

# Odvozování klíčů (3) – z passwordů

Klíče hrají v moderní počítačové a internetové kryptografii zásadní roli. Vývojáři aplikací často stojí před problémem, jak je vytvářet. V předchozích dvou číslech Sdělovací techniky byly články věnovány odvozování klíčů z master klíče, který byl náhodný. Často je však k dispozici jen heslo uživatele, z něhož je nutné vše odvodit. Opět se k tomu použije funkce KDF (key derivation function), která umí ze vstupního hesla uživatele (passwordu,  $P$ ) generovat libovolné množství odvozených klíčů.

## KDF pro nenáhodné klíče

Funkce KDF pro odvozování klíčů z passwordů a pro odvozování z master klíčů se liší, protože hesla uživatelů mají nižší entropii než náhodný master klíč. Aby se tyto funkce odlišily, nazývají se PBKDF (password based key derivation functions). Zabývá se jimi např. norma PKCS#5 a nejnověji doporučení NIST [1], staré jen několik měsíců. Věnujme se [1], neboť pomocí této normy lze z passwordu vygenerovat tzv. master klíč (MK), který je pak možné použít podle postupů popsaných v minulých číslech ST. Využijme toho, že v nich byly podle platných norem také definovány dvě pseudonáhodné funkce PRF, a to PRF na bázi blokové šifry – PRF (klíč, data) = CMAC (klíč, data), nebo na bázi hašovací funkce – PRF (klíč, data) = HMAC (klíč, data). Ve funkci PBKDF vystupuje místo klíče proměnná  $P$  (password) a navíc ještě tzv. sůl ( $S$ ), čísla iterací ( $C$ ) a požadovaná délka klíče  $MK$  v bitech ( $k_{Len}$ ).  $MK$  se vygeneruje takto:

$$MK = \text{PBKDF}(\text{PRF}, C, P, S, k_{Len})$$

Upozorníme, že bezpečné klíče by podle NIST měly začínat na hodnotě  $k_{Len}$  větší nebo rovné 112 bitům.

## Sůl proti útočníkům

Hodnota soli se přidává do vzorce pro generování klíčů z passwordů proto, aby se zabránilo předgenerování tabulek pro všechny možné nebo nejpravděpodobnější hodnoty passwordů. Jak známo, tyto techniky jsou stále dosti účinné, neboť pohodlnost uživatelů se nijak nezmenšila. Přidáním soli je útočník donucen generovat případnou tabulku až po zjištění hodnoty soli. Proto má být hodnota soli podle NIST náhodný řetězec o délce alespoň 128 bitů.

## Počet iterací – miliony?!

Dalším opatřením proti útočníkům, kteří musí pro konkrétní případ zjistit konkrétní hodnotu soli (ukládá se) a zkusit vytvářet  $MK$  pomocí uvedeného vzorce, je

nutnost, aby tento vzorec byl velmi složitý. Počet iterací ( $C$ ) je počet volání PRF, než se vytvoří klíč  $MK$ . Uživatel má výhodu v tom, že zmíněný výpočet projde jen jednou; např. při přihlášení k počítači zadá password a od něho se již uvedeným vzorcem odvozuje klíč pro šifrování pevného disku. V tomto případě může výpočet trvat třeba sekundu. Útočník je na tom mnohem hůře, protože musí vyzkoušet co nejvíce passwordů, ale vyzkoušení každého passwordu mu zabere (na stejné výpočetní technice) jednu sekundu. Proto není schopen vyzkoušet příliš mnoho passwordů, což je cílem tohoto opatření. Čím větší je počet iterací, tím je technika bezpečnější. NIST doporučuje co nejvyšší hodnotu akceptovatelnou uživatelem, nejméně však 100 000(!), ale není-li otázka času kritická, doporučuje např. hodnotu 10 000 000.

## Nová funkce PBKDF

V kryptologii je nutné postupovat stejně jako u jiných funkcí, tj. pamatovat na možnost jejich aktualizace. Následující defini-

For  $i = 1$  to  $len$

$$\left\{ \begin{array}{l} T_i = 0; \\ U_0 = S \parallel \text{Int}(i); \\ \text{For } j = 1 \text{ to } C \\ \quad \left\{ \begin{array}{l} U_j = \text{HMAC}(P, U_{j-1}) \\ T_i = T_i \oplus U_j \end{array} \right. \end{array} \right.$$

Return  $MK = T_1 \parallel T_2 \parallel \dots \parallel T_{len} <0\dots r-1>$

Obr. 1 Funkce PBKDF pro generování master klíče ( $MK$ ) z passwordu ( $P$ ), soli ( $S$ ) a počtu iterací ( $C$ )

ce PBKDF používá starý dobrý vzorec podle PKCS#5 s tím, že se právě aktualizovala jen použitá hašovací funkce v HMAC (PRF). Dříve to byla SHA-1, která je již nevhodná, a byla ve vzorci nahrazena jakoukoliv jinou schválenou hašovací funkcí (např. SHA-256 nebo SHA-512). Délka výstupu hašovací funkce v bitech se značí  $h_{Len}$  a hodnota záleží na tom, jaká funkce bude zvolena. U uvedených příkladů by to bylo 256 a 512.

Poznamenejme, že ve vzorci jsou čísla  $len$  a  $r$  zvolena tak, aby bylo možné pomocí výstupu HMAC o délce  $h_{Len}$  po  $len+1$  aplikacích vygenerovat požadovaných  $k_{Len}$  bitů, přičemž z posledního  $h_{Len}$ -bitového kódu HMAC se vezme jen  $r$  potřebných bitů ( $k_{Len} = (len-1) * h_{Len} + r$ ). Vzorec vypadá složitě, neboť výsledek vzniká iterativním procesem, ale není to tak hrozné.

Jde v podstatě o to, že se generuje tolik řetězců  $T_i$ , kolik je třeba, přičemž každý z řetězců  $T_i$  je xor-sumou  $C$  hodnot HMAC, kde  $C$  je onen počet iterací. Hodnoty HMAC mají vždy klíč roven passwordu  $P$  a datový vstup se iterativně mění. Na počátku je to datový vstup  $U_0$ , což je sůl zřetězená s číslem  $i$  (vyjádřeným ve čtyřech bajtech  $\text{Int}(i)$ ). Potom v každém dalším kroku je  $U$  rovno HMAC od předchozího  $U$ . Iterativně se tak vytvoří  $C$  hodnot  $U$  a ty



se xor-sečtou. Tento proces se opakuje pro tolik hodnot  $i$ , kolik je třeba. Připomeňme článek v ST o možnosti degenerace hašovací funkce při iterativním volání – zde to přichází v úvahu u hodnot  $U$ . Protože však všechny mezivýsledky jsou xor-sčítány, k významné ztrátě entropie nedojde.

## Použití master klíče

Klíč  $MK$  je možné použít přímo jako klíč k ochraně dat (data protection key,  $DPK$ ), tj.  $DPK = MK$ . Další možnosti jsou jen komplikovanější varianty, kdy  $DPK$  se teprve z  $MK$  vytvoří, tj.  $DPK = \text{KDF}(MK)$ . Nebo jsou klíče  $DPK$  již vytvořené a uloženy (zašifrované) a pomocí  $MK$  se odšifrují pro použití. Tato varianta může být použita např. při šifrování celého disku, kdy vlastní šifrovací klíč pro šifrování disku (tj.  $DPK$ ) může být generován náhodně, přičemž na disku (v master boot recordu) je uložen zašifrovaně – právě pomocí  $MK$ , který je vypočítán z passwordu funkcí PBKDF. Výhodou je, že při změně passwordu není nutné dešifrovat a znovu zašifrovat celý disk, ale pouze v master boot recordu se vymění záznam  $E_{oldP}(DPK)$  za  $E_{newP}(DPK)$ .

Vlastimil Klíma,  
vlastimil.klima@knzsro.cz

## LITERATURA

[1] NIST Special Publication 800-132: Recommendation for Password-Based Key Derivation –Part 1: Storage Applications, Computer Security Division, ITL, NIST, June 2010.