1 Collision-Resistant Hash functions (CRHF's)

Pro podpis libovolně dlouhé zprávy nejprve aplikujeme hash h a pak až digitální podpis. Požadavky:

- Výstup h je ostře menší než vstup: $|h(m)| << |m|$.
- Funkce h je efektivně spočitatelná.
- Je těžké efektivně nalézt kolizi: $(m, m') : m \neq m'$ a $h(m) = h(m')$.

Definice 1 $H = \left\{h_i : \{0,1\}^{\ell_d(i)} \to \{0,1\}^{\ell_r(i)}\right\}_{i \in I}$ *je* kolekcí CRHF's *pokud platí:*

1. *Generování: existuje PPT* $G : i \leftarrow G(1^n), i \in I$.
2. *Komprese:* $\ell_d(i) > \ell_r(i), \forall i \in I$.
3. *Efektivita:* $h_i(x)$ *lze vyhodnotit v čase poly*(i) *pro všechny* $i \leftarrow G(1^n)$ *a* $x \in \{0,1\}^{\ell_d(i)}$.
4. *Collision Resistent: Pro všechny PPT* A *existuje negligible funkce* ε *taková, že*
$$\Pr[A(i) = (x, x') : x \neq x', h_i(x) = h_i(x')] \leq \varepsilon(n),$$
kde pravděpodobnost je přes $i \leftarrow G(1^n)$ *a náhodné mince* A.
5. *Technická podmínka:* $n \leq$ *poly*(i) *pro všechny* $i \leftarrow G(1^n)$.

- Typické nastavení: $\ell_d(i)$ je libovolné a $\ell_r(i) = n$.
- Možné útoky:
  1. Hádání: $\ell_d = 2n$ a $\ell_r = n$.
     - Vybereme náhodně $m, m' \leftarrow \{0,1\}^{2n}$ $\left(\Pr[m = m'] = \frac{1}{2^{2n}}\right)$.
     - Kolizi uhádneme s pravděpodobností vyšší než $\frac{1}{2^n} - \frac{1}{2^{2n}}$.
  2. Birthday Attack:
     - Vybereme t náhodných zpráv a hledáme kolizi.
     - Pravděpodobnost, že najdeme kolizi je počet dvojic krát pravděpodobnost kolize v náhodné dvojici. Tedy pravděpodobnost nalezení kolize je větší než:
$$\binom{t}{2}\left(\frac{1}{2^n} - \frac{1}{2^{2n}}\right) \approx \frac{t^2}{2^{n+1}}.$$
     - Pro $t = \Theta\left(2^{\frac{n}{2}}\right)$ pak máme konstantní pravděpodobnost nalezení kolize.

2 Universal One-way Hash Functions

Collision-resistent je silný předpoklad, někdy stačí následující:

1. A zvolí $x \in \{0,1\}^{\ell_d(i)}$.
2. $i \leftarrow G(1^n)$.
3. A vrátí $x' : x \neq x'$ a $h_i(x) = h_i(x')$.

- UOWHF's existují pokud existují jednosměrné funkce.
- CRHF's nelze zkonstruovat pomocí black-box metod ani z jednosměrných permutací.

3 Merkle-Damgård Transformace

Definice CRHF's dává kompresi o 1 bit, nyní ukážeme kompresi o libovolný počet bitů.

1. Kompresní funkce $h : \{0,1\}^{\ell+n} \to \{0,1\}^n$.
2. Pro zprávu $M \in \{0,1\}^*$:
   (a) Rozděl M do bloků délky ℓ:

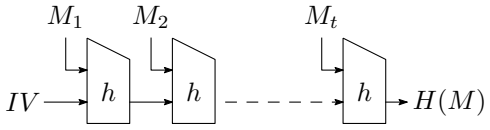
$$M = M_1||M_2||M_3||\dots||M_t.$$

   Blok M_t je doplněn 0 na délku ℓ. Definujme funkci H následovně

$$H(M) = h\Big(M_t||h\big(M_{t-1}||h(M_{t-2}\dots h(M_1||IV)\dots)\big)\Big),$$

   kde IV je inicializační vektor nastaven například na samé 0.

Schéma transformace je zobrazeno na následujícím obrázku.



Tvrzení 1 *Pokud je h kolekcí CRHF's pak i H je kolekcí CRHF's.*

4 Konstrukce CRHF's

Věta 1 *Kolekce CRHF's existují pokud platí předpoklad o obtížnosti faktorizace přirozených čísel nebo nalezení diskrétního logaritmu.*

Myšlenka důkazu. Mějme generátor g pro grupu $\mathbb{Z}_p^*$. Nechť $y \leftarrow \mathbb{Z}_p^*$. Problém diskrétního logaritmu je nalézt $x \in \mathbb{Z}_{p-1}$ takové, že $g^x \equiv y \ (mod\ p)$. Definujme

$$h_{p,g,y} : \mathbb{Z}_{p-1} \times \{0,1\} \to \mathbb{Z}_p^*,$$
$$h_{p,g,y}(x,b) = y^b \cdot g^x \ (mod\ p).$$

Ukážeme, že kolize $h_{p,g,y}$ nám dává diskrétní logaritmus y. Mějme PPT A, který najde s velkou pravděpodobností kolizi $(x,b) \neq (x',b')$ a $h_{p,g,y}(x,b) = h_{p,g,y}(x',b')$. Rozlišíme dva případy. Předpokládejme, že $b = b'$, pak ukážeme, že nenastane kolize. Pokud $y^b \cdot g^x \equiv y^b g^{x'} \ (mod\ p)$, pak také $g^x \equiv g^{x'} \ (mod\ p)$. Prvek g je ale generátor grupy $\mathbb{Z}_p^*$, tedy $x = x'$.

Nechť tedy $b \neq b'$ a bez újmy na obecnosti $b = 0$ a $b' = 1$. Pak tedy máme

$$g^x \equiv y g^{x'} \ (mod\ p)$$
$$y \equiv g^{x-x'} \ (mod\ p).$$

Tedy $x - x'$ je diskrétní logaritmus y vzhledem ke g. □

5 Hashovací Funkce v Praxi

Schéma	**Návrh**	**Délka Výstupu** (v bitech)	**Útok**	**Poznámka**
MD4	Rivest, 1990	128	1995	
MD5	1992	128	1998	
SHA1	1994	160	2005	• Kolize v čase 2^{60} • Lepší než birthday attack
SHA2/SHA256	2005	256	Zatím neprolomen	• Heuristický kandidát • Používá se v bitcoinu

6 Hash-then-Sign

Nyní ukážeme, že pokud existují kolekce CRHF's pak existují schémata elektronického podpisu pro libovolně dlouhé zprávy. Nechť (G,S,V) je one-time elektronický podpis pro $M = \{0,1\}^n$ a H je kolekcí CRHF's pro $\{0,1\}^*$ s výstupy délky n. Zkonstruujeme one-time elektronický podpis (G',S',V') pro zprávy libovolné délky.

G'**:** $pk' = (pk,i), sk' = (sk,i)$, kde $(pk,sk) \leftarrow G(1^n)$ a pro generátor G_H z H: $i \leftarrow G_H(1^n)$.

S': $S'_{sk}(m) = S_s k\big(h_i(m)\big)$.

V': $V'_{pk}(m, \sigma) = V_{pk}\big(h_i(m), \sigma\big)$.

Věta 2 *Pokud (G, S, V) je secure one-time elektronický podpis pro $M = \{0,1\}^n$ a H je kolekce CRHF's pro $\{0,1\}^*$ pak (G', S', V') je secure one-time elektronický podpis pro $M' = \{0,1\}^*$.*

Důkaz. Mějme PPT A, který padělá podpis pro (G', S', V') s pravděpodobností alespoň ε (non-negligible). Algoritmus $A(pk')$ pošle nejvýše jeden dotaz m na $S'_{sk'}(\cdot)$ a dostane odpověď $\sigma' \leftarrow S'_{sk'}(m)$. Algoritmus A uspěje pokud vrátí (m', σ') takové, že $m \neq m'$ a $V'_{pk'}(m', \sigma') = accept$.

7 Další Aplikace CRHF's

- Fingerprinting
- Deduplikace dat – test, zda soubor již není na serveru

7.1 Merkle Hash Tree

Mějme $h : \{0,1\}^{2\ell} \to \{0,1\}^{\ell}$ a data $X = \{0,1\}^*$. Rozdělíme X do bloků velikosti ℓ, tedy

$$X = x_1 || x_2 || \dots || x_t,$$

kde $|x_i| = \ell$. Můžeme předpokládat, že $|X| = \ell t$ a t je mocnina 2, jinak použijeme padding nulami. Mějme binární strom s t listy. Ke každému vrcholu v přiřadíme hodnotu $c_X(v) \in \{0,1\}^{\ell}$. Do listů umístíme bloky x_i a následně počítáme hashe od listu nahoru. Mějme vrchol v jehož synové v_1 a v_2 již mají spočítané hodnoty $c(v_1)$ a $c(v_2)$ pak hodnota na vrcholu v bude $c_X(v) = h\big(c_X(v_1)||c_X(v_2)\big)$. Označme $MT_h(X)$ hodnotu v kořeni stromu.

Příklad použití: chceme uložit data na serveru a mít možnost ověřit, že server data nezměnil bez toho aniž bychom si daná data pamatovali.

1. Spočítáme $MT_h(X)$ a odešleme data na server.
2. Chceme získat x_i a ověřit, že server data nezměnil.
3. Nechť P je cesta z listu, kde je uloženo x_i, do kořene stromu.
4. Server pošle hodnoty všech vrcholů z P a všech jejich bratrů.
5. Nyní můžeme ověřit, zda hodnoty na cestě P jsou správně spočítané a zda hodnota v kořeni je rovna $MT_h(X)$ – je potřeba $O(\log t)$-krát spočítat h.
6. Aby server podvrhl x_i musí umět nalézt kolizi h.

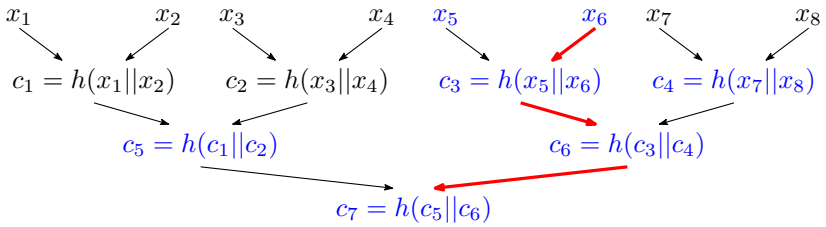


Figure 1: Příklad Merkle hash tree pro data s 8 bloky. Při vyžádání bloku x_6 ze serveru je červeně vyznačená cesta P z listu x_6 do kořene a modře jsou označeny data, která server odešle.

Lemma 1 *Nechť $MT_h(X) = MT_h(X')$ pro $X \neq X' \in \{0,1\}^{t\ell}$. Pak musí existovat vrchol v ve stromu se syny v_1 a v_2 takový, že*

$$c_X(v_1)||c_X(v_2) \neq c_{X'}(v_1)||c_{X'}(v_2),$$
$$h\big(c_X(v_1)||c_X(v_2)\big) = h\big(c_{X'}(v_1)||c_{X'}(v_2)\big).$$

Důkaz. Postupujme od listů nahoru. Mějme x_i a x'_i bloky X a X' takové, že $x_i \neq x'_i$. Vezměme list v_1, kde jsou uloženy x_i a x'_i a jeho bratra v_2. Pro vrcholy v_1 a v_2 jistě platí, že

$$c_X(v_1)||c_X(v_2) \neq c_{X'}(v_1)||c_{X'}(v_2).$$

Nechť v je jejich otec. Pokud pro v platí, že $c_X(v) = c_{X'}(v)$, pak jsme skončili. Jinak vezmeme $v_1 := v$ a v_2 je bratr v a celý proces opakujeme. Jelikož pro kořen stromu r platí, že $c_X(r) = c_{X'}(r)$ tak jistě najdeme požadované vrcholy v, v_1 a v_2.