HMAC - seminarska naloga pri predmetu KITK2

Andrej Tolič, 63040299, IŠRM

16.01.2011

Povzetek

Ena od osnovnih potreb v informacijski dobi je način, kako preveriti integriteto informacij, ki jih prenašamo po (ali hranimo na) nezanesljivih medijih. Rešitev tega problema so t.i. MAC (message authentication code). Poznamo več vrst MAC-ov. CBC-MAC npr. temelji na bločni šifri. HMAC je osnovan na zgoščevalnih funkcijah tipa Merkle-Damgard. Na začetku predstavimo osnove zgoščevalnih funkcij, nato pridemo do NMAC ter HMAC. Pogledamo varnost vključno z novejšim dokazom varnosti. Poleg tega predstavimo še alternativo, t.i. HMAC brez drugega ključa ter na koncu krajši opis protokolov, ki uporabljajo HMAC.

Kazalo

1 Uvod

V seminarski nalogi predstavimo HMAC, MAC mehanizem, ki temelji na zgoščevalnih funkcijah. Zgoščevalne funkcije kot take, nam omogočajo preverjanje integritete podatkov s pomočjo izračuna zgostitve (ang. hash) oz. „prstnega odtisa“ podatkov. Ne zagotavljajo pa nam avtentikacije in preprečevanja ponarejanja, saj bi lahko npr. ponarejevalec spremenil podatke in zraven še hash vrednost. Za take primere, se uporabljajo MAC-i (ang. message authentication code). MAC-e so na začetku gradili iz bločnih šifer, vendar so ti postopki ponavadi počasnejši za računanje in ponekod tudi omejeni zakonsko. Zato so avtorji v [2] predlagali uporabo kriptografskih zgoščevalnih funkcij kot osnovo za MAC. Takemu MAC pravimo HMAC, krajša definicija in analiza varnosti je na voljo tudi kot RFC[1].

V 2. razdelku predstavimo zgoščevalne funkcije, osnovne varnostne pojme, napad s paradoksom rojstnih dnevov in Merkle-Damgard konstrukcijo.

V 3. razdelku se lotimo MAC-ov in načinov vpeljave ključa v iterativne zgoščevalne funkcije.

V 4. razdelku pridemo do HMAC kot derivata NMAC. Predstavimo konstrukcijo, osnovno analizo varnosti, možne napade, novejši dokaz in spregovorimo malo o implementacijskih detajlih.

V 5. razdelku opišemo alternativo HMAC, in sicer t.i. HMAC brez drugega ključa oz. H^2-MAC. Pogledamo si konstrukcijo ter analizo, kot je predstavljena v [6].

6. razdelek na kratko opiše uporabo HMAC v protokolih kot sta IPsec in TLS.

2 Osnovno o zgoščevalnih funkcijah

2.1 Definicija zgoščevalne funkcije

Kriptografske zgoščevalne funkcije preslikajo podatke poljubne velikosti v vrednosti neke fiksne dolžine. Tem vrednostim pravimo **zgostitev** (ang. hash) ali pa izvleček sporočila(ang. message digest) saj podatkom pravimo tudi sporočilo (zaradi pogoste uporabe v integriteti komunikacij, čeprav gre lahko za poljubne podatke npr. zgostitev iz video datoteke). Ko zapišemo zgoščevalna funkcija, mislimo kriptografsko zgoščevalno funkcijo z neomejeno domeno (deluje nad podatki poljubne velikosti), za tisto z omejeno domeno pa bomo uporabili izraz kompresijska funkcija. Zgostitev je v trenutno uporabljanih funkcijah, kot sta npr. MD5 in SHA-1, tipično med 128 in 512 biti. V 5 finalistkah za novi hash standard, je dolžina izhoda tipično 224, 256, 384 ali 512 bitov[11, 12].

Naj bo h neka kriptografska zgoščevalna funkcija. Za potrebe integritete podatka x izračunamo njegovo zgostitev $y = h(x)$ in jo hranimo na varnem mestu. Če se je podatek spremenil v x' pričakujemo, da se je spremenila tudi zgostitev in pričakujemo seveda, da se kakršne koli korelacije med x in x' ne odražajo v pripadajočih zgostitvah. V tem primeru govorimo o fiksni zgoščevalni funkciji h, lahko pa bi imeli tudi družino funkcij H, izmed katere bi posamezne funkcije izbrali npr. na podlagi nekega ključa K. Tako funkcijo označimo s h_K. Ta način je uporaben zlasti, ko ne moremo varno shraniti zgostitve, ampak jo prenašamo npr. po nekem nevarnem komunikacijskem kanalu. Če povzamemo definicijo po [3]:

Definicija 1. *Družina zgoščevalnih funkcij je četverica* $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$*, tako da velja:*

1. $\mathcal{X}$ *je množica možnih sporočil*
2. $\mathcal{Y}$ *je končna množica možnih zgostitev*
3. $\mathcal{K}$ *je končna množica možnih ključev oz. prostor ključev*
4. *za vsak* $K \in \mathcal{K}$ *obstaja zgoščevalna funkcija* $h_K \in \mathcal{H}$*. Vsak* $h_K : \mathcal{X} \to \mathcal{Y}$*.*

Pri $\mathcal{X}$ gre lahko za končno ali neskončno množico, če je končna govorimo o kompresijski funkciji, v tem primeru gre tipično za sporočila dolžine 512 + velikostIzhoda(h) bitov. Zakaj je tako, bomo videli kasneje, ko bomo govorili o iterativnih zgoščevalnih funkcijah. Tam bomo načeloma govorili tudi o kompresijskih funkcijah z 2 vhodoma, zato uporaba znaka + pri opisu dolžine vhoda. Fiksne zgoščevalne funkcije si lahko predstavljamo kot družino z enim samim ključem.

2.2 Varnost zgoščevalnih funkcij

V mislih imamo zgoščevalne funkcije brez ključa. Definirajmo 3 ključne probleme varnosti.

Definicija 2 (**Praslika (ang. preimage)**). *Za dano zgoščevalno funkcijo $h : \mathcal{X} \to \mathcal{Y}$ in dan $y \in \mathcal{Y}$ poiščimo tak $x \in \mathcal{X}$, da je $h(x) = y$.*

Definicija 3 (**2. praslika (ang. 2nd preimage)**). *Za dano zgoščevalno funkcijo $h : \mathcal{X} \to \mathcal{Y}$ in dan $x \in \mathcal{X}$ poiščimo tak $x' \in \mathcal{X}$, da je $x' \neq x$ in $h(x') = h(x)$.*

Definicija 4 (**Trk (ang. collision)**). *Za dano zgoščevalno funkcijo $h : \mathcal{X} \to \mathcal{Y}$ poiščimo taka $x, x' \in \mathcal{X}$, da je $h(x') = h(x)$.*

Rečemo, da je funkcija odporna na trčenja, če je računsko izjemno težko poiskati dve ali več sporočil z isto zgostitvijo. Naj bo (ε, Q)-algoritem Las Vegas algoritem, katerega povprečna uspešnost (povprečno po vseh možnih zgoščevalnih funkcijah, njihovih vhodih in izhodih) je ε in ki naredi največ Q klicev zgoščevalni funkciji.

Algorithm 1 Find-Collision(h,Q)

naključno izberi $\mathcal{X}_0 \subseteq \mathcal{X}$, da je $|\mathcal{X}_0| = Q$
for all $x \in \mathcal{X}_0$ **do**
  $y_x \leftarrow h(x)$
end for
if $y_x = y_{x'}$ za nek $x' \neq x$ **then**
  return (x, x')
else
  return failure
end if

Algoritem 1 predstavlja preprost algoritem tipa Las Vegas za iskanje trkov. Naj bo $M = |\mathcal{Y}|$. Rojstnodnevni paradoks si v kontekstu algoritma 1 lahko predstavljamo, kot da je h funkcija, ki vrača rojstni dan oseb, torej $M = 365$. Paradoks pravi, da če v sobi „preverimo“ rojstni dan 23 oseb, bosta imeli 2 osebi isti rojstni dan z verjetnostjo večjo od 50%. Torej je v tem primeru približno $Q = 23$, $\varepsilon = 0.5$ in $M = 365$. Nas pa zanima analiza za splošne parametre. Velja sledeči izrek.

Izrek 1. *Za poljuben $\mathcal{X}_0 \subseteq \mathcal{X}$, da je $|\mathcal{X}_0| = Q$, je verjetnost uspeha algoritma 1:*

$$\varepsilon = 1 - \left(\frac{M-1}{M}\right)\left(\frac{M-2}{M}\right)\cdots\left(\frac{M-Q+1}{M}\right).$$

Dokaz 1. *Naj bo $\mathcal{X}_0 = \{x_1, x_2, \ldots, x_Q\}$. Za vsak $1 \leq i \leq Q$, naj bo E_i dogodek, da $h(x_i) \notin \{h(x_1), \ldots, h(x_{i-1})\}$. Upoštevajoč osnovne verjetnosti ugotovimo, da je $\mathbf{Pr}[E_1] = 1$ in*

$$\mathbf{Pr}[E_i | E_1 \wedge E_2 \wedge \ldots \wedge E_{i-1}] = \frac{M-i+1}{M},$$

kadar $2 \leq i \leq Q$. *Preko formule za pogojno verjetnost, vemo da velja:*

$$\mathbf{Pr}[E_1 \wedge E_2 \wedge \ldots \wedge E_Q] = \mathbf{Pr}[E_Q | E_1 \wedge E_2 \wedge \ldots \wedge E_{Q-1}] \cdots \mathbf{Pr}[E_2|E_1]\mathbf{Pr}[E_1]$$

in potem

$$\mathbf{Pr}[E_1 \wedge E_2 \wedge \ldots \wedge E_Q] = \left(\frac{M-1}{M}\right)\left(\frac{M-2}{M}\right)\cdots\left(\frac{M-Q+1}{M}\right).$$

Verjetnost vsaj enega trka je potem $1 - \mathbf{Pr}[E_1 \wedge E_2 \wedge \ldots \wedge E_Q]$. □

Z nekaj algebraične spretnosti, upoštevajoč da za male x velja $1 - x \approx e^{-x}$, lahko zgornjo verjetnost ε pretvorimo v $1-e^{\frac{-Q(Q-1)}{M}}$. Če sedaj Q izrazimo kot funkcijo ε in M in spet uporabimo nekaj poenostavitev, dobimo oceno:

$$Q \approx \sqrt{2M\ \ln\frac{1}{1-\varepsilon}}.$$

Podrobnosti pretvorbe so v [3, str. 126]. Če želimo 50% verjetnost odkritja trka, postavimo $\varepsilon = 0.5$ in dobimo $Q \approx 1.17\sqrt{M}$. Npr. za rojstnodnevni paradoks bi za $M = 365$ dobili $Q = 22.3$.

Ta ocena je pomembna, saj nam pove, da rojstnodnevni napad s pričakovano uspešnostjo 50% zahteva izračun zgostitve za nekaj več kot $\sqrt{M}$ naključnih sporočil. V praksi to pomeni, da se varnost zgoščevalne funkcije proti izčrpnem preiskovanju kvadratno koreni oz. razpolovi v bitih. Torej bi za varnost nekaj več kot 80 bitov potrebovali funkcijo, ki generira 160 bitne zgostitve. Tak primer je SHA-1. Varnost MD5 je iz vidika zgoraj prikazanega napada potem nekaj več kot 64 bitov.

Preprosta Las Vegas algoritma, ki bi reševala problem praslike in 2. praslike, sta računsko težja, saj je tam uspešnost glede na Q enaka

$$\varepsilon = 1 - (1 - \frac{1}{M})^Q$$

za prasliko, in

$$\varepsilon = 1 - (1 - \frac{1}{M})^{Q-1}$$

za 2. prasliko. V [3, str. 127] je pokazano, da se da problem trkov brez težav prevesti na problem 2. praslike. Algoritem 2 s pomočjo algoritma za prasliko Oracle-Preimage(h, y) rešuje problem trka:

Izrek 2. *Naj bo* $h : \mathcal{X} \to \mathcal{Y}$ *zgoščevalna funkcija in naj velja* $|\mathcal{X}| \geq 2|\mathcal{Y}|$. *Naj bo Oracle-Preimage* $(1, Q)$*-algoritem za reševanje praslike za dani* h. *Potem je algoritem 2* $(\frac{1}{2}, Q + 1)$*-algoritem za iskanje trkov za dani* h.

Dokaz izreka je v [3, str. 128]. Torej odpornost na trke implicira tudi odpornost na iskanje praslike in 2. praslike.

2.3 Iterativne zgoščevalne funkcije in Merkle-Damgard konstrukcija

Merkle in kasneje neodvisno Damgard, sta predstavila način za gradnjo kriptografskih zgoščevalnih funkcij, ki procesirajo sporočila poljubne dolžine na osnovi kompresijskih funkcij, torej funkcij, ki procesirajo krajša sporočila fiksne dolžine. Dokazala sta tudi, da je taka konstrukcija odporna

Algorithm 2 Collision-to-Preimage(h)

```
naključno izberi x ∈ 𝒳
y ← h(x)
if (Oracle-Preimage(h,y)=x′) AND (x′ ≠ x) then
  return (x, x′)
else
  return failure
end if
```

na trke, če je kompresijska funkcija odporna na trke. To je pomembno, saj omogoča razvijalcem, da se osredotočijo na kreiranje funkcij, ki delujejo na fiksni velikosti vhoda. Naj f predstavlja neko kompresijsko funkcijo, F pa zgoščevalno funkcijo za poljubno dolga sporočila. Funkcijo F sestavimo tako, da iterativno uporablja kompresijsko funkcijo f in s tem procesira sporočila poljubne dolžine. Način, kako točno sestavimo iterativno zgoščevalno funkcijo F iz kompresijske funkcije f, imenujemo Merkle-Damgard konstrukcija. Trenutno popularni zgoščevalni funkciji MD5 in SHA-1 sta tipa Merkle-Damgard.

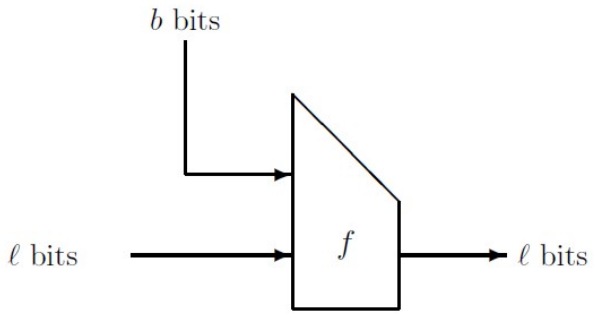
Kompresijska funkcija f sprejme 2 vhoda, spremenljivko za veriženje (ang. chaining variable) dolžine ℓ bitov, ter blok podatkov dolžine b bitov. Pri MD5 in SHA-1 je $b = 512$ in $\ell = 128$ pri MD5 ter $\ell = 160$ pri SHA1. V Merkle-Damgard konstrukciji mora biti ℓ enake dolžine kot izhod f. Slika 1 prikazuje kompresijsko funkcijo.

Iterativna konstrukcija (glej sliko 2) uporablja kot prvo spremenljivko za veriženje neko začetno vrednost IV dolžine ℓ bitov. Najprej neformalno predstavimo iterativno konstrukcijo po fazah.

- V prvi fazi se vhodno sporočilo na poseben način razširi do večkratnika števila b. Ta način ponavadi vsebuje dolžino vhoda. Temu se reče tudi Merkle-Damgard krepitev (ang. Merkle-Damgard strengthening) in je pomembno pri dokazu varnosti. To se vidi tudi na sliki 2), kjer $|x|$ kot vhod v zadnji klic prestavlja dolžino sporočila.

- V drugi fazi se potem izvede iteracija, kjer se na vsakem koraku po vrsti v kompresijsko funkcijo pošilja ℓ bitov spremenljivke za veriženje, v prvem koraku je to IV, kasneje pa zadnji izhod f. Poleg tega se na vhod pošlje trenutni blok (sporočilo po prvi fazi razdelimo na bloke dolžine b bitov) sporočila, pri čemer je zadnji blok, kot smo že omenili rezultat zakodiranja dolžine sporočila in dopolnitve do večkratnika b.

- V neobvezni tretji fazi se lahko zgodi še naknadno procesiranje. En tak primer je, da v



Slika 1: Kompresijska funkcija

končnem rezultatu zanemarimo nekaj zadnjih bitov.

Sedaj pa zgornji postopek formaliziramo. Naj bo sporočilo $x = x_1, x_2, \ldots, x_n$ dolžine n blokov, kjer je vsak blok dolg b bitov. Sledi še blok $x_{n+1} = |x|$, ki predstavlja dolžino sporočila x. Tipično gre za 64-bitno zakodiranje dolžine, pred tem pa pripnemo zaporedje 10^*, kjer je število ničel toliko, da je celoten x_{n+1} dolžine b. Zgostitev iterativne funkcije F na sporočilu x je potem $F(x) = h_{n+1}$ kjer velja:

$$\begin{aligned} h_0 &= \text{IV} \\ h_i &= f(h_{i-1}, x_i),\ 1 \leq i \leq n+1 \end{aligned}$$

3 MAC ter vpeljava ključa v zgoščevalne funkcije

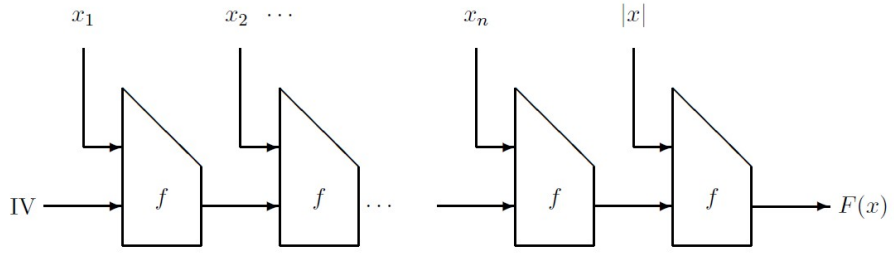
Predstavimo osnovne ideje vpeljave ključa preko inicializacijskega vektorja (IV) zgoščevalne funkcije. Pokažemo zakaj to ni v redu. Pridemo do pojmov varnega MAC, šibke odpornosti proti trkom. Nato opišemo drugačno idejo vpeljave ključa preko inicializacijskega vektorja (IV) zgoščevalne funkcije z uporabo gnezdenja, torej NMAC kot teoretični temelj HMAC.

3.1 Uvod

MAC (ang. message authentication code) je angleška kratica, ki pomeni koda za avtentikacijo sporočil. Kot smo nakazali že v pri predstavitvi zgoščevalnih funkcij, nam te same nudijo le preverjanje integritete, in še to pri pogojih, da potencialni napadalec nima dostopa do zgostitve - torej dostopa v smislu konkretnega prostora, kjer hranimo naš izračun zgostitve, zato da nam je ne bi spremenil. Po drugi strani pa nam MAC omogoča integriteto in avtentikacijo (kdo je avtor sporočila) z uporabo skrivnega ključa. Na začetku so bili MAC zgrajeni predvsem na osnovi bločnih šifer, potem pa so se raziskovalci obrnili h kriptografskim zgoščevalnim funkcijam. Prednost teh je hitrost (zlasti v programskih implementacijah) in pa lažja dostopnost, saj ni raznih izvoznih omejitev s strani vlad, kot to velja za bločne šifre. Glavna slabost v primerjavi z bločnimi šiframi pa je, da zgoščevalne funkcije tipično niso zgrajene za uporabo skrivnega ključa. Zato je treba premisliti, kako varno in učinkovito vpeljati ključ v zgoščevalne funkcije, da bi dobili MAC. Ravno to je pravzaprav bistvo HMAC.

3.2 Definicija MAC, varnost MAC s poskusi vpeljave ključa

MAC je neka funkcija, ki sprejme skrivni ključ k ter sporočilo m in vrne oznako (ang. tag) $\text{MAC}_k(m)$. Nasprotnik vidi pare sporočil in pripadajočih oznak: $(m_1, a_1), \ldots, (m_q, a_q)$. Tukaj



Slika 2: Iterativna konstrukcija F na osnovi f

izraz „nasprotnik vidi“ pomeni, da jih ima na voljo. Lahko jih je dobil s prestrezanjem komunikacije, lahko jih je zahteval sam po principu napada z izbranim čistopisom ipd.

Rečemo, da je nasprotnik razbil MAC, če mu uspe za neko sporočilo $m \notin m_1, \ldots, m_q$, generirati veljavno oznako $a = MAC_k(m)$ glede na nek fiksni njemu neznani ključ k. Očitno lahko nasprotnik razbije MAC, če pozna ključ, a načeloma govorimo o načinih uporabe znanih parov za kreiranje novih sporočil in pripadajočih oznak.

O varnosti MAC bomo govorili v smislu verjetnosti uspeha generičnega napada v okviru časovnih omejitev t in q veljavnih parov (sporočilo-oznaka).

Definicija 5. *Predpostavimo, da napadalec nima dostopa do ključa k, je omejen s časom (številom operacij) t in ima na voljo največ q veljavnih parov (ali pa q poizvedb) s sporočili dolžine največ L. Rečemo, da je nek MAC* **(ε, t, q, L)-varen MAC**, *kadar verjetnost uspešnega napada za opisanega nasprotnika ni večja od ε.*

Naj bo naš prvi poskus vpeljave ključa v iterativno zgoščevalno funkcijo F preko inicializacijskega vektorja, in sicer $IV = k$. To je konstrukcija pripenjanja pred sporočilo (ang. prepend-only construction). O njej in napadu nanjo bo še govora v razdelku 4.4. Naš MAC je potem kar:

$$MAC_k(m) = F_k(m)$$

V tem primeru je potem glede na definicije iz prejšnjega razdelka $|k| = l$, če je ključ daljši, ga po nekem vnaprej dogovorjenem in znanem postopku skrajšamo na ℓ bitov. Zaradi poenostavitve naj bo dolžina sporočila večkratnik števila b. Kompresijska funkcija f naj slika iz $\{0,1\}^{l+b}$ v $\{0,1\}^l$.

Predpostavimo, da ima nasprotnik na voljo veljaven par (m, a). Nato zgenerira poljuben niz m' dolžine b bitov. Sedaj poskuša izračunati oznako za niz $m \parallel m'$, kjer $\parallel$ pomeni spoj nizov. Zaradi iterativne konstrukcije, pomeni dodatni blok m' samo še eno izvajanje kompresijske funkcije f z vhodoma m' in $F_k(m)$. Torej:

$$MAC_k(m \parallel m') = F_k(m \parallel m') = f(F_k(m), m') = f(MAC_k(m), m')$$

To je torej zelo realen in računsko nezahteven napad. Tudi, če bi uporabili kakšno razširitev v predprocesiranju, bi se še vedno dalo preprosto z nekajkratno uporabo kompresijske funkcije izračunati oznako novega sporočila, ki ga dobimo tako, da staremu pripnemo niz dolžine b. V [3, str. 140] najdemo primer. Enak napad je možen, če bi pripeli osnovnemu sporočilu sporočilo iz več blokov, preprosto bi se večkrat uporabila f. Zgornja konstrukcija se lahko simulira tudi s pripenjanjem ključa pred vhodne podatke. Napad je znan tudi kot razširitveni napad (ang. extension attack), saj lahko za razširitve sporočil, za katere nasprotnik pozna njihovo oznako, preprosto izračuna novo oznako, saj oznaka originala predstavlja spremenljivko za veriženje. Vse skupaj je seveda možno zaradi iterativne konstrukcije.

Kljub zgornjemu napadu, še vedno ostajamo pri ideji vključevanja ključa preko IV, le da bomo ubrali malo drugačen pristop. Poleg tega bomo videli, da se zaradi iterativne konstrukcije, vpeljava skozi IV lahko simulira tudi brez dejanske modifikacije F in njenega IV. Na take funkcije s ključem lahko potem gledamo kot na družine funkcij. Naj bo $f_k(x) = f(k, x)$ kompresijska funkcija s ključem, kjer je $|k| = l$ in $|x| = b$. Z neko konkretno iterativno konstrukcijo potem povežemo družino funkcij $\{F_k\}$, kjer so elementi izbrani glede na ključ. Prostor ključev so vsi

ℓ-bitni nizi. V tem primeru je:

$$\begin{aligned} F_k(x) &= k_{n+1} \\ k_i &= f_{k_{i-1}}(x_i) \\ k_0 &= k \\ x_{n+1} &= |x| \end{aligned}$$

kjer je $x = x_1 x_2 \dots x_n$ ter $i = 1, \dots, n+1$.
Sedaj pa razširimo pojem odpornosti na trke še na družino zgoščevalnih funkcij s ključem.

Definicija 6. *Imejmo nasprotnika, ki nima dostopa do ključa k, ki je omejen s časom (številom operacij) t in ki ima na voljo izračun F_k na največ q različnih sporočilih dolžine največ L. Rečemo, da je družina zgoščevalnih funkcij s ključem $\{F_k\}$* **(ε, t, q, L)-šibko odporna na trke**, *kadar verjetnost, da bo nasprotnik odkril dve različni sporočili m in m', za kateri bo veljalo $F_k(m) = F_k(m')$, ni večja od ε.*

Tukaj je treba komentirati dejstvo, da je v praksi odpornost iz zgornje definicije močnejša kot klasična odpornost proti trkom fiksnih zgoščevalnih funkcij, ki jih lahhko vidimo kot posebne elemente družine s ključem, kjer je ključ kar IV. Razlog je v tem, da ima napadalec ponavadi omejen dostop do izračuna funkcije pri žrtvi - omejen tako po številu q kot tudi v smislu hitrosti. Po drugi strani pa so pri fiksnih zgoščevalnih funkcijah te omejitve mnogo manjše, možna je tudi paralelizacija. Torej je odpornost iz definicije 6 zares šibkejša verzija odpornosti na problem iz definicije 4.

3.3 NMAC

Kot smo že videli, je klasičen način z zamenjavo IV s ključem oz. s pripenjanjem ključa pred vhod (zaradi iterativne zgradbe je oboje ekvivalentno) izpostavljeno preprostem razširitvenemu napadu. Zato so avtorji v [2] za osnovo nove konstrukcije vzeli t.i. gnezdeni MAC (ang. Nested MAC), kar je v angleščini s kratico NMAC. Kot smo že opisali, naj bo f_k kompresijska funkcija s ključem k, njena iteriranka pa F_k. Naj bo ključ za NMAC dvojica ključev za F_k, torej $k = (k_1, k_2)$. Potem definiramo NMAC kot:

$$NMAC_k(x) = F_{k_1}(F_{k_2}(x))$$

Opazimo, da zunanji klic v bistvu pomeni eno izvajanje kompresijske funkcije f_{k_1} na $F_{k_2}(x)$ razširjenem do dolžine bloka po pravilu razširitve za F. Pomembno je tudi, da taka konstrukcija zahteva le eno dodatno izvajanje kompresijske funkcije napram računanju klasične zgostitve.

Za začetek varnostne analize, zapišimo glavni varnostni izrek za NMAC.

Izrek 3. *Če je kompresijska funkcija s ključem f (ε_f, q, t, b)-varni MAC na sporočilih dolžine b in če je iterativna funkcija s ključem F (ε_F, q, t, L)-šibko odporna na trke, potem je NMAC $(\varepsilon_f + \varepsilon_F, q, t, L)$-varni MAC.*

Izrek pravi, da nasprotnik, ki se loti razbijanja NMAC, nima več kot dvakratne možnosti proti nasprotniku, ki išče trke za F ali nasprotniku, ki poizkuša razbiti f_k kot MAC. Vsi imajo na voljo enak čas in število parov sporočil in pripadajočih oznak.

Dokaz 2. *Imejmo napadalca A_N, ki želi razbiti NMAC. Na voljo naj ima q poizvedb, čas t in maksimalno dolžino sporočil L. Recimo, da je njegova verjetnost uspeha ε_N. ε_f pa naj bo verjetnost uspeha najboljšega možnega napadalca proti F_{k_2} pri čemer ima na voljo enake vire (q,t,L). Z uporabo A_N bomo zgradili napadalca A_f, ki za MAC funkcijo f_{k_1} na vhodih b bitov, z uporabo q poizvedb in t časa. Napadalec bo imel verjetnost uspeha s spodnjo mejo $\varepsilon_f \geq \varepsilon_N - \varepsilon_F$. Tako bomo dokazali, da karšenkoli napadalec A_N nima možnosti uspeha večje od $\varepsilon_f + \varepsilon_F$.*

V algoritmu A_f bomo uporabili A_N kot podprogram. Če torej še enkrat ponovimo delovanje napadalca A_N z opisanimi parametri. Napadalec pošlje q poizvedb funkciji $NMAC_k(x)$, katere k ne pozna. Omejen je s časom t sporočili dolžine L. Na koncu ali vrne par (x, y), tako da je $y = MAC_k(x)$ in da $x \neq x_i$ za $i = 1, \ldots, q$, ali pa vrne neuspeh. Poleg tega bo za nek niz m dolžine ℓ, niz $\overline{m}$ pomenil niz m podaljšan do dolžine b. Algoritem 3 prikazuje napad.

Algorithm 3 A_f-with-A_N(q, A_N)

naključno izberi k_2
for $i = 1, \ldots, q$ **do**
  $A_N \rightarrow x_i$
  $A_N \leftarrow f_{k_1}(\overline{F_{k_2}(x_i)})$
end for
$A_N \rightarrow (x, y)$
return $(\overline{F_{k_2}(x)}, y)$

A_f torej najprej izbere naključno ključ k_2 za F_{k_2}. Potem pokliče A_N, ki generira q poizvedb za NMAC, na katere odgovori A_f. Ta odgovori tako, da je isto kot, da bi nanje odgovoril NMAC s ključem (k_1, k_2), kar se vidi iz psevdo kode. Za vsak x_i, ki ga generira A_N, A_f izračuna $z_i = F_{k_2}(x_i)$, to potem razširi v $\overline{z_i}$ in pošlje kot poizvedbo v f_{k_1}. Po q poizvedbah, A_N vrne ponaredek (tako imenujemo par dobljen z napadom) (x, y). To uporabi A_f in vrne ponaredek $(\overline{F_{k_2}(x)}, y)$.

Napad ne uspe v 2 primerih. Prvi je, kadar A_N kot podprogram ne uspe. Drugi je, ko za x ki ga vrne A_N velja $\overline{F_{k_2}(x)} = \overline{F_{k_2}(x_i)}$ oz. $F_{k_2}(x) = F_{k_2}(x)$ za nek $i = 1, \ldots, q$.

Sedaj seštejemo verjetnosti obeh dogodkov. Še enkrat poudarimo, da so odgovori na poizvedbe A_N, kot jih daje A_f enaki kot v splošnem napadu A_N na nek $NMAC_k$. Čeprav smo k_2 izbrali v A_f, smo ga izbrali naključno in je iz vidika A_N to enako kot v napadu na nek $NMAC_k$. Torej je verjetnost prve napake po definicij največ $1 - \varepsilon_N$. V drugem primeru, lahko uporabimo dejstvo, da bi A_f lahko uporabili za iskanje trkov F_{k_2}, le da bi v tem primeru izbrali naključno k_1 in ne k_2. Po definiciji je navečja verjetnost za uspeh najboljšega napadalca na F_{k_2} ε_F.

Po definiciji je verjetnost neuspeha A_f $1 - \varepsilon_f$. Po največ $(1 - \varepsilon_N) + \varepsilon_F$. Ko preuredimo dobimo neenakost in s tem končamo dokaz:

$$\varepsilon_N \leq \varepsilon_f + \varepsilon_F$$

□

Nekaj komentarjev dokaza:

- Dokaz, kot smo ga opisali, je konstruktivne narave, saj podaja algoritem napada.
- Gre za generični napad, torej vse možne napade, tudi take, ki jih še ne poznamo.
- Pokazali smo razmerje varnosti (v smislu uspešnosti napada) med NMAC in zgoščevalno funkcijo, ki jo NMAC uporablja.

- Predpostavke o varnosti so v praktičnem smislu premočne. Npr. zahteva, da je F_k šibko odporna na trke, bi se lahko preuredila v zahtevo, da je F_k odporna na trke proti nasprotnikom, ki vidijo njen izhod šele po tem, ko je zgoščen še enkrat z novim ključem k_1.
- Varnost NMAC na napad z izčrpnim preiskovanjem ni $2l$-bitov ampak ℓ-bitov. Razlog bo znan kasneje ob opisu deli-in-vladaj napada.
- Že v izvirnem članku ([2]) so avtorji zapisali intuicijo, da so varnostne predpostavke glede kompresijske funkcije in njene iteriranke v nekem smislu premočne. Novejši dokaz iz leta 2006 to potrjuje.

4 HMAC

4.1 Uvod v HMAC

NMAC ima nekaj slabosti, oz. lastnosti, ki bi se jim v praktični uporabi želeli izogniti. Ena takih je potreba po dostopu do implementacije kompresijske funkcije z namenom, da se namesto IV uvede skrivni ključ v postopek. Temu se želimo izogniti, saj je lepše imeti mehanizem, ki uporablja zgoščevalne funkcije kot črne škatle. Tako lahko uporabimo že spisane knjižnice brez modificiranja, ali pa celo strojne implementacije zgoščevalnih funkcij.

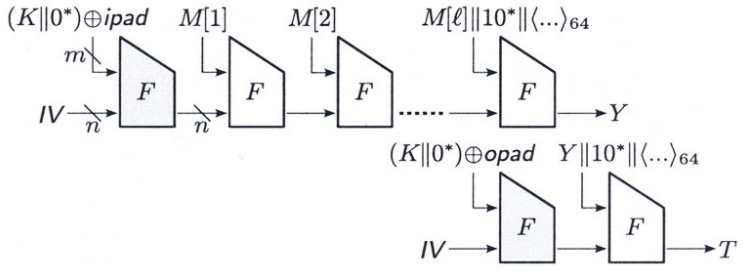
Druga slabost pa je potreba po dveh ključih, saj to spet oteži implementacijo z vidika upravljanja ključev. Zato so avtorji NMAC uvedli adaptacijo imenovano HMAC.

4.2 Definicija HMAC

Naj bo F iterativna zgoščevalna funkcija brez ključa inicializirana s svojim standardnim IV. Imejmo en skrivni ključ k dolžine ℓ bitov. Kot v prejšnem razdelku, ℓ je število izhodnih bitov zgoščevalne funkcije ter velikost vhoda za veriženje, b pa je velikost bloka na vhodu. HMAC za poljubno dolgo sporočilo x je potem definiran takole:

$$HMAC_k(x) = F(\overline{k} \oplus opad \parallel F(\overline{k} \oplus ipad \parallel x))$$

V enačbi $\overline{k}$ pomeni dopolnitev k do b bitov z ničlami, *opad* in *ipad* sta fiksni b-bitni konstanti. V



Slika 3: Prikaz delovanja HMAC. V tem primeru je $|M[l]| \leq b - 65$.

heksadecimalnem zapisu je *opad* definiran kot bajt 0x36 ponovljen do dolžine b bitov. Analogno je *ipad* 0x5c. Znak $\oplus$ pomeni bitni ekskluzivni ali. Postopek prikazuje tudi Slika 3, le da je na sliki kompresijska funkcija označena s F in ne f (slika je vzeta iz drugega članka). Poleg tega prikazuje tudi detajle Merkle-Damgard konstrukcije, kot je dopolnitev bitov do večkratnika velikosti b ter 64 bitno kodiranje dolžine sporočila z oznako $\langle \ldots \rangle_{64}$.

4.3 Varnost HMAC

Za začetek razprave o varnosti, si je dobro pogledati relacijo med HMAC in NMAC. HMAC je v bistvu NMAC, kjer sta ključa $k_1 = f(\overline{k} \oplus opad)$ in $k_2 = f(\overline{k} \oplus ipad)$. V tem detajlu se skriva bistvo uporabe varnosti NMAC za HMAC. Varnost NMAC namreč predpostavlja, da sta ključa izbrana naključno in neodvisno. Pri HMAC torej zahtevamo od kompresijske funkcije f dodatno lastnost, da je čim bolj psevdo-naključna, saj želimo, da napadalec ne bi mogel prepoznati povezave med $f(\overline{k} \oplus opad)$ in $f(\overline{k} \oplus ipad)$ oz. da bi se mu zdela naključna. K temu naj bi pripomogla tudi izbira *opad* in *ipad*. Izbrana sta tako, da imata rezultata mešanja s ključem čim večjo hammingovo razdaljo ter da sta lahka za zapis zaradi zmanjšanja implementacijskih napak.

Za f ne potrebujemo sicer izredno močne psevdo-naključnosti, saj napadalec nikoli ne vidi ključev. Strogo gledano so napadi, ki delujejo na HMAC in ne na NMAC možni, vendar bi to kazalo na hude pomankljivosti kompresijske funkcije.

Pomembno je tudi vedeti, da bi tipična implementacija NMAC uporabljala nek psevdo-naključni generator za generiranje obeh ključev in torej tudi tam ključa ne bi bila zares naključna. V primeru HMAC pa je psevdo-naključni generator nekako „vgrajen“ v samo zasnovo preko uporabe kvalitetne kompresijske funkcije in izbire *opad* ter *ipad*.

4.4 Napadi na HMAC

Preverimo nekaj znanih napadov na kriptografske zgoščevalne funkcije in kako se obnašajo v odnosu do HMAC.

Rojstnodnevni napad (ang. birthday attack) smo že omenjali kot osnovo za iskanje trkov v zgoščevalnih funkcijah. Izkaže pa se, da je napad uporaben tudi proti MAC-om temelječim na iterativnih funkcijah. To velja tudi za NMAC in HMAC. Pomembnost napada je v tem, da zelo izboljša izčrpno preiskovanje, saj nekje razpolovi število bitov. Še vedno pa je to nepraktično za MAC-e, kjer so ključi daljši od 64 bitov (kar velja za praktično vse danes popularne MAC-e), zlasti ker je v nasprotju z napadom na klasične zgoščevalne funkcije tukaj potrebna interakcija z „žrtvijo“, da nam izračuna ogromno število MAC-ov in pa ni možna paralelizacija. Tako da, čeprav je napad občutno izboljšanje izčrpnega preiskovanja, je za MAC-e še vedno daleč od praktičnega.

Primer napada iz [4, raz. 6]: Izberemo si poljubno sporočilo x dolžine b (b je dolžina bloka). Nato pošiljamo poizvedbe na MAC za sporočila $a_i \parallel x$, kjer je a_i poljubno sporočilo dolžine b. To počnemo dokler ne najdemo različnih a_i in a_j, da je $MAC_k(a_i \parallel x) = MAC_k(a_j \parallel x)$, kjer je k neznani ključ. Sedaj je zelo verjetno (za HMAC, če se je trk zgodil zaradi trka v notranjem klicu zgoščevalne funkcije), da je $MAC_k(a_i) = MAC_k(a_j)$. Napadalec to potem izkoristi tako, da za poljuben y pošlje poizvedbo za MAC oznako za $a_i \parallel y$. Recimo, da je vrnjena oznaka m. Potem velja tudi, da je $MAC_k(a_j \parallel y) = m$.

Tem napadom v osnovni obliki se da izogniti z randomizacijo konstrukcije MAC-a. Več o tem je na voljo v [4, raz. 5].

Napadi s trkom na zgoščevalno funkcijo (ang. collision attacks on keyless hash functions) izkoriščajo dejstvo, da so napadi s trkom na zgoščevalne funkcije brez ključa mnogo bolj realni (tudi z uporabo analitičnih metod) kot na MAC-e s neznanimi ključi. Da se jih paralelizirati, ni potrebna interakcija s tarčo. Npr. če pogledamo preprost MAC, kjer ključ uporabimo tako, da ga pripnemo na konec sporočila, t.i. „append-only“ konstrukcija. Formalno, $MAC_k(x) = F(x \parallel k)$. Če poznamo dve sporočili x in x', ki trčita v F, potem trčita tudi pripadajoči MAC oznaki. Kot rečeno, sta HMAC in NMAC v tem pogledu varna, je pa verz-

ija NMAC, kjer v notranjih funkciji ne uporabimo ključa, precej bolj izpostavljena temu napadu.

Razširitveni napad (ang. extension attack) se pojavi v okviru „prepend-only“ konstrukcije, kjer je $MAC_k(x) = F(k \parallel x)$. To konstrukcijo in napad nanjo smo že opisali v razdelku 3.2. NMAC prepreči ta napad z zunanjim apliciranjem F_{k_1}.

Deli in vladaj napad (ang. divide and conquer attack) opišemo v kontekstu „envelope“ konstrukcije, kjer je $MAC_{k_1,k_2}(x) = F(k_1 \parallel x \parallel k_2)$. Napad, ki išče celoten ključ, ne potrebuje čas eksponenten v skupni dolžini obeh ključev, ampak čas reda eksponenten v dolžini enega ključa. Napad ni praktičen ampak kaže na dejstvo, da je moč MAC-a odvisna od enkratne dolžine ključa. Podobno velja za NMAC. Vidimo tudi, da uporaba enega ključa v HMAC ne zmanjša varnosti proti izčrpnem preiskovanju napram NMAC.

4.5 Novejši dokaz varnosti za HMAC

Eden od avtorjev HMAC je v letu 2006 objavil novejši dokaz varnosti za NMAC in HMAC[5]. Do te točke so dokazi varnosti temeljili na predpostavki, da je iterirana zgoščevalna funkcija šibko odporna na trke. Poleg tega so predpostavljali, da je kompresijska funkcija varna kot MAC, kar je posledično pomenilo MAC varnost za NMAC in HMAC. Tak dokaz je v razdelku 3.3. Lahko pa se zahteva, da je kompresijska funkcija PRF (psevdo-naključna funkcija), kar je težja zahteva, vendar ima potem tudi NMAC oz. HMAC PRF varnost, kar je močnejša varnost kot MAC varnost. Z napadi na MD5 in SHA-1 je postalo jasno, da ti dve funkciji (ki sta najpogostejši osnovi za HMAC) nista šibko odporni na trke in s tem dokaz varnost pade. Je pa že v [2] opozorjeno, da je HMAC najbrž varen že s šibkejšimi zahtevami, vendar ni bil predstavljen dokaz.

Dokaz iz [5] pokaže, da je NMAC PRF že samo na predpostavki, da je kompresijska funkcija PRF. To je pomembno, saj dokaz ne predpostavlja ničesar o iterativni zgoščevalni funkciji, ampak le o kompresijski funkciji. Za kompresijski funkciji MD5 in SHA-1 pa niso znani napadi, ki bi ovrgli predpostavko, da sta PRF. Dokaz najprej pokaže, da če je kompresijska funkcija PRF, potem je njena iteriranka računsko skoraj univerzalna (ang. computationally almost universal, cAU). Potem pa avtor pokaže, da je kompozicija PRF in cAU (za to uporabi posplošeni NMAC, GNMAC, kjer zanemari detajle o pad funkciji) spet PRF. Avtor predstavi še dodaten dokaz, ki temelji na šibkejši predpostavki za kompresijsko funkcijo, namreč da je zasebnost ohranjajoči MAC (ang. privacy preserving MAC, PP-MAC) ter da je iterativna funkcija cAU in da je potem NMAC varni MAC. To je pomembno, saj je PP-MAC bistveno šibkejša zahteva od PRF, varni MAC pa je zadostna varnost za MAC - ne potrebujemo PRF. Definicija PP-MAC je v [5, raz. 4].

4.6 Implementacijski detajli HMAC

HMAC je v primerjavi z NMAC počasnejši, saj uporablja 2 dodatna izračuna zgostitve, za blok $(\overline{k} \oplus opad)$ in $(\overline{k} \oplus ipad)$. Pri sporočilih velike dolžine se to ne pozna preveč, lahko pa se opazi pri uporabi na kratkih sporočilih. Odvisno je torej kakšen delež predstavljata ta 2 dodatna izračuna kompresijske funkcije v skupnem številu izračunov kompresijske funkcije. Implementacija lahko ta dva izračuna opravi vnaprej in jih shrani kot ključa k_1 in k_2. V tem primeru mora implementacija imeti dostop do IV f in ga nastaviti na ti dve vrednosti ob primernem času.

HMAC bi lahko uporabljal tudi različne dolžine ključev, vendar se uporaba ključev krajših od ℓ bitov odsvetuje. Daljši ključi bi se pa skrajšali na ℓ bitov in v tem smislu ne bi prinašali večje varnosti. Je pa možna večja varnost v smislu naključnosti in neodvisnosti ključa k in

kasneje obeh ključev izračunanih iz njega. Kot tudi drugod v kriptografiji je seveda pomembna varna hramba ključa. Svetuje se tudi periodično spreminjanje ključa, saj to pripomore k večji praktični varnosti.

5 H²MAC - HMAC brez drugega ključa

5.1 Uvod v H^2MAC

Pomankljivost HMAC algoritma je v načinu uporabe skrivnega ključa. Algoritem namreč zahteva dostop do ključa na začetku in na koncu operacije. Predvsem je problem v drugem dostopu, saj to pomeni, da mora biti skrivni ključ na voljo celoten čas operacije, ali pa mora proces dvakrat dostopati do neke varne shrambe. Alternativa, predstavljena v [6], poskuša to pomankljivost odpraviti in ohraniti glavne prednosti HMAC, predvsem uporabo iterativnih zgoščevalnih funkcij kot črnih škatel ter dokaz varnosti, ki temelji na predpostavki, da je kompresijska funkcija PRF (psevdo naključna funkcija) oz. neka modifikacija PRF zahteve.

Rešitev predstavlja MAC algoritem imenovan H^2MAC, ki je definiran kot

$$T = H(H(K \parallel pad \parallel M)),$$

kjer je $H : \{0,1\}^* \to \{0,1\}^n$ (prej smo to označili s F) iterativna zgoščevalna funkcija, n je dolžina izhoda (prej smo to označili z ℓ). Skrivni ključ je $K \in \{0,1\}^n$, T pomeni oznako za sporočilo poljubne dolžine M. Nadalje je $pad \in \{0,1\}^{m-n}$ fiksna konstanta, kjer m pomeni dolžino bloka (prej smo to označili z b).

Dodatna prednost zgornjega algoritma je en izvajanje kompresijske funkcije manj, saj pred zadnjim klicem ne vpeljemo drugega ključa, kot je to pri HMAC. Ta prednost se pozna pri krajših sporočilih.

V dokazu, da je nova konstrukcija PRF, uporabimo predpostavko za kompresijsko funkcijo, imenovano PRF-AX (ang. PRF with an affix). Gre za manjšo modifikacijo standardne PRF zahteve, ki bo razložena kasneje. PRF zahtevo za HMAC v dokazu varnosti iz [5] imenujemo PRF-KD (ang. PRF with key derivation) in ni ne močnejša ne šibkejša od PRF-AX zahteve.

5.2 Uvodne definicije

Glede operacij nad biti uporabljamo standardno notacijo, kot v predhodnem delu besedila. Na novo gre omeniti zapis $\langle n \rangle_{64}$, ki pomeni 64-bitno kodiranje števila n. Kadar je vrednost n implicitna, zapišemo $\langle \ldots \rangle_{64}$.

Kot smo že uvodoma nakazali, m pomeni velikost bloka, n velikost izhoda. Kompresijsko funkcijo bomo sedaj označevali s $F : \{0,1\}^{n+m} \to \{0,1\}^n$. Zahtevamo, da je $n + 65 \leq m$, kar drži za večino sodobnih kompresijskih funkcij. Časovno zahtevnost q izračunov F bomo zapisali kot $\text{Time}_F(q)$.

Iterativno zgoščevalno funkcijo na osnovi kompresijske funkcije F označimo kot F^*_V, kjer V pomeni inicializacijsko vrednost za F. Preko tega, potem dobimo Merkle-Damgard zgoščevalno funkcijo H, ki smo jo že definirali v 2.3. Z $M[i]$ označimo i-ti blok sporočila M. HMAC je definiran kot v 4.2 le z drugimi imeni spremenljivk, delovanje pa prikazuje tudi slika 3, kjer so imena spremenljivk enaka kot v tem razdelku.

Psevdo naključne funkcije. Naj bo $\mathcal{O}$ oznaka za črno skrinjico (ang. oracle). A naj bo nasprotnik (v obliki algoritma), ki vrača odgovore 1 ali 0. $A^{\mathcal{O}}$ je odgovor, ki ga vrne A po interakciji s črno skrinjico $\mathcal{O}$.

Črna skrinjica vrača naključne odgovore na poizvedbe, le da za isto poizvedbo, vrne vedno enak odgovor. Mi bomo uporabili 2 črni skrinjici. Prva je t.i. „prava“ črna skrinjica $G_K : \{0,1\}^* \to \{0,1\}^n$, ki za neko poizvedbo M izbere ključ K naključno (funkcija pa je fiksna, npr. HMAC) in potem vrne izhod za to poizvedbo, glede na izbrani ključ in funkcijo, ki jo ta črna skrinjica predstavlja, npr. HMAC. Operacijo naključnega izbora iz prostora n bitov v spremenljivko K označimo z $K \xleftarrow{\$} \{0,1\}^n$.

Druga je t.i. „idealna“ črna skrinjica $\mathcal{R} : \{0,1\}^* \to \{0,1\}^n$, ki za vsako poizvedbo izbere naključno neko funkcijo, ki slika iz $\{0,1\}^*$ v $\{0,1\}^n$.

Ideja je torej imeti 2 črni skrinjici, eno ki predstavlja funkcijo (natančno, družino funkcij, glede na nek parameter, v našem primeru ključ K), ki jo študiramo in drugo, ki predstavlja naključne funkcije, ki delujejo nad enakimi množicami kot prva. Nasprotnik A pa je verjetnostni algoritem, ki se trudi razlikovati odgovore iz teh dveh črnih skrinjic. V kontekstu PRF torej poskuša razlikovati neko zgoščevalno funkcijo od naključne funkcije. V splošnem to počne tako, da pošilja poizvedbe črnima skrinjicama, in potem glede na odgovore in lastne algoritme in tudi naključne odločitve, vrača odgovore.

Prednost (ang. advantage) opisanega nasprotnika A definiramo kot:

$$\mathsf{Adv}_G^{prf} = \Pr[A^{G_K^{(\cdot)}} = 1] - \Pr[A^{\mathcal{R}^{(\cdot)}} = 1],$$

kjer so verjetnosti glede na izbiro K, izbiro $\mathcal{R}$ in naključne izbire znotraj A. Dodatno še definiramo:

$$\mathsf{Adv}_G^{prf}(t,q,l) = \max_A \mathsf{Adv}_G^{prf}(A),$$

kjer gre maksimum po vseh nasprotnikih A, ki porabijo ne več kot t časa, naredijo največ q poizvedb črnim skrinjicam, kjer nobena poizvedba ni daljša od ℓ blokov.

Družine črnih multi-skrinjic (ang. multi-oracle families). Tehnika je predstavljena v [4]. Namesto G sedaj govorimo o konkretni funkciji $F_K : \{0,1\}^m$ v $\{0,1\}^n$, to je verzija kompresijske funkcije s ključem. Ključ pripnemo na začetku, tako da je $F_K(\cdot) = F(K \parallel \cdot)$. Imamo še „idealno“ funkcijo $\mathcal{R} : \{0,1\}^m$ v $\{0,1\}^n$.

Črna multi-skrinjica $F \otimes \cdots \otimes F$ izbere naključno q neodvisnih ključev $K_1, \ldots, K_q \xleftarrow{\$} \{0,1\}^n$. Za poizvedbo (i,x) potem vrne $F_{K_i}(x)$.

Analogno, črna multi-skrinjica $R \otimes \cdots \otimes R$ izbere naključno q neodvisnih funkcij $R_1, \ldots, R_q$ iz prostora vseh funkcij, ki slikajo m bitov v n bitov. Za poizvedbo (i,x) vrne $R_i(x)$. Spet definiramo prednost:

$$\mathsf{Adv}^{prf}_{F\otimes\cdots\otimes F}(A) = \Pr[A^{F\otimes\cdots\otimes F} = 1] - \Pr[A^{R\otimes\cdots\otimes R} = 1].$$

Analogno kot prej definiramo $\mathsf{Adv}^{prf}_{F\otimes\cdots\otimes F}(t,q)$, q pomeni skupno število poizvedb po vseh i, ℓ parametra pa nimamo, saj je F kompresijska funkcija, ki deluje nad enim samim blokom in so torej vse poizvedbe dolžine 1 bloka. Zapišimo lemo, katere dokaz najdemo v [4].

Lema 1. *Če je F PRF, potem je tudi $F \otimes \cdots \otimes F$ PRF. Konkretno*

$$\mathsf{Adv}^{prf}_{F\otimes\cdots\otimes F}(t,q) \le q \cdot \mathsf{Adv}^{prf}_F(t',q),$$

kjer je časovna zahtevnost t' približno $t + \mathrm{Time}_F(q)$.

Algorithm 4 $H^2MAC_K(M)$

$Y \leftarrow H(K \parallel pad \parallel M)$
$T \leftarrow H(Y)$
return $T \in \{0,1\}^n$

5.3 Definicija H^2MAC

Definicijo H^2MAC prikazuje algoritem 4 ter slika 4.
Definirajmo še

$$pad = 1 \parallel 0^{m-n-65} \parallel \langle n \rangle_{64} \in \{0,1\}^{m-n}.$$

Zaradi lažje analize predpostavimo, da je ključ K dolžine n bitov, v praksi bi lahko deloval lepo tudi z drugačnimi dolžinami. Sledi analiza varnosti.

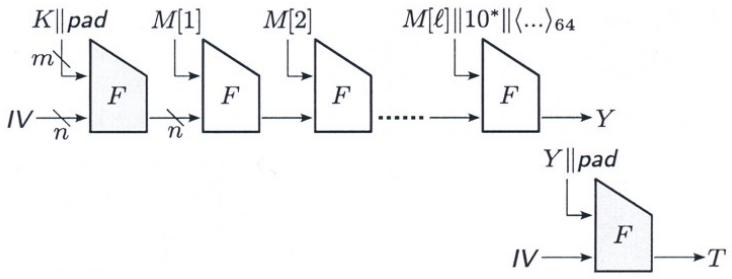
5.4 Varnost H^2MAC

Na začetku podrazdelka pojasnimo PRF-AX domnevo. Nato sledi obsežnejši del, kjer zapišemo glavni izrek varnosti ter njegov dokaz. Po dokazu zapišemo še primerjavo med PRF-AX in PRF-KD, domnevo, ki jo uporablja dokaz varnosti za običajni HMAC.

PRF-AX. PRF-AX (ang. PRF with an affix) domneva je podobna klasični PRF, le da ima tukaj nasprotnik dostop do dodatne informacije - t.i. priponke (ang. affix). Imejmo spet kompresijsko funkcijo s ključem pripetim na začetku $F_K(\cdot) = F(K \parallel \cdot)$. Poleg dostopa do črnih skrinjic za F_K in R ima nasprotnik še dostop do črne skrinjice za priponko (ang. affix oracle). Ta skrinjica vrača $F(IV \parallel K \parallel pad)$, kjer je treba poudariti, da je K enak kot tisti pri črni skrinjic za $F_K(\cdot)$, torej napadalec dostopa do obeh skrinjic sočasno. To tudi pomeni, da so zahteve na to skrinjico brez kakšnih dodatnih sporočil oz. da skrinjica ne sprejme vhoda podatkov. Cilj nasprotnika je torej razlikovati ta par črnih skrinjic (skrinjica za F_K in skrinjica za priponko) od para skrinjice za $R : \{0,1\}^m \to \{0,1\}^n$ in skrinjice za naključni niz $r \xleftarrow{\$} \{0,1\}^n$. Za opisanega nasprotnika A definiramo prednost (tokrat za okolje prf-ax) podobno kot prej:

$$\mathsf{Adv}_F^{prf-ax}(A) = \Pr[A^{F_K(\cdot),F(IV\|K\|pad)} = 1] - \Pr[A^{R(\cdot),r} = 1].$$

Analogno prejšnim definicijam uporabljamo zapise $\mathsf{Adv}_F^{prf-ax}(t,q)$, $\mathsf{Adv}_{F\otimes\cdots\otimes F}^{prf-ax}(A)$ in $\mathsf{Adv}_{F\otimes\cdots\otimes F}^{prf-ax}(t,q)$. Sledi zapis glavnega izreka in njegov dokaz.



Slika 4: Prikaz delovanja H^2MAC.

Izrek 4. *H^2MAC algoritem ima PRF varnost, če za kompresijsko funkcijo zgoščevalne funkcije velja PRF-AX domneva varnosti. Velja*

$$\mathsf{Adv}^{prf}_{H^2MAC}(t,q,l) \leq (lq+1)\cdot \mathsf{Adv}^{prf-ax}_{F}(t',q),$$

kjer je časovna zahtevnost t' približno $t + \mathrm{Time}_F(lq+q)$.

Dokaz razdelimo v tri dele.

Prvi del: od H^2MAC do $\tilde{H}^2$. Vpeljemo modificirano različico algoritma imenovano $\tilde{H}^2$. Razlika je v prvi fazi, kjer skrivno spremenljivko za veriženje $F(IV \parallel K \parallel pad)$ nadomestimo s ključem K. Postopek prikazuje algoritem 5.

Algorithm 5 $\tilde{H}^2_K(M)$

$x[1]\cdots x[l] \leftarrow K \parallel pad \parallel M \parallel 10^* \parallel \langle \ldots \rangle_{64}$
$Y \leftarrow F^*_K(x[2]\cdots x[l])$
$T \leftarrow H(Y)$
return $T \in \{0,1\}^n$

Lema, ki sledi, zreducira varnost H^2MAC na varnost $\tilde{H}^2$ pod enakimi predpostavkami.

Lema 2. *Če je kompresijska funkcija F varna glede na PRF-AX domnevo in če je $\tilde{H}^2$ algoritem varen glede na PRF, potem je tudi H^2MAC algoritem varen glede na PRF. Velja*

$$\mathsf{Adv}^{prf}_{H^2MAC}(t,q,l) \leq \mathsf{Adv}^{prf-ax}_{F}(t',q) + \mathsf{Adv}^{prf}_{\tilde{H}^2}(t,q,l),$$

kjer je časovna zahtevnost t' približno $t + \mathrm{Time}_F(lq)$.

Dokaz 3. *Naj bo A nasprotnik za H^2MAC. Predpostavimo, da je časovna zahtevnost A največ t, da naredi največ q poizvedb in dolžina vsake poizvedbe je največ ℓ blokov. Sestavimo nasprotnika B, ki napade F v PRF-AX smislu z uporabo nasprotnika A. Delovanje B kaže algoritem 6.*

Algorithm 6 Nasprotnik B

Naredimo poizvedbo na črno skrinjico za priponko F (affix oracle) in dobimo vrnjeno $V \in \{0,1\}^n$
Poženemo A in odgovorimo na njegovo zahtevo črni skrinjici M na naslednji način:
$x[1]\cdots x[\lambda] \leftarrow 0^m \parallel M \parallel 10^* \parallel \langle \ldots \rangle_{64}$
Posreduj $H(F^*_V(x[2]\cdots x[\lambda]))$ A-ju
return izhod(A)

Opazimo, da se $B^{\cdots,F(IV\|K\|pad)}$ obnaša točno tako, kot prava črna skrinjica za $H^2MAC_K(\cdot)$. Prav tako je $B^{\cdots,r}$ točna simulacija črne skrinjice za $\tilde{H}^2(\cdot)$. Zato sledi

$$\begin{aligned}
\mathsf{Adv}^{prf-ax}_{F}(t',1) &\geq \mathsf{Adv}^{prf-ax}_{F}(B)\\
&= \Pr[B^{\cdots,F(IV\|K\|pad)}=1] - \Pr[B^{\cdots,r}=1]\\
&= \Pr[A^{H^2MAC_K(\cdot)}=1] - \Pr[A^{\tilde{H}^2_K(\cdot)}=1]\\
&= \Pr[A^{H^2MAC_K(\cdot)}=1] - \Pr[A^{\mathcal{R}(\cdot)}=1]\\
&\quad -\Pr[A^{\tilde{H}^2_K(\cdot)}=1] + \Pr[A^{\mathcal{R}(\cdot)}=1]\\
&= \mathsf{Adv}^{prf}_{H^2MAC}(A) - \mathsf{Adv}^{prf}_{\tilde{H}^2}(A),
\end{aligned}$$

kjer je časovna zahtevnost t' približno $t + \mathrm{Time}_F(lq)$. To konča dokaz leme 2. □

Drugi del: od $\tilde{H}^2$ do $F \otimes \cdots \otimes F$. Zreduciramo varnost $\tilde{H}^2$ na varnost družine črnih multi-skrinjic (ang. multi-oracle families). Dokažemo sledečo lemo.

Lema 3. *Algoritem $\tilde{H}^2$ je PRF-varen, če je $F \otimes \cdots \otimes F$ družina črnih multi-skrinjic varna glede na PRF-AX domnevo. Velja*

$$\mathsf{Adv}^{prf}_{\tilde{H}^2}(t, q, l) \leq \ell \cdot \mathsf{Adv}^{prf-ax}_{F \otimes \cdots \otimes F}(t', q),$$

kjer je časovna zahtevnost t' približno $t + \mathrm{Time}_F(lq)$.

Dokaz 4. *Vpeljemo funkcije G_i, $i = 0, 1, \ldots, l$. Te funkcije sprejmejo naključno funkcijo $g : \{0,1\}^* \to \{0,1\}^n$ kot ključ. Delovanje G_i prikazuje algoritem 7.*

Algorithm 7 $G_i(M)$

Na vhod dobimo funkcijo g in sporočilo M
$x[1] \cdots x[\lambda] \leftarrow M \parallel 10^* \parallel \langle \ldots \rangle_{64}$
if $\lambda < i$ **then**
  $T \leftarrow g(x[1] \cdots x[\lambda])$
end if
if $\lambda = i$ **then**
  $Y \leftarrow g(x[1] \cdots x[\lambda])$
  $T \leftarrow H(Y)$
end if
if $\lambda > i$ **then**
  $V \leftarrow g(x[1] \cdots x[i])$
  $T \leftarrow H(F^*_V(x[i+1] \cdots x[\lambda]))$
end if
return $T \in \{0,1\}^n$

Naj bo A nasprotnik za H^2MAC. Predpostavimo, da je časovna zahtevnost A največ t, da naredi največ q poizvedb in dolžina vsake poizvedbe je največ ℓ blokov. Definiramo verjetnosti $P_i = \Pr[A^{G_i(\cdot)} = 1]$ za $i = 0, 1, \ldots, l$, glede na izbiro g. Prednost napadalca je potem

$$\mathsf{Adv}^{prf}_{\tilde{H}^2}(A) = \Pr[A^{\tilde{H}^2_k(\cdot)} = 1] - \Pr[A^{\mathcal{R}(\cdot)} = 1] = P_0 - P_l = \sum_{i=0}^{l-1} (P_i - P_{i+1}).$$

Sedaj poskušamo omejiti $(P_i - P_{i+1})$ z uporabo nasprotnikov B_i za $i = 0, 1, \ldots, l-1$. B_i je nasprotnik za družino črnih multi-skrinjic $F \otimes \cdots \otimes F$ in uporablja nasprotnika A kot podprogram. Nasprotnika B_i prikazuje algoritem 8.

Opazimo, da se $B_i^{F \otimes \cdots \otimes F}$ obnaša točno tako, kot funkcija G_i. Prav tako je $B_i^{R \otimes \cdots \otimes R}$ točna simulacija funkcije G_{i+1}. Zato sledi

$$\begin{aligned} P_i - P_{i+1} &= \Pr[A^{G_i(\cdot)} = 1] - \Pr[A^{G_{i+1}(\cdot)} = 1] \\ &= \Pr[B_i^{F \otimes \cdots \otimes F} = 1] - \Pr[B_i^{R \otimes \cdots \otimes R} = 1] \leq \mathsf{Adv}^{prf-ax}_{F \otimes \cdots \otimes F}(B_i). \end{aligned}$$

Opazimo, da vsak B_i opravi največ q poizvedb in ima časovno zahtevnost največ t', kar je približno $t + \mathrm{Time}_F(lq)$. Od tod sledi

$$\mathsf{Adv}^{prf}_{\tilde{H}^2}(A) \leq \sum_{i=0}^{l-1} \mathsf{Adv}^{prf-ax}_{F \otimes \cdots \otimes F}(B_i) \leq l \cdot \mathsf{Adv}^{prf-ax}_{F \otimes \cdots \otimes F}(t', q),$$

kar konča dokaz leme 3. □

Algorithm 8 B_i

Števec $c \leftarrow 0$
Poženi A in odgovori na njegovo α-ito poizvedbo M^α na naslednji način:
$x^\alpha[1] \cdots x^\alpha[l] \leftarrow M^\alpha \parallel 10^* \parallel \langle \ldots \rangle_{64}$
if $\ell < i$ **then**
  $T \xleftarrow{\$} \{0,1\}^n$
  Posreduj T do A
else
  if $x^\alpha[1] \cdots x^\alpha[l] = x^\beta[1] \cdots x^\beta[l]$ za neko prejšnjo poizvedbo $\beta < \alpha$ **then**
    $s^\alpha \leftarrow s^\beta$
  else
    $c \leftarrow c + 1$
    $s^\alpha \leftarrow c$
  end if
  if $\ell = i$ **then**
    Naredi poizvedbo za priponko svoji s^α-ti skrinjici in odgovor shrani v T
    Posreduj T A-ju.
  end if
  if $\ell = i + 1$ **then**
    Naredi poizvedbo $x[l]$ svoji s^α-ti skrinjici in odgovor shrani v Y
    Posreduj $H(Y)$ A-ju.
  end if
  if $\ell \geq i + 2$ **then**
    Naredi poizvedbo $x[i+1]$ svoji s^α-ti skrinjici in odgovor shrani v V
    Posreduj $H(F_V^*(x^\alpha[i+2] \cdots x^\alpha[l]))$ A-ju.
  end if
end if
return izhod(A)

Tretji del: od $F \otimes \cdots \otimes F$ do F. V zadnjem koraku modificiramo lemo 1 za PRF-AX domnevo namesto standardne PRF. Tako dobimo

Lema 4. *Če je F varna glede na PRF-AX, potem je tudi $F \otimes \cdots \otimes F$ varna glede na PRF-AX. Konkretno*

$$\mathsf{Adv}^{prf-ax}_{F\otimes\cdots\otimes F}(t,q) \leq q \cdot \mathsf{Adv}^{prf-ax}_{F}(t',q),$$

kjer je časovna zahtevnost t' približno $t + \mathrm{Time}_F(q)$.

Torej smo samo zamenjali predpostavko PRF s PRF-AX. Sledi dokaz izreka 4.

Dokaz 5. *Glede na leme 2, 3 in 4, izračunamo*

$$\begin{aligned}
\mathsf{Adv}^{prf}_{H^2MAC}(t,q,l) &\leq \mathsf{Adv}^{prf-ax}_{F}(t',1) + \mathsf{Adv}^{prf}_{\tilde{H}^2}(t,q,l) \\
&\leq \mathsf{Adv}^{prf-ax}_{F}(t',1) + \ell \cdot \mathsf{Adv}^{prf-ax}_{F\otimes\cdots\otimes F}(t',q) \\
&\leq \mathsf{Adv}^{prf-ax}_{F}(t',1) + \ell q \cdot \mathsf{Adv}^{prf-ax}_{F}(t'',q) \\
&\leq (lq+1) \cdot \mathsf{Adv}^{prf-ax}_{F}(t'',q),
\end{aligned}$$

kjer je časovna zahtevnost t' približno $t + \mathrm{Time}_F(lq)$ in t'' približno $t + \mathrm{Time}_F(lq + q)$. To konča dokaz izreka 4. □

Za konec si še pogledamo primerjavo med PRF-AX domnevo, ter PRF-KD domnevo iz novejšega dokaza varnosti za HMAC[5]. Tudi PRF-KD, tako kot PRF-AX, dovoljuje nasprotniku dodatne informacije, povezane z izpeljavo ključa. V tem smislu sta obe zahtevi močnejši kot standardni PRF. Glavna razlika pa je v tem, da je pri PRF-AX uporabljan enak ključ tako za $F_K(\cdot)$ črno skrinjico, kot za skrinjico za priponko. Pri PRF-KD sta ta dva ključa neodvisna, vendar ne moremo reči, da je ena zahteva močnejša od druge. Avtor predpostavi, da sta PRF-AX in PRF-KD v praksi enakovredni standardni PRF in da bi torej težko pokazali, da so kompresijske funkcije sodobnih zgoščevalnih funkcij varne kot PRF, ne pa kot PRF-AX ali PRF-KD.

6 Uporaba HMAC v protokolih

HMAC je postal sestavni del glavnih protokolov (oz. skupkov protokolov) za varnost na področju internetnih komunikacij kot sta IPsec in TLS. V tem razdelku ju na kratko opišemo ter vlogo HMAC znotraj njiju.

6.1 IPsec

Po definiciji iz [7], je IPsec skupek protokolov za zaščito IP komunikacij. Omogoča avtentikacijo in tajnost IP paketov. Poleg tega omogoča tudi avtentikacijo udeležencev na začetku seje in dogovor o ključu. Deluje na internetni plasti in je kot tak transparenten za uporabniške aplikacije. Ščiti lahko komunikacijo med dvema gostiteljema, dvema omrežjema ali med gostiteljem in omrežjem. Konkretno uporablja naslednje protokole:

- Internet Key Exchange (IKE and IKEv2) je protokol za dogovor o poteku seje in izmenjavo in generiranje ključa za sejo
- Authentication Header (AH) protokol omogoča integriteto ter avtentikacijo izvora.

- Encapsulating Security Payload (ESP) protokol omogoča tajnost podatkov, integriteto ter avtentikacijo izvora.

RFC 4835 [9] iz leta 2007 opisuje zahteve za implementacijo AH in ESP. V delu, ki govori o avtentikacijskih algoritmih (tako za AH kot ESP), je tabela, ki za posamezen algoritem opisuje zahtevo z „must“(mora biti), „should“(naj bi bil), „may“(je lahko). HMAC na osnovi MD5 je označen z „may“, HMAC na osnovi SHA-1 pa je obvezen.

6.2 TLS - Transport Layer Security

Kot že kratica pove, ta skupek protokolov deluje na transportni plasti omrežja. SSL (Secure Sockets Layer) je njegov predhodnik, dostikrat se izraza zamenjujeta. Po opisu glede na [8] TLS ščiti podatke nad transportno plastjo z uporabo simetričnih šifer za tajnost in MAC za zanesljivost podatkov. Za razliko od IPsec, se aplikacije zavedajo uporabe TLS.

Večina verzij TLS za MAC uporablja HMAC na osnovi MD5 in SHA-1. Najnovejši RTF 5246 [10] iz leta 2008 pa navaja tudi uporabo HMAC z SHA-2, konkretno SHA-256, SHA-384 in SHA-512.

7 Zaključek

V seminarski nalogi smo opisali HMAC, skupaj z osnovami zgoščevalnih funkcij. Predstavili smo nekatere dokaze in razložili zakaj verjamemo, da je zaenkrat HMAC še vedno varen, tudi ko ga uporabimo s funkcijami kot so MD5, katere ne smatramo več za varne. Malo večji razdelek smo namenili alternativi HMAC in njeni analizi.

Na koncu smo opisali uporabo HMAC v IPsec in TLS protokolih, kjer tudi v najnovejšem RFC 4835[9] o zahtevah za algoritme v IPsec, HMAC ostaja edini „vsestranski“ avtentikacijski protokol. RFC za avtentikacijo sicer omenja tudi AES-XCBC-MAC, vendar se tega lahko uporabi le v kombinaciji z AES v CBC načinu, RFC 4835 pa za enkripcijo zahteva tudi 3DES-CBC ter dovoljuje AES-CTR.

Literatura

[1] H. Krawczyk, M. Bellare, R. Canetti. HMAC: Keyed-hashing for message authentication. RFC 2104, 1997.

[2] M. Bellare, R. Canetti, H. Krawczyk. Keying hash functions for message authentication. Advances in Cryptology - Crypto 96 Proceedings, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed, Springer-Verlag, 1996.

[3] D.R. Stinson. Cryptography: Theory and practice, third edition. Chapman and Hall/CRC, 2006.

[4] M. Bellare, R. Canetti, H. Krawczyk. Pseudorandom functions revisited: The cascade construction and its security. Proceedings 37th Annual Symposium on the Foundations of Computer Science, IEEE, 1996.

[5] M. Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. Advances in Cryptology - Crypto 2006 Proceedings, Lecture Notes in Computer Science Vol. 4117, C. Dwork ed, Springer-Verlag, 2006.

[6] K. Yasuda. HMAC without the „second“ key. ISC '09 Proceedings of the 12th International Conference on Information Security, 2009.

[7] IPsec. http://en.wikipedia.org/wiki/IPsec, Januar 2011.

[8] TLS. http://en.wikipedia.org/wiki/Transport_Layer_Security, Januar 2011.

[9] RFC 4835. http://tools.ietf.org/html/rfc4835, Januar 2011.

[10] RFC 5246. http://tools.ietf.org/html/rfc5246, Januar 2011.

[11] The Skein Hash Function Family. http://www.skein-hash.info, Januar 2011.

[12] Hash function Groestl. http://www.groestl.info, Januar 2011.