Ministerstvo financií
Slovenskej republiky

EFQM
Recognised for excellence
5 star - 2012

Hašovacie funkcie a MAC

M. Stanek

KPMG
cutting through complexity™

Hašovacie funkcie a MAC

Martin Stanek

Obsah

1. Hašovacie funkcie
   - Vlastnosti
   - Narodeninový útok
   - Konštrukcie
2. Autentizačné kódy správ (MAC)
   - Vlastnosti
   - Generické útoky
   - Konštrukcie
3. KDF (Funkcie pre odvodenie kľúčov)

Hašovacie funkcie – úvod

- (kryptografické) hašovacie funkcie vypočítajú pre správu/dokument (takmer) ľubovoľnej dĺžky odtlačok pevnej dĺžky
- Deterministická funkcia h: $X \rightarrow Y$
  - Efektívnosť (teda dostatočne rýchla)
  - Nepoužíva kľúč
- Zvyčajne

  $X = \{0,1\}^*$, $X = \{0,1\}^{\leq 2^{64}}$, $X = \{0,1\}^{\leq 2^{128}}$ a pod.

  $Y = \{0,1\}^{160}$ (napr. pre SHA-1)

  $Y = \{0,1\}^{256}$ (napr. pre SHA-256) a pod.

Použitie hašovacích funkcií

- Schémy pre digitálne podpisy
  - Podpisuje resp. overuje sa odtlačok správy
- Konštrukcia jednorazových podpisových schém
- Výplňové schémy pre asymetrické šifrovanie (OAEP)
- Overovanie integrity dát
  - Neúmyselné zmeny údajov („náhodné“ chyby pri prenose)
  - Úmyselné zmeny & útočník nevie zmeniť odtlačok (off-line)
- Inštancia náhodných orákul a pseudonáhodných funkcií
- Odvodzovanie kryptografických kľúčov (KDF konštrukcie)
- Konštrukcie MAC (HMAC)
- Ukladanie používateľských hesiel atď.

Základné vlastnosti (neformálne)

- Odolnosť vzoru (jednosmernosť):

  pre dané $y \in h(X)$ je ťažké nájsť $x \in X$ také, že $h(x) = y$

- Odolnosť 2. vzoru:

  pre dané $x \in X$ je ťažké nájsť $x' \in X$ také, že
  $x \neq x'$ a zároveň $h(x) = h(x')$

- Odolnosť voči kolíziám:

  je ťažké nájsť x, $x' \in X$ také, že
  $x \neq x'$ a zároveň $h(x) = h(x')$

Poznámky k vlastnostiam

- V praxi $|X| >> |Y|$, inak je hašovacia funkcia nepoužiteľná
  $\Rightarrow$ *h* má veľmi veľa kolízií (len ich musí byť ťažké nájsť)
- *Y* je konečné a *h* je deterministické – teda napr. efektívny algoritmus pre nájdenie kolízií musí existovať
  - Hodnoty „zakódované“ v programe
  - Formálne definície pre vlastnosti h.f. sú (trocha) zložitejšie
  - Pre praktické účely postačuje intuitívna predstava
- Rôzne konštrukcie vyžadujú rôzne vlastnosti h.f.
  - Napr. pre bezpečnosť HMAC nie je potrebná odolnosť voči kolíziám
- Rôzne vlastnosti h.f.:
  - Pre, Sec, Coll, (aPre, ePre, aSec, eSec), MAC, Prf, Pro, TCR, CTFP, ...

Vzťahy medzi vlastnosťami

- Odolnosť voči kolíziám $\Rightarrow$ odolnosť 2. vzoru
  - Ak vieme nájsť 2. vzor, tak máme kolíziu
- Vo všeobecnosti:
  - Odolnosť voči kolíziám ***neimplikuje*** odolnosť vzoru
  - Odolnosť voči 2. vzoru ***neimplikuje*** odolnosť vzoru

- Za „obvyklých“ okolností ...
  - ak vieme invertovať aj na neinjektívnej časti *h* ... kolízie

Hľadanie vzoru

- Generický útok (pre dané $y \in Y$):
  1. Zvolíme náhodne alebo systematicky $x \in X$
  2. Ak $h(x) = y$ tak sme našli vzor, inak postup opakujeme

- Pre $Y = \{0,1\}^n$ je očakávaná zložitosť $\sim 2^n$
- Pre hľadanie 2. vzoru rovnako ...

Narodeninový útok – idea

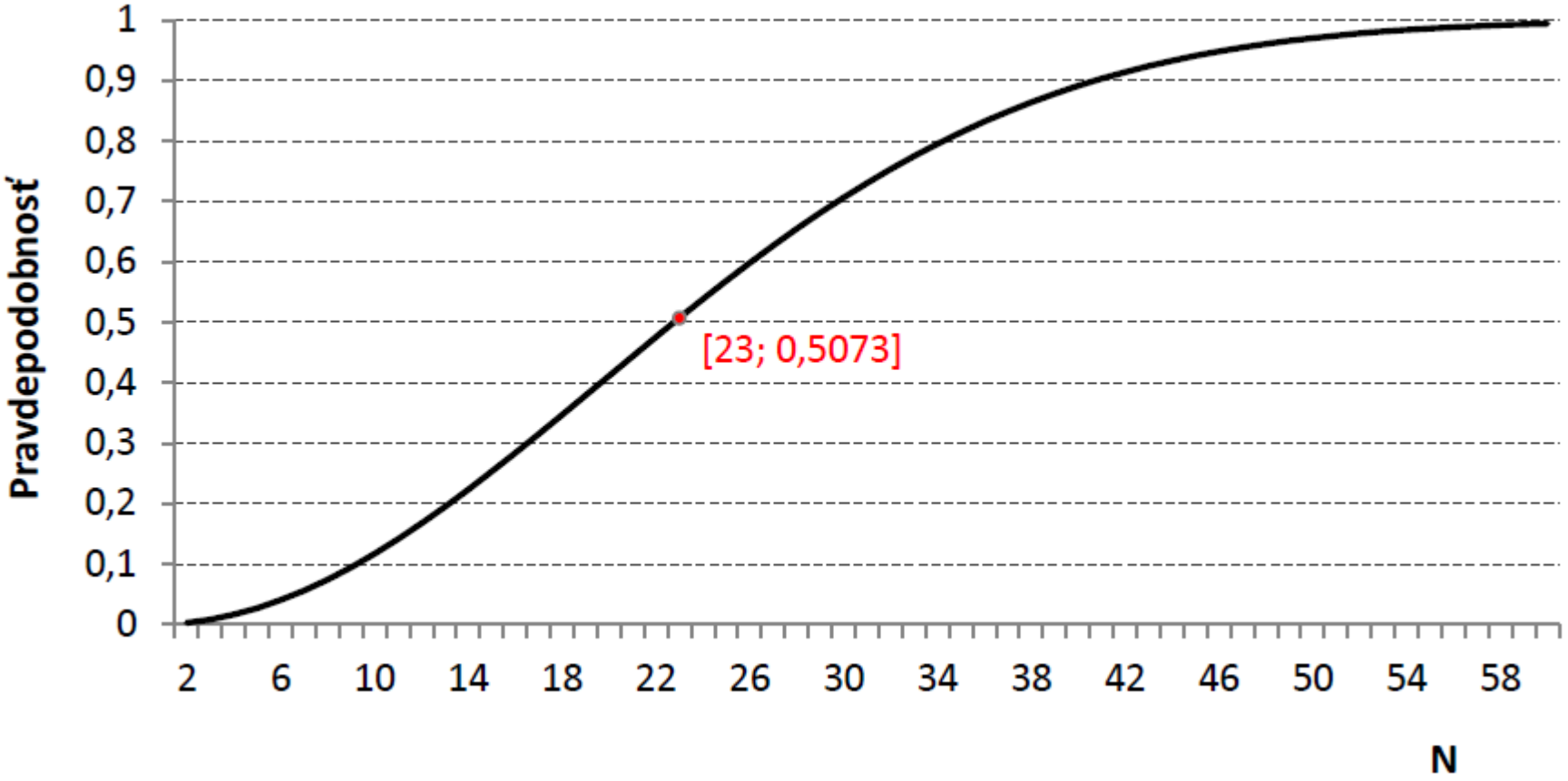
- Generický útok na nájdenie kolízie v h.f.
- Aká je pravdepodobnosť, že aspoň dve osoby v miestnosti majú narodeniny v ten istý deň?

$$\Pr_k = 1 - (365 \cdot 364 \cdot \ldots \cdot (365 - k + 1)) / 365^k$$

- 23 osôb stačí na dosiahnutie pravdepodobnosti aspoň ½
- H.f. zobrazuje ľudí na dni v roku, $|Y| = 365$

- Idea: vytvoreme odtlačky k veľkému počtu (rôznych) správ a hľadajme medzi nimi rovnaké

Narodeninový paradox – pravdepodobnosť pre N ľudí



Narodeninový útok – priebeh

- Priebeh:
  1. Zvolíme k rôznych hodnôt $x_1, x_2, ..., x_k \in X$
  2. Vypočítame odtlačky $y_i = h(x_i)$ pre $i = 1, ..., k$, pričom hľadáme zhodu (kolíziu)

- Predpokladajme $Y = \{0,1\}^n$
- Očakávané k ak chceme pravdepodobnosť úspechu aspoň:

50%	$k \approx 1{,}177 \cdot 2^{n/2}$
90%	$k \approx 2{,}146 \cdot 2^{n/2}$
99%	$k \approx 3{,}035 \cdot 2^{n/2}$

- Čím je h.f. „nerovnomernejšia“, tým je pr. úspechu vyššia
  - pri rovnakom počte k

Dôsledky narodeninového útoku

- Realizovateľný na ľubovoľnú h.f.
- Ak chceme „*n*-bitovú bezpečnosť“, výstup h.f. musí mať dĺžku aspoň 2*n* bitov.

- Štandardizované parametre pre AES a sadu h.f. SHA-2:

dĺžky kľúča AES		128	192	256
dĺžky výstupu pre SHA-2	224	256	384	512

„Zmysluplné“ kolízie

- Vstup pre h.f. ako dokument (nie náhodná hodnota).
- Pripravíme *m*, *m’* s *t* miestami, ktoré môžu byť zmenené bez zmeny významu dokumentu
  - Jedna vs. dve medzery, synonymá a pod.
- 2^t variantov *m* aj *m’* – vstupy do narodeninového útoku
  - Hľadáme kolíziu medzi dvoma množinami variantov
- Zložitosť asymtoticky rovnaká ako v „klasickom“ variante

Kolízie v konštantnej pamäti

- Klasický narodeninový útok vyžaduje pomerne veľkú pamäť:

  $\approx 1{,}177 \cdot n \cdot 2^{n/2}$ (počítajúc len veľkosť odtlačkov)

  napr. pre $n = 80$ (12,9 TB), $n = 128$ ($3{,}5 \cdot 10^8$ TB)

- Postup:
  1. Zvolíme náhodné $x \in X$
  2. Počítame (x_i, x_{2i}) for $i \geq 1$: $x_i = h(x_{i-1})$, $x_{2i} = h(h(x_{2(i-1)}))$
  3. Ak $x_i = x_{2i}$, tak x_{i-1} a $h(x_{2(i-1)})$ s vysokou pravdepodobnosťou tvoria kolíziu
- Pamäť: len aktuálna a predchádzajúca dvojica x_i a x_{2i}
- Rovnaká asymptotická zložitosť (iná konštanta)
- Bez možnosti riadiť podobu kolidujúcich vstupov

Konštrukcie hašovacích funkcií

- Konštrukcie využívajúce ťažké problémy
  - Teoreticky možné, dokázateľné vlastnosti, prakticky nepoužiteľné
- Konštrukcie z blokových šifier
- Špecifické konštrukcie

- Dlhé vstupy – iterované konštrukcie

Konštrukcie z blokových šifier

- Vstup rozdelený na bloky: $m = m_1, m_2, ..., m_k$
- h_0 – inicializačný vektor; h_i – medzihodnota pre $i = 1, ..., k$
- Postupné spracovanie blokov (iterácia)
- Príklady tzv. kompresnej funkcie (E je bloková šifra):
  - Matyas, Meyer, Oseas: $h_i = E_{g(h_{i-1})}(m_i) \oplus m_i$
  - Davies, Meyer: $h_i = E_{m_i}(h_{i-1}) \oplus h_{i-1}$
  - Miyaguchi, Preneel: $h_i = E_{g(h_{i-1})}(m_i) \oplus h_{i-1} \oplus m_i$
- Výstup: $h(m) = h_k$ (ostatný mezivýsledok)
- Problém: štandardné blokové šifry majú krátky blok
  - Špecifické šifry (SHACAL pre SHA-1, W šifra pre Whirlpool atď.)
  - Konštrukcie so zdvojenou dĺžkou bloku (Hirose, Tandem-DM atď.)

Špecifické konštrukcie

- Rýchlosť, koncentrácia na požadované vlastnosti h.f.
- Iterované konštrukcie:
  - Výplň vstupnej správy, rozdelenie na bloky rovnakej dĺžky
  - Postupné spracovanie blokov (začína sa s IV)
  - Výstup ostatnej iterácie je odtlačok (niekedy po dodatočnej transformácii)

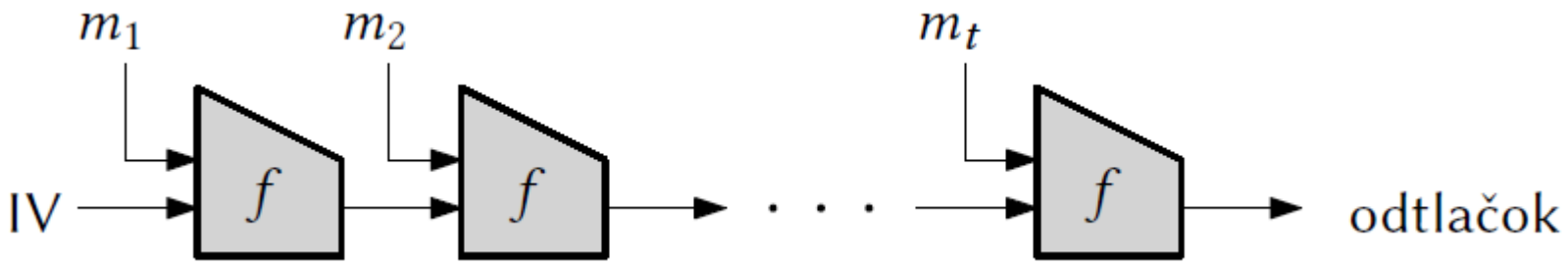
- Najčastejšie spôsoby konštrukcie:
  - Merkleho-Damgårdova konštrukcia (SHA-1, sada SHA-2)
  - „Špongiová“ konštrukcia (SHA-3)

Parametre používaných h.f.

sada	funkcia	dĺžka [počet bitov]		
		max. vstup	výstup	blok
	MD-5	$2^{64} - 1$	128	512
	SHA-1	$2^{64} - 1$	160	512
	Whirlpool	$2^{256} - 1$	512	512
SHA-2	SHA-256	$2^{64} - 1$	256	512
	SHA-384	$2^{128} - 1$	384	1024
	SHA-512	$2^{128} - 1$	512	1024
SHA-3	SHA3-256	∞	256	1088
	SHA3-384	∞	384	832
	SHA3-512	∞	512	576

Merkleho-Damgårdova konštrukcia

- Iterácia kompresnej funkcie *f* (pevná dĺžka vstupu)
- Odolnosť *f* voči kolíziám $\Rightarrow$ odolnosť *h* voči kolíziám
- Bloky vstupu po výplni: m_1, m_2, ..., m_t



Zvyčajná výplň

- Obvykle:
  - Pridáme bit 1 a doplníme nulami tak, aby sa na záver posledného bloku zmestil zápis dĺžky vstupu (v bitoch)
  - MD-zosilnenie – zápis dĺžky vstupu do výplne

- Príklad: nech *m* má 62 bajtov (teda 496 bitov)

SHA-1: $m \,||\, 1 \underbrace{0 \ldots 0}_{463} \,||\, \underbrace{0 \ldots 0111110000}_{64}$ ← 2 bloky (2 x 512 bitov)

SHA-512: $m \,||\, 1 \underbrace{0 \ldots 0}_{399} \,||\, \underbrace{0 \ldots 0111110000}_{128}$ ← 1 blok (1 x 1024 bitov)

Na zamyslenie

Predpokladajme, že výplň by spočívala len v pridaní reťazca 00...00 na zarovnanie na násobok dĺžky bloku kompresnej funkcie. Aké problémy by to znamenalo pre Merkleho-Damgårdovu konštrukciu?

Sada hašovacích funkcií SHA-2

- Štandard FIPS PUB 180-4
- SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 a SHA-512/256
- Merkleho-Damgårdova konštrukcia
- Podobná štruktúra SHA-256 (32-bitové slová, dĺžka bloku 512) a SHA-512 (64-bitové slová, dĺžka bloku 1024 bitov)
- Iné varianty sú skrátený výstup s inými IV

SHA-256 (kompresná funkcia)

- 32-bitové slová
- Vstupy:
  - Predchádzajúci medzivýsledok (256 bitov): H_0, H_1, ..., H_7
  - Blok vstupnej správy (16 slov): M_0, M_1, ..., M_{15}
- Výpočet (zjednodušene):
  1. Expanzia bloku: $(M_0, M_1, ..., M_{15}) \rightarrow (W_0, W_1, ..., W_{63})$
  2. $(a, b, c, d, e, f, g, h) \leftarrow (H_0, H_1, ..., H_7)$
  3. Pre $t = 0 ... 63$:
     1. $T_1 = F_1(e, f, g, h, t, W_t)$
     2. $T_2 = F_2(a, b, c)$
     3. $(a, b, c, d, e, f, g, h) \leftarrow (T_1 + T_2, a, b, c, d + T_1, e, f, g)$
  4. $(H_0, H_1, ..., H_7) \leftarrow (a + H_0, b + H_1, ..., h + H_7)$

- iný pohľad: bloková šifra SHACAL-2 + Davies-Meyer

Ilustračné výkonové charakteristiky

Hašovacia funkcia	MB/s
MD5	680
SHA-1	707
SHA-256	212
SHA-512	331

- Veľkosť bloku 8192 B
- CPU i7-2600 @ 3.4 GHz
- Implementácia openssl 1.0.1

Problémy s MD konštrukciou

- Predlžovanie vstupu (nadviazanie na správu a výplň)
  - Problém pre zle konštruované MAC
  - 1 kolízia – ľahko konštruovať veľa ďalších kolízií
- Multikolízie s menšou zložitosťou ako pri RO
  - Veľa správ s rovnakým odtlačkom
- Hľadanie 2. vzoru pre dlhé správy je rýchlejšie ako úplné preberanie (generický útok)

a pod.

- Riešenie: napr. wide-pipe konštrukcia (medzivýsledky dvojnásobnej dĺžky)

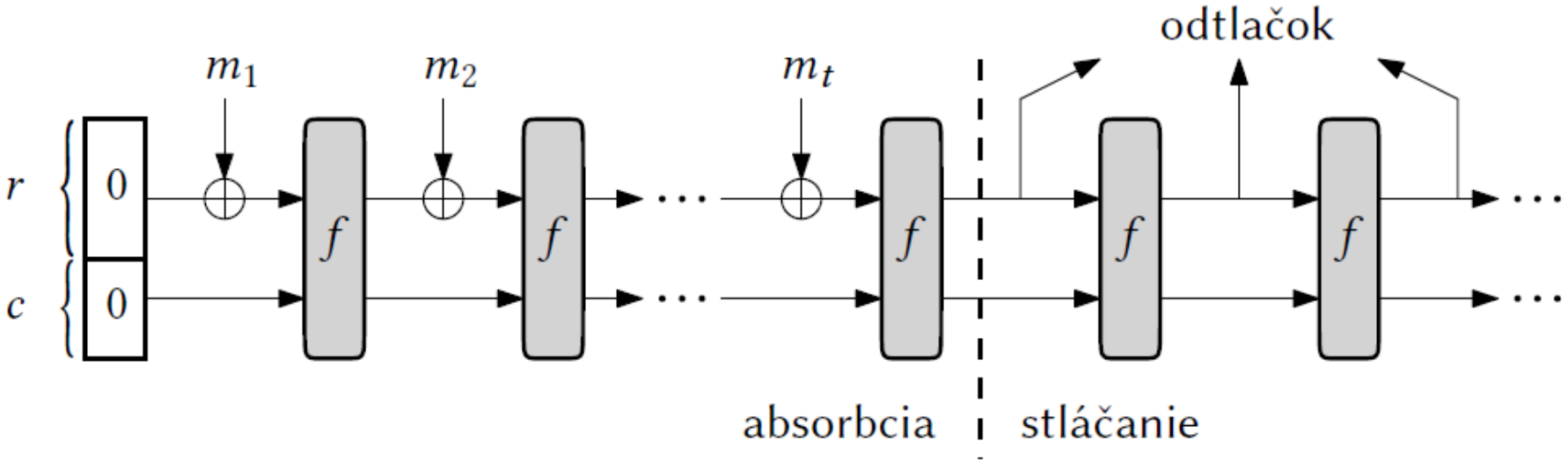
SHA-3

- Verejný výber nového štandardu
- Víťaz (2012): h.f. Keccak
- Aktuálny stav (k 27.10.2014):
  - Draft FIPS 202, *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions* (May 2014)
- 4 algoritmy h.f. s pevnou dĺžkou odtlačku:
  - SHA3-224, SHA3-256, SHA3-384, SHA3-512
- 2 funkcie s voliteľnou dĺžkou výstupu
  - XOF – extendable-output functions
  - SHAKE128, SHAKE256

- Nie je to MD konštrukcia!

Špongiová konštrukcia

- f – permutácia množiny $\{0,1\}^{r+c}$
- Parametre:
  - r – rýchlosť (pre SHA3-256 je 1088)
  - c – kapacita (pre SHA3-256 je 512)
- Výplň: m || 01 || 10*1 (návrh pre SHA3-256)



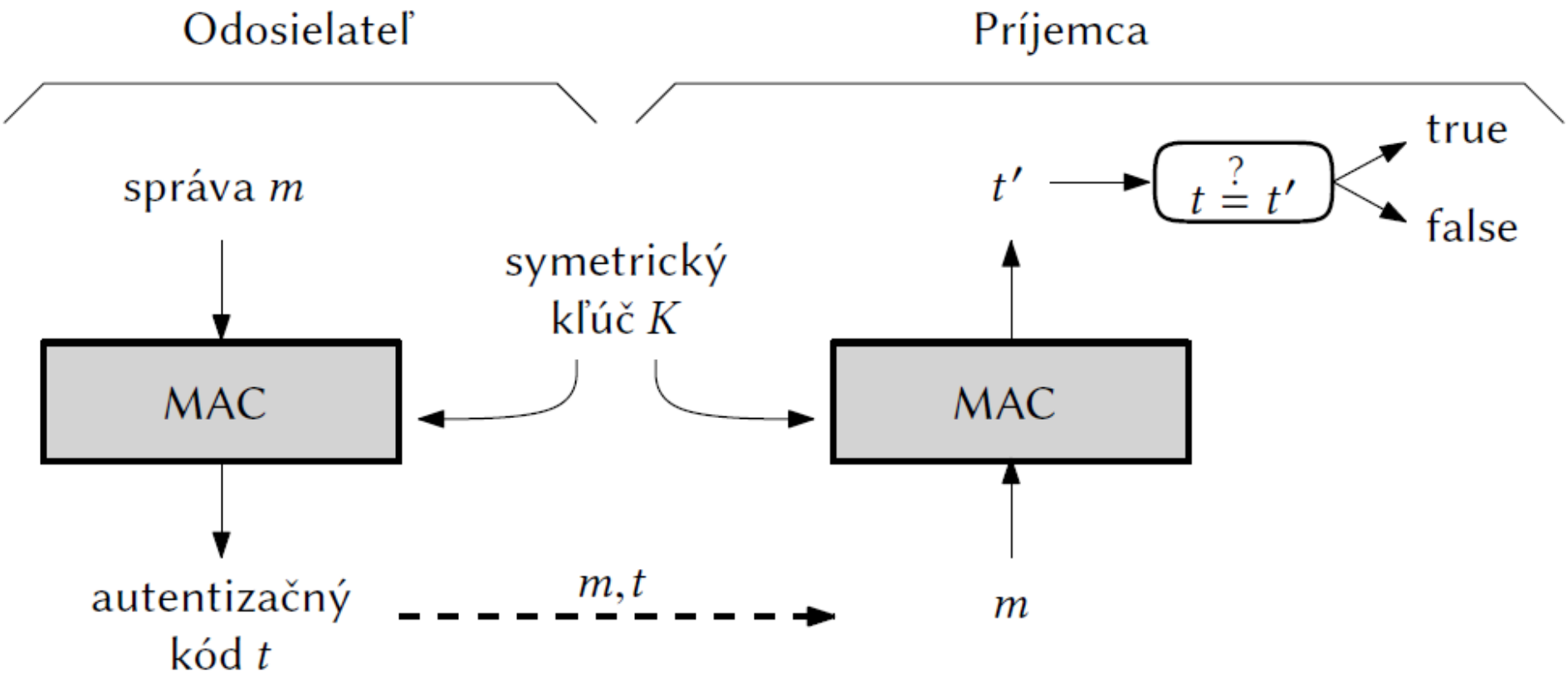
Bezpečnosť hašovacích funkcií

- MD5
  - Ľahké generovanie kolízií (v podstate okamžité)
  - Kolidujúce dokumenty (PS, XLS, ...) – aj jedna kolízia stačí
  - Generovanie falošného X.509 certifikátu CA (chosen prefix collision attack)
- SHA-1
  - Zatiaľ nie sú známe konkrétne kolízie (odhadovaná zložitosť 2^{61})
  - Neodporúča sa používať (ak je potrebná odolnosť voči kolíziám)
- Sada SHA-2
  - Zatiaľ len útoky na redukované verzie (menší počet kôl)

- Zlyhanie konkrétnej h.f. vzhľadom na nejakú vlastnosť neznamená rozbitie všetkých konštrukcií, kde je použitá.
  - Bezpečnosť HMAC nezávisí na odolnosti použitej h.f. voči kolíziám

Autentizačné kódy správ

- MAC (Message Authentication Code)
- Zabezpečenie integrity a autentickosti dát
  - Obvykle v protokoloch ako TLS, SSH, IPSec
- Symetrická konštrukcia
  - Tajný kľúč pozná príjemca aj odosielateľ
  - Kľúč je nevyhnutný pre výpočet a verifikáciu MAC
  - Nezabezpečujú nepopierateľnosť autorstva (!)
- Rýchlejšie ako digitálne podpisy
- MAC ~ hašovacia funkcia s kľúčom
- Požiadavky ako obvykle:
  - efektívnosť
  - bezpečnosť


Odosielateľ
Príjemca
správa m
symetrický
kľúč K
MAC
MAC
t'
$t \stackrel{?}{=} t'$
true
false
autentizačný
kód t
m, t
m

Bezpečnosť MAC

- Generické útoky:
  - Uhádnuť kľúč $\sim 2^k$ (pre náhodný kľúč dĺžky k bitov)
  - Uhádnuť kód $\sim 2^t$ (pre kód dĺžky t bitov)

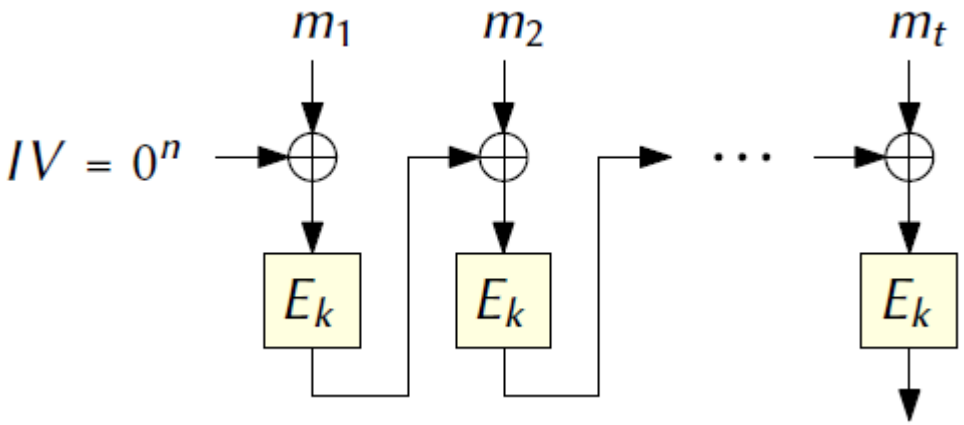
- MAC používa (tajný) kľúč, narodeninový útok vo všeobecnosti nie je aplikovateľný
  - Výstup MAC môže byť kratší ako výstup hašovacej funkcie
  - Napr. IPSec: HMAC-SHA1-96 (skrátený HMAC)

Bezpečnosť MAC (formálnejšie)

- MAC je konštrukcia zahŕňajúca tri algoritmy:
  - Generovanie kľúča, Výpočet autentizačného kódu, Verifikácia
- Útočníkovi (opäť efektívny) umožníme prístup k Mac_K orákulu (pre náhodné K)
- Bezpečný MAC: Existenciálne nefalšovateľné pri útoku s adaptívnou voľbou správ.
- Útočník nevie nezanedbateľnou pravdepodobnosťou vypočítať (m, t) také, že $\text{Vrf}_K(m, t) = \text{true}$, pričom sa nepýtal na MAC hodnotu správy m.

CBC-MAC

- MAC konštruovaný z blokovej šifry



- Bezpečné pre správy (vstupy) pevnej dĺžky
  - Predpoklad: *E* je pseudonáhodná permutácia
- Nie je bezpečné pre správy s variabilnou dĺžkou

CBC-MAC – problém s variabilnou dĺžkou

Útočník A:

- spýta sa Mac_K orákula na 1-blokové správy m a m':

  $t = \text{Mac}_K(m) = E_K(m)$ a $t' = \text{Mac}_K(m') = E_K(m')$

- spýta sa orákula na 2-blokovú správu $m \,||\, x$:

  $t^* = E_K(E_K(m) \oplus x)$

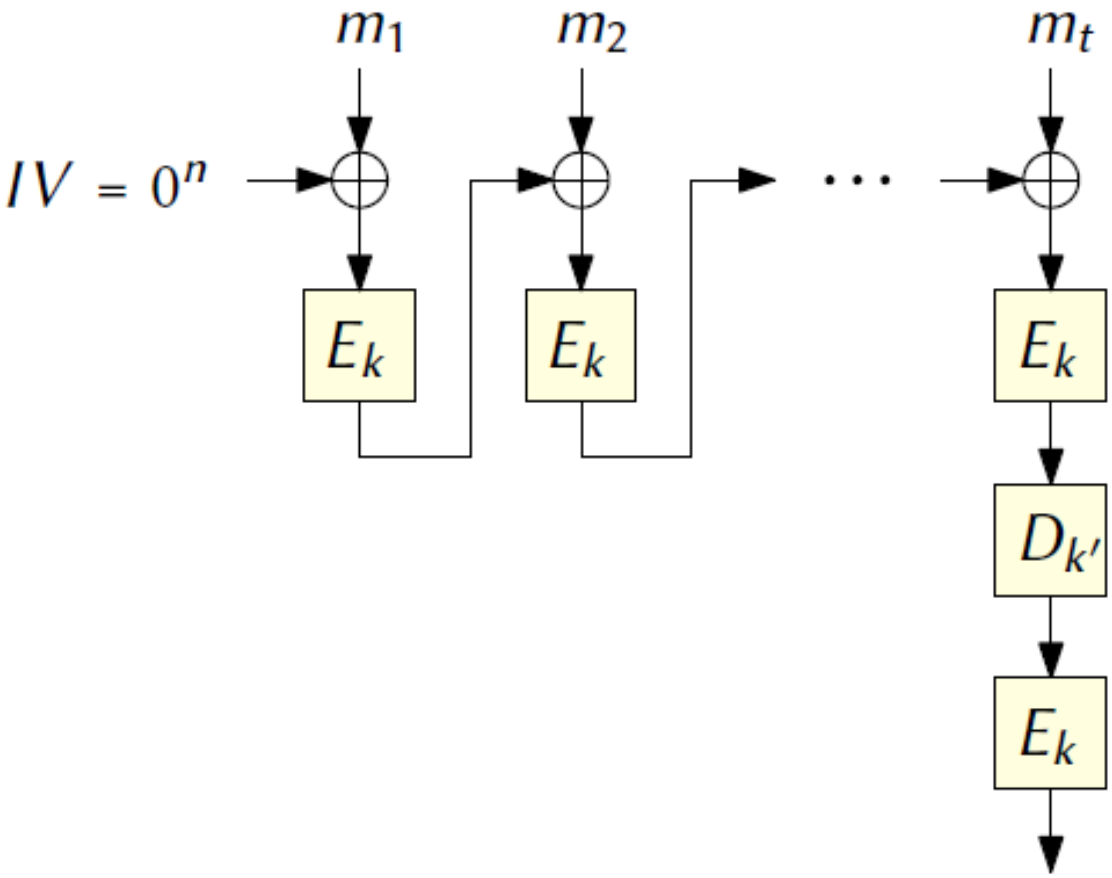
- MAC pre 2-blokovú správu $m' \,||\, E_K(m) \oplus E_K(m') \oplus x$:

$E_K(E_K(m') \oplus E_K(m) \oplus E_K(m') \oplus x) = E_K(E_K(m) \oplus x) = t^*$

teda A pozná korektný MAC pre túto správu bez pýtania sa orákula

Potenciálna oprava CBC

- Dva rôzne kľúče (potenciálne odvodené z jedného kľúča)



CMAC

- Autentizačný mód blokovej šifry, schválený NIST (SP 800-38B)
- Zjednodušená prezentácia
  - Predpokladajme, že dĺžka vstupu je násobkom dĺžky bloku E
- $m = m_1, ..., m_t$
- $l = E_K(0)$; $K_1 = MSB(l)\ ?\ (l << 1) \oplus R : l << 1$
  kde R je konštanta na dĺžke bloku, napr. $R_{128}=0^{120}10000111$
- Posledný blok sa predspracuje takto: $m'_t = m_t \oplus K_1$

- CBC spracovanie ($C_0 = 0$):

1. $C_i = E_K(C_{i-1} \oplus m_i)$ pre $i = 1,..., t - 1$
2. $C_t = E_K(C_{t-1} \oplus m'_t)$ záverečné kolo
3. výstup: C_t (môže byť aj skrátený)

Konštrukcie z hašovacích funkcií

- Jednoduché nápady nefungujú
- $\mathrm{Mac}_K(m) = H(K \,||\, m)$
  - Ak je H iterovaná h.f., napr. MD-konštrukcia, útočník vie predĺžiť m a dopočítať MAC
- $\mathrm{Mac}_K(m) = H(m \,||\, K)$
  - Pre niektoré iterované h.f., napr. MD-konštrukcie, nájdenie kolízie znamená kolíziu v MAC
  - Bezpečnosť znížená na odolnosť voči kolíziám
- Ľahké je skúsiť „fixnúť“, napr. $H(K \,||\, m \,||\, K)$
  - Vieme dokázať bezpečnosť konštrukcie pre nejaké zmysluplné predpoklady o H?

HMAC

- Najpopulárnejšia konštrukcia MAC z h.f.
  - SSL/TLS, SSH, IPSec, ...
- Dokázateľne bezpečná
  - Predpoklad: kompresná funkcia hašovacej funkcie *H* je PRF

$$\mathrm{HMAC}_K(m) = H(K \oplus \mathrm{opad} \,||\, H(K \oplus \mathrm{ipad} \,||\, m))$$

- opad/ipad – vonkajšia/vnútorná konštanta
  - opad ≠ ipad, typicky 0x5c5c... / 0x3636 ...
  - teda napr. 64 bajtov pre MD5, SHA-1 alebo SHA-256
- Takmer rovnako rýchle ako *H* (len 3 bloky navyše)
- Často sa výstup skracuje (napr. v IPSec)

Kombinovaná konštrukcia

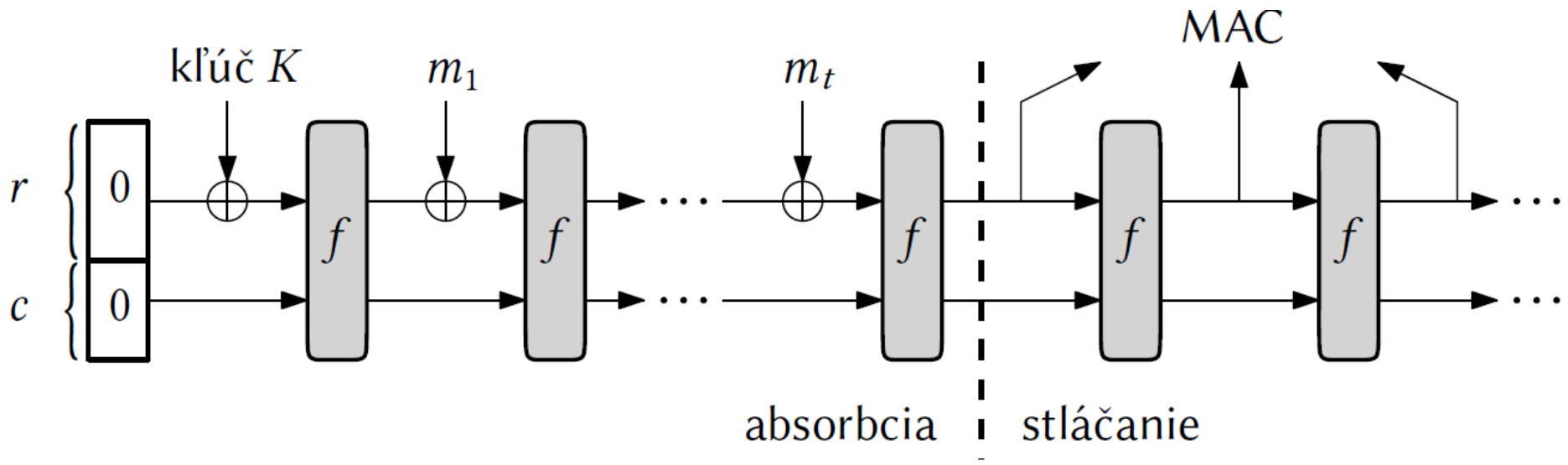
- Ďalší jednoduchý nápad, spojenie h.f. a blokovej šifry

$$\mathrm{Mac}_K(m) = E_K(H(m))$$

- Dokázateľná bezpečnosť
  - Predpoklady: *E* je PRP a *H* je odolná voči kolíziám
- Nevýhody:
  - Silnejšie predpoklady ako HMAC (teda bez motivácie používať)
- Blokové šifry zvyčajne s krátkou dĺžkou bloku *n* a kvôli kolíziám zodpovedá bezpečnosť *n*/2 bitom
  - Napr. AES so 128-bitovým blokom (and skráteným odtlačkom *H*) vedie k 64 bitovej bezpečnosti

MAC zo špongiovej konštrukcie

- Jednoduchá konštrukcia MAC
- Neštandardizované (!)



Bezpečný komunikačný kanál

- Dôvernosť a autentickosť naraz
  - Autentizované šifrovanie
  - Kombinácia symetrickej šifry a MAC
- Nezávislé kľúče K_1 (šifrovanie) a K_2 (MAC)
- Šifruj potom MAC (napr. IPSec):

  c, $\text{Mac}_{K_2}(c)$, kde $c = E_{K_1}(m)$
- MAC potom šifruj (napr. SSL/TLS):

  $E_{K_1}(m \,||\, \text{Mac}_{K_2}(m))$
- Šifruj a MAC (napr. SSH):

  $E_{K_1}(m)$, $\text{Mac}_{K_2}(m)$

Bezpečný komunikačný kanál (2)

- Podľa teórie je bezpečná konštrukcia „šifruj potom MAC“
- Iné konštrukcie môžu byť bezpečné pri vhodnej implementácii

- MAC nezabezpečuje ochranu pred opakovaním správ
  - Číslovanie správ / paketov, časové údaje a pod.

Implementačné problémy

- Obvyklý scenár: server porovná prijatý MAC s vypočítanou hodnotou
- Cieľ útočníka: získať korektný MAC k vlastnej správe
- Čo je zlé na tomto kóde (Python):

```
return self.Sign(msg) == sig_bytes
```

  - msg – prijatá správa
  - sig_bytes – prijatý MAC

- alebo na tomto kóde (Java):

```
public static boolean
isEqual(byte digesta[], byte digestb[]) {

  if (digesta.length != digestb.length)
    return false;
  for (int i = 0; i < digesta.length; i++) {
    if (digesta[i] != digestb[i])
      return false;
  }
  return true;
}
```

Rekonštrukcia MAC – timing attack

- Ako rýchlo server zareaguje na poslaný MAC?
  - Ak je prvých 0, 1, 8 or 15 bajtov správnych?
- Rekonštrukcia MAC na základe času získania odpovedí servera
- 4. bajt:

```
71 A0 89 00 00 ... 00
71 A0 89 01 00 ... 00
...
71 A0 89 4A 00 ... 00    dlhší čas?
...
71 A0 89 FF 00 ... 00
```

- Zvyčajne viacero meraní potrebných na redukciu šumu a rozpoznanie správnej hodnoty (štatistické vyhodnotenie meraní)

KDF

- Odvodenie symetrických kľúčov
  - z hesla – napr. pre šifrovanie, MAC
  - z iných hodnôt dohodnutých napr. v protokoloch
- Častokrát špecifické KDF pre konkrétny účel/protokol
  - IKEv1, IKEv2, TLS (rôzne verzie), SSH, SNMP, TPM, ...
  - Pozri napr. NIST SP 800-135 Revision 1 Recommendation for Existing Application-Specific Key Derivation Functions
- TLS
- PBKDF2
  - NIST SP 800-132 Recommendation for Password-Based Key Derivation

KDF v TLS 1.0 a 1.1

- Master secret (48B):
  MS = PRF(preMS, „master secret“, CRnd || SRnd)
- Kľúče, príp. IV:
  PRF(MS, „key expansion“, SRnd || CRnd)

- PRF(secret, label, seed) – pseudonáhodná funkcia

  PRF(secret, label, seed) =
  P_MD5(S1, label || seed) ⊕ P_SHA1(S2, label || seed)

  - S1 a S2 sú l’avá a pravá polovica secret

KDF v TLS 1.0 a 1.1 (2)

P_hash(secret, seed) – expanzná funkcia

P_hash(secret, seed) = HMAC_hash(secret, A(1) || seed) ||
HMAC_hash(secret, A(2) || seed) ||
HMAC_hash(secret, A(3) || seed) ||
...

A(0) = seed

A(i) = HMAC_hash(secret, A(i-1))

TLS 1.2: hash = SHA256
PRF(secret, label, seed) = P_hash(secret, label || seed)

PBKDF2

- vstupy: heslo *P*, soľ *S*, počítadlo *c*, požadovaná dĺžka *d* výstupu v bajtoch
- soľ
  - obvykle náhodný reťazec bitov, nemusí byť tajný
  - k jednému heslu zodpovedá potenciálne veľa kľúčov
  - znemožňuje predvypočítať všetky kľúče k slovníkovým heslám
  - ak nie je možné generovať soľ náhodne, dá sa vytvoriť deterministicky, napr. KDF(*P*, *M*), kde *M* je správa (nie však, ak je priestor správ malý)
- počítadlo
  - zvyšovanie zložitosti funkcie a teda aj útoku, napr. 1000

PBKDF2 (2)

- výstup: $T_1 \,||\, T_2 \,||$ ... koľko je potrebné
- max. dĺžka výstupu $(2^{32} - 1) \cdot t$, kde t je dĺžka výstupu h.f.
  - napr. pre SHA-1 je to cca. 86 GB
- výpočet: $T_i = F(P, S, c, i)$, kde

  $F(P, S, c, i) = U_1 \oplus U_2 \oplus \ldots \oplus U_c$

  $U_1 = PRF(P, S \,||\, INT(i))$ kde INT vráti 4B reprezentáciu

  $U_2 = PRF(P, U_1)$

  ...

  $U_c = PRF(P, U_{c-1})$

- štandardná PRF je HMAC-SHA-1:
  - $PRF(a, b) = HMAC_a(b)$
  - bežne sa používajú aj HMAC s SHA-256 a pod.
- alternatívy PBKDF2: bcrypt, scrypt (vyššia pamäťová zložitosť)

Ďakujem za pozornosť