

Kriptografska zgoščevalna funkcija JH

Ivan Verdonik¹

¹Fakulteta za gradbeništvo, Univerza v Mariboru

E-pošta: ivan.verdonik@um.si

Cryptographic hash function JH

Cryptographic hash functions are very important elements in contemporary cryptography. Their most important usage is manipulation detection during data transfer. Main construction method for these functions is still old Merkle-Damgard construction. However, there are improvements like HAIFA and also sponge functions. For transforming arbitrary long messages into fixed size output, compression functions are used. The most known are Davies-Meyer, Matyas-Meyer-Oseas and Miyaguchi-Preneel. Cryptographic hash functions must mandatory resist at least to collision, pre-image and secondary pre-image attacks. Function JH was SHA-3 competition finalist. Competition was announced because the most widely used cryptographic hash functions (SHA-1, MD5) are vulnerable (especially MD5 - collision) and SHA-2 is constructed similarly to SHA-1. But now SHA-2 was found to be stronger than it was thought.

1 Uvod

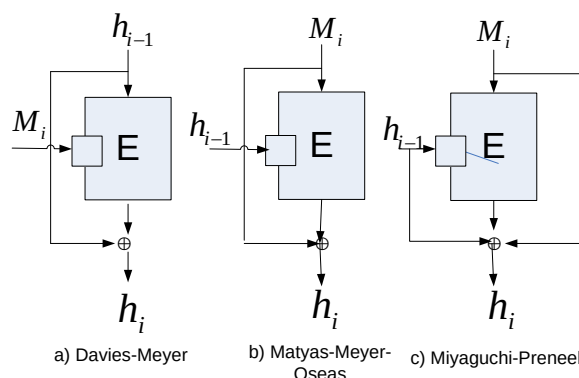
Kriptografske zgoščevalne funkcije so nepogrešljiv del šifriranja. Z njimi pridobimo neke vrste prstni odtis (elektronske) vsebine, besedila, datotek itd. Kot take so med drugim uporabljena v elektronskih podpisih, pri postopkih overjanja, generiranju psevdonaključnih števil, hranjenju gesel itd.

V zadnjem desetletju so odkrili več ranljivosti v največ uporabljenih kriptografskih zgoščevalnih funkcijah (MD5 in SHA-1 [3]). Edina varna funkcija je preostala SHA-2. Ker pa je družina kriptografskih zgoščevalnih funkcij SHA-2 zgrajena podobno kot SHA-1, je leta 2007 NIST razpisal SHA-3 natečaj [6] za izbiro nove kriptografske zgoščevalne funkcije. Raziskovalci so prijaviili več kot 50 idej. Po več letih preverjanja se je pet funkcij prebilo v finale, to so bile: Blake, Groestl, JH, Keccak in Skein. Septembra 2012 je NIST izbral zmagovalko, funkcijo Keccak. Ironija je, da se je družina funkcij SHA-2 zdaj pokazala za varnejšo, kot se je mislilo in SHA-3 trenutno (2014) služi bolj za rezervno.

2 Kriptografske zgoščevalne funkcije

Pri kriptografski funkciji moramo vsebino poljubne dolžine (npr. do $2^{64} - 1$) v fiksno izhodno dolžino npr. 256 bitov. K temu pristopamo tako, da vhod razdelimo na bloke fiksne dolžine in jih obdelujemo enega za drugim, to je tako imenovana Merkle-Damgardova

konstrukcija [7]. Pri njej je zanimiva varnostna vrzel, namreč če zadnji blok ni poln, funkcija ni varna ne glede na vse ostalo. Zato mora biti zadnji blok dopolnjen (to je t.i. Merkle-Damgard Strengthening [7]). Del funkcije, ki procesira vhodne bloke je kompresijska funkcija, znotraj nje pa imamo funkcijo runde (to je običajno (prirejani) simetrični bločni šifrirni algoritem) [8]. Varnostno občutljive so tudi kompresijske funkcije, glede na njihovo konstrukcijo. Največ uporabljamo konstrukcije: Davies-Meyer, Matyas-Meyer-Oseas in Miyaguchi-Preneel [9], ki so preverjeno varne (Glej Sliko 1).

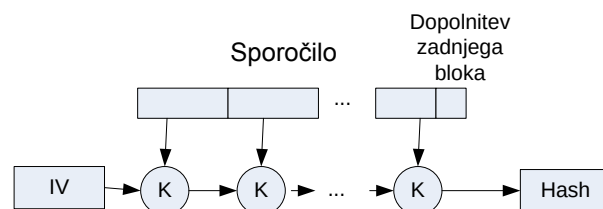


Slika 1: Največ uporabljane kompresijske funkcije

Osnovna Merkle-Damgard konstrukcija je predstavljena v Enačbi 1 in Sliki 2:

$$C_{MD} : \{0,1\}^h \times \{0,1\}^n \rightarrow \{0,1\}^h \quad (1)$$

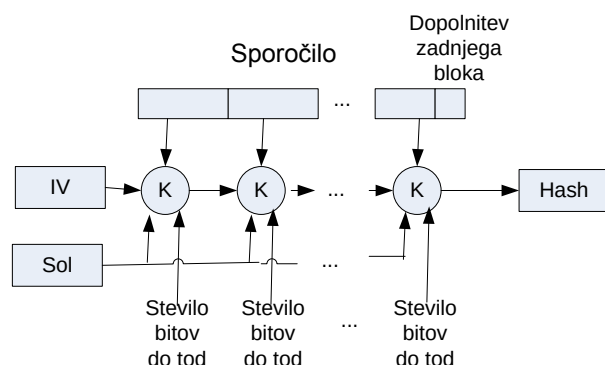
Binarni niz dolžine h (trenutno notranje stanje) kombiniramo z binarnim sporočilom dolžine n (blokom) in dobimo novo notranje stanje (oziroma zgoščeno vrednost) dolžine h bitov [7].



Slika 2: Konstrukcija Merkle-Damgard

Dodatno varnost omogoča HAIFA konstrukcija kompresijske funkcije [2] (glej Sliko 3). Z njo k vhodu dodamo še t.i. sol (salt, to je/so fiksna(e) vrednost(i) znana(e) le napravi(am), ki računajo zgoščeno vrednost,

pogosto kar vnaprej pripravljen(a)(e) konstant(a)(e)) in dinamični števec (običajno je to število do sedaj sprocesiranih bitov v sporočilu).



Slika 3: Konstrukcija HAIFA

HAIFA konstrukcija je predstavljena še v Enačbi 2:

$$C_{HAIFA} : \{0,1\}^h \times \{0,1\}^n \times \{0,1\}^b \times \{0,1\}^h \rightarrow \{0,1\}^h \quad (2)$$

Binarni niz dolžine h (trenutno notranje stanje) kombiniramo z binarnim sporočilom dolžine n (blokom), stanjem števca dolžine b bitov in soljo (salt) dolžine h bitov in dobimo novo notranje stanje (oziroma zgoščeno vrednost) dolžine h bitov[2].

Novo notranje stanje (ali na koncu zgoščena vrednost) je tako izračunana, kot kaže Enačba 3 [2]:

$$h_i = (h_{i-1}, M_i, \#b, s) \quad (3)$$

Operacija s predhodnim notranjim stanjem (trenutno zgoščeno vrednostjo), trenutnim blokom sporočila, števcem bitov in soljo tvori novo notranje stanje oz. zgoščeno vrednost.

Kriptografske zgoščevalne funkcije morajo biti odporni vsaj na generične napade [4]. To so kolizija (collision), napad na predpodobo (preimage) in napad na sekundarno predpodobo (second preimage) [4]. Pri koliziji skuša napadalec odkriti dve različni sporočila, ki se pretvorita v isto zgoščeno vrednost. Pri napadu na predpodobo skuša napadalec iz zgoščene vrednosti (rezultata) odkriti vhodno sporočilo. Pri napadu na sekundarno predpodobo, pa pri izbranem sporočilu odkriti drugo, različno sporočilo, ki ima isto zgoščeno vrednost, kot izbrano. Zato da je funkcija varna ne smejo obstajati napadi, ki bi bili učinkovitejši od napadov z golo silo (brute force). Za kolizijo je to $2^{n/2}$, za predpodobo in sekundarno predpodobo pa 2^n . Pri tem je n dolžina zgoščene vrednosti v bitih.

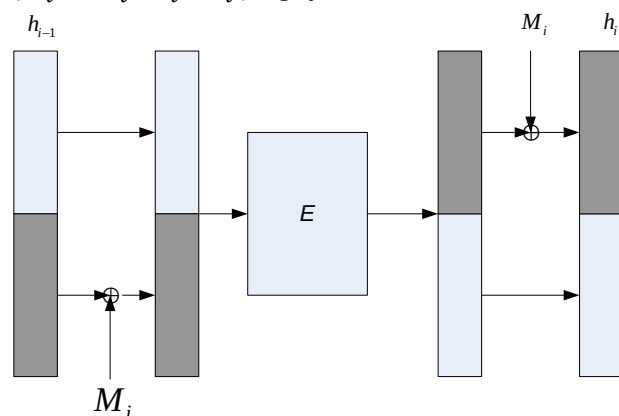
3 JH

Kriptografsko zgoščevalno funkcijo JH je ustvaril Hongjun Wu iz Institute for Infocomm Research Singapore. Ta funkcija ima enostavno implementacijo tako programsko, kot strojno ter tako na 1-bitnem, kot

na 128-bitnem procesorju. Večina vsebine te sekcije izhaja iz **Napaka! Vira sklicevanja ni bilo mogoče najti.** Ideja na kateri je zgrajen JH je posplošitev bločnega simetričnega šifrnega algoritma AES iz dveh dimenzij v več dimenzij d (na primer v 8- dimenzij). Nadaljnja izboljšava je gradnja kompresijske funkcije iz velikega bločnega šifrnega algoritma s ključem iz konstante (bijektivna funkcija). Velikost šifrnega bloka in zgoščene vrednosti je $2n$ bitov, medtem ko sta velikosti bloka sporočila in njegov izveček (digest) dolga n bitov.

Sporočilo je dopolnjeno na večkratnike 512 bitov. Na konec sporočila dodamo eno bitno enico, kateri sledi $383 - (dolžina\ sporočila\ v\ bitih\ po\ modulu\ 512)$ bitov ničel ter 128 bitna vrednost dolžine sporočila. Sporočilo je kodirano v obliki BigEndian.

To tako dopolnjeno sporočilo razbijemo v N 512 bitnih blokov, ki jih nadalje razdelimo v 4 128-bitne besede $(M_i^0, M_i^1, M_i^2, M_i^3)$ – glej Sliko 4.



Slika 4: Kompresijska funkcija JH

V kriptografski zgoščevalni funkciji JH so združene najboljše značilnosti funkcij Serpent (SPN, bit-slice implementation) [5][10] in AES (SPN and MDS code)[11].

JH vsebuje naslednje funkcije:

Zamenjave (S-boxes)

JH uporablja dve tabeli zamenjav, eno za lihe runde in eno za sode (Glej Tabela 1). Each S-box consists of 4x4 bits.

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

Tabela 1: Tabela zamenjav

Linearna transformacija (Linear transformation) L

Ta linearna transformacija izvaja MDS (Maximum Distance Separable) na polju $GF(2^4)$ z množenjem binarnih polinomov po modulu s polinomom $x^4 +$

$x + 1$ (to množenje označimo z $\bullet$). Če so A, B, C, D štiribitne besede, potem:
 $(C, D) = L(A, B) = (5 \bullet A + 2 \bullet B, 2 \bullet A + B)$

$$A = A^0 x^3 + A^1 x^2 + A^2 x + A^3$$

$$2 \bullet A = A^1 x^3 + A^2 x^2 + (A^0 + A^3) x + A^0$$

$$B = P'_d(A)$$

$$D^0 = B^0 \oplus A^1; \quad D^1 = B^1 \oplus A^2;$$

$$D^2 = B^2 \oplus A^3 \oplus A^0; \quad D^3 = B^3 \oplus A^0;$$

$$C^0 = A^0 \oplus D^1; \quad C^1 = A^1 \oplus D^2;$$

$$C^2 = A^2 \oplus D^3 \oplus D^0; \quad C^3 = A^3 \oplus D^0;$$

Permutacija (Permutation) P_d

Permutacija P_d je naslednja operacija, ki jo izvaja JH. Sestavljena je iz treh pod-permutacij (π_d, P'_d in Φ_d) nad 2^d vhodnimi elementi.

1. Permutacija π_d $B = \pi_d(A)$ je:

$$b_{4i+0} = a_{4i+0} \quad \text{for } i = 0 \text{ to } 2^{d-2} - 1;$$

$$b_{4i+1} = a_{4i+1} \quad \text{for } i = 0 \text{ to } 2^{d-2} - 1;$$

$$b_{4i+2} = a_{4i+3} \quad \text{for } i = 0 \text{ to } 2^{d-2} - 1;$$

$$b_{4i+3} = a_{4i+2} \quad \text{for } i = 0 \text{ to } 2^{d-2} - 1;$$

2. Permutacija P'_d $B = P'_d(A)$ je:

$$b_i = a_{2i} \quad \text{for } i = 0 \text{ to } 2^{d-1} - 1;$$

$$b_{i+2^{d-1}} = a_{2i+1} \quad \text{for } i = 0 \text{ to } 2^{d-1} - 1;$$

3. Permutacija Φ_d $B = \Phi_d(A)$ je:

$$b_i = a_i \quad \text{for } i = 0 \text{ to } 2^{d-1} - 1;$$

$$b_{2i+0} = a_{2i+1} \quad \text{for } i = 2^{d-2} \text{ to } 2^{d-1} - 1;$$

$$b_{2i+1} = a_{2i+0} \quad \text{for } i = 2^{d-2} \text{ to } 2^{d-1} - 1;$$

Glavna permutacija P_d je $P_d = \Phi_d \circ P'_d \circ \pi_d$

3.1 Funkcija runde (Round function) R_d

Funkcija runde je sestavljena iz tabela zamenjav (S-box), linearne transformacije in permutacije P_d . Vhod in izhod funkcije runde sta 2^{d+2} bitov dolga. R_d vsebuje tudi konstante rund $C_r^{(d)}$. R_d je sestavljena tako: $B = R_d(A, C_r^{(d)})$

$$\text{for } i = 0 \text{ to } 2^d - 1,$$

{

$$\text{if } C_r^{(d),i} = 0, \text{ then } v_i = S_0(a_i);$$

$$\text{if } C_r^{(d),i} = 1, \text{ then } v_i = S_1(a_i);$$

}

$$(w_{2i}, w_{2i+1}) = L(v_{2i}, v_{2i+1}) \quad \text{for } 0 \leq i \leq 2^{d-1} - 1;$$

$$(b_0, b_1, \dots, b_{2^d-1}) = P_d(w_0, w_1, \dots, w_{2^d-1});$$

3.2 Bijektivna funkcija (Bijective function) E_d

Ta funkcija je zgrajen na AES zasnovi posplošeni na več dimenzij (na primer 8) z uporabo MDS in SPN na d -dimenzionalni tabeli. Če imamo 8 dimenzij, se funkcija runde na isti dimenziji ponovi z vsako osmo iteracijo (funkcija runde se premika po dimenzijah). E_d ima $5(d-1)$ R_d rund ter dodatno tabelo zamenjav (S-box). Če je Q_r 2^{d+2} -bitna beseda in je $q_{r,i}$ 4-bitna beseda in je R_d^* funkcija R_d brez linearne transformacije in permutacije, potem je:

$$B = E_d(A)$$

1. grupiraj bite A v 2^d 4-bitne elemente za Q_0 ;
2. for $r = 0$ to $5(d-1) - 1$, $Q_{r+1} = R_d(Q_r, C_r^{(d)})$;
3. $Q_{5(d-1)+1} = R_d(Q_{5(d-1)}, C_{5(d-1)}^{(d)})$;
4. Razgrupiraj 2^d 4-bitne elemente v $Q_{5(d-1)+1}$ za B ;

3.3 Kompresijska funkcija (Compression function) F_d

Ta funkcija temelji na bijektivni funkciji E_d in iz predhodne zgoščene vrednosti in trenutnega bloka sporočila (dolžine 2^{d+1}) konstruira naslednjo zgoščeno vrednost (dolžine 2^{d+2}). Posplošen načrt algoritma je sledeč:

$$h_i = F_d(h_{i-1}, M_i)$$

Kompresijska funkcija F_d za $d=8$

$$A^j = h_{(i-1)j} \oplus M_{ij}; \text{ for } 0 \leq j \leq 511;$$

$$A^j = h_{(i-1)j} \text{ for } 512 \leq j \leq 1023;$$

$$B = E_8(A);$$

$$h_{ij} = B^j; \text{ for } 0 \leq j \leq 511;$$

$$h_{ij} = B^j \oplus M_{ij-512}; \text{ for } 512 \leq j \leq 1023;$$

Začetna zgoščena vrednost h_0 je določena z dolžino izvlečka (digest size). Prva dva bajta h_{-1} vsebujeta dolžino izvlečka. Postopek izračuna končne zgoščene vrednosti je naslednji:

For $i = 1$ to N

$$h_i = F_8(h_{i-1}, M_i)$$

Na koncu pridobimo izvleček sporočila, tako da odrežemo zadnjih (224, 256, 384 ali 512) bitov iz

končne zgoščene vrednosti h_N .

Najbolj inovativna značilnost kriptografske zgoščevalne funkcije JH je večdimenzionalnost notranjega stanja, pa tudi različna S-box substitucija za lihe in sode runde. Uporablja elemente dveh simetričnih bločnih šifirnih algoritmov: AES in Serpent.

Zaključek

Kriptografske zgoščevalne funkcije so zelo pomemben del računalniških šifirnih algoritmov, imenujejo jih tudi delovni konji šifriranja. Njihov razvoj se nadaljuje predvsem zaradi slabosti starih funkcij (MD5, SHA-1). Enako pomemben, kot razvoj novih funkcij je tudi kriptanaliza obstoječih. Najpomembnejši metodi sta linearna kriptanaliza in diferencialna kriptanaliza, v času SHA-3 natečaja, je nastalo največ novih zgoščevalnih funkcij, pa tudi največ novih metod kriptanalize.

Razvoj šifirnih algoritmov je že precej daleč in sedaj, ko imamo še nove kriptografske zgoščevalne funkcije iz SHA-3 natečaja, zna biti, da bodo sedanje funkcije (Keccak (SHA-3), AES, RSA) vzdržale še več desetletij. Glavna nevarnost je lahko Shor-ov postopek kriptanalize asimetričnih šifirnih algoritmov (RSA, diskretni logaritem) s primerno močnim kvantnim računalnikom.

Literatura

- [1] Hongjun Wu: The Hash Function JH, September 2009
- [2] Eli Biham, Orr Dunkelman: A Framework for Iterative Hash Functions - HAIFA, 2008
- [3] [Xiaoyun Wang](#), Yiqun Lisa Yin and Hongbo Yu: Finding Collisions in the Full SHA-1, Crypto 2005
- [4] J Upadhyay: Generic Attacks on Hash Functions
- [5] Ross Anderson, Eli Biham, Lars Knudsen Serpent
- [6] NIST: Cryptographic Hash Algorithm Competition, <http://csrc.nist.gov/groups/ST/hash/sha-3/index.html>, 2007
- [7] I. Damgard: A Design Principle for Hash Functions Crypto 89, 1989
- [8] B. Preneel, R. Govaerts and J. Vandewalle: Hash Functions Based on Block Ciphers, Crypto 93, 1993
- [9] J. Coron, Y. Dodis, C. Malinaud, P. Puniya: Merkle-Damgard Revisited: How to Construct a Hash Function
- [10] Ross Anderson, Eli Biham, Lars Knudsen, Serpent: A Proposal for the Advanced Encryption Standard
- [11] Joan Daemen, Vincent Rijmen: AES proposal: Rijndael, 1998