Hašovací funkce – nejaktuálnější téma kryptologie

Jiří Hörner♣, Václav Rozhoň♠, Martin Vančura♠,
Gymnázium Pelhřimov♣, Gymnázium Jana Valeriána Jirsíka♠,
horner.jiri@gmail.com, jeniceknestaval@seznam.cz, marti.vancura@gmail.com

Abstrakt

Hašovací funkce jsou „zázračné“ funkce, které z dlouhých zpráv vyrábějí jejich krátké otisky (anglicky hash). Chovají se jako tzv. náhodná orákula, tedy otisky jsou jakoby náhodně vybírány. Dále musí funkce splňovat, že hašování, tedy získávání otisku zprávy, je rychlé, zatímco k danému otisku je výpočetně příliš náročné najít zprávu, která by měla tento otisk. Hašování má v současné době velmi široké využití (např. při přihlašování do emailu, digitálním podepisování, porovnávání velkých objemů dat atd.) Dokladem, že jde o téma vysoce aktuální, je fakt, že se do finále blíží soutěž o nový hašovací standard SHA-3. Vítěz by měl být vybrán koncem roku 2012.

1 Hašovací funkce

Na hašovací funkce jsou kladeny 3 základní požadavky:

- Jednocestnost
- Bezkoliznost
- Odolnost vůči hledání druhého vzoru

Definice 1 (jednocestná funkce)**.** *Funkci $f : X \to Y$ nazveme* jednocestná (one-way)*, pokud je f odolná vůči hledání vzoru: pro náhodně vybrané $y \in Y$ je výpočetně nemožné najít její* **vzor***, tj. $x \in X$ tak, že $y = f(x)$.*

Při použití hašování v kryptografii je tento požavek zásadní – znemožňuje zjištění původní zprávy. Stále to ale musí být funkce – pro stejný vstup musí generovat stejnou haš.

Definice 2 (funkce odolná vůči hledání druhého vzoru)**.** *Funkci $f : X \to Y$ nazveme* odolnou vůči hledání druhého vzoru*, pokud pro náhodně vybrané $x \in X$ je výpočetně nemožné najít* **druhý vzor** *$y = f(x)$, tj. $x' \in X,\ x' \neq x$ tak, že $y = f(x')$.*

Pokud je lehké najít druhý vzor, může dojít i k vážnému zneužití. Na obrázku 1 je vidět teoretická možnost podstrčení falešného dokumentu (má stejnou haš jako dokument, který byl elektronicky podepsán).

Poznámka 1. *Protože hašovací funkce většinou zobrazují velkou množinu X na malou množinu Y, druhé vzory v takovém případě vždy existují. (Funkce není prostá.)*

****

CONTRACT
At the price of **$176,495** Alf Blowfish
sells his house to Ann Bonidea ...

****

CONTRACT
At the price of **$276,495** Alf Blowfish
sells his house to Ann Bonidea ...

Obrázek 1: V roce 1996 H. Dobbertin prezentoval na konferenci FSE 1996 metodu nalézání kolizí u algoritmu MD4.

Definice 3 (bezkolizní funkce)**.** *Funkci* $f : X \to Y$ *nazveme* bezkolizní (collision-free)*, pokud je výpočetně nemožné najít* $x, x' \in X,\ x \neq x'$ *tak, že* $f(x) = f(x')$*.*

Pokud funkce není bezkolizní, není považována za kryptograficky bezpečnou, ačkoliv možnost zneužití je spíše teoretická. Snadné nalézání kolizí svědčí o některých slabinách algoritmu. Na hašovací funkce jsou kladeny ještě některé další nároky. Především se funkce má chovat jako *náhodné orákulum* (resp. pseudonáhodné orákulum), takže i při malé změně vzoru (1 bit) můžeme dostat zcela jiný výstup.

Na hašování jsou kladeny velké nároky ohledně rychlosti, a tak se v hašovacích funkcích používají jen jednoduché bitové operace – např.

- xor – pro každou dvojici bitů vrací jedna pro různé nebo nula pro stejné bity,
- add – probíhá jako normální sčítání dvou stejně dlouhých čísel, pokud je binární délka výsledku delší, zanedbáme první číslici,
- cyklický bitový posun – každý bit ve zprávě se posune o určitý počet míst doleva nebo doprava, bity, které by byly ze zprávy vyřazeny se posunou na začátek nebo na konec zprávy.

Výše zmíněné principy se dají prezentovat na *narozeninovém paradoxu.* (O paradox se v podstatě nejedná, jde spíše o neočekávaný výsledek pravděpodobnostního počtu, který dobře koresponduje s realitou.)

Věta 1 (kolize)**.** *Mějme k lidí na party, poté pravděpodobnost, že 2 lidé mají narozeniny ve stejný den, je:*

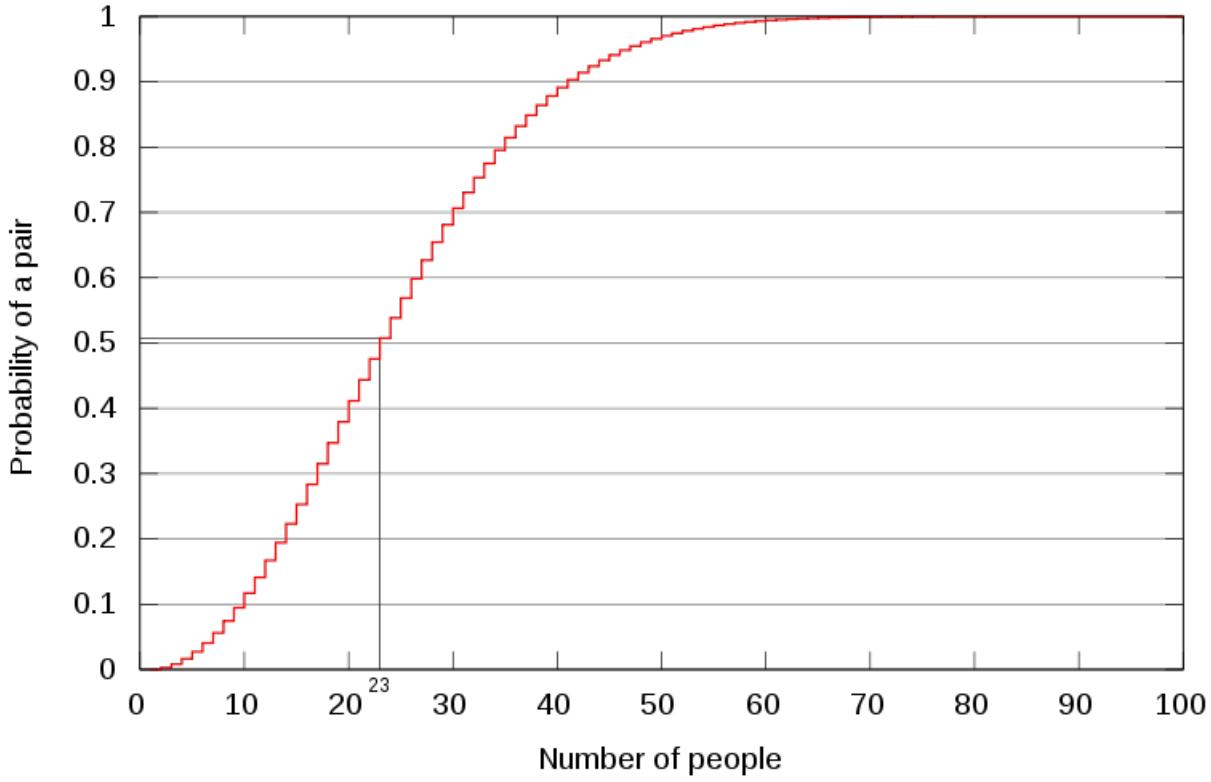
$$P(k) = 1 - \frac{365 \cdot 364 \cdot \cdots \cdot (365 - k + 1)}{365^k}.$$

Tato pravděpodobnost je překvapivě vysoká – proto paradox.

Věta 2 (druhý vzor)**.** *Mějme k lidí na party, poté pravděpodobnost, že někdo jiný má narozeniny ve stejný den jako já, je:*

$$P(k) = 1 - \frac{364^k}{365^k}.$$

Tato pravděpodobnost je daleko nižší než v prvním případě – proto je jednodušší hledat kolize než nalézt druhý vzor.



Obrázek 2: $P(365, 23) = 0,507$ a $P(365, 30) = 0,706 \Rightarrow$ ve skupině 23 lidí najdeme s pravděpodobností 50% dvojici slavící narozeniny ve stejný den, ve skupině 30 lidí s pravděpodobností 70%. Pro skupiny větší než 60 osob se pravděpodobnost blíží 100%.

2 Použití hašovacích funkcí

Hašovací funkce se využívají na mnoha místech. Podle užití se liší nároky, které na ně klademe: největší požadavky na bezkoliznost atp. jsou v kryptografii, v jiných oborech je spíše hlavní rychlost [3].

2.1 Autentizace uživatele

Hašování se používá v elektronické komunikaci k ověření totožnosti uživatele. Odesílatel ke zprávě přiloží text zprávy zahešovaný spolu s klíčem, který sdílí s adresátem. Ten po přijetí zprávu také zahešuje se sdíleným klíčem a ověří totožnost haše od odesílatele s tou, kterou vytvořil. Pokud by útočník změnil text zprávy, haše by se neshodovaly, a tím by byla zfalšovaná zpráva odhalena.

Pokud se jedná o pouhé ověření totožnosti uživatele, dotazovatel (server) odešle uživateli tzv. *challenge*, který uživatel opět spolu se sdíleným klíče zahašuje a haš odešle zpět.

2.2 Shodnost dat

Dalším využití hašování je kontrola kopírovaných dat, tedy jako ochrana před náhodnými chybami. K tomu se využívají speciální hašovací funkce, na které nejsou kladeny bezpečnostní nároky jako při autentizaci. Jednou z těchto hašovacích funkcí je *CRC* (*Cyclic Redun-*

dancy Check). Funkce vytváří tzv. kontrolní součet, který je vzhledem k velikosti souboru miniaturní, ale již při nepatrném rozdílu v souboru se změní. Srovnává se tedy kontrolní součet původního souboru a přeneseného. V případě shody jsou téměř jistě soubory identické, v případě neshody jednoznačně odlišné.

2.3 Vyhledávání

Při vyhledávání, například telefonních čísel v telefonním seznamu, se používají hašovací funkce k tvorbě *indexů* (ukazatelů) z takzvaných klíčů, například jmen v adresáři. Vyhledávácí algoritmus nemusí porovnávat klíč po klíči, ale pomocí indexu nalezne v tabulce příslušnou buňku. Číselné indexy jsou seřazeny, takže vyhledávání je mnohem rychlejší. V případě shodných indexů se na buňku může navázat další, spočítat index pomocí jiné funkce, nebo posunout soubor do další buňky určené nějakým algoritmem. Tyto postupy ovlivňují rychlost vyhledávání, která se také výrazně mění v závislosti na zaplnění tabulky.

3 Nový standard SHA-3

Hašovací funkce stojí často na nedokázaných principech, proto je přirozené, že tu a tam dojde k jejich prolomení. Roku 2004 byl prolomen hašovací standard MD5 (o dva roky později se podařilo českému kryptologovi Vlastimilu Klímovi najít kolize během jedné minuty). I u pozdějšího standardu SHA-1 se kolize umějí hledat rychleji, než je považováno za bezpečné. Proto americký úřad pro standardizaci (NIST) [2] vydal další standardní algoritmus - SHA-2. Ten je však třikrát pomalejší než SHA-1, a tak se NIST v listopadu roku 2007 rozhodl vyhlásit soutěž o hašovací funkci, která by byla dostatečně rychlá a spolehlivá, aby mohla SHA-2 nahradit.

Této soutěže se dohromady zúčastnilo 64 hašovacích funkcí od 191 autorů. Mezi nimi byly i dvě funkce s českými spoluautory - EDON-R a BMW (Blue Midnight Wish - podobnost s názvem německé automobilky je čistě náhodná). Na obou dvou programech pracovali (kromě jiných) Makedonec Danilo Gligoroski a již zmíněný významný český kryptolog Vlastimil Klíma. Oba dva programy patřily mezi nejrychlejší, ukázalo se však, že funkce EDON-R není dostatečně odolná vůči hledání vzoru, a tak do dalšího kola postoupilo spolu s dalšími třinácti kandidáty pouze BMW. V prosinci 2010 vyhlásil NIST pět finalistů, mezi které se již Blue Midnight Wish nedostal, přestože byl jeden z nejrychlejších a nebyly u něj nalezeny žádné slabiny [1]. Mezi pět postupujících funkcí patřily algoritmy BLAKE, Gröstl, JH, Keccak a Skein. Na 31. prosince letošního roku je naplánováno vyhlášení celkového vítěze soutěže o nový hašovací standard. Vítězný algoritmus pak bude zpřístupněn pro širší veřejnost, bude doporučen pro utajenou komunikaci mezi úřady atd.

4 Závěr

V rámci miniprojektu jsme se zabývali kombinatorickou pravděpodobností (složitost hledání vzoru, druhého vzoru a kolizí hašovacích funkcí), naprogramovali jsme jednoduchou hašovací funkci podle vlastního návrhu, nastudovali jsme využití hašování a blíže jsme se seznámili s právě probíhající soutěží o nový hašovací standard SHA-3.

Poděkování

Děkujeme FJFI ČVUT za umožnění realizace a Ing. L'ubomíře Balkové, Ph.D. za dohled, uvedení do problému a výraznou pomoc s miniprojektem.

Reference

[1] Klíma V. *Jak dopadla soutěž SHA-3?*, Crypto-World **10** (2010), 2 – 10.

[2] http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/documents/Round2_Report_NISTIR_7764.pdf (zdůvodnění NISTu k výběru 5 finalistů)

[3] http://en.wikipedia.org/wiki/Hash_function