Pregled SHA-3 standarda

Šaravanja, Lucija

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

SVEUČILIŠNI STUDIJ

PREGLED SHA-3 STANDARDA

Završni rad

Lucija Šaravanja

Osijek, 2019.

Sadržaj

1. UVOD

Kriptografija je znanost koja se bavi metodama logičke promjene podataka te je njen cilj učiniti povjerljive podatke nemogućim za dešifriranje onima koji nad tim podacima nemaju ovlašten pristup. Smatra se da prvi oblici kriptografije potječu još od prije 3000 godina. Međutim, danas kada je korištenje računala i interneta u svrhu razmjene podataka postalo normalna stvar bez koje se teško zamišlja život, značajna se uloga u razmjeni podataka pripisuje kriptografiji. Zbog javne dostupnosti podataka na internetu, kojim se koristi velik broj ljudi, dolazi do potrebe šifriranja povjerljivih podataka koji zahtijevaju određenu sigurnost. Stalni razvoj i napredak računalnih tehnologija vremenom dovode do sve veće nesigurnosti sigurnosnih sustava i algoritama. *Secure Hash* algoritmi su jedne od najvažniji kriptografskih funkcija koje su osmišljene za očuvanje sigurnosti podataka. *Hash* funkcije su funkcije koje za ulaznu poruku proizvoljne duljine vraćaju izlaznu, šifriranu poruku fiksne duljine. Dakle, ove funkcije od neke informacije proizvoljne duljine prave „sažetak“ točno određene duljine. Iz tog je sažetka nemoguće doći do prvobitne, originalne poruke budući da je jedno od svojstava ovih funkcija izostanak inverznih funkcija. U ovome je radu pobliže objašnjen jedan od standarda već navedenih *Secure Hash* algoritama, odnosno najnoviji član ovih algoritama, SHA-3 standard. Također su prikazani prethodnici SHA-3 standarda, povijest njegova nastanka, dizajn, te njegove prednosti i načini upotrebe.

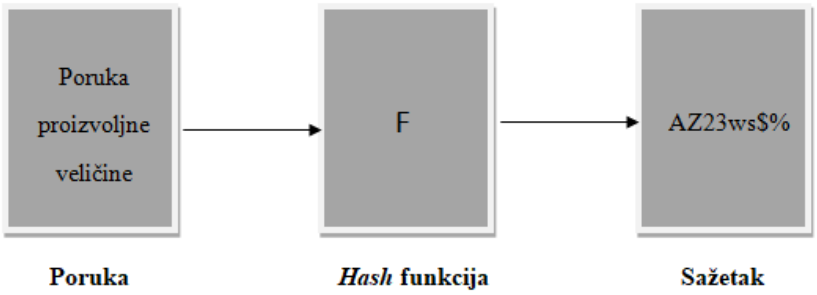
Nakon uvoda, u drugom su poglavlju navedena i objašnjena svojstva *Hash* funkcija općenito. Treće poglavlje govori o klasifikaciji *Secure Hash* algoritama, odnosno o prethodnicima SHA-3 standarda te o samoj povijesti njegova nastanka. U četvrtom je poglavlju dan opis SHA-3 standarda, njegov dizajn te način njegova kriptiranja. Zadnje, peto poglavlje govori o upotrebi SHA-3 standarda te o njegovim prednostima.

1.1. Zadatak završnog rada

U ovom završnom radu potrebno je dati pregled dosadašnjih *hash* standarda i potrebu novih verzija. Također je potrebno objasniti način kriptiranja SHA-3 algoritma, njegovu primjenu i prednosti.

2. HASH FUNKCIJE

Osim što čine sastavni dio mnogih sigurnosnih aplikacija poput kodova za provjeru autentičnosti poruka, integriteta podataka, pohrane lozinke, digitalnog potpisa i generiranja slučajnih brojeva, *hash* funkcije također se smatraju i jednom od najvažnijih komponenata skoro svih kriptografskih protokola. Ubrajaju se među najkorisnije kriptografske funkcije zbog razine sigurnosti koju nude. Omogućuju očuvanje integriteta poruke, kako bi primatelj dobio jednaku poruku kakvu je pošiljatelj poslao, odnosno pružaju sigurnost od izmijene poruke od strane napadača. Uz to se koriste i kao temeljni element sigurnosnih transakcija i kripto valuta. Ove su funkcije jednosmjerne te za dani ulazni podatak proizvoljne duljine vraćaju izlaz fiksne duljine. Omogućuju preslikavanje dugih ulaznih nizova podataka na kraće izlaze koji se upravo zbog toga često nazivaju sažetkom. Slika 2.1. prikazuje ulaz proizvoljne duljine (nazvan poruka ili običan tekst) koji koristi jednosmjernu *hash* funkciju, te izlaz fiksne duljine (koji se naziva sažetak).



Sl 2.1. *Hash funkcija* [1]

Kao što je ranije navedeno, *hash* funkcije koriste se u raznim sigurnosnim aplikacijama poput digitalnog potpisa, provjere autentičnosti poruka te očuvanja integriteta podataka. Dalje u tekstu ukratko su objašnjene ove tri sigurnosne aplikacije.

Povijesno gledano, digitalni je potpis bio prva upotreba sigurnih kriptografskih *hash* funkcija. Digitalni je potpis matematička shema koja se koristi u slučajevima kada je od presudne važnosti otkrivanje izmijenjenih podataka i u bilo kojoj financijskog transakciji. Izvodi se pomoću javnog i privatnog ključa koje digitalni potpis koristi. Nakon potpisa, napadaču je teško kopirati podatak koji je druga osoba prethodno potpisala. [1]

Očuvanje integriteta poruke glavni je cilj *hash* funkcija budući da omogućuje otkrivanje promjene podataka. Očuvanje integriteta poruke radi na sljedeći način: pošiljatelj zajedno sa ulaznim podatkom (porukom) primatelju šalje i sažetak (izlazni podatak) koji je sam prethodno izračunao. S druge strane, primatelj će samostalno iz primljene poruke (ulaznog podatka) izračunati sažetak te će ga usporediti s onim sažetkom kojeg je prethodno primio od strane pošiljatelja. Ukoliko su sažetci jednaki, primljena je poruka jednaka kao i prvobitno poslana. Ovo vrijedi pod pretpostavkom da je šansa da su i poruka i sažetak izmijenjeni zanemariva- sažetak izmijenjene poruke također je izmijenjen! [2]

Provjera identičnosti poslane i primljene poruke može se učiniti koristeći kodom za provjeru autentičnosti poruke (engl. *Message Authentication Code- MAC*). *MAC* koristi dva podatka- poruku i tajni ključ koji smiju znati samo pošiljatelj i primatelj. To omogućuje primatelju poruke potvrđivanje integriteta poruke. Pošiljatelj i primatelj moraju imati identičan ključ, pa obje strane koriste jednake funkcije i algoritme za stvaranje ključa kako bi se eliminirala svaka šansa za manipulacijom nad podatcima. Ukoliko se ključevi podudaraju šansa za izmjenom podataka trebala bi biti vrlo mala. Kako bi se osigurala učinkovitost računanja *MAC-a* potrebno je koristiti *hash* funkciju velike brzine. [2]

Matematička definicija *hash* funkcije (H) definirana je kao:

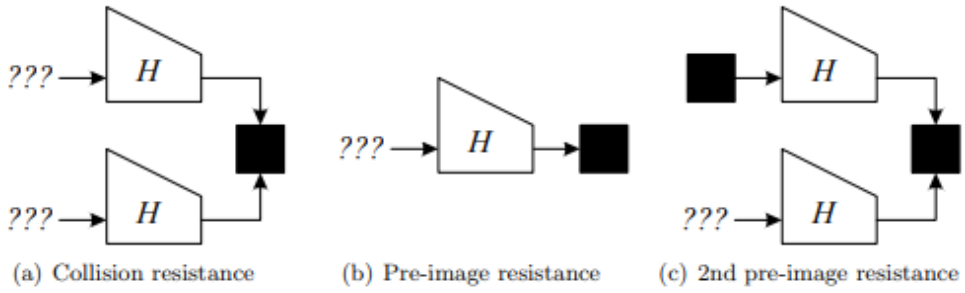
$$H: \{0,1\}^* \rightarrow \{0,1\}n.$$

gdje se $\{0,1\}^*$ odnosi na skup binarnih elemenata proizvoljne duljine uključujući prazan skup, dok se $\{0,1\}n$ odnosi na skup binarnih elemenata duljine n. Dakle, skup binarnih elemenata proizvoljne duljine preslikava se na skup binarnih elemenata fiksne duljine koristeći *Hash* funkciju.[1]

Postoji nekoliko različitih klasa *hash* funkcija, a neke od najčešće korištenih su MD5 (enlg. *Message Digest 5)*, RIPEMD (engl. *RACE Integrity Primitives Evaluation Message Digest*), te obitelj SHA (enlg. *Secure Hash Algorithm)* algoritama. Iako su sve ove funkcije na prvi pogled vrlo slične, postoje razlike u načinu na koji svaki algoritam kreira izlaz na osnovu danog ulaza. Također se razlikuju i po duljini sažetka odnosno izlazne poruke. Unatoč razlikama, postoje svojstva koja bi svaka od ovih funkcije trebala ispunjavati.

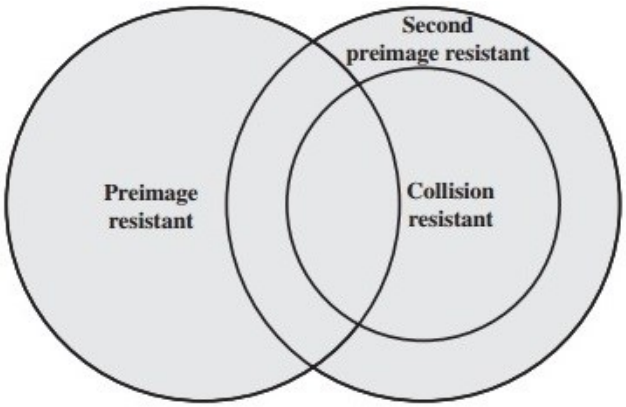
2.1. Svojstva *hash* funkcija

Unatoč tome što se *hash* funkcije mogu međusobno razlikovati po nekim svojstvima, postoji ono što ih sve povezuje, a to su osnovna svojstva bez kojih se niti jedna od ovih funkcija ne može smatrati sigurnom. Osnovna svojstva koja svaka *hash* funkcija treba ispunjavati su: otpornost na sudare, *pre-image* otpornost i *second pre-image* otpornost. Na slici 2.2. vidljiva su sva tri osnovna svojstva *hash* funkcija. Ova se svojstva smatraju univerzalnima te onima koja su za siguran rad ovih funkcija neophodna. Međutim, postoji mogućnost potrebe za dodatnim sigurnosnim svojstvima zahtijevanim na bazi aplikacije koja za svoj rad koristi ove funkcije.



Sl.2.2. *Grafički prikaz osnovnih svojstava hash funkcija: a) otpornost na sudare, b) pre-image otpornost i c) second pre-image otpornost* [3]

Što se tiče međusobne povezanosti ovih svojstava, funkcija koja je otporna na sudare uvijek zadovoljava i svojstvo *second pre-image* otpornosti, ali obratno ne mora uvijek biti istina. No funkcija koja je otporna na sudare ne mora ispunjavati *pre-image* svojstvo otpornosti i suprotno. Na slici 2.3. prikazan je grafikon međusobnih odnosa svojstava *hash* funkcija. [4]



Sl. 2.3. *Prikaz odnosa između svojstava hash funkcija* [4]

Kada se govori o svojstvima *hash* funkcija često se koristi pojam napada te probijanja funkcije. Zbog toga se prije opisa navedenih svojstava treba definirati na što se odnose ta dva pojma. Općenito, kada se kaže da je napad uspješno probio *hash* funkciju, to ne znači nužno da se funkcija u praksi smatra pokvarenom. Ako napadač uspije dokazati da se *hash* funkcija može probiti radom koji je manji od složenosti rada same funkcije, funkcija se smatra nesigurnom, čak i ako napadač u praksi ne može izvesti ono što je dokazano u teoriji. [5]

2.1.1 Otpornost na sudare

Prvo svojstvo koje bi sigurne *hash* funkcije trebale zadovoljavati je otpornost na sudare. Otpornost na sudare prvi je definirao Ivan B. Damgåard, a nekad se koristi i naziv otpor na snažne sudare (engl. *Strong collision resistance).* S obzirom na to da postoje različiti termini vezani uz ovo svojstvo, mora se poznavati razlika između sudara s više blokova i višestrukog sudaranja. Neki autori koriste izraz- sudar s više blokova (enlg. *Multi-block collision)* pri čemu se taj pojam odnosi na dvije poruke koje se sudaraju, a obje se sastoje od najmanje 2 bloka. Ovaj se izraz ne treba poistovjećivati s tzv. višestrukim sudaranjem (engl. *multi-collision)* koji se odnosi na sudar više poruka bez obzira na njihovu veličinu. [3]

Kada se kaže da je *hash* funkcija otporna na sudare, misli se na to da vjerojatnost pronalaska dva različita ulaza koji daju jednak izlaz mora biti vrlo mala, odnosno treba biti nemoguće pronaći takve ulaze. S obzirom na to ulazni podatci mogu biti proizvoljne duljine postoji beskonačno mnogo mogućih ulaza koji se mogu unijeti u *hash* funkciju. Međutim, izlazi su ove funkcije fiksne duljine, što dovodi do zaključka da postoji ograničen broj izlaza koje ova funkcija može proizvesti. Dakle, postoji sigurnost da će funkcija za više različitih ulaza proizvesti jednak izlaz. Zbog toga za funkciju H vrijedi da je otporna na sudare ukoliko je računski nemoguće pronaći

bilo koje dvije poruke M i $M^{'}$ takve da su $H(M) = H(M^{'})$ dok je $M \neq M^{'}$; gdje su M i M' ulazni podatci, dok se $H(M)$ i $H(M')$ odnosi na izlazne podatke *hash* funkcije. Napadačev je cilj pronaći dva ulaza za koje funkcija proizvodi jednak izlaz. Za sigurnu *hash* funkciju, napadačev najbolji pronalazak ne smije biti bolji od složenosti rada same funkcije $2^{\frac{n}{2}}$ koja izvodi n-bitne hash vrijednosti. [1], [5]

2.1.2 *Pre-image* otpornost

Kako bi mogle ispunjavati potrebna svojstva, *hash* funkcije ne smiju biti reverzibilne, pa se često nazivaju i jednosmjernim funkcijama. Kada se za funkciju kaže da ne smije biti reverzibilna misli se na to da njezin izlaz ne smije otkrivati nikakve podatke o ulazu. Nakon što je poruka kriptirana, izlaz *hash* funkcije ne smije biti reverzibilan kako iz njega ne bi bilo moguće doći do izvorne poruke. To se svojstvo naziva *pre-image* otpornost, a često se pojednostavljeno koristi i naziv- jednosmjernost funkcije.

Dakle, za danu *hash* funkciju $H(M)$, čiji je izlaz Y poznat, vrijedi da ispunjava *pre-image* otpornost ako je računski neizvedivo pronaći ili generirati bilo koju poruku M takvu da vrijedi $H(M) = Y$. [6]

Kako bi *hash* funkcija imala svojstvo *pre-image* otpornosti mora vrijediti da najbolji napad protiv funkcije, tzv. grubi napad (engl. *brute force attack)*, nije bolji od složenosti rada same funkcije (2^n za *hash* funkciju s veličinom izlaza n). [1]

2.1.3 *Second pre-image* otpornost

Zadnje svojstvo, od tri osnovna, naziva se *second pre-image* otpornost. Za *hash* funkciju vrijedi da zadovoljava ovo svojstvo ukoliko je za danu funkciju $H(M)$, čiji je ulazni podatak M također poznat, nemoguće pronaći bilo koju poruku M' takvu da vrijedi: $H(M) = H(M')$. Kod ovog se svojstva pretpostavlja da su napadaču poznati i prva ulazna poruka M i izlazni podatak $H(M)$. Ukoliko nije tako, napadač sam može izračunati ulazni podatak. Ponekad se postojanje poznatog ulaznog podatka M zanemari iako se treba paziti da traženi podatak M' ne bude u potpunosti jednak prvoj ulaznoj poruci. Međutim, nasumično birajući poruke, uz pretpostavku da je domena *hash* funkcije puno veća od kodomene, vjerojatnost se pronalaska druge poruke (M') koja je jednaka prvoj (M) svodi na zanemarivu, pa se ova mogućnost zapravo odbacuje. [1]

3. SECURE HASH ALGORITMI

Secure Hash algoritmi obitelj su kriptografskih *hash* funkcija objavljenih od strane *NIST*-a (enlg. *National Institute of Standards and Technology*). Ovi su algoritmi dizajnirani za pružanje mogućnosti preslikavanja podataka proizvoljnih duljina u sažetke fiksne duljine i za ispunjavanje određenih sigurnosnih svojstava. Obitelj ovih algoritama sastoji se od sljedećih: SHA-0, SHA-1, SHA-2 i SHA-3. Svaka je od ovih funkcija uspješno dizajnirana sa sve jačom sigurnosnom razinom za obrane od napada. *Secure hash* algoritmi međusobno se razlikuju prema svojoj konstrukciji kao i prema duljini bita sažetaka. [1]

SHA-0 zapravo je naziv koji se trenutno koristi za originalnu verziju 160-bitne *hash* funkcije objavljenje 1993. godine pod nazivom SHA. Ovu je verziju *NSA* (engl. *National Security Agency*) povukla vrlo brzo nakon objave jer je otkrivena značajna mana. Nakon nekih promjena, objavljena je nova verzija ovog algrotima, danas poznata kao SHA-1. [7]

1995. godine *NIST* je objavio novu verziju svoga prethodnika, SHA-1 algoritam. Ova se verzija od prethode razlikuje samo po jednostrukoj rotaciji u rasporedu poruke svoje kompresijske funkcije. SHA-1 koristi Merkle- Damgåard konstrukciju, kao i MD5, i generira 160-bitni sažetak za ulaznu poruku proizvoljne veličine. Od njegovog objavljivanja, ovaj su algoritam usvojili mnogi vladini i industrijski sigurnosni standardi, posebice standardi digitalnih potpisa za koje je potrebna *hash* funkcija otporna na sudare. Osim uporabe u digitalnim potpisima, SHA-1 je također implementiran kao važna komponenta u različitim kriptografskim protokolima kao što su provjera autentičnosti korisnika ili generiranje pseudoslučajnih brojeva. Međutim, 2005. godine X. Wang, Y. L. Yin i H. Yu objavili su uspješan napad i na ovu verziju. Napad je pronađen u cijeloj verziji ovog algoritma te zahtjeva rad manji od 2^{69} (dok je brutalni napad zahtijevao rad od 2^{80}). Stoga je *NIST* objavio eliminaciju SHA-1 korak po korak. Nakon 2010. godine standard za većinu kriptografskih korištenja nije odobren. [8]

2002. godine *NIST* dodaje nove algoritme obitelj SHA. SHA-2 kao i njegov prethodnik koristi Merkle- Damgåard konstrukciju koja je međutim kompleksnija u ovom slučaju budući da je nelinearna funkcija dodana kompresijskoj. SHA-2 obitelj sastoji se od 6 *hash* funkcija: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256, čiji sufiksi kao i kod SHA-3 funkcije označavaju fiksnu duljinu sažetka. Bitcoin, kao najpopularnija kripto valuta, koristi jednu od ovih funkcija- SHA-256 koja mu pruža sigurnost nad transakcijama. Obitelj SHA-2 također se suočila s napadima koji su djelom proizašli iz natjecanja za SHA-3 algoritam. Pronađeni su samo napadi sudara praktične složenosti i niti jedan još ne koristi cijeli set rundi

kako je predviđeno za SHA-2. Neki od napada, zajedno s nazivom i kompleksnošću navedeni su u tablici 3.1. [9]

Tab. 3.1. *Napadi na SHA-2 algoritam, njihovi nazivi i kompleksnost rada* [9]

ATTACKS MADE ON SHA-2 VARIANTS

Paper	Year	Attack Method	Variant of SHA-2	Collision
New Collision attacks Against Up To 24-step SHA-2	2008	Deterministic Collision	SHA-256 SHA-512	In 24/64 rounds with $2^{28.5}$ complexity In 24/80 rounds with $2^{32.5}$ complexity
Preimages for step-reduced SHA-2	2009	Preimage , Meet-in-the-middle	SHA-256	In 42/64 rounds with $2^{251.7}$ complexity In 43/64 rounds with $2^{254.9}$ complexity
			SHA-512	In 42/80 rounds with $2^{502.3}$ complexity In 46/80 rounds with $2^{511.5}$ complexity
Advanced meet-in-the-middle preimage attacks	2010	Preimage , Meet-in-the-middle	SHA-256 SHA-512	In 42/64 rounds with $2^{248.4}$ complexity In 42/80 rounds with $2^{494.6}$ complexity
Higher-Order Differential Attack on Reduced SHA-256	2011	Pseudo Collision , Differential	SHA-256	In 46/64 rounds with 2^{178} complexity In 46/64 rounds with 2^{46} complexity
Bicliques for Pre-images: Attacks on Skein-512 and the SHA-2 family	2011	Preimage , Biclique	SHA-256 SHA-512 SHA-256 SHA-512	In 45/64 rounds with $2^{555.5}$ complexity In 50/80 rounds with $2^{511.5}$ complexity In 52/64 rounds with 2^{555} complexity In 57/80 rounds with 2^{511} complexity

3.2. Povijest nastanka SHA-3 standarda

Iako novi standard nije trebao zamijeniti SHA-2, 2007. godine *NIST* je službeno najavio natječaj za SHA-3 standard otvorenim pozivom za podnošenje kandidatskih *hash* funkcija. Nakon određenog vremena, primljeno je 64 prijave od kriptografa širom svijeta. Prvi krug natjecanja prošlo je 51 kandidata, među kojima je bio i KECCAK. 2009. godine drugi je krug obilježilo 14 polu finalista, dok je u zadnji krug, u prosincu 2010. godine, prošlo njih 5. Finalisti ovog natjecanja bili su: BLAKE, Skein, Grøstl, JH i KECCAK. Između četiri ostala finalista, *NIST* je odabrao KECCAK za pobjednika natjecanja zbog njegovog elegantnog dizajna, jake sigurnosti, dobrih općih performansi, izvrsne učinkovitosti u implementaciji hardvera i zbog fleksibilnosti. Tablica 3.2. prikazuje tijek natjecanja za SHA-3 s pripadajućim datumima svakog kruga ovoga natjecanja. [10]

Tab. 3.2. *Tijek natjecanja za SHA-3 algoritam* [11]

Date	Event	Candidates Left
11/2/2007	Call for Proposals published, competition began	
10/31/2008	SHA3 submission deadline	64
12/10/2008	First-round candidates announced	51
2/25/2009	First SHA3 workshop in Leuven, Belgium	51
7/24/2009	Second-round candidates announced	14
8/23/2010	Second SHA3 workshop in Santa Barbara, CA	14
12/9/2010	SHA3 finalists announced	5
3/22/2012	Third SHA3 workshop in Washington, DC	5
10/2/2012	Keccak announced as the SHA3 winner	1

KECCAK koristi novi način povezivanja- spužvastu konstrukciju, koja se temelji na fiksnoj permutaciji, te omogućuje izuzetnu fleksibilnost radi mogućnosti proširenja izlaznih podataka. KECCAK dobro nadopunjuje postojeću SHA-2 obitelj *hash* algoritama, što je bilo vrlo važno budući da je *NIST* ostao uvjeren u njegovu sigurnost. Još jedna od prednosti KECCAK algoritma je ta što postoje veće razlike u dizajnu i svojstvima u odnosu na SHA-2, što će budućim dizajnerima aplikacija i protokola omogućiti veću fleksibilnost u pronalasku algoritma koji više odgovara njihovim zahtjevima. [10]

4. SHA-3 STANDARD

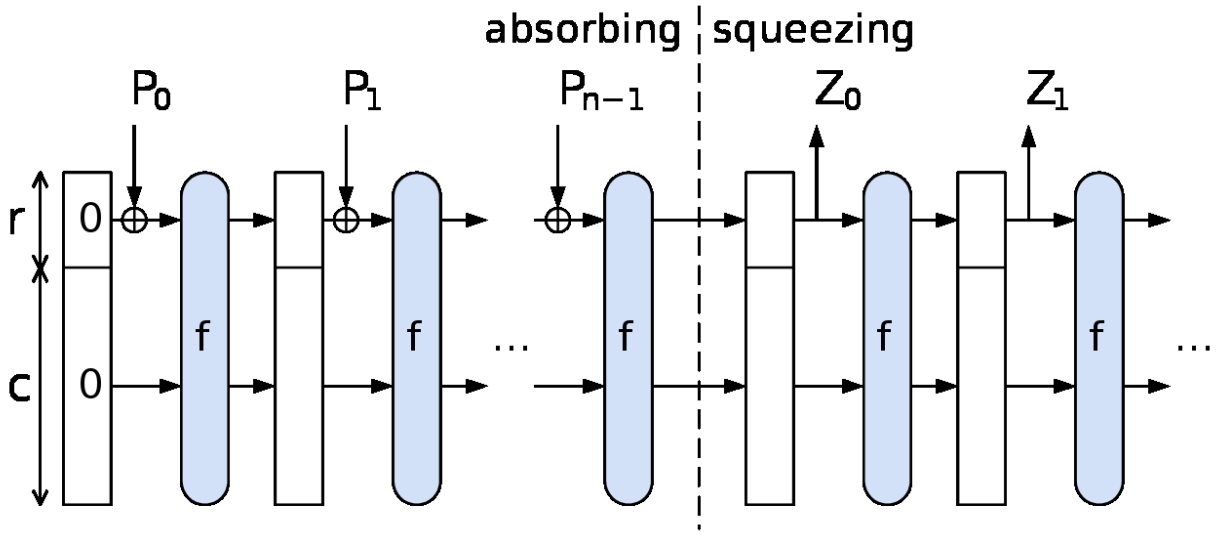
Nakon nekoliko uspješnih napada, *NIST* je 2007. godine objavio javno natjecanje za razvijanje potpuno novog *hash* algoritma- SHA-3. Svrha SHA-3 standarda nije zamjena SHA-2 standarda, međutim zbog uspješnih napada na MD5, SHA-0 i SHA-1, *NIST* je uočio potrebu za novim, različitim standardom koji je postao SHA-3. SHA-3 standard predstavlja obitelj funkcija koje nadopunjuju SHA-1 i SHA-2 funkcije. Ova obitelj temelji se na KECCAK algoritmu kojeg je *NIST* odabrao kao pobjednika natjecanja za SHA-3 kriptografski *hash* algoritam. Sastoji se od četiri kriptografske *hash* funkcije: SHA3-224, SHA3-256, SHA3-384 i SHA3-512, te dvije funkcije s proširivim izlazom koje se zovu SHAKE128 i SHAKE256. Iako ih je moguće koristiti na sličan način, uz uvjet izravnog prilagođavanja aplikacijama, funkcije proširivog izlaza (engl. *extendable–output functions*) razlikuju se od *hash* funkcija. Ove funkcije nazivaju se još i spužvaste funkcije zbog svoje strukture- spužvasta konstrukcija. [12]

Kod četiri kriptografske funkcije iz SHA-3 obitelji, SHA3-224, SHA3-256, SHA3-384 i SHA3-512, sufiks iza crtice označava fiksnu duljinu sažetka, npr. SHA3-256 proizvodi 256-bitni sažetak. Funkcije iz SHA-2 obitelji daju jednak niz duljina sažetaka zbog čega se SHA-3 funkcije mogu koristiti kao zamjena za funkcije iz SHA-2 obitelji i obratno. Za razliku od *hash* funkcija koje imaju izlaze fiksne duljine, izlazi funkcija s proširivim izlazom mogu biti prošireni na željenu duljinu. Sufiksi u nazivima (SHAKE128 i SHAKE256) funkcija s proširivim izlazom označavaju snagu sigurnosti koju ove funkcije mogu podržati. Duljine sažetaka *hash* funkcija su 160, 224, 256, 384 i 512 bita. Funkcije s proširivim izlazom logičan su izbor kada aplikacija zahtjeva kriptografsku *hash* funkciju koja nema standardiziranu duljinu sažetka. [12]

4.1. Spužvasta konstrukcija

Kao što je ranije navedeno, dvije funkcije iz SHA-3 obitelji razlikuju se od drugih, između ostalog, i zbog svoje konstrukcije. Radi se o funkcijama proširivog izlaza koje imaju konstrukciju spužve. U kontekstu kriptografije, spužvaste funkcije omogućuju generalizaciju *hash* funkcija na općenitije funkcije čija je duljina izlaza proizvoljna. Radi se o funkcijama koje za ulazne podatke promjenjive duljine vraćaju izlazne podatke također promjenjive duljine na temelju transformacije izlaza fiksne duljine. [12]

Spužvasta konstrukcija je jednostavna iterativna (ponavljajuća) konstrukcija za izgradnju funkcije s ulazom promjenjive duljine i proizvoljnom duljinom izlaza, a temelji se na transformaciji fiksne duljine ili na permutaciji koja djeluje na fiksni broj bita b, gdje se b naziva širina. Konstrukcija spužve djeluje prema sljedećoj formuli: $b = r + c$ bita. Ovdje r predstavlja brzinu prijenosa, dok c označava kapacitet. Na slici 4.1. vidi se konstrukcija spužve. [13]



Sl. 4.1. *Prikaz spužvaste konstrukcije: Pi predstavlja ulazne podatke, dok Zi predstavlja izlazne podatke* [14]

Prvo se svi bitovi postavljaju na nulu. Ulazna je poruka postavljena i izrezana na blokove koji se sastoje od r bita. Spužvasta se konstrukcija zatim odvija u dvije faze: faza apsorbiranja (upijanja) i faza cijeđenja. U prvoj fazi, fazi apsorbiranja ili upijanja, r-bitni ulazni blokovi prolaze kroz XOR funkciju koja ih postavlja u prve r bitove stanja, te se prepliću s aplikacijama funkcije f. Kada svi blokovi funkcije prođu kroz ovu fazu, konstrukcija spužve prelazi u fazu cijeđenja. U ovoj se fazi prvi r bitovi stanja vraćaju kao izlazni blokovi, isprepleteni s aplikacijama funkcije f. Broj izlaznih blokova bira korisnik. [13]

4.2. KECCAK algoritam

KECCAK je obitelj spužvastih funkcija koje su standardizirane u obliku funkcija proširivog izlaza SHAKE128 i SHAKE256 te SHA3-224 i SHA3-512 *hash* funkcija. Ovaj je algoritam djelo Guida Bertonija, Joan Daemena, Michaela Peetersa i Gillesa Van Asschea. U KECCAK-u osnovna je funkcija permutacija, izabrana iz seta od sedam KECCAK-$f[b]$ permutacija, gdje $b \in \{25, 50, 100, 200, 400, 800, 1600\}$ predstavlja širinu permutacije. Širina permutacije također je i širina stanja u spužvastoj konstrukciji. Stanje je organizirano kao niz od 5×5 traka, gdje je svaka traka duljine $w \in \{1, 2, 4, 8, 16, 64\}$, a $b = 25w$. [15]

4.3. KECCAK-p permutacija

U SHA-3 standardu način rada specificira KECCAK-p permutacija koja osigurava fleksibilnost za izmjenu veličine i sigurnosnih parametara. KECCAK-p permutacije određene su s dva parametra:

1.) fiksne duljine nizova koji se permutiraju, a nazivaju se širina permutacije
2.) broj iteracija unutarnje transformacije, nazvane runda (engl. *round).*

Širina permutacije označena je s b, a broj rundi iteracije označen je s n_r. KECCAK-p permutacija s n_r rundi i širinom b označava se kao KECCAK-p $[b, n_r]$. Permutacija KECCAK-p definirana je za bilo koji $b \in \{25, 50, 100, 200, 400, 800, 1600\}$ i za bilo koji pozitivni broj n_r. [12]

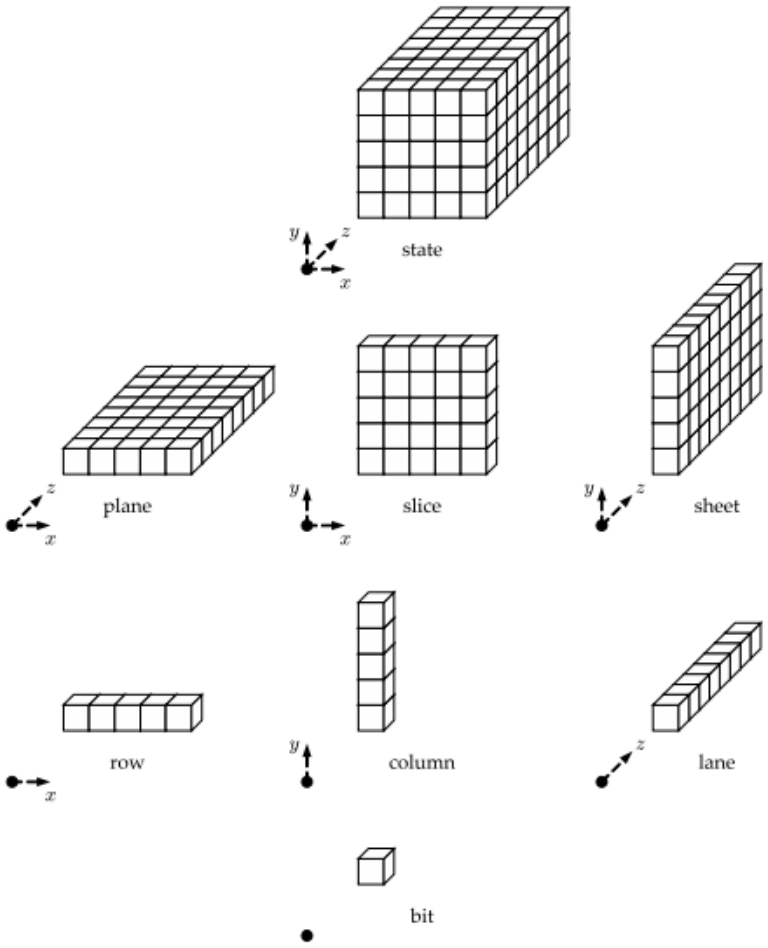
Runda KECCAK-p permutacije, označena s Rnd, sastoji se od niza od pet transformacija, koji se nazivaju koracima preslikavanja. Permutacija se određuje u smislu niza vrijednosti b bitova koji se opetovano ažurira i naziva se stanje. Stanje se u početku postavlja na ulazne vrijednosti permutacije. Stanje za KECCAK-p $[b, n_r]$ permutaciju sastoji se od b bita. Specifikacije ovoga standarda sadrže još dvije veličine vezane uz b, a to su: $b/25$ i $\log_2(b/25)$, označene s w i l. U tablici 4.1. prikazano je sedam mogućih vrijednosti ovih varijabli koje su definirane za KECCAK-p permutacije. [12]

Tab. 4.1. *Vrijednosti varijabli b,w i l definiranih za KECCAK-p*

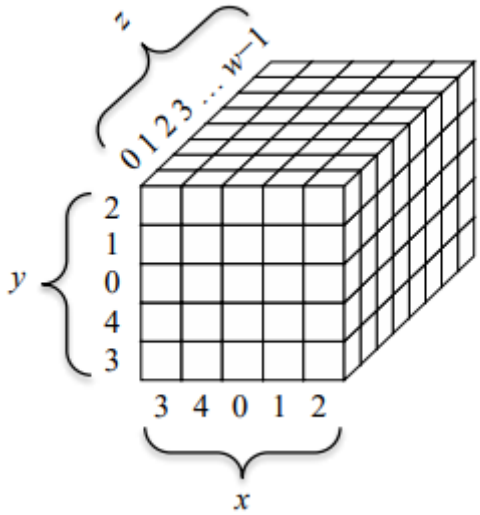
b	25	50	100	200	400	800	1600
w	1	2	4	8	16	32	64
l	0	1	2	3	4	5	6

Ulazna i izlazna stanja permutacija mogu se prikazati kao b-bitni nizovi. S označava niz koji predstavlja stanje, a njegovi su bitovi indeksirani od 0 do $b-1$: $S = S[0]\,||S[1]||\,...\,||\,S[b-2]||\,S[b-1]$. Ulazna i izlazna stanja koraka preslikavanja prikazuju se kao *5-5-w* (enlg. *5-by-5-by-w*) redovi bitova, a označeni su s $\boldsymbol{A}$. Njegovi su indeksi cjelobrojne trojke (*x, y, z*), te vrijedi: $0 \leq x < 5,\ 0 \leq y < 5$ i $0 \leq z < w$. Matrica stanja (enlg. *State array*) prikaz je stanja s trodimenzionalnim redovima koji su indeksirani na ovaj način. [12]

Slika 4.2. prikazuje matricu stanja za KECCAK-p permutaciju i njezine dvodimenzionalne podnizove, za slučaj kada je $b = 200$, iz čega slijedi $w = 8$. Dvodimenzionalni podnizovi nazivaju se listovi, ravnine i režnjevi (enlg. *sheets, planes, slices*), dok se jednodimenzionalni podnizovi nazivaju redovi, stupci i trake (engl. *row, column, lane*). Slika 4.3. prikazuje način označavanja matrice stanja. [12]



Sl. 4.2. *Prikaz matrice stanja KECCAK-p permutacije* [12]



Sl. 4.3. *Označavanje x, y i z koordinata matrice stanja* [12]

4.3.1. Pretvaranje nizova u matricu stanja

Ako S označava niz od b bitova te predstavlja stanje KECCAK-$p[b, n_r]$ permutacije, odgovarajuća matrica stanja, označena s $\boldsymbol{A}$, definirana je kako slijedi:

za sve trojke (x, y, z), gdje je $0 \leq x < 5$, $0 \leq y < 5$ i $0 \leq z < w$,

$$\boldsymbol{A}[\mathrm{x}, \mathrm{y}, \mathrm{z}] = S[\mathrm{w}(5\mathrm{y} + \mathrm{x}) + \mathrm{z}] \quad [12]$$

Primjer pretvaranja niza u matricu stanja:

- ako je $b = 200$, onda je $w = 8$

$\boldsymbol{A}[0, 0, 0] = S[0]$ $\quad$ $\boldsymbol{A}[1, 0, 0] = S[8]$ $\quad$ $\boldsymbol{A}[3, 0, 0] = S[24]$
$\boldsymbol{A}[0, 0, 1] = S[1]$ $\quad$ $\boldsymbol{A}[1, 0, 1] = S[9]$ $\quad$ $\boldsymbol{A}[3, 0, 1] = S[25]$
$\boldsymbol{A}[0, 0, 2] = S[2]$ $\quad$ $\boldsymbol{A}[1, 0, 2] = S[10]$ $\quad$ $\boldsymbol{A}[3, 0, 2] = S[26]$
$\vdots$ $\quad$ $\vdots$ $\quad \cdots \quad$ $\vdots$
$\boldsymbol{A}[0, 0, 6] = S[6]$ $\quad$ $\boldsymbol{A}[1, 0, 6] = S[14]$ $\quad$ $\boldsymbol{A}[3, 0, 6] = S[30]$
$\boldsymbol{A}[0, 0, 7] = S[7]$ $\quad$ $\boldsymbol{A}[1, 0, 7] = S[15]$ $\quad$ $\boldsymbol{A}[3, 0, 7] = S[31]$

i

$\boldsymbol{A}[0, 1, 0] = S[40]$ $\quad$ $\boldsymbol{A}[1, 1, 0] = S[48]$ $\quad$ $\boldsymbol{A}[3, 1, 0] = S[64]$
$\boldsymbol{A}[0, 1, 1] = S[41]$ $\quad$ $\boldsymbol{A}[1, 1, 1] = S[49]$ $\quad$ $\boldsymbol{A}[3, 1, 1] = S[65]$
$\boldsymbol{A}[0, 1, 2] = S[42]$ $\quad$ $\boldsymbol{A}[1, 1, 2] = S[50]$ $\quad$ $\boldsymbol{A}[3, 1, 2] = S[66]$
$\vdots$ $\quad$ $\vdots$ $\quad \cdots \quad$ $\vdots$
$\boldsymbol{A}[0, 1, 6] = S[46]$ $\quad$ $\boldsymbol{A}[1, 1, 6] = S[54]$ $\quad$ $\boldsymbol{A}[3, 1, 6] = S[70]$
$\boldsymbol{A}[0, 1, 7] = S[47]$ $\quad$ $\boldsymbol{A}[1, 1, 7] = S[55]$ $\quad$ $\boldsymbol{A}[3, 1, 7] = S[71]$

…

4.3.2. Pretvaranje matrice stanja u nizove

Ako $\boldsymbol{A}$ označava matricu stanja, onda je odgovarajući niz S, koji može biti od ravnina i traka (enlg. *planes and lanes*), definiran kako slijedi:

- Za svaki par brojeva (i, j), gdje su $0 \leq i < 5$ i $0 \leq j < 5$ niz traka-*lane*(i, j) je:
  lane $[i, j] = \boldsymbol{A}[i, j, 0] \,||\, \boldsymbol{A}[i, j, 1] \,||\, \ldots \,||\, \boldsymbol{A}[i, j, w\text{-}2] \,||\, \boldsymbol{A}[i, j, w\text{-}1]$ [12]

Primjer za $b = 200$, pa je $w = 8$:

lane$(0, 0) = \boldsymbol{A}[0, 0, 0] \,||\, \boldsymbol{A}[0,0,1] \,||\, \ldots||\, \boldsymbol{A}[0, 0, 6] \,||\, \boldsymbol{A}[0, 0, 7]$
lane$(1, 0) = \boldsymbol{A}[1, 0, 0] \,||\, \boldsymbol{A}[1,0,1] \,||\, \ldots \,||\, \boldsymbol{A}[1, 0, 6] \,||\, \boldsymbol{A}[1, 0, 7]$
lane$(2, 0) = \boldsymbol{A}[2, 0, 0] \,||\, \boldsymbol{A}[2,0,1] \,||\, \ldots||\, \boldsymbol{A}[2, 0, 6] \,||\, \boldsymbol{A}[2, 0, 7]$

- Za svaki broj j, takav da je $0 \leq j < 5$ niz ravnina- *plane* definiran je kao:

plane(j) = *lane* $(0, j)$ || *lane* $(1, j)$ || *lane* $(2, j)$ || *lane* $(3, j)$ || *lane* $(4, j)$

$\rightarrow$ S = *plane* (0) || *plane* (1) || *plane* (2) || *plane* (3) || *plane* (4) || [12]

Primjer za za $b = 200$, pa je $w = 8$:

$S = \boldsymbol{A}[0, 0, 0] \,||\, \boldsymbol{A}[0, 0, 1] \,||\, \ldots||\, \boldsymbol{A}[0, 0, 6] \,||\, \boldsymbol{A}[0, 0, 7]$
$||\, \boldsymbol{A}[1, 0, 0] \,||\, \boldsymbol{A}[1, 0, 1] \,||\, \ldots||\, \boldsymbol{A}[1, 0, 6] \,||\, \boldsymbol{A}[1, 0, 7]$
$||\, \boldsymbol{A}[2, 0, 0] \,||\, \boldsymbol{A}[2, 0, 1] \,||\, \ldots||\, \boldsymbol{A}[2, 0, 6] \,||\, \boldsymbol{A}[2, 0, 7]$
$||\boldsymbol{A}[3, 0, 0] \,||\, \boldsymbol{A}[3, 0, 1] \,||\, \ldots||\, \boldsymbol{A}[3, 0, 6] \,||\, \boldsymbol{A}[3, 0, 7]$
$||\boldsymbol{A}[4, 0, 0] \,||\, \boldsymbol{A}[4, 0, 1] \,||\, \ldots||\, \boldsymbol{A}[4, 0, 6] \,||\, \boldsymbol{A}[4, 0, 7]$

$||\boldsymbol{A}[0, 1, 0] \,||\, \boldsymbol{A}[0, 1, 1] \,||\, \ldots||\, \boldsymbol{A}[0, 1, 6] \,||\, \boldsymbol{A}[0, 1, 7]$
$||\boldsymbol{A}[1, 1, 0] \,||\, \boldsymbol{A}[1, 1, 1] \,||\, \ldots||\, \boldsymbol{A}[1, 1, 6] \,||\, \boldsymbol{A}[1, 1, 7]$
$||\boldsymbol{A}[2, 1, 0] \,||\, \boldsymbol{A}[2, 1, 1] \,||\, \ldots||\, \boldsymbol{A}[2, 1, 6] \,||\, \boldsymbol{A}[2, 1, 7]$
$||\boldsymbol{A}[3, 1, 0] \,||\, \boldsymbol{A}[3, 1, 1] \,||\, \ldots||\, \boldsymbol{A}[3, 1, 6] \,||\, \boldsymbol{A}[3, 1, 7]$
$||\boldsymbol{A}[4, 1, 0] \,||\, \boldsymbol{A}[4, 1, 1] \,||\, \ldots||\, \boldsymbol{A}[4, 1, 6] \,||\, \boldsymbol{A}[4, 1, 7]$

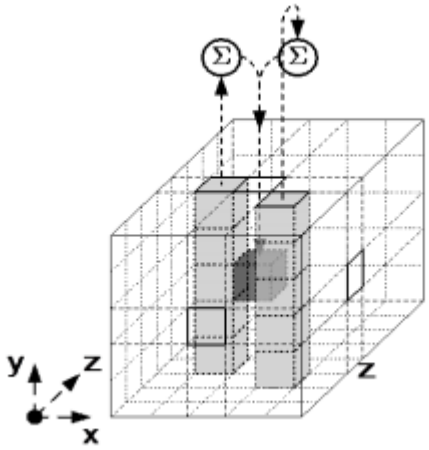
$\vdots$

$$\|\boldsymbol{A}[0, 4, 0] \| \boldsymbol{A}[0, 4, 1] \| \ldots \| \boldsymbol{A}[0, 4, 6] \| \boldsymbol{A}[0, 4, 7]$$
$$\|\boldsymbol{A}[1, 4, 0] \| \boldsymbol{A}[1, 4, 1] \| \ldots \| \boldsymbol{A}[1, 4, 6] \| \boldsymbol{A}[1, 4, 7]$$
$$\|\boldsymbol{A}[2, 4, 0] \| \boldsymbol{A}[2, 4, 1] \| \ldots \| \boldsymbol{A}[2, 4, 6] \| \boldsymbol{A}[2, 4, 7]$$
$$\|\boldsymbol{A}[3, 4, 0] \| \boldsymbol{A}[3, 4, 1] \| \ldots \| \boldsymbol{A}[3, 4, 6] \| \boldsymbol{A}[3, 4, 7]$$
$$\|\boldsymbol{A}[4, 4, 0] \| \boldsymbol{A}[4, 4, 1] \| \ldots \| \boldsymbol{A}[4, 4, 6] \| \boldsymbol{A}[4, 4, 7]$$

4.3.3. Koraci preslikavanja

Kao što je ranije navedeno, runda KECCAK-p premutacije sastoji se od niza od pet transformacija, a nazivaju se koraci preslikavanja. Oznake ovih pet koraka su: θ, ρ, π, χ i ι. Algoritam svakog koraka preslikavanja za ulaz uzima matricu stanja označenu s $\boldsymbol{A}$, a kao izlaz vraća promijenjenu matricu stanja označenu s $\boldsymbol{A'}$. Veličina stanja b je parametar koji se iz zapisa izostavlja budući da je b uvijek određen pri pozivanju koraka preslikavanja. Korak ι preslikavanja ima drugi ulaz- broj koji se naziva kružni indeks (engl. *round indeks*) i označen je s i_r. Ostali koraci preslikavanja ne ovise o kružnom indeksu. [12]

U koraku θ svaki se bit stanja *„XOR-a“* s paritetima dva stupca u polju. Točnije, za bit $\boldsymbol{A}[\mathrm{x}_0, \mathrm{y}_0, \mathrm{z}_0]$, x koordinata jednog od stupaca je $(x_0 - 1) mod\ 5$, sa istom z koordinatom, dok je x koordinata drugog stupca $(x_0 + 1) mod\ 5$, sa z koordinatom $(z_0 - 1) mod\ w$. Slika 4.4. prikazuje ilustraciju koraka θ preslikavanja. [16]



Sl. 4.4. *Prikaz koraka θ preslikavanja gdje simbol Σ predstavlja paritet* [16]

Algoritam koraka θ:

Ulaz: matrica stanja $\boldsymbol{A}$

Izlaz: matrica stanja $\boldsymbol{A'}$

Koraci: 1. Za sve parove (x, z) takve da su $0 \leq x < 5$ i $0 \leq z < w$ neka je:

$$C[x, z] = \boldsymbol{A}[x, 0, z] \oplus \boldsymbol{A}[x, 1, z] \oplus \boldsymbol{A}[x, 2, z] \oplus \boldsymbol{A}[x, 3, z] \oplus \boldsymbol{A}[x, 4, z]$$

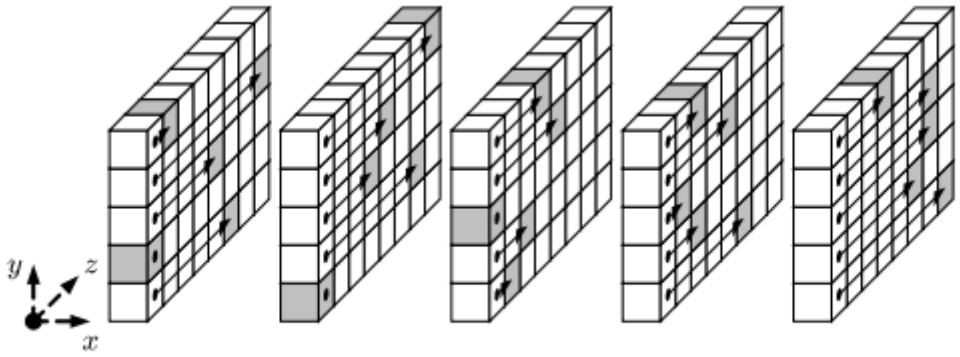
2. Za sve parove (x, z) takve da su $0 \leq x < 5$ i $0 \leq z < w$ neka je:

$$D[x, z] = C[(x\text{-}1) \bmod 5, z] \oplus C[(x\text{+}1) \bmod 5, (z\text{-}1) \bmod w]$$

3. Za sve trojke (x, y, z) takve da su su $0 \leq x < 5$, $0 \leq y < 5$ i $0 \leq z < w$ neka je:

$$\boldsymbol{A'}[x, y, z] = \boldsymbol{A}[x, y, z] \oplus D[x, z] \quad [12]$$

Zadatak koraka ρ je rotacija bitova svake trake po dužini, nazvane *offset*, koja ovisi o fiksni x i y koordinatama trake. Na slici 4.5. prikazana je ilustracija koraka ρ za slučaj kada je $b = 200$. Označavanje koordinata x i y na ovoj slici jednako je označavanju koje je ranije prikazano na slici 4.3. Primjerice, traka $\boldsymbol{A}[0, 0]$ nacrtana je u sredini srednjeg lista, a traka $\boldsymbol{A}[2, 3]$ nacrtana je na dnu krajnje desnog lista. Na svakoj traci sa slike crna točka označava bit čija je z koordinata 0, a osjenčana kocka predstavlja poziciju tog bita nakon izvršenja koraka ρ. [12]



Sl. 4.5. *Prikaz koraka ρ preslikavanja, za slučaj kada je b=200, a w=8* [16]

Algoritam koraka ρ :

Ulaz: matrica stanja $\boldsymbol{A}$

Izlaz: matrica stanja $\boldsymbol{A'}$

Koraci: 1. Za sve z takve da je $0 \leq z < w$ neka je: $\boldsymbol{A'}[0, 0, z] = \boldsymbol{A}[0, 0, z]$

2. Neka je $(x, y) = (1, 0)$

3. Za sve t 0 do 23:

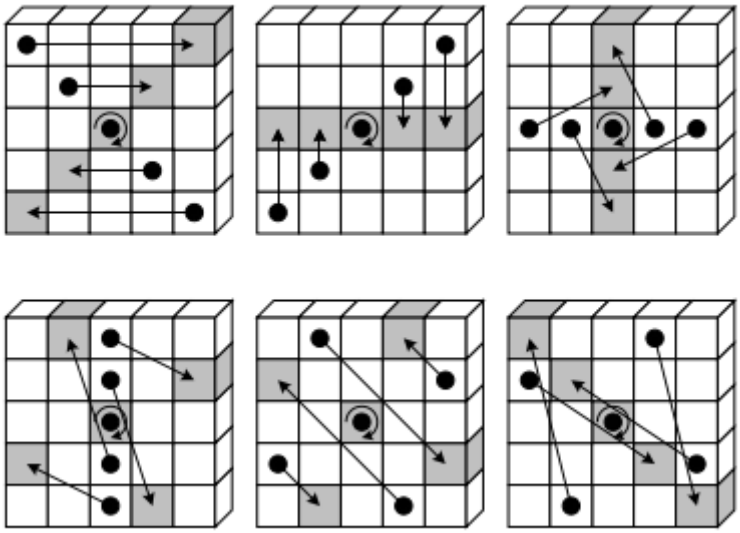
a) za sve z koji su $0 \leq z < w$ neka je:

$\boldsymbol{A'}[x, y, z] = \boldsymbol{A}[x, y, (z-(t+1)(t+2)/2) \bmod w]$

b) neka je: $(x, y) = (y, (2x+3y) \bmod 5)$

4. Vrati $\boldsymbol{A'}$ [12]

U koraku π preuređuju se pozicije traka kako je prikazano na slici 4.6. Označavanje je također jednako kao na slici 4.3., pa je primjerice bit s koordinatama $x=y=0$ nacrtan na sredini režnja. [12]



Sl. 4.6. *Prikaz koraka π primijenjenog na pojedini režanj* [16]

Algoritam koraka π :

Ulaz: matrica stanja ***A***

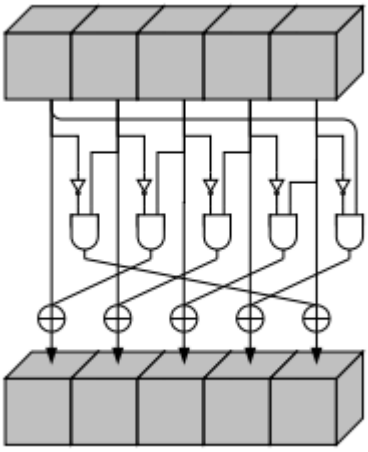
Izlaz: matrica stanja ***A′***

Koraci: 1. Za sve trojke (*x, y, z*) takve da su su $0 \leq x < 5$, $0 \leq y < 5$ i $0 \leq z < w$ neka je:

$$A'[x, y, z] = A[(x+3y) \bmod 5, x, z]$$

2. Vrati ***A′*** [12]

Korak χ *„XOR-a“* svaki bit sa nelinearnom funkcijom sa sa ostala dva bita toga reda kako je prikazano na slici 4.7. [12]



Sl. 4.7. *Prikaz koraka χ primijenjenog na pojedini red* [16]

Algoritam koraka χ :

Ulaz: matrica stanja ***A***

Izlaz: matrica stanja ***A′***

Koraci: 1. Za sve trojke (*x, y, z*) takve da su su $0 \leq x < 5$, $0 \leq y < 5$ i $0 \leq z < w$ neka je:

$$A'[x, y, z] = A[x, y, z] \oplus ((A[(x+1) \bmod 5, y, z] \oplus 1) \cdot A[(x+2) \bmod 5, y, z]).$$

2. Vrati ***A′*** [12]

Zadatak koraka ι je izmijeniti neke od bitova trake [0, 0] na način koji ovisi o kružnom indeksu i_r. Ovaj korak ne utječe na ostale trake. [12]

4.4. Razlika između KECCAK-p i KECCAK-f

KECCAK-f obitelj permutacija zapravo predstavlja specijalizaciju KECCAK-p permutacija za poseban slučaj kada je $n_r = 12 + 2l$. Iz toga slijedi da je:

$$\text{KECCAK-}f\ [b] = \text{KECCAK-}p\ [b, 12 + 2l]$$

Runde KECCAK-f permutacija indeksirane su od 0 do $11 + 2l$. [12]

5. UPOTREBA SHA-3 STANDARDA

Iako SHA-3 predstavlja zadnji dostupni *Secure Hash* algoritam, njegov prethodnik, SHA-2, ostaje u upotrebi nekih od najčešće korištenih sigurnosnih aplikacija. Važno je napomenuti da *NIST* ovaj algoritam nije namijenio kao zamjenu za njegovog prethodnika, već kao dopunu za SHA-2 budući da se ovaj algoritam još uvijek smatra sigurnim. Bez obzira na činjenicu da SHA-2 koristi jednaku konstrukciju kao SHA-1, između ova dva algoritma postoje bitne razlike. Zbog korištenja veće duljine bita te većih ulaznih i izlaznih podataka, SHA-2 je u odnosu na SHA-1 pokazao sposobnost pružanja bolje zaštite, pa ga još uvijek koriste brojne sigurnosne aplikacije poput američkih vladinih aplikacija za zaštitu osjetljivih podataka i Bitcoina. Zbog uspješnih napada na SHA-1 algoritam, *NIST* je objavio eliminaciju ove verzije te je ista postupno zamijenjena SHA-2 algoritmom. S obzirom da je SHA-3 tada već postojao došlo je do pitanja zašto se ne prebaciti odmah na najnoviju verziju obitelji ovih algoritama. Odgovor na to pitanje leži u činjenici da većina svjetskih računala nema verzije softvera niti hardvera koji podržavaju ovaj algoritam. Što se tiče brzine SHA-3 standarda, on se od SHA-2 smatra softverski sporijim. Međutim, kada je riječ od hardveru SHA-3 puno je brži od svojih prethodnika. Ova činjenica svakako ide u prilog najnovijem standardu budući da kriptografi sve više koriste hardverske komponente. SHA-3 funkcije također su dizajnirane tako da pružaju otpor i drugim napadima poput napada produljene duljine (engl. *lenght-extensions attacks*). Ovim se napadima odupire slučajna funkcija iste izlazne duljine. Dakle, iako je prethodnik SHA-3 još uvijek u uporabi, ovaj se algoritam smatra boljim i fleksibilnijim jer nudi novosti poput mogućnosti proširenja izlaza, samo još nema jednako široku upotrebu. [17]

5.1. Napad kvantnih računala

Kvantna su računala ona računala koja za rješavanje bilo kakvih problema, odnosno za računanje, koriste kvantno-mehaničke fenomene. Ti se fenomeni konkretno odnose na superpoziciju stanja i kvantno sprezanje. U klasičnom se računalu količina podataka mjeri u bitovima, dok se u kvantnom računalu podatci mjere kvantnim bitovima- qubitovima (engl. *quantum bit*). Zbog izvedbe na vrlo malom broju kvantnih bitova, ova računala postižu izuzetne brzine. Groverov algoritam pokazuje da kvantna računala mogu izvesti strukturirani *pre-image* napad radom od $\sqrt{2^d} = 2^{d/2}$, dok klasični napad grube sile zahtjeva rad od 2^d. Strukturirani *pre-image* napad podrazumijeva *second pre-image* napad, a samim time i napad sudara. Dokazano je da se Merkle- Damgåard konstrukcija, koju koristi SHA-2, urušava kao posljedica napada kvantnih računala. Međutim, za spužvastu konstrukciju, koju koristi SHA-3, pronađeno je

nedovoljno dokaza da bi ovakva konstrukcija mogla biti urušena. Dakle, SHA-3 se smatra još uvijek otpornom na napad kvantnih računala. Zbog toga se ovaj standard smatra daleko naprednijim od svoga prethodnika. [14], [18]

6. ZAKLJUČAK

U prvom dijelu završnog rada obrađen je pojam *hash* funkcija općenito te su navedena i opisana njihova svojstva. *Hash* funkcije su jednosmjerne funkcije koje za dani ulazni podatak proizvoljne duljine vraćaju izlazni podatak fiksne duljine. *Hash* funkcije, koje se smatraju sigurnima, ispunjavaju nekoliko svojstava. Tri najvažnija svojstva ovih funkcija su otpornost na sudare, *pre-image* i *second pre-image* otpornost.

Prikazani su članovi obitelji *Secure Hash* algoritama, te je ukratko objašnjen njihov razvoj kroz povijest. Radi razumijevanja kronologije nastanka SHA-3 algoritma, ukratko su opisani njegovi prethodnici. SHA-0, kao i njegova novija verzija- SHA-1, algoritmi su koji danas više nisu u upotrebi zbog dokazanih uspješnih napada na njih. Nasuprot tome, SHA-2 se i danas koristi u raznim sigurnosnim aplikacijama. Iako SHA-3, kao najnoviji član ove obitelji funkcija, nije trebao zamijeniti svoga prethodnika, *NIST* je 2007. godine objavio otvoreni natječaj za ovaj standard. Od prvobitno 64 prijavljenih kandidata, 2012. godine KECCAK je izabran za pobjednika NIST-ova natjecanja.

Najnoviji standard obitelji *Secure Hash* algoritama, SHA-3, sastoji se od četiri kriptografske *hash* funkcije te dvije funkcije proširivog izlaza. Ove funkcije zasnivaju se na drugačijoj konstrukciji nego njeni prethodnici, na spužvastoj konstrukciji zbog čega se često nazivaju i spužvaste funkcije. Spužvaste su funkcije omogućile poopćenje *hash* funkcija na funkcije čija je duljina izlaznih podataka proizvoljna. KECCAK algoritam predstavlja obitelj upravo spužvastih funkcija koje su standardizirane u obliku funkcija proširivog izlaza i *hash* funkcija. Način rada SHA-3 standarda specificiran je KECCAK-p permutacijom koja je odgovorna za mogućnost fleksibilnosti izmjene veličina parametara.

Iako SHA-3 standard nije bio namijenjen kao zamjena svog prethodnika koji se još uvijek i koristi, ovaj algoritam pruža svojstva koja će biti sve korisnija u budućnosti. Budući da se smatra sigurnijim za rad, bržim u hardverskim komponentama, fleksibilnijim zbog novosti poput mogućnosti proširenja izlaznih podataka te za sada još uvijek otpornim na napad kvantnih računala, zaključeno je da će, iako još nije u širokoj upotrebi, ovaj algoritam jednoga dana vjerojatno u potpunosti zamijeniti svog prethodnika.

LITERATURA

[1] A. Maetouq, S. Daud, N. Ahmad, N. Maarop, N. N. A. Sjarif, H. Abas, *„Comparsion of Hash Function Algorithms Against Attacks: A Review“*, International Journal of Advanced Computer Science and Applications, br. 8, sv. 9, 2018. , dostupno na: https://pdfs.semanticscholar.org/6ed2/50d11a5c80f550bd8efcc673606c3cae34b7.pdf, pristup ostvaren 10. kolovoza 2018.

[2] E. Harmoush, *„Message Integrity“*, Practical Networking, 2015., dostupno na: https://www.practicalnetworking.net/series/cryptography/message-integrity/, pristup ostvaren 28. kolovoza 2019.

[3] S. Al-Kuwari, J. H. Davenport, R. J. Bradford, *„Cryptographic Hash Functions: Recent Design Trends and Security Notions“*, Department of Computer Science, University of Bath, Bath, dotupno na: https://eprint.iacr.org/2011/565.pdf, pristup ostvaren 28. kolovoza 2019.

[4] *“Cryptographic and network security principles and practice- Requirements and Security“*, Brain Kart, dostupno na: https://www.brainkart.com/article/Requirements-and-Security_8448/, pristup ostvaren 28. kolovoza 2019.

[5] *„Cryptographic Hash Functions Explained: A Beginner's Guide“*, Komodo Platform, 2018., dostupno na: https://komodoplatform.com/cryptographic-hash-function/, pristup ostvaren 28. kolovoza 2019.

[6] R. Purohit, U. Mishra, A. Bansal, *„A Survey on Recent Cryptographic Hash Function Design“*, International Journal of Emerging Trends & Technology in Computer Science, 2013., dostupno na: https://pdfs.semanticscholar.org/7223/a289315f1df906b3dde23b1775b30e70d9a7.pdf, pristup ostvaren 30. kolovoza 2019.

[7] *„Secure Hash Algorithms“*, Wikipedia, dostupno na: https://en.wikipedia.org/wiki/Secure_Hash_Algorithms, pristup ostvaren 30. kolovoza 2019.

[8] X. Wang, Y. L. Yin , H. Yu, *„Finding Collision int he Full SHA-1“*, 2005., dostupno na: https://link.springer.com/content/pdf/10.1007%2F11535218_2.pdf, pristup ostvaren: 2. rujna 2019.

[9] N. Kishore, B. Kapoor, *„Attacks on and Advances in Secure Hash Algorithms“,* IAENG International Journal of Computer Science, kolovoz 2016., dostupno na:

http://www.iaeng.org/IJCS/issues_v43/issue_3/IJCS_43_3_08.pdf, pristup ostvaren 13. rujna 2019.

[10] *„SHA-3 Selection Announcement“*, NIST, dostupno na: https://csrc.nist.gov/CSRC/media/Projects/Hash-Functions/documents/sha-3_selection_announcement.pdf, pristup ostvaren 15. rujna 2019.

[11] J. Kelsey, *„SHA-3: Past, Present and Future“*, NIST, 2013., dostupno na: https://csrc.nist.gov/CSRC/media/Projects/Hash-Functions/documents/kelsey_ches2013_presentation.pdf?fbclid=IwAR0HHh7-pLn9YVbCNtfjpwu_XVYZR-vBeP0vFkfpfNtn0wLVTSaSmiN8dh8 , pristup ostvaren 17. rujna 2019.

[12] *„Sha-3 Standard: Permutation-Based Hash And Extendable Output Functions“,* Federal Information Processing Standards Publication, kolovoz 2015., dostupno na: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf, pristup ostvaren 11. lipnja 2019.

[13] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, *„Cryptographic sponge functions“*, 2011., dostupno na: https://keccak.team/files/CSF-0.1.pdf, pristup ostvaren 2. rujna 2019.

[14] *„SHA-3“*, Wikipedia, dostupno na: https://en.wikipedia.org/wiki/SHA-3#Security_against_quantum_attacks, pristup ostvaren 2. rujna 2019.

[15] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, R. Van Keer, S. Hoffert, *„KECCAK specifications summary“*, dostupno na: https://keccak.team/keccak_specs_summary.html, pristup ostvaren 2. rujna 2019.

[16] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, *„The KECCAK reference“*, 2011., dostupno na: https://keccak.team/files/Keccak-reference-3.0.pdf, pristup ostvaren 11. rujna 2019.

[17] S. Jones, *„What You Need to Know About SHA-3 for Embedded System Security“*, Electronic Design, 2019., dostupno na: https://www.electronicdesign.com/embedded-revolution/what-you-need-know-about-sha-3-embedded-system-security, pristup ostvaren 17. rujna 2019.

[18] I. Rončević, *„Kako radi kvantno računaloi zašto je toliko mahnito brže? Hint- jer je lukavo“*, Ideje.hr, 2017., dostupno na: http://ideje.hr/radi-kvantno-racunalo-zasto-toliko-mahnito-brze-hint-lukavo/, pristup ostvaren 17. rujna 2019.

SAŽETAK

U završnom radu opisan je SHA-3 standard, njegov nastanak, dizajn, način njegova rada, te njegove upotrebe i prednosti u odnosu na prethodnike. Prikazana su svojstva *hash* funkcija općenito te razni primjeri njihove primjene. Prikazana je kronologija nastanka SHA-3 standarda te su navedeni i ukratko opisani njegovi prethodnici (SHA-0, SHA-1 i SHA-2). Prikazana je konstrukcija ovoga standarda- spužvasta konstrukcija, koja mu omogućuju veću fleksibilnost. Objašnjen je način rada ove funkcije koji specificiran KECCAK-p permutacija te su prikazani koraci preslikavanja od pet transformacija koje ova permutacija koristi. Navedena je upotreba SHA-3 standarda zajedno s njegovim prednostima. Zbog navedenih prednosti s obzirom na prethodnike, zaključeno je da će u budućnosti ovaj standard vjerojatno imati još značajniju ulogu u kriptografiji.

Ključne riječi: *hash* funkcija, KECCAK, kvantno računalo, spužvasta konstrukcija

ABSTRACT

Title: Review of SHA-3 standard

In this bachelor thesis SHA-3 standard, this standard's origin, design, method of its operating, as well as its uses and advantages over its predecessors were described. The properties of hash functions in general and various examples of their application are shown. The chronology of the formation of the SHA-3 standard is presented and its predecessors (SHA-0, SHA-1, and SHA-2) are listed and shortly described. The construction of this standard is presented - a sponge construction, which allows it more flexibility. The mode of operation of this function, which is specified by the KECCAK-p permutation, is explained, and the mapping steps of the five transformations used by this permutation are shown. The use of the SHA-3 standard, along with its benefits, is outlined. Because of these advantages in view of itspredecessors, it is concluded that in the future this standard is likely to play an even more significant role in cryptography.

Keywords: *hash* function, KECCAK, quantum computer, sponge construction

ŽIVOTOPIS

Lucija Šaravanja rođena je 22. srpnja 1996. godine u Osijeku. U Osijeku završava OŠ „Sveta Ana“ nakon čega 2011. godine upisuje Prirodoslovno-matematičku gimnaziju. 2015. godine ostvaruje upis na Fakultet elektrotehnike, računarstva i informacijskih tehnologija, smjer elektrotehnika.

Potpis:

_____