nedeljainformatike v5.0

Heširanje: kako predvideti budućnost

Dimitrije Erdeljan

Matematička gimnazija

NEDELJA INFORMATIKE v5.0

21. decembar 2018.

Definicija

Heš funkcija: $H \in \{0,1\}^* \to \{0,1\}^l$

- Mapira proizvoljne podatke na konačan skup
- Uniformna distribucija izlaza
- (Idealno) efikasna za računanje

Za bezbedne (“kriptografske”) heš funkcije:

- Otporna na kolizije: teško je naći $x \neq y$ tako da $H(x) = H(y)$

Kriptografski heš?

- Za neke primene, dobra distribucija je dovoljna (poređenje stringova, heš tabele, ...)
- Šta ovde moze da bude problem?

[1] Loša ideja.

Kriptografski heš?

- Za neke primene, dobra distribucija je dovoljna (poređenje stringova, heš tabele, ...)
- Šta ovde moze da bude problem?
- Ako napadač bira podatke: *denial of service*
- Za druge primene, otpornost na kolizije je neophodna – na primer, identifikacija fajlova
  - Videćemo još primera kasnije
- MD5[1], SHA-1[1], SHA-2, SHA-3, ...

[1] Loša ideja.

Formalno

- $h \in \{0,1\}^* \to \{0,1\}^n$ je otporna na kolizije (*collision resistance*) ako ne postoji polinomni algoritam koji pronalazi par (x, y) za koji $H(x) = H(y)$ sa ne-zanemarivom verovatnoćom
- Tehnički detalji:
  - H mora da ima još jedan parametar s (tražimo $H_s(x) = H_s(y)$) – u suprotnom postoji $\mathcal{O}(1)$ resenje
  - “ne-zanemarivo”: manje od svakog polinoma $p(n)$

Druge osobine

- Otpornost na kolizije: teško je naći x i y gde $H(x) = H(y)$
- *Second-preimage resistance*: za dato x teško je naći y gde $H(x) = H(y)$
- *Preimage resistance*: za dato h teško je naći x gde $H(x) = h$

Traženje kolizija

Koliko je teško traženje kolizija?

- Dovoljno je ljudi u prostoriji da bi verovatnoća da dvoje dele rođendan bila 50%

Traženje kolizija

Koliko je teško traženje kolizija?

- Dovoljno je 23 ljudi u prostoriji da bi verovatnoća da dvoje dele rođendan bila 50%
- Za uniformno nasumične vrednosti iz $1, 2, \ldots, n$: $\mathcal{O}(\sqrt{n})$ $\rightarrow \mathcal{O}(2^{\frac{n}{2}})$ za traženje kolizija

$x = y = x_0$
repeat
  $x_{prev} = x, y_{prev} = H(y)$
  $x = H(x)$
  $y = H(H(y))$
until $H(x) \neq H(y)$
return (x_p, y_p)

Primene

- Kratak opis podataka (*message digest*): provera integriteta fajla, manje podataka za potpisivanje, ...
- Identifikator za objekat (npr. `git` fajlovi i commit-ovi)
- Čuvanje lozinki
- Potpisivanje (simetrično – *hash message authentication code*) podataka
- Dokaz da znamo neke podatke (obavezivanje)
- ...

Kompresujuća funkcija

Polazna tacka: kompresujuća funkcije $C \in \{0,1\}^{2n} \to \{0,1\}^n$

Kako izmisliti kompresujuću funkciju?

- Davis-Meyer konstrukcija – od funkcije za enkripciju $E(K, M)$:

$$C(K, M) = E(K, M) \oplus M$$

  - SHA-1, SHA-2, MD5, ...
- Postoji još sličnih metoda [2]
- “Ni iz čega”?

[2] Preneel, Bart, René Govaerts, and Joos Vandewalle. “Hash functions based on block ciphers: A synthetic approach.”

Merkle-Damgård konstrukcija

- Imamo: $C \in \{0,1\}^{2n} \to \{0,1\}^n$
- Pravimo: $H \in \{0,1\}^* \to \{0,1\}^n$
- Ideja: “nadovezujemo” blokove dužine n

$x = x_0||x_1||x_2||\ldots||x_B$
$x_{B+1} = len(x)$
$z_0 = 0^n$
for $i = 1..B+1$ **do**
    $z_i = C(z_{i-1}, x_i)$
end for
return z_{B+1}

Proširivanje

- Scenario: ne znamo x, ali imamo $H(x)$ (Merkle-Damgård heš H)
- Ako znamo dužinu x (obeležimo sa L), možemo izračunati heš za

$$x||0^{n-1-(L \bmod n)}||L||y$$

  za proizvoljan "nastavak" y!
- Jednostavno: znamo stanje nakon bloka sa L – produžimo konstrukciju

U praksi...[3]

- Flickr API 2009-e (“potpis” za zahtev sa parametrima $p_1 = v_1, p_2 = v_2, \ldots$, sa tajnim ključem K):

  sortiraj $(p_1, v_1), (p_2, v_2), \ldots$
  $x = \text{concat}(K, p_1, v_1, p_2, v_2, \ldots)$
  return MD5(x)

- Na primer:
  `http://flickr.com/services/auth/?api_key=`**`[key]`**
  `&perms=`**`read`**`&api_sig=`**`[sig]`**
- Imamo heš sledećeg stringa (nazovimo ga **`[sig]`**):
  `[KLJUČ]``api_key`**`[key]`**`perms`**`read`**`padding`

[3] Duong, Thai, and Juliano Rizzo. “Flickr’s API signature forgery vulnerability.”

U praksi...

- `[KLJUČ]api_key[key]permsreadpadding`
- Možemo produžiti ovaj string sa:
  `...api_key[key]requestevil`
- Novi potpis: `[sig2]`
- Sada imamo validan upit:
  `http://.../?a=pi_key...&api_key=[key]`
  `&request=evil&api_sig=[sig2]`

HMAC

- Bolji način za potpisivanje poruka heš funkcijom H: *Hash-based message authentication code*
- Cilj: potpisati poruku m ključem K
- $H(K||H(m))$ je konstantne dužine $\rightarrow$otporno na proširivanje

$$\text{HMAC}_K(X) = H(K \oplus o_{pad}||h(K \oplus i_{pad}||m))$$

Višestruke kolizije

- U opštem slucaju, naći “trostruku koliziju” $H(x) = H(y) = H(z)$ je teže nego “običnu” (*second-preimage*)

Višestruke kolizije

- U opštem slucaju, naći “trostruku koliziju” $H(x) = H(y) = H(z)$ je teže nego “običnu” (*second-preimage*)
- Za Merkle-Damgård heš funkcije, ako možemo naći jednu koliziju, možemo i **mnogo**:
  - Nađemo $H(x) = H(x')$
  - Postavimo inicijalnu vrednost (nulti blok) na $H(x)$ i nađemo $H_x(y) = H_x(y')$
  - Sad imamo $H(x||y) = H(x||y') = H(x'||y) = H(x'||y')$
  - ...

Višestruke kolizije

- U opštem slucaju, naći “trostruku koliziju” $H(x) = H(y) = H(z)$ je teže nego “običnu” (*second-preimage*)
- Za Merkle-Damgård heš funkcije, ako možemo naći jednu koliziju, možemo i **mnogo**:
  - Nađemo $H(x) = H(x')$
  - Postavimo inicijalnu vrednost (nulti blok) na $H(x)$ i nađemo $H_x(y) = H_x(y')$
  - Sad imamo $H(x||y) = H(x||y') = H(x'||y) = H(x'||y')$
  - ...
  - Za 2^m kolizija: $\mathcal{O}(m \cdot 2^{\frac{n}{2}})$
- Posledica: kolizije za $H_1(x)||H_2(x)$ u manje od $\mathcal{O}(2^{\frac{n_1+n_2}{2}})$

Obavezivanje

- Cilj: protokol gde dokazujemo da znamo poruku m koju ćemo objaviti kasnije
  - Na primer, garantovano poštena lutrija (nemoguće promeniti ishod)
- Jednostavno “rešenje”: objavimo $H(m)$
- Ako hoćemo da objavimo lažnu poruku m': *second-preimage* napad
- Trebalo bi nam $\mathcal{O}(2^n)$...

“Nostradamus napad”[4]

- Ideja: potrošimo puno truda **pre objavljivanja** da nađemo m za koje ćemo lako naći m'
- Generišemo 2^k inicijalnih vrednosti $x_1, x_2, \ldots$
- Nađemo kolizije (dužine jednog bloka) za parove $(x_1, x_2), (x_3, x_4), \ldots$
  - Sada imamo 2^{k-1} vrednosti

[4] Kelsey, John, and Tadayoshi Kohno. “Herding hash functions and the Nostradamus attack.”

“Nostradamus napad”[4]

- Ideja: potrošimo puno truda **pre objavljivanja** da nađemo m za koje ćemo lako naći m'
- Generišemo 2^k inicijalnih vrednosti $x_1, x_2, \ldots$
- Nađemo kolizije (dužine jednog bloka) za parove $(x_1, x_2), (x_3, x_4), \ldots$
  - Sada imamo 2^{k-1} vrednosti
- Ponovimo ovo dok ne dobijemo puteve dužine k od svake inicijalne vrednosti do nekog m_0

[4] Kelsey, John, and Tadayoshi Kohno. “Herding hash functions and the Nostradamus attack.”

“Nostradamus napad”[4]

- Ideja: potrošimo puno truda **pre objavljivanja** da nađemo m za koje ćemo lako naći m'
- Generišemo 2^k inicijalnih vrednosti $x_1, x_2, \ldots$
- Nađemo kolizije (dužine jednog bloka) za parove $(x_1, x_2), (x_3, x_4), \ldots$
  - Sada imamo 2^{k-1} vrednosti
- Ponovimo ovo dok ne dobijemo puteve dužine k od svake inicijalne vrednosti do nekog m_0
- Dodamo padding-blok na m_0 i objavimo dobijeno m
- Čekamo...

[4] Kelsey, John, and Tadayoshi Kohno. “Herding hash functions and the Nostradamus attack.”

“Nostradamus napad”

- ...dočekamo m' koje smo “predvideli”
- Tražimo p tako da je krajnje stanje (bez padding-a) za $m'||p$ jedno od $x_1, x_2, \ldots$

“Nostradamus napad”

- ...dočekamo m' koje smo “predvideli”
- Tražimo p tako da je krajnje stanje (bez padding-a) za $m'||p$ jedno od $x_1, x_2, \ldots$
- Našli smo x_i – imamo niz blokova $z_1, z_2, \ldots, z_k$ koji vodi od x_i do m

“Nostradamus napad”

- ...dočekamo m' koje smo “predvideli”
- Tražimo p tako da je krajnje stanje (bez padding-a) za $m'||p$ jedno od $x_1, x_2, \ldots$
- Našli smo x_i – imamo niz blokova $z_1, z_2, \ldots, z_k$ koji vodi od x_i do m
- Objavimo “originalnu” poruku $m'||p||z_1|| \ldots ||z_k||padding$
- Cena: $\mathcal{O}(2^{\frac{n+k}{2}})$ unapred i $\mathcal{O}(2^{n-k})$ nakon objavljivanja