# Kryptologie pro praxi – funkce HMAC

## Hašovací funkce a HMAC

Hašovací funkce se používají velmi hojně v celé symetrické i asymetrické kryptografii. V moderních standardech a protokolech (například SSL/TLS) se to jimi a klíčovanými hašovými kódy (Keyed-Hash Message Authentication Code, HMAC) jen hemží. Hašovací funkce zpracovávají libovolně dlouhá vstupní data ($M$) na výstupní hašový kód (*haš*, *hash*) $h(M)$, který má předem danou pevnou délku, obvykle stovky bitů. Hašovací funkce musí být jednocestná (one-way) a bezkolizní (collission-free). Jednocestnost znamená, že z $M$ lze $h(M)$ vypočítat jednoduše, ale naopak to není výpočetně zvládnutelné. Tato vlastnost se využívá ke kontrole hesel v operačních systémech. Hesla zde nejsou uložena přímo, ale pouze jejich haše. Pokud nějaký útočník získá *haš* hesla nějakého uživatele, díky jednocestnosti není z této hodnoty schopen určit uživatelovo heslo (pochopitelně, vyloučíme-li triviální slovníkový útok). Bezkolizност požaduje, aby výpočetně nezvládnutelné bylo i nalezení libovolných dvou různých zpráv (byť naprosto nesmyslných) $M$ a $M'$ se stejným hašovým kódem $h(M)=h(M')$. Tuto situaci nazýváme kolizí. Protože zpráv je mnohonásobně více, než hašových kódů, kolize musí teoreticky existovat. Pointa hašovacích funkcí je v tom, že díky výpočetní složitosti je nejsme schopni prakticky nalézt. Bezkolizност tak umožňuje místo zprávy $M$ (například celý obsah 80 GB HDD) podepsat pouze její *haš* $h(M)$, tedy řádově stovky bitů. U digitálních podpisů se proto *haš* vstupní zprávy často nazývá výtah zprávy (message digest) nebo digitální otisk (digital fingerprint). Díky bezkolizности haš identifikuje zprávu $M$. Dále, kdokoliv se může přesvědčit, že zpráva nebyla narušena znovuvypočtením jejího hašového kódu $h(M)$, proto se haše mohou používat stejně jako kódy detekující chybu, jsou ovšem mnohem pomalejší (než například *CRC*). Malou modifikací haše lze vytvářet klíčované hašové autentizační kódy zpráv (*HMAC*), které mají tu vlastnost, že společně se zprávou $M$ hašují i nějaký tajný klíč $K$. Výsledkem je zabezpečovací kód $HMAC(M,K)$, který nejen detekuje chybu, ale případnému útočníkovi znemožňuje pozměnit zprávu a současně i její *HMAC*. Bez znalosti tajného klíče nový *HMAC* nevypočítá, což v případě čisté haše udělat mohl. *HMAC* může být proto chápán jako nepadělatelný integritní kód. V různých protokolech funguje *HMAC* jako průkaz znalosti tajného sdíleného tajemství ($K$) pokud na nějakou výzvu *challenge* obdržíme odpověď $response=HMAC(challenge,K)$.

## Princip hašovacích funkcí

Princip většiny hašovacích funkcí vychází z tzv. kompresní funkce $f$ a je následující. Vstupní zpráva $M$ (téměř neomezeně dlouhá) je určitým jednoznačným způsobem doplněna bitem 1, určitým počtem nulových bitů a nakonec binárním vyjádřením délky původního vstupu tak, že vytváří celistvý násobek $L$bitových bloků $M(1)$, ... $M(n)$, přičemž $L$ bývá 512 nebo 1024. Potom se v cyklu pro $i$=1 až $n$ zpracovává vždy běžný blok $M(i)$ tak, že $H(i)=f(H(i–1), M(i))$, kde $H(i)$ je průběžný kontext hašovací funkce a $H(0)$ je nastaven na definovanou konstantu. Kontextem bývá obvykle několik 16bitových nebo 32bitových slov. Po zpracování bloku $M(n)$ se jako výstup celé hašovací funkce definuje kontext $H(n)$ nebo jeho definovaná část.

## Nejznámější hašovací funkce

Mezi nejznámější hašovací funkce patřily donedávna *MD*2, *MD*4, *MD*5 [2] a *SHA*–1 [3]. Od používání funkcí *MD* (se 128bitovým výstupním kódem) se z bezpečnostních důvodů upouští, u *MD*4 byly nalezeny kolize a u *MD*2 a *MD*5 byly nalezeny kolize jejich kompresních funkcí. V současné době je nejpoužívanější hašovací funkcí *SHA*–1 (se 160bitovou haší), která je už osm let považována za zcela bezpečnou. Nedávno byly však vydány nové standardy, definující další tři hašovací funkce nové generace *SHA*-256, *SHA*-384 a *SHA*-512 [3], které nabízí delší výstupní kód – 256, 384 a 512 bitů. Na okraj poznamenejme, že třídu funkcí *SHA* navrhla americká tajná služba NSA, zodpovědná za ochranu vládních systémů, takže jejich kvalita by měla být dostatečně garantována.

## Narozeninový paradox

Narozeninový paradox je základem pro posuzování bezpečnosti hašovacích funkcí a říká, kolik zpráv bychom museli zhašovat, abychom se dostali ke kolizi, tj. nalezli dvě různé zprávy se stejnou haší. Odpověď vyplývá z následujícího tvrzení. Mějme množinu $A$ o $n$ různých prvcích. Je-li $n$ dostatečně velké a $k$ rovno přibližně $(2n \cdot \ln 2)^{1/2}$, potom v množině o $k$ prvcích, které vybíráme z $A$ náhodně (s vracením), se přibližně s 50% pravděpodobností naleznou dva prvky shodné. Narozeninový paradox dostalo toto tvrzení z aplikace pro čísla $n$=365 a $k$=23. Říká, že postačí skupina náhodně vybraných 23 lidí k tomu, aby se mezi nimi (s asi 50% pravděpodobností) našla dvojice, slavící narozeniny ve stejný den (pro 30 lidí je to asi 70 %). Tvrzení se zdá paradoxní proto, že většinou ho chápeme ve smyslu kolik lidí je potřeba, aby se k danému člověku našel jiný, slavící narozeniny ve stejný den, tedy nehledáme jakékoliv shodné narozeniny, ale pouze jedny konkrétní. Podobné je to i s hašemi, kde $A$ odpovídá množině všech možných výsledků dané hašovací funkce. Je-li hašový kód 128bitový, $n=2^{128}$, a postačí paradoxně zhašovat „pouze“ $k=2^{64}$ zpráv, abychom s 50% pravděpodobností našli kolizi. Přitom $2^{64}$ zpráv je sice hodně, ale tento počet je už prakticky dosažitelný, i když s ohromnými náklady. Proto se z bezpečnostních důvodů upouští i od funkce *MD*5 a celé třídy *MD*, neboť jejich haše jsou pouze 128bitové. *SHA*–1 nabízí složitost vyšší, tj. $2^{80}$, což je dnes chápáno pro mnoho aplikací jako zcela dostačující, ale pro budoucnost a pro aplikace bezpečnostně kritické už jako nedostačující. Naproti tomu složitost nalezení kolizí u funkcí nové generace *SHA*–256, 384 a 512 je po řadě $2^{128}$, $2^{192}$, $2^{256}$, což je chápáno za prvé jako dostačující a za druhé to také odpovídá složitosti prolomení 128, 192 a 256bitových klíčů u nového šifrovacího standardu AES [1]. Nové hašovací funkce byly navrženy právě proto, aby jejich prolomení odpovídalo přibližně prolomení klíčů u AES.

## Klíčovaný hašový autentizační kód

Klíčovaný hašový autentizační kód zprávy $HMAC(M,K)$ je funkčně podobný autentizačnímu kódu zprávy *MAC* [5], ale místo blokové šifry využívá hašovací funkci. Označuje se konkrétně podle toho, jakou hašovací funkci používá, např. $HMAC–SHA–1(M,K)$. Je definován ve standardu FIPS 198 [4] (je tam o něco obecněji popsán, než v RFC 2104 a ANSI X9.71) a jeho definice závisí na délce bloku kompresní funkce v bajtech (např. u *SHA*–1 a *SHA*–256 je to $B$=64 bajtů, u *SHA*–384 a *SHA*–512 je to $B$=128 bajtů) a na délce hašového kódu dané hašovací funkce $H$. Pro $H=SHA–1$ a klíč $K$ máme blok $B$=64 bajtů a dále definujeme *ipad* jako řetězec $B$ bajtů s hodnotou 0×36 a *opad* jako řetězec $B$ bajtů s hodnotou 0×5C. Klíč $K$ se doplní nulovými bajty do plného bloku délky $B$ a poté $HMAC–H(M,K)=H((K \text{ xor } opad)||H((K \text{ xor } ipad)||M))$, kde $||$ označuje zřetězení. Použití *HMAC* jsme zmínili výše.

Vlastimil Klíma, Tomáš Rosa,
klima@lec.cz, trosa@ebanka.cz

*LITERATURA*

*[1] Kryptologie pro praxi – operační mód , ST 11, s. 16, 2003.*
*[2] http://www.rfc-editor.org/*
*[3] http://csrc.nist.gov/CryptoToolkit/tkhash.html*
*[4] HMAC: definován ve standardu FIPS PUB 198, viz [3] , resp. jako RFC 2104*
*[5] Kryptologie pro praxi – používané šifry, ST 9, s. 16, 2003.*
*[6] Archiv autorů: http://cryptography.hyperlink.cz a http://crypto.hyperlink.cz*