Zgoščevalna funkcija z diskretnim logaritmom

Varnost Chaum, Van Heijst in Pfitzmannove zgoščevalne funkcije je zasnovana na varnosti diskretnega logaritma.

Ni dovolj hitra, da bi jo uporabljali v praksi, je pa zato vsaj primerna za študij varnosti.

Naj bosta p in $q=(p-1)/2$ veliki praštevili, α in β pa dva primitivna elementa v $\mathbb{Z}_p$, za katera je vrednost $\log_\alpha\beta$ zasebna.

Zgoščevalno funkcijo

$$h\ :\ \{0,\dots,q-1\}\times\{0,\dots,q-1\}\longrightarrow\mathbb{Z}_p\backslash\{0\}$$

definirajmo z

$$h(x_1,x_2)=\alpha^{x_1}\beta^{x_2} \bmod p.$$

Pokazali bomo, da je ta funkcija **krepko brez trčenj** (strongly collision-free).

Predpostavimo obratno: za $(x_1,x_2)\neq(x_3,x_4)$ velja

$$h(x_1,x_2)=h(x_3,x_4)$$

oziroma

$$\alpha^{x_1}\beta^{x_2}\ \equiv\ \alpha^{x_3}\beta^{x_4}\pmod p$$

ali

$$\alpha^{x_1-x_3}\ \equiv\ \beta^{x_4-x_2}\pmod p.$$

Če je $d=D(x_4-x_2,p-1)$, potem imamo zaradi $p-1=2q$ natanko štiri možnost za d:

$$\{1,2,q,p-1\}.$$

Če je $d=1$, definiramo

$$y=(x_4-x_2)^{-1} \mod (p-1)$$

in dobimo

$$\beta\equiv\beta^{(x_4-x_2)y}\equiv\alpha^{(x_1-x_3)y}\pmod p,$$

iz česar znamo izračunati diskretni logaritem

$$\log_\alpha\beta\equiv(x_1-x_3)(x_4-x_2)^{-1} \mod (p-$$

Če je $d=2$, je $d=D(x_4-x_2,q)=1$, tako da lahko definiramo

$$y=(x_4-x_2)^{-1} \mod q$$

in dobimo za $(x_4-x_2)y=kq+1$, kjer je $k\in\mathbb{Z}$,

$$\beta^{(x_4-x_2)y}\equiv\beta^{kq+1}\equiv(-1)^k\beta\equiv\pm\beta\pmod p$$

zaradi $\beta^q\equiv-1\pmod p$.

Torej imamo

$$\pm\beta\equiv\beta^{(x_4-x_2)y}\equiv\alpha^{(x_1-x_3)y}\pmod p,$$

od koder znamo izračunati diskretni logaritem

$$\log_\alpha\beta\equiv(x_1-x_3)y \bmod (p-1)$$

ali pa

$$\log_\alpha\beta\equiv(x_1-x_3)y+q \mod (p-1).$$

Primer $d=q$ ni možen, saj iz

$$0\leq x_2\leq q-1 \quad\text{in}\quad 0\leq x_4\leq q-1$$

sledi

$$-(q-1)\leq x_4-x_2\leq q-1.$$

Končno si poglejmo še primer $d=p-1$, kar se lahko zgodi le za $x_2=x_4$. Potem velja

$$\alpha^{x_1}\ \equiv\alpha^{x_3}\pmod p$$

oziroma $x_1=x_3$ in $(x_1,x_2)=(x_3,x_4)$. Protislovje! ■

Razširitev zgoščevalne funkc

Doslej smo študirali zgoščevalne funkcije s domeno.

Sedaj pa pokažimo, kako lahko razširimo zgo funkcije, ki so krepko brez trčenj in imajo domeno, do zgoščevalnih funkcij, ki so krep trčenj in imajo neskončno domeno.

Tako bomo lahko podpisovali sporočila p dolžine.

Naj bo h^* zgoščevalna funkcija za katero je $|X| = \infty$.

Naj bo zgoščevalna funkcija $h : (\mathbb{Z}_2)^m \longrightarrow (\mathbb{Z}_2)^t$, $m \geq t+1$ krepko brez trčenj.

Potem bomo za $X = \cup_{i=m}^{\infty} (\mathbb{Z}_2)^i$ definirali zgoščevalno funkcijo

$$h^* : X \longrightarrow (\mathbb{Z}_2)^t,$$

ki bo tudi krepko brez trčenj.

Elementi množice X so zaporedja bitov, $|x|$, $x \in X$, pa naj predstavlja dolžino elementa x, tj. število bitov x-a. Z $x \,||\, y$ označimo spetje zaporedij x in y.

1.primer: $m \geq t+2$. Naj bo $|x| = n > m$ in x spetje $x_1 \,||\, x_2 \,||\, \dots \,||\, x_k$, kjer je

$$|x_1| = |x_2| = \cdots = |x_{k-1}| = m - t - 1$$

in $|x_k| = m - t - 1 - d$, pri čemer je $0 \leq d \leq m - t - 2$. Torej je

$$k = \left\lceil \frac{n}{m-t-1} \right\rceil.$$

Funkcijo $h^*(x)$ definiramo z naslednjim algoritmom:

1. **for** $i = 1$ **to** $k-1$ **do** $y_i = x_i$
2. $y_k = x_k \,||\, 0^d$
3. naj bo y_{k+1} število d v dvojiškem sistemu
4. $g_1 = h(0^{t+1} \,||\, y_1)$
5. **for** $i = 1$ **to** k **do** $g_{i+1} = h(g_i \,||\, 1 \,||\, y_{i+1})$
6. $h^*(x) = g_{k+1}$

Spetje $y_1 \,||\, y_2 \,||\, \dots \,||\, y_{k+1}$ smo dobili tako, da smo x_k-ju na desni pripeli d ničel, zaporedju y_{k+1} pa smo pripeli ničle na levi, tako da je $|y_{k+1}| = m - t - 1$.

Izrek: *Naj bo $h : (\mathbb{Z}_2)^m \longrightarrow (\mathbb{Z}_2)^t$, $m \geq t$*
zgoščevalna funkcija krepko brez trčenj.
Potem je zgoraj def. zgoščevalna funkcija
$h^* : X \longrightarrow (\mathbb{Z}_2)^t$ tudi krepko brez trčenj.

Dokaz: Predpostavimo, da smo našli $x \neq x'$
je $h^*(x) = h^*(x')$ in pokažimo, da lahko poi
polinomskem času trčenje za h. Vzamemo $|x|$

Naj bo $y(x) = y_1|| \dots ||y_{k+1}$ in $y(x') = y'_1|| \dots$
kjer sta x in x' dopolnjena z d ničlami po 2. k
so $g_1, \dots, g_{k+1}$ in $g'_1, \dots, g'_{j+1}$ zaporedoma v
ki jih izračunamo v korakih 4 in 5.

Če je $|x| \not\equiv |x'|$ (mod $m-t-1$), potem je $d \neq d'$ in od tod $y_{k+1} \neq y'_{j+1}$, torej dobimo trčenje iz

$$h(g_k||1||y_{k+1}) = g_{k+1} = h^*(x) = h^*(x') = g'_{j+1} = h(g'_j||1||y'_{j+1}).$$

Zato smemo sedaj privzeti, da $m-t-1$ deli $|x| - |x'|$. Iz $y_{k+1} = y'_{j+1}$, tako kot v prejšnjem primeru, sledi

$$h(g_k||1||y_{k+1}) = h(g'_j||1||y'_{j+1}).$$

Če je $g_k \neq g'_j$, smo našli trčenje, v nasprotnem primeru pa je

$$h(g_{k-1}||1||y_k) = g_k = g'_j = h(g'_{j-1}||1||y'_j).$$

Če na ta način s postopnim vračanjem ne pridemo do trčenja, dobimo na koncu

$$h(0^{t+1}||y_1) = g_1 = g'_{j-k} = \begin{cases} h(0^{t+1}||y'_1), & \text{če je } |x| = |x'| \\ h(g'_{j-k}||1||y'_1), & \text{sicer} \end{cases}$$

Za $|x| = |x'|$ oziroma $k = j$ nam da $y_1 \neq y'_1$ trčenje, sicer pa je $y_i = y'_i$ za $1 \leq i \leq k+1$. Od tod $y(x) = y(x')$, toda potem je $x = x'$, saj je preslikava $x \mapsto y(x)$ injekcija. Dobili smo protislovje s predpostavko, da je $x \neq x'$.

Končno v primeru, ko je $m-t-1$ deli $|x| - |x'| \neq 0$ dobimo trčenje, ker je $(t+1)$-vi bit v spetju $0^{t+1}||y_1$ enak 0, $(t+1)$-vi bit v spetju $g'_{j-k} \,||\, 1 \,||\, y'_1$ pa 1. ■

2.primer: $m = t+1$. Naj bo $|x| = n > m$ in definirajmo funkcijo f z $f(0) = 0$ in $f(1) = 01$.

Zgoščevalno funkcijo $h^*(x)$ definiramo z algoritmom:

1. $y = y_1 y_2 \dots y_k := 11||f(x_1)||f(x_2)|| \cdots ||f(x_n)$
2. $g_1 = h(0^t \,||\, y_1)$
3. **for** $i = 1$ **to** $k-1$ **do** $g_{i+1} = h(g_i \,||\, y_{i+1})$
4. $h^*(x) = g_k$

Funkcija $x \mapsto y = y(x)$ iz prvega koraka je injekcija.

Izrek: *Naj bo $h : (\mathbb{Z}_2)^{t+1} \longrightarrow (\mathbb{Z}_2)^t$*
zgoščevalna funkcija krepko brez trčenj.
Potem je zgoraj def. zgoščevalna funkcija
$h^* : X \longrightarrow (\mathbb{Z}_2)^t$ tudi krepko brez trčenj.

Dokaz: Predpostavimo, da smo našli $x \neq x'$
je $h^*(x) = h^*(x')$ in naj bo $y(x) = y_1|| \dots |$
$y(x') = y'_1|| \dots ||y'_{j+1}$, kjer sta x in x' dopoln
ničlami po 2. koraku.

Če je $k = j$, dobimo (kot pri prejšnjem dokazu
trčenje za zgoščevalno funkcijo h bodisi $y = y'$.
nam da $x = x'$, kar pa je protislovje!

Sedaj pa privzemimo, da je $k \neq j$ oziroma kar $j > k$. Če ne pride do trčenja, dobimo naslednje zaporedje enakosti:

$$y_k = y'_j, \quad y_{k-1} = y'_{j-1}, \ldots, y_1 = y'_{j-k+1},$$

kar pa ni možno, saj za $x \neq x'$ ter poljubno zaporedje z velja $y(x) \neq z||y(x')$, kajti zaporedni enici se pojavita izključno na začetku zaporedja $y(x)$.

Od tod zaključimo, da je h^* krepko brez trčenj. ■

Za računanje funkcije h^* smo uporabili funkcijo h kvečjemu

$$\left(1 + \left\lceil \frac{n}{m-t-1} \right\rceil\right) - \text{krat} \quad \text{za} \quad m \geq t+2$$

in

$$(2n+2) - \text{krat} \quad \text{za} \quad m = t+1.$$

Zgoščevalne funkcije iz kriptosistemov

Naj bo $(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ računsko varen kriptosistem in naj bo

$$\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_n,$$

kjer je $n \geq 128$, da bi preprečili napad z rojstnim dnevom.

Ta pogoj izključi **DES** (pa tudi DES-ov čistopis ni tako dolg kot DES-ov ključ).

Naj bo dano zaporedje

$x_1 \,||\, x_2 \,||\, \ldots \,||\, x_k$, kjer je $x_i \in (\mathbb{Z}_2)^n$, $1 \leq$

Če število bitov v zaporedju ne bi bilo ve
števila n, bi lahko dodali nekaj ničel...

Začnemo z neko začetno vrednostjo $g_0 = $ IV
(initial value) in nato konstruiramo zaporedje

$$g_i = f(x_i, g_{i-1}),$$

kjer je f šifrirna funkcija izbranega kripto
Potem je $h(x) = g_k$.

Definiranih je bilo veliko takih funkcij in mnoge med njimi so razbili (tj. dokazali, da niso varne), ne glede na to, ali je ustrezna šifra varna ali ne.

Naslednje štiri variacije pa zaenkrat izgledajo varne:

$$\begin{aligned} g_i &= e_{g_{i-1}}(x_i) \oplus x_i, \\ g_i &= e_{g_{i-1}}(x_i) \oplus x_i \oplus g_{i-1}, \\ g_i &= e_{g_{i-1}}(x_i \oplus g_{i-1}) \oplus x_i, \\ g_i &= e_{g_{i-1}}(x_i \oplus g_{i-1}) \oplus x_i \oplus g_{i-1}. \end{aligned}$$

Zgoščevalna funkcija MD4

Preglejmo nekaj hitrih zgoščevalnih funkcij.

MD4 je predlagal Rivest leta 1990, njeno izboljšano verzijo **MD5** pa leta 1991.

Funkcija **Secure Hash Standard** (**SHS**) iz leta 1992/93 je bolj komplicirana, a je zasnovana na istih principih. Njeno "tehnično napako" pa so odpravili šele leta 1994 (**SHA-1**).

Iz danega zaporedja bitov x najprej sestavimo zaporedje

$$M = M[0]\,M[1]\,\ldots\,M[N-1],$$

kjer je $M[i]$ 32-bitna beseda in je $N \equiv 0 \bmod 16$.

1. $d = (447 - |x|) \bmod 512$,
2. naj bo j binarna reprezentacija števila $x \bmod 2^{64}$, pri čemer je $|j| = 64$,
3. $M = x \,||\, 1 \,||\, 0^d \,||\, j$.

1. $A = 67452301$ (hex), $B = efcdab89$ (he
   $C = 98badcfe$ (hex), $D = 10325476$ (hex
2. **for** $i = 1$ **to** $N/16 - 1$ **do**
3. **for** $j = 0$ **to** 15 **do**
4. $X[j] = M[16i + j]$.
5. $AA = A, \ldots, DD = D$.
6. 1. krog
7. 2. krog
8. 3. krog
9. $A = A + AA, \ldots, D = D + DD$.

Osnovne operacije:

$X \wedge Y$	po bitih
$X \vee Y$	po bitih
$X \oplus Y$	XOR po bitih
$\neg X$	negacija
$X + Y$	seštevanje po modulu 2^{32}
$X \lll s$	ciklični pomik v levo za s mest

V *big-endian* arhitekturi (kot npr. Sun SPARC postaja) predstavimo število na naslednji način

$$a_1 2^{24} + a_2 2^{16} + a_3 2^8 + a_4,$$

v *little-endian* arhitekturi (kot npr. Intel 80xxx), ki jo je privzela funkcija MD4 pa z

$$a_4 2^{24} + a_3 2^{16} + a_2 2^8 + a_1.$$

V 1., 2. in 3. krogu funkcije **MD4** uporabimo zaporedoma funkcije f, g, in h, definirane spodaj.

$$\begin{aligned} f(X,Y,Z) &= (X \wedge Y) \vee ((\neg X) \wedge Z) \\ g(X,Y,Z) &= (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z) \\ h(X,Y,Z) &= X \oplus Y \oplus Z \end{aligned}$$

1. krog

1. $A = (A + f(B,C,D) + X[0]) \lll 3$
2. $D = (D + f(A,B,C) + X[1]) \lll 7$
3. $C = (C + f(D,A,B) + X[2]) \lll 11$
4. $B = (B + f(C,D,A) + X[3]) \lll 19$
5. $A = (A + f(B,C,D) + X[4]) \lll 3$
6. $D = (D + f(A,B,C) + X[5]) \lll 7$
7. $C = (C + f(D,A,B) + X[6]) \lll 11$
8. $B = (B + f(C,D,A) + X[7]) \lll 19$
9. $A = (A + f(B,C,D) + X[8]) \lll 3$
10. $D = (D + f(A,B,C) + X[9]) \lll 7$
11. $C = (C + f(D,A,B) + X[10]) \lll 11$
12. $B = (B + f(C,D,A) + X[11]) \lll 19$
13. $A = (A + f(B,C,D) + X[12]) \lll 3$
14. $D = (D + f(A,B,C) + X[13]) \lll 7$
15. $C = (C + f(D,A,B) + X[14]) \lll 11$
16. $B = (B + f(C,D,A) + X[15]) \lll 19$

2. krog

1. $A = (A + g(B,C,D) + X[0] + 5A827999) \lll 3$
2. $D = (D + g(A,B,C) + X[4] + 5A827999) \lll 5$
3. $C = (C + g(D,A,B) + X[8] + 5A827999) \lll 9$
4. $B = (B + g(C,D,A) + X[12] + 5A827999) \lll 13$
5. $A = (A + g(B,C,D) + X[1] + 5A827999) \lll 3$
6. $D = (D + g(A,B,C) + X[5] + 5A827999) \lll 5$
7. $C = (C + g(D,A,B) + X[9] + 5A827999) \lll 9$
8. $B = (B + g(C,D,A) + X[13] + 5A827999) \lll 13$
9. $A = (A + g(B,C,D) + X[2] + 5A827999) \lll 3$
10. $D = (D + g(A,B,C) + X[6] + 5A827999) \lll 5$
11. $C = (C + g(D,A,B) + X[10] + 5A827999) \lll 9$
12. $B = (B + g(C,D,A) + X[14] + 5A827999) \lll 13$
13. $A = (A + g(B,C,D) + X[3] + 5A827999) \lll 3$
14. $D = (D + g(A,B,C) + X[7] + 5A827999) \lll 5$
15. $C = (C + g(D,A,B) + X[11] + 5A827999) \lll 9$
16. $B = (B + g(C,D,A) + X[15] + 5A827999) \lll 13$

3. krog

1. $A = (A + h(B,C,D) + X[0] + 6ED9EBA1) \lll 3$
2. $D = (D + h(A,B,C) + X[8] + 6ED9EBA1) \lll 9$
3. $C = (C + h(D,A,B) + X[4] + 6ED9EBA1) \lll 11$
4. $B = (B + h(C,D,A) + X[12] + 6ED9EBA1) \lll 15$
5. $A = (A + h(B,C,D) + X[2] + 6ED9EBA1) \lll 3$
6. $D = (D + h(A,B,C) + X[10] + 6ED9EBA1) \lll 9$
7. $C = (C + h(D,A,B) + X[6] + 6ED9EBA1) \lll 11$
8. $B = (B + h(C,D,A) + X[14] + 6ED9EBA1) \lll 15$
9. $A = (A + h(B,C,D) + X[1] + 6ED9EBA1) \lll 3$
10. $D = (D + h(A,B,C) + X[9] + 6ED9EBA1) \lll 9$
11. $C = (C + h(D,A,B) + X[5] + 6ED9EBA1) \lll 11$
12. $B = (B + h(C,D,A) + X[13] + 6ED9EBA1) \lll 15$
13. $A = (A + h(B,C,D) + X[3] + 6ED9EBA1) \lll 3$
14. $D = (D + h(A,B,C) + X[11] + 6ED9EBA1) \lll 9$
15. $C = (C + h(D,A,B) + X[7] + 6ED9EBA1) \lll 11$
16. $B = (B + h(C,D,A) + X[15] + 6ED9EBA1) \lll 15$

Zgoščevalna funkcija **MD4** še ni bila razbita, vendar pa je ni težko razbiti, če bi opustili prvi ali pa zadnji krog.

Zato zgoščevalna funkcija **MD5** uporablja 5 krogov, a je 30% počasnejša (.9Mbytes/sec na SPARC-u).

Zgoščevalna funkcija **SHA** je še počasnejša (0.2Mbytes/sec na SPARC-u).
Opisali bomo le nekaj njenih modifikacij:

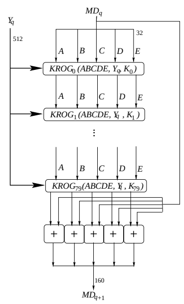
1. **SHS** privzame *big-endian* arhitekturo
little-endian.
2. **SHS** dobi 160-bitni rezultat (5 registrov)
3. **SHS** obdela 16 besed naenkrat, vendar ji
razširi v 80 besed, potem pa uporabi zapor
operacij na vsaki besedi.

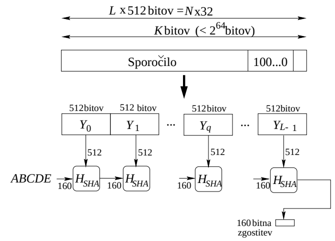
   $X[j] = X[j-3] \oplus X[j-8] \oplus X[j-14] \oplus$

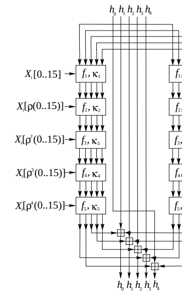
   za $16 \le j \le 79$.
4. **SHA-1** pa uporabi

   $X[j] = X[j-3] \oplus X[j-8] \oplus X[j-14] \oplus X[j$

   za $16 \le j \le 79$.







RIPEMD-160

HMAC

(Keyed-Hashing for Message Authentication)

Prednosti:

1. Kripto. zgoščevalne funkcije so v splošner v softwaru kot pa simetrične šifre (kot n DES).
2. Knjižnice zgoščevalnih funkcij so široko (medtem ko so bločne šifre, tudi l uporabljene samo za MAC, omejene izvoznih dovoljenj).

Za design objectives v HMAC algoritmu in njegovo varnost glej:

M. Bellare, R. Canetti in H. Krawczyk, CRYPTO'96

(in `http://wwwcse.ucsd.edu/users/mihir`),

ki ga trenutno poskušajo vključiti v IETF (Internet Engineering Task Force).

Časovne oznake/žigi (Timestamping)

Potrebujemo pričo o obstoju določenih podatkov ob določenem času, na primer na področju

- zaščite intelektualne lastnine (angl. intelectual property - IP), ali pa
- zanesljivega servisa za preprečevanje zanikanja (za dokaz, da je bil digitalni podpis generira v času veljavnosti ustreznega javnega ključa).

Če želi Bojan imeti dokaz o obstoju podatkov x ob nekem določenem času, potem naredi naslednje:

1. najprej izračuna zgostitev $z = h(x)$,
2. nato še zgostitev spoja
$$z' = h(z \,||\, \text{javna\_informacija}),$$
3. rezultat podpiše $y = \text{sig}_K(z')$, in
4. naslednji dan v časopisu objavi podatke
$$(z, \text{javna\_informacija}, y).$$

Časovne oznake s TS

Časovne žige omogoča pooblaščena organiza podpise, (angl. **Timestamper - TS**) elektronski notar (angl. trusted timestamping oz. center zaupanja (TTP, angl. trusted third

Bojan najprej izračuna

$$z = h(x), \quad y = \text{sig}_K(x)$$

in pošlje par (z, y) notarju TS,

ki doda še datum D in podpiše trojico $(z, y, I$

Zgornji algoritem je varen le pod pogojem, če je notar nepodkupljiv.

Potencialno se TS sooči z enormno odgovornostjo, če so časi kompromitirani.

Na primer, uporabnik lahko zanika vse podpise, ki jih je opravil kdaj koli.

Morda lahko nastane hujša škoda kot kompromitacija CA-jevega zasebnega ključa, o katerem bomo govorili v naslednjem poglavju.

Sicer pa si pomagamo z naslednjim algoritmom:

1. TS najprej izračuna
   $L_n = (t_{n-1}, \mathrm{ID}_{n-1}, z_{n-1}, y_{n-1}, h(L_{n-1}))$,
2. nato še $C_n = (n, t_n, z_n, y_n, \mathrm{ID}_n, L_n)$,
3. rezultat podpiše $s_n = \mathrm{sig}_{\mathrm{TS}}(h(C_n)))$, ter
4. pošlje $(C_n, s_n, \mathrm{ID}_{n+1})$ osebi ID_n.