Razvoj i optimizacija heš funkcija korišćenjem sistema ostataka brojeva

Milija Pavlović, Negovan Stamenković
Odsek za informatiku
Prirodno-matematički fakultet
Kosovska Mitrovica, Srbija
milija.pavlovic@pr.ac.rs, negovan.stamenkovic@pr.ac.rs

Boris Damjanović
Fakultet za informacione tehnologije i inženjerstvo
Univerzitet Union – Nikola Tesla
Beograd, Srbija
boris.damjanovic@fpsp.edu.rs

Sažetak — **U savremenoj kriptografiji, heš funkcije predstavljaju jedan od osnovnih alata. Njihov koncept se zasniva na kompresiji ulaznih podataka promenljive dužine u izlaz fiksne dužine. Njihova primena obuhvata autentifikaciju podataka, digitalne potpise, generisanje ključeva, kao i bezbedno čuvanje lozinki. Heš zasnovan na RNS aritmetici koristi modularnu aritmetiku i binarne operacije za generisanje heš vrednosti. Poruka se deli na blokove od 64 bita, obrađuje kroz niz modula, dok se međuvrednosti kombinuju operacijom XOR i konvertuju u binarni oblik. Konačna heš vrednost dobija se spajanjem rezultujućih binarnih zapisa. Predloženi algoritam je dizajniran s fokusom na sigurnost i otpornost na kolizije, uz modularnu strukturu koja omogućava prilagodljivost i optimizaciju. Rad uključuje matematički opis i pseudokod za praktičnu primenu u različitim kontekstima kriptografske zaštite.**

Ključne riječi - heš funkcije; RNS; moduli; kolizije;

I. Uvod

Heš funkcije se ubrajaju u ključne koncepte u savremenim digitalnim tehnologijama, uzimajući u obzir da služe kao osnova za veliki broj različitih primena kao što su osiguravanje integriteta podataka, sigurnost komunikacije i efikasnost pretraživanja. Iako je njihov koncept relativno jednostavan, dizajn i primena heš funkcija nije ni malo lak posao i zahteva duboko poznavanje računarskih i matematičkih principa. Heš funkcije su algoritmi koji pretvaraju poruke proizvoljne dužine u heš vrednosti fiksne veličine [1].

Heš možemo opisati na sledeći način:

1. Heš se može primeniti na blok podataka promenljive veličine.

2. Heš proizvodi izlaz fiksne dužine.

3. Funkciju H(x) je relativno jednostavno izračunati za bilo koje dato x, što čini primenu u hardveru i softveru praktičnom. (x predstavlja ulaz koji unosite u heš).

4. Za bilo koju datu vrednost h, računski je neizvodljivo pronaći x tako da je H(x) = h. Ova osobina se u literaturi naziva svojstvom jednosmernosti.

5. Za bilo koji dati blok x, računski je neizvodljivo pronaći y $\neq$ x tako da je H(y) = H(x). Ova osobina se naziva slabom otpornošću na kolizije.

6. Računski je neizvodljivo pronaći bilo koji par x i y takav da je H(x) = H(y). Ova osobina se naziva jakom otpornošću na kolizije [2].

Postoje dva tipa heš funkcija: sa ključem i heš funkcije bez ključa. Heš funkcije sa ključem koriste poruku i tajni ključ, dok heš funkcije bez ključa koriste samo ulaznu poruku. Kod funkcija sa ključem napadačima je izuzetno teško da generišu identične heš vrednosti ako ne poseduju tajni ključ. Ipak, razvojem računarske snage i kriptoanalitičkih alata stariji heš algoritmi postali su ranjiviji na napade. Očuvanje bezbednosnih standarda i zaštita od napada zahtevaju razvoj novih, robusnijih kriptografskih algoritama [3].

Funkcionisanje većine uobičajenih heš algoritama medju kojima su najznačajniji MD4, MD5, SHA-1, SHA-2 i SHA-3 leži u primeni različitih aritemtičkih, logičkih i algebarskih operacija. Ovi algoritmi su se vremenom pokazali kao nepouzdani, pokazujući odredjene slabosti u pogledu otpornosti na napade zasnovane na kolizijama. Pored ovoga, primena lienarnih operacija u odredjenim algoritmima heš funkcija može uticati na njihovu ranjivost na odredjene kriptografske napade [4].

Kada govorimo o primeni heš funkcija, jedna od najčešćih primena heš funkcija je u oblastima kriptografije, gde imaju ključnu ulogu u obezbeđivanju poverljivosti i autentičnosti podataka. Kriptografske heš funkcije, među kojima se posebno izdvajaju SHA-2 i SHA-3, dizajnirane su tako da zadovolje stroge kriterijume. Jedan od ključnih kriterijuma je otpornost na kolizije, što znači da je praktično nemoguće pronaći dva različita unosa koja proizvode isti heš. Zbog ovih svojstava oni su postali nezamenljivi u protokolima za bezbednu komunikaciju, digitalnim potpisima i autentifikaciji poruka. Pored toga, heš funkcije se široko koriste u organizaciji i efikasnom pristupu velikim bazama podataka. Tehnike kao što su heš tabele omogućavaju brzo pretraživanje i upravljanje podacima, čime se postiže značajna ušteda u pogledu vremena i resursa. Ova primena ima izuzetan značaj u oblastima kao što su pretraživači, skladišta podataka i vremenski osetljive aplikacije. Razvoj i optimizacija heš funkcija takođe se susreću sa izazovima koji proizlaze iz eksponencijalnog rasta količine

podataka u savremenom svetu. Sa sve većom potrebom za obradom podataka u raspodeljenim sistemima i oblacima, važnost skalabilnosti i efikasnosti heš funkcija postaje sve izraženija. Osim toga, pojava novih bezbednosnih pretnji zahteva konstantno usavršavanje kriptografskih algoritama, kako bi se odgovorilo na izazove koje donose napredni napadi. Danas, inovacije u oblasti heš funkcija ne obuhvataju samo njihovu primenu u tradicionalnim sistemima, već i u novim tehnologijama kao što su blokčejn i kriptovalute. Na primer, heš funkcije predstavljaju osnovu mehanizama kao što je „proof of work“ u kriptovalutama poput bitkoina, gde osiguravaju nepovratnost i proverljivost transakcija u raspodeljenom okruženju. Isto tako, u mašinskom učenju i veštačkoj inteligenciji, tehnike heširanja se primenjuju za kompresiju podataka i ubrzanje algoritama [5].

Postojeći heš algoritmi, poput SHA-2 i SHA-3, imaju ograničenja u pogledu računske zahtevnosti, otpornosti na napade sa sporednih kanala i efikasnosti u hardverskim implementacijama. Predloženi pristup zasnovan na sistemu ostataka (RNS) poboljšava efikasnost izvođenjem modularnih operacija paralelno i povećava otpornost na određene napade. Osim toga, RNS-heš funkcije mogu se optimizovati za hardversko ubrzanje, što ih čini pogodnim za ograničena okruženja.

II. HEŠIRANJE ZASNOVANO NA TRADICIONALNIM MATEMATIČKIM PRINCIPIMA

Heš funkcije spadaju u fundamentalne algoritme u kriptografiji i generalno u računarskim naukama, a njihova primarna uloga jeste pretvaranje ulaznih poruka proizvoljne dužine u izlazne vrednosti fiksne dužine. Neke od najpopularnijih heš funkcija koje su korišćene kroz istoriju, pa sve do danas su: MD4, MD5, RIPEMD, SHA-1, SHA-2, SHA-3. Neki su proizašli jedni iz drugih, dok su drugi utemeljeni na sasvim različitim pristupima. Kada su u pitanju operacije koje se koriste za obradu poruke u heš funkcijama, najčešće su to:

• Bitovne operacije - AND, OR ili XOR, koriste se za upravljanje bitovima na način koji je optimizovan za procesorsku efikasnost.

• Rotacije i pomeranje bitova – pomažu da se informacije ravnomerno distribuiraju kroz heš

• Dopunjavanje poruke (padding) – dodavanje bitova na kraj poruke kako bi njena dužina bila deljiva sa veličinom bloka.

• Mešanje i permutacije – redosled bitova u poruci se meša kroz različite runde kako bi se povećala otpornost na analitičke napade.

• Aritmetičke operacije – koriste se za kombinovanje delova poruka na efikasan i nelinearan način.

• Kompresione funkcije – specijalne matematičke funkcije koje smanjuju veličinu podataka na fiksnu dužinu [6].

III. HEŠIRANJE ZASNOVANO NA RNS ARITMETICI

Neka je data poruka $\boldsymbol{M}$ dužine N. Najpre se poruka dopunjava nulama kako bise osiguralo da njena dužina n bude deljiva sa 64. Formalno, ako je $\lceil N/64 \rceil$ najmanji ceo broj veći ili jednak , onda je:

$$n = 64 \cdot \lceil N/64 \rceil$$

Dopunjena poruka $\boldsymbol{M}^*$ ima dužinu n, a njen sadržaj je:

$$\boldsymbol{M}^* = \boldsymbol{M} \,||\, 0_{n-N}$$

gde je 0_{n-N} niz nula dužina $n-N$, a $||$ označava konkatenaciju.

Zatim, se poruka $\boldsymbol{M}^*$ deli na t blokova $\boldsymbol{B_i}$ dužine 64 bita:

$$\boldsymbol{M}^* = \boldsymbol{B}_1 \,||\, \boldsymbol{B}_2 \,||\, \ldots \,||\, \boldsymbol{B}_t, \qquad t = \frac{n}{64}$$

A. Obrada blokova pomoću modula

Za svaki blok $\boldsymbol{B_i}$, koristi se skup modula $\{x_1, x_2, ..., x_8\}$, gde su:

$x_1 = 211, x_2 = 223, x_3 = 227, x_4 = 229, x_5 = 233, x_6 = 239,$
$x_7 = 241, x_8 = 251.$

Nakon izbora modula. Računa se ostatak i količnik za svaki modul:

$$a_{i,j} = \boldsymbol{B_i} \bmod x_i, \; A_{i,j} = \left\lfloor \frac{\boldsymbol{B_i}}{x_i} \right\rfloor, \; j \in \{1,2,...,8\}$$

B. Iterativna obrada

Nakon izračunavanja $\{a_{i,j}, A_{i,j}\}$, vrši se dalja iterativna obrada kroz sledeće korake:

1. **Računanje novih vrednosti za ostatke i količnike**:

- Za ostatke:
  $b_{i,j} = \left(A_{i,j} + a_{i,j}\right) \bmod x_j, \; j \in \{1,2,...,8\}.$
- Za količnike:
  $$\boldsymbol{B}_{i,j} = \left\lfloor \frac{\left(A_{i,j} + a_{i,j}\right)}{x_j} \right\rfloor, \quad j \in \{1,2,...,8\}.$$

2. **Nastavak obrade sa prethodnim rezultatima**:

Iz prethodno navedenih izraza, daljom obradom dobija se:

$$c_{i,j} = \left(\boldsymbol{B}_{i,j} + b_{i,j} \right) \mod x_j,\ C_{i,j} = \left\lfloor \frac{\left(\boldsymbol{B}_{i,j} + b_{i,j} \right)}{x_j} \right\rfloor.$$

Proces se ponavlja za

$$\left\{ d_{i,j}, D_{i,j} \right\},\ \left\{ e_{i,j}, E_{i,j} \right\},\ \left\{ f_{i,j}, F_{i,j} \right\},\ \left\{ g_{i,j}, G_{i,j} \right\},\ \left\{ h_{i,j}, H_{i,j} \right\}.$$

C. Formalni izraz za opšti korak

Za svaki korak k (gde je $k \geq 1$), računa se :

$$x_{i,j}^{(k)} = \left(X_{i,j}^{(k-1)} + x_{i,j}^{(k-1)} \right) \mod x_j,$$

$$X_{i,j}^{(k)} = \left\lfloor \frac{\left(X_{i,j}^{(k-1)} + x_{i,j}^{(k-1)} \right)}{x_j} \right\rfloor,$$

gde su $x_{i,j}^{(0)} = a_{i,j}$ i $X_{i,j}^{(0)} = A_{i,j}$.

Ovaj postupak se nastavlja do završnog koraka gde dobijamo rezultate $\left\{ h_{i,j}, H_{i,j} \right\}$ za svaki blok $\boldsymbol{B}_i$.

D. Algoritam I deo

ULAZ: Poruka M, dužina N

POSTAVI: x = [211, 223, 227, 229, 233, 239, 241, 251] // Moduli

POSTAVI: n = CEIL(N / 64) * 64 # Prilagodi dužinu poruke tako da je deljiva sa 64

DODAJ nule na kraj poruke M dok dužina ne postane n

PODELI poruku M na t blokova B[1], B[2], ..., B[t] od po 64 bita

// Inicijalizacija vrednosti za računanje

ZA svaki blok B[i], gde je i od 1 do t:

ZA svaki modul x[j], gde je j od 1 do 8:

// Korak 1: Računanje početnih vrednosti

a[i][j] = B[i] MOD x[j]

A[i][j] = FLOOR(B[i] / x[j])

// Iterativno računanje

b[i][j] = (A[i][j] + a[i][j]) MOD x[j]

B1[i][j] = FLOOR((A[i][j] + a[i][j]) / x[j])

c[i][j] = (B1[i][j] + b[i][j]) MOD x[j]

C[i][j] = FLOOR((B1[i][j] + b[i][j]) / x[j])

d[i][j] = (C[i][j] + c[i][j]) MOD x[j]

D[i][j] = FLOOR((C[i][j] + c[i][j]) / x[j])

e[i][j] = (D[i][j] + d[i][j]) MOD x[j]

E[i][j] = FLOOR((D[i][j] + d[i][j]) / x[j])

f[i][j] = (E[i][j] + e[i][j]) MOD x[j]

F[i][j] = FLOOR((E[i][j] + e[i][j]) / x[j])

g[i][j] = (F[i][j] + f[i][j]) MOD x[j]

G[i][j] = FLOOR((F[i][j] + f[i][j]) / x[j])

h[i][j] = (G[i][j] + g[i][j]) MOD x[j]

H[i][j] = FLOOR((G[i][j] + g[i][j]) / x[j])

IZLAZ: h[i][j] i H[i][j] za sve i u {1, ..., t} i j u {1, ..., 8}

E. Računanje konačnih ostataka putem XOR operacije

Za svaki modul x_j $(j = 1,...,8)$ i odgovarajuće vrednosti $a_j, b_j, c_j, d_j, e_j, f_j, g_j, h_j, H_j$, konačni ostatak $ostaci_j$ definiše se kao:

$$ostaci_j = a_j \oplus b_j \oplus c_j \oplus d_j \oplus e_j \oplus f_j \oplus g_j \oplus h_j \oplus H_j$$

Ovde je $\oplus$ simbol za logičku operaciju *XOR*.

F. Pretvaranje rezultata u binarni zapis sa 8 bita

Svaki ostatak $ostaci_j$ konvertuje se u binarni zapis sa tačno 8 bita, dodavanjem vodećih nula ako je potrebno. Označavamo ovaj zapis sa b_ostaci_j :

$$b\_ostaci_j = bin\left(ostaci_j \right),$$

uz dodatak modećih nula da bi se postigla dužina od 8 bita.

G. Spajanje binarnih zapisa

Konačni heš *hesF* dobija se spajanjem binarnih zapisa

$$hesF = b\_ostaci_1 \,||\, b\_ostaci_2 \,||\, b\_ostaci_3 \,||\, b\_ostaci_4 \,||$$

$$||\, b\_ostaci_5 \,||\, b\_ostaci_6 \,||\, b\_ostaci_7 \,||\, b\_ostaci_8 .$$

Ovde || označava operaciju konkatenacije (spajanja) binarnih nizova.

H. Rezultat

Konačni heš *hesF* je binarni niz dužine 64 bita, koji predstavlja spoj svih pojedinačnih binarnih ostataka.

I. Algoritam II deo

```
1. Računanje konačnih ostataka putem XOR operacije

Za svaki j od 1 do 8 uradi sledeće:

   ostaci[j] = a[j] XOR b[j] XOR c[j] XOR d[j] XOR e[j] XOR f[j] XOR g[j] XOR h[j] XOR H[j]



// 2. Pretvaranje rezultata u binarni zapis sa 8 bita

Za svaki j od 1 do 8 uradi sledeće:

   b_ostaci[j] = konvertuj_u_binarni_sa_duzinom_8(ostaci[j])



// 3. Spajanje binarnih zapisa

binary_spojeno = ""

Za svaki j od 1 do 8 uradi sledeće:

   binary_spojeno = binary_spojeno + b_ostaci[j]

// 4. Dodela konačnog heša

hesF = binary_spojeno

// Funkcija za konvertovanje broja u binarni zapis sa dužinom od 8 bita

funkcija konvertuj_u_binarni_sa_duzinom_8(broj):

   binarni_zapis = konvertuj_u_binarni(broj) // Standardna binarna konverzija

   ako dužina(binarni_zapis) < 8:
      dodaj_vodeće_nule(binarni_zapis, 8 - dužina(binarni_zapis))

   povratna_vrednost = binarni_zapis
```

ZAKLJUČAK

Heš funkcije predstavljaju ključni alat u savremenoj kriptografiji, osiguravajući autentičnost, integritet i bezbednost podataka u različitim aplikacijama. RNS heš algoritam je zasnovan na modularnoj aritmetici, binarnim operacijama i višestepenoj obradi, što ga čini jednostavnim i efikasnim u generisanju heš vrednosti. Naročito je važno što ovakav pristup omogućava dobru otpornost na kolizije i prilagodljivost različitim veličinama ulaznih podataka, pa ovaj algoritam čini široko primnljivim u savremenim sistemima. Ovaj algoritam obezbedjuje visok stepen entropije u rezultujućoj heš vrednosti, kombinovanjem modularnih operacija sa binarnim formatiranjem i spajanjem. Ovo utiče na veću sigurnost u pogledu odbrane od pokušaja obrnutog inženjeringa ili pronalaženja kolizija. Jednostavna implementacija čini ovaj algoritam dodatno atraktivnim za različite softverske i hardverske platforme, bez potrebe za kompleksnim prilagođavanjem.

Za potpunu evaluaciju predloženog algoritma potrebno je analizirati njegovu otpornost na kriptoanalitičke napade, uključujući pretraživanje grubom silom, napade na kolizije i analizu distribucije heš vrednosti. Pored toga, njegove performanse i sigurnosne prednosti mogu se proceniti kroz poređenje sa postojećim standardima, poput SHA-2 i SHA-3, kao i kroz analizu njegove efikasnosti u različitim primenama [3]. Poseban fokus može biti stavljen na njegovu upotrebu u blokčejn tehnologijama, gde su brzina, sigurnost i otpornost na napade ključni faktori. Dodatno, optimizacija za hardverske arhitekture poput FPGA i ASIC mogla bi poboljšati njegovu efikasnost i omogućiti brže implementacije u resursno ograničenim okruženjima, što bi ga učinilo pogodnim za široku upotrebu u distribuiranim sistemima i decentralizovanim aplikacijama.

Možemo zaključiti da ovaj algoritam ima potencijal da nadje primenu u različitim oblastima, kao što su digitalni potpisi, autentifikacija korisnika, bezbedno čuvanje lozinki i integritet podataka. Uz dodatnu optimizaciju i analizu, mogao bi da predstavlja koristan doprinos oblasti kriptografije i bezbednosti informacija.

ZAHVALNICA

Autori izražavaju duboku zahvalnost Ministarstvu nauke, tehnološkog razvoja i inovacija Republike Srbije na podršci (Ugovor br. 451-03-65/2024-03/200123). Ovaj rad je delimično finansiran od strane Univerziteta u Prištini sa privremenim sedištem u Kosovskoj Mitrovici, Prirodno-matematičkog fakulteta, u okviru projekta IJ-2302 (Optimizacija neuronskih mreža).

LITERATURA

[1] Eldin, S. M. S., Abd El-Latif, A. A., Chelloug, S. A., Ahmad, M., Eldeeb, A. H., Diab, T. O., Al Sobky, W. I., & Zaky, H. N. (2023). Design and Analysis of New Version of Cryptographic Hash Function Based on Improved Chaotic Maps With Induced DNA Sequences. IEEE Access, 11, 101694–101709.

[2] Stallings, W. (2016). Network Security Essentials: Applications and Standards (6. izd.). Pearson Education.

[3] Menezes, A. J., Van Oorschot, P. C., & Vanstone, S. A. (1997). Handbook of Applied Cryptography. Boca Raton, FL: CRC Press.

[4] Hassan, M., Vliegen, J., Picek, S., & Mentens, N. (2024). A Systematic Exploration of Evolutionary Computation for the Design of Hardware-oriented Non-cryptographic Hash Functions. Proceedings of the Genetic and Evolutionary Computation Conference, 1255–1263.

[5] Yang, Y., & Zhang, X. (2022). A Novel Hash Function Based on Multi-iterative Parallel Structure. Wireless Personal Communications, 127, 2979–2996.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] S, C., G K, S., Rangaiah, L., & R, S. (2024). Design and Implementation of an Efficient Collision Resistant Novel Hash Function for IoT. Tuijin Jishu/Journal of Propulsion Technology, 45(01).

ABSTRACT

In modern cryptography, hash functions are fundamental tools for compressing variable-length input data into fixed-length output. They are used for data authentication, digital signatures, key generation, and secure password storage. A hash based on RNS arithmetic uses modular arithmetic and binary operations to generate values by processing 64-bit message blocks through modules, combining results with XOR, and converting them to binary. The final hash is obtained by merging these binary values. The algorithm emphasizes security, collision resistance, and adaptability, providing a mathematical description and pseudocode for cryptographic applications.

DEVELOPMENT AND OPTIMIZATION OF HASH FUNCTIONS USING THE RESIDUE NUMBER SYSTEM

Milija Pavlović, Boris Damjanović, Negovan Stamenković