant tiksliai K aplankymų sudėtinių sumų atsitiktiniame kelyje. Atsitiktinio kelio ciklu, vadinama žingsnių seka, susidedanti iš atsitiktinio pradžios taško ir sekos pabaigoje grįžimo į jį. Šio bandymo tikslas yra nustatyti ar atitinkamoje būsenoje atliktų ciklų skaičius yra panašus į atsitiktinės sekos rezultatus. Šis testas iš tikrųjų yra atskirų 8 testų (ir jų išvadų) seka. Vienas testas ir jo išvada kiekvienai būsenai: -4, -3, -2, -1 ir +1, +2, +3, +4

Kreipimasis į funkciją

RandomExcursions(n), kur:

n bitų srauto ilgis

Papildomas įvesties duomuo, kurį naudoja funkcija, bet yra pateikiamas paties testo:

ε testuojama bitų seka, kuri yra sugeneruota RNG arba PRNG; funkcijos kvietimo metu, pateikiama tokiu pavidalu: $\varepsilon = \varepsilon_1, \varepsilon_2, \dots, \varepsilon_n$.

Testo statistika ir reikšmių pasiskirstymas

$X^2(obs)$ skaičius, nurodantis, kaip tiksliai konkrečioje būsenoje ciklų skaičius atitinka atsitiktinės sekos rezultatus.

Testo statistikos reikšmių pasiskirstymas yra X^2 skirstinys.

Testo aprašas

1. Suformuojama normalizuota $(-1, +1)$ seka X: testui duotos sekos (ε) nuliai ir vienetai yra konvertuojami į atitinkamai į X_i reikšmes, pagal $X_i = 2\varepsilon_i - 1$ priskiriant reikšmes -1 arba $+1$.
2. Apskaičiuojamos dalinės sumos $S_i = S_{i-1} + X_i$, kurių kiekvieną pradedama X_1. Suformuojamas dalinių sumų rinkinys $S = \{S_i\}$.
3. Sekos pradžioje ir pabaigoje pridedant nulius, suformuojama nauja seka $S' = 0, S_1, S_2, \dots, S_n, 0$.
4. Apibrėžiamas J – nulių skaičius sekoje S'. Šis parametras taip pat parodo, kiek sekoje yra ciklų – nulių pradedamų ir pabaigiamų eilučių.
5. Kiekvienam ciklui ir nenulinei būsenos reikšmei x ($-4 \le x \le -1$ ir $1 \le x \le 4$), apskaičiuojama kiekvieno x kiekviename iš ciklų dažnis.

6. Kiekvienai iš aštuonių x būsenų, apskaičiuojamas visų ciklų, kuriuose būsena x atsiranda lygiai k kartų, kiekis $\nu_k(x)$, kai $k = 0,1, \dots ,5$ (kai $k = 5$, visi dažniai saugomi $\nu_5(x)$). Pažymima, kad $\sum_{k=0}^{5} \nu_k(x) = J$.
7. Apskaičiuojama kiekvienos iš aštuonių būsenų x testo statistikos reikšmė $X^2(obs) = \sum_{k=0}^{5} \frac{(\nu_k(x) - J\pi_k(x))^2}{J\pi_k(x)}$, kur $J\pi_k(x)$ yra tikimybė, kad būsena x atsitiktiniame skirstinyje pasirodys k kartų.
8. Apskaičiuojama kiekvienos iš aštuonių būsenų x $P - \text{reikšmė} = \text{igamc}\left(\frac{5}{2}, \frac{X^2(obs)}{2}\right)$. Gaunamos 8 $P - \text{reikšmės}$.

Apsisprendimo taisyklė (1% lygmenyje)

Jeigu apskaičiuota $P - \text{reikšmė} < \alpha$, kur $\alpha \in [0{,}001, 0{,}01]$, tai darome išvadą, kad seka nėra atsitiktinė, priešingu atveju, kad seka yra atsitiktinė.

Išvados ir gautų rezultatų interpretavimas

Jeigu $X^2(obs)$ reikšmės yra per didelės, tai rodo sekos nuokrypį nuo teorinio pasiskirstymo visuose cikluose konkrečioje būsenoje x.

Reikalavimai įvesties duomenims

Rekomenduojama, kad kiekviena testuojama seka būtų bent 1000000 bitų ilgio (t.y. $n \geq 10^6$).

15. Atsitiktinių ekskursijų variantų testas

Testo tikslas

Šiame teste dėmesys kreipiamas į tai, kiek kartų atsitiktinio sudėtinių sumų kelio metu yra aplankoma tam tikra būsena. Šio bandymo tikslas yra nustatyti nuokrypį nuo atsitiktinės sekos rezultatų. Šis testas iš tikrųjų yra atskirų 18 testų (ir jų išvadų) seka. Vienas testas ir jo išvada kiekvienai būsenai: -9, -8, ..., -1 ir +1, +2, ..., +9

Kreipimasis į funkciją

RandomExcursionsVariant(n), kur:

n bitų srauto ilgis

Papildomas įvesties duomuo, kurį naudoja funkcija, bet yra pateikiamas paties testo:

ε testuojama bitų seka, kuri yra sugeneruota RNG arba PRNG; funkcijos kvietimo metu, pateikiama tokiu pavidalu: $\varepsilon = \varepsilon_1, \varepsilon_2, \dots, \varepsilon_n$.

Testo statistika ir reikšmių pasiskirstymas

ξ duotai būsenai x, skaičius, nurodantis, kiek kartų ta būsena buvo aplankyta viso atsitiktinio kelio metu, kaip nurodyta algoritmo 4-ajame žingsnyje.

Testo statistikos reikšmių pasiskirstymas yra pusiau normalusis skirstinys (dideliems n).

Testo aprašas

1. Suformuojama normalizuota $(-1, +1)$ seka X: testui duotos sekos (ε) nuliai ir vienetai yra konvertuojami į atitinkamai į X_i reikšmes, pagal $X_i = 2\varepsilon_i - 1$ priskiriant reikšmes -1 arba $+1$.
2. Apskaičiuojamos dalinės sumos $S_i = S_{i-1} + X_i$, kurių kiekvieną pradedama X_1. Suformuojamas dalinių sumų rinkinys $S = \{S_i\}$.
3. Sekos pradžioje ir pabaigoje pridedant nulius, suformuojama nauja seka $S' = 0, S_1, S_2, \dots, S_n, 0$.
4. Kiekvienam iš 18 nenulinių būsenų x, apskaičiuojama testo statistika $\xi(x)$ - būsenos x patekimo į J ciklus kiekis. J – nulių skaičius sekoje S', bei ciklų skaičius sekoje.
5. Kiekvienai $\xi(x)$ apskaičiuojama $P - \text{reikšmė} = \text{erfc}\left(\frac{|\xi(x) - J|}{\sqrt{2J(4|x| - 2)}}\right)$. Gaunama 18 $P - \text{reikšmių}$.

Apsisprendimo taisyklė (1% lygmenyje)

Jeigu apskaičiuota P − reikšmė $< \alpha$, kur $\alpha \in [0{,}001, 0{,}01]$, tai darome išvadą, kad seka nėra atsitiktinė, priešingu atveju, kad seka yra atsitiktinė.

Išvados ir gautų rezultatų interpretavimas

Jeigu $\xi(x)$ reikšmės yra per didelės, tai rodo sekos nuokrypį nuo teorinio pasiskirstymo visuose cikluose konkrečioje būsenoje x.

Reikalavimai įvesties duomenims

Rekomenduojama, kad kiekviena testuojama seka būtų bent 1000000 bitų ilgio (t.y. $n \geq 10^6$).

Priedas Nr. 5

BLAKE su pastovia druska	BLAKE su kintančia druska	Groestl	JH	Keccak	Skein
0,099703	0,177477	0,15358	0,529349	0,524617	0,531808
0,100458	0,176647	0,156785	0,524982	0,515671	0,534885
0,111568	0,18185	0,159191	0,52543	0,521271	0,535378
0,123068	0,184814	0,158509	0,509142	0,517818	0,53371
0,11978	0,185896	0,157825	0,502586	0,523403	0,531873
0,122627	0,185401	0,161867	0,491051	0,527388	0,535155
0,126761	0,189805	0,162855	0,480729	0,522696	0,53001
0,133211	0,191403	0,161381	0,474758	0,532109	0,531296
0,134171	0,190643	0,16181	0,475988	0,527497	0,524708
0,13482	0,190819	0,163577	0,461282	0,534807	0,52348
0,142673	0,183274	0,155435	0,458402	0,533386	0,514914
0,152582	0,177093	0,157423	0,449707	0,53271	0,521888
0,15348	0,174581	0,153297	0,441101	0,542054	0,523447
0,150815	0,174278	0,14971	0,446722	0,540353	0,513254
0,148406	0,174578	0,148645	0,444104	0,555705	0,501926
0,15174	0,169199	0,149349	0,436937	0,54494	0,490917
0,149159	0,172319	0,14394	0,439951	0,546851	0,493997
0,147651	0,177979	0,13697	0,440151	0,549201	0,496518
0,144143	0,177093	0,132785	0,436252	0,548065	0,503392
0,145436	0,174581	0,132649	0,444843	0,553961	0,499448
0,142622	0,174278	0,127497	0,438017	0,550446	0,493705
0,143878	0,174578	0,133569	0,44279	0,554087	0,484775
0,146084	0,169199	0,131981	0,448332	0,550145	0,49473
0,142312	0,172319	0,132628	0,453921	0,546865	0,49922
0,142389	0,177979	0,126769	0,455355	0,54155	0,50273
0,14492	0,177093	0,126731	0,455871	0,530953	0,499442
0,150313	0,174581	0,130178	0,454729	0,525466	0,498503
0,148372	0,174278	0,131592	0,467532	0,530863	0,4998
0,148754	0,174578	0,131355	0,47941	0,524764	0,505605
0,14902	0,169199	0,137133	0,486705	0,518808	0,507773