A. Utajená míra složitosti?

Vlastimil Klíma, kryptolog konzultant, KNZ, s.r.o., Praha
(http://cryptography.hyperlink.cz, vlastimil.klima@knzsro.cz)

Článek navazuje na příspěvek v čísle 4 Crypto-Worldu 2010, v němž jsme chtěli stimulovat výzkum složitosti kandidátů na SHA-3. Navrhli jsme vyjít z algebraické normální formy. Protože ta je dost složitá - například jen po jedné jediné operaci ADD32 dvou 32bitových sčítanců vzniká přes 4 miliardy termů - navrhli jsme počítat počet (jednobitových) operací AND a XOR a počty meziproměnných (de facto paměťových buněk). V tomto článku navrhneme míru jednodušší, a to pouze jako minimální počet (jednobitových) operací AND, s kterými lze danou funkci obvodově realizovat. Tato míra se zdá být velmi dobrá, neboť je současně jednoduchá a současně dostatečně vypovídající. Původní složitost operace ADD klesne ze 4 miliard na hodnotu 31. Danou měrou jsme spočítali složitost nejrychlejších kandidátů na SHA-3 a byli jsme velmi překvapeni, jak jsou si tito kandidáti v naší míře blízcí! Jakoby tady pracovalo nějaké tajné pravidlo, které jejich návrháři dodržovali. Vzhledem k tomu, že tyto týmy pracovali nezávisle a v tajnosti, zanechává tento výsledek spíše více otázek, než jich řeší.

Míra složitosti

Mírou složitosti je tedy minimální počet operací AND. To lze u složitějších funkcí obvykle špatně spočítat. Stačí si zkusit toto spočítat pro malé substituční boxy 4 bity na 4 bity. Nicméně u operace ADD32 jsme v čísle 4 Crypto-Worldu tuto složitost spočítali, a je rovna 31, viz též následující rovnice.
$a = b + c$,
kde a, b, c jsou 32bitové proměnné, sčítání je v modulu 2^{32}.

Máme
pro bit i = 0: $a_0 = b_0 \oplus c_0$, $carry_1 = b_0$ AND c_0,
$a_i = b_i \oplus c_i \oplus carry_i$, i = 1, ..., 31,
kde $carry_{i+1} = (b_i \oplus carry_i)$ AND $(c_i \oplus carry_i) \oplus carry_i$ pro i = 1, ..., 30.
Tato míra složitosti hodně souvisí s praktickou elektronickou realizací, což je možná jedna z příčin oné utajené shody, viz tabulka.

Hašovací funkce typu XAR a ostatní

Složitost se nám dobře počítala pro funkce typu XAR, tj. funkce, používající pouze operace XOR, ADD a ROT (SHIFT). Zde stačí spočítat počet operací ADD a vynásobit složitostí 31. U ostatních funkcí může být výpočet jejich složitosti velice netriviální. Musíme umět prokázat, že máme vyjádření s nejmenším možným počtem operací AND, a to je obtížné. Proto hodnoty pro tyto funkce jsou jen orientační a je tu velké pole jak pro jejich přesnější výpočet, tak pro důkaz minima.

Algorit-mus	Používané operace	Rychlost (cyklů/bajt)	Počet operací AND na jeden bit zprávy	Koeficient rych-lost/složitost
SHA-1	XAR	9	17	1,89
BMW	XAR	7	24	3,43
BLAKE	XAR	9	29	3,22
Shabal	XAR a modulární násobení	10	13	1,30
CubeHash	XAR	13	992	76,31
SIMD	XAR a různé operace	12	23	1,92
Skein	XAR	21	26	1,24
SHA-2	XAR	20	40	2,00

Tabulka: Nejrychlejší kandidáti na SHA-3 a jejich míra složitosti

V tabulce je velice pěkně vidět, že SHA-2 je cca dvakrát složitější než SHA-1, za což platí svojí poloviční rychlostí. Avšak přece jen je trochu dokonalejší v tom, že za složitost algoritmu (počtu operací AND) se neplatí tolik ztrátou rychlosti jako u SHA-1, i když rozdíl je malý, viz koeficient rychlost/složitost v tabulce, což je (počet operací AND v algoritmu na jeden bit zprávy)/ (počet cyklů na jeden bit zprávy). Při přechodu od SHA-2 k SHA-3 však NIST chce tento koeficient ještě zvýšit, neboť požaduje současně vyšší bezpečnost (složitost), tj. většího čitatele, a současně menšího jmenovatele, tedy vyšší rychlost zpracování dat. Za normálních okolností bychom řekli, že je to nesmysl. Pouštíme se teď na tenkou půdu filozofování, ale je těžké si představit, že nějaký algoritmus, když ho máme před sebou "zadrátovaný" do logických operací, zesílíme tím, když nějaké operace AND z něho vyjmeme a nahradíme je operacemi XOR. Jistě, že bychom uměle takový příklad zkonstruovali, aby s vybranými operacemi AND vznikaly nějaké lineární závislosti mezi dílčími nelineárními bloky, zatímco, když je nahradíme operacemi XOR, tak tyto lineární závislosti zrušíme, a tím naopak zvýšíme složitost výsledného schématu. To je ale v případě umělého schématu, kde ve skutečnosti uměle vytváříme slabinu, abychom ji také uměle mohli odstranit. Jenže schémata, o nichž je řeč, mají být bez evidentních slabin, tj. taková, kde řešení příslušné soustavy booleovských rovnic je jedinou cestou, kterou kryptologové vidí, jak příslušný problém řešit. A zde je počet operací AND potom odrazem skutečné složitosti. Pokud bychom pokračovali dlouhým a rozsáhlým výzkumem, možná bychom nalezli hraniční hodnotu koeficientu složitost/rychlost pro současnou technologii. NIST svými požadavky na SHA-3 řekl, že chce,

aby výzkum tento koeficient zvýšil. Místo 50 let výzkumu jednoho týmu využil několik desítek týmů po dobu 5 let. Tímto způsobem pak přirozeně může vybrat nejefektivnějšího kandidáta. I když tento koeficient také souvisí se složitostí realizace (plochou křemíku apod.), určitě to nebude dělat jen podle něho, takže článek nesměřuje k tomu, že nejlepším kandidátem je BMW (odhlédneme-li od zcela vybočujícího CubeHash), ale spíše k pohledu na kandidáty z jiného úhlu, než je nabízen obvykle.
Poznamenejme, že značně odlišná čísla u CubeHash zatím nedokážeme vysvětlit, a možná, že někde je chyba v uvedených úvahách. Také počet operací u SIMD je hrubě odhadnutý a zasloužil by si řádný výpočet. Nicméně u ostatních algoritmů by měly být výpočty správné, což je podstatné, neboť shoda na složitosti je velmi překvapující.

Výpočet složitosti u kandidátů

V následujícím uvedeme, jak jsme dospěli k číslům v tabulce. Pro všechny kandidáty jsme posuzovali složitost jejich kompresní funkce, protože při prodlužování zprávy se složitost na bit pro kompresní funkci velmi rychle blíží složitosti na bit celé hašovací funkce. A použili jsme vždy variantu hašovací funkce, poskytující 256 bitů výstupu. Většinou jsou použity 32bitové operace, sčítání dvou 32bitových argumentů značíme proto místo ADD32 krátce ADD a sčítání 64bitových argumentů jako ADD64. Zkratka AND značí operaci AND dvou jednobitových argumentů, v níž složitost počítáme.

BMW-256

- blok je 512bitů (16 32bitových slov)
- funkce f0: 16 operací v f0, jedna operace má čtyři ADD (SUB) = 64 ADD, poté 16 ADD
- funkce f1: 2 rundy typu 1 + 14 rund typu 2 = 2*(18) + 14*(18) = 16*18=288 ADD
- funkce f2: 8+16 = 24 ADD
- celá kompresní funkce =64+16+288+24=392 ADD na 1 blok 512 bitů,
- složitost kompresní funkce = 392*31 = 12152,
- složitost na bit = 12152/512=**24**

Skein-256

- používá vždy 64bitová slova
- pro Skein-256 má blok 512bitů (4 slova)
- kompresní funkce se skládá z přičtení dat na průběžnou haš = 4 ADD
- potom 9 velkých rund, každá se skládá z 16 sčítání 64bitových slov a dvou přičítání průběžné haše
- celá kompresní funkce = 4 + 9*(8rund)= 4+9*(8+8+2*inject_key) = 4+9*(8+8+2*4) = 4+9*24 = 216 ADD64
- složitost kompresní funkce = 216*63=13608 AND
- složitost na bit = 13608/512=**26**

CubeHash

- nezávisí na délce haše
- na každý bajt nejprve provádí xor bajtu zprávy na vnitřní stav haše, stav má 1024 bitů (jakožto 32 32bitových slov)
- potom stav zpracuje 8 velkými rundami
- 1 velká runda má 10 kroků
- sčítají se vždy 32bitová slova, v jedné velké rundě je 32 ADD
- 8*32=256 operací ADD na jeden bajt zprávy
- složitost na 1 bit = 8*32*31/8= **992**

BLAKE

- Pro 256bitovou haš používá 16 32bitových slov zprávy (512bitový blok) a 8 32bitových slov průběžné haše
- kompresní funkce zpracovává průběžnou haš pomocí rund, v každé rundě je 8 operací G po šesti ADD, které promíchávají stav se dvěma slovy zprávy, má 10 rund
- celkem 10*8*6 = 480 operací ADD
- složitost je 480*31=14880
- složitost na 1 bit = 14880/512=**29**

Shabal

- používá mj. operace 3*x a 5*x modulo 2^{32},
- pracuje s 32bitovými slovy, blok zprávy je 512 bitů
- používá registry B (16 slov), C (16 slov), A (12 slov)
- Shabal-256 je typu double-pipe (Shabal-512 je single pipe), odlišnost je pouze v tom, kolik bitů se vezme z výsledku
- kompresní funkce se skládá z permutace P(A, B, C, M) a z počátečního a závěrečného přičítání,
- vstupem permutace je B = B + M, M se poté od výstupu permutace odečítá, tj. před a po permutaci je to 16 + 16 operací ADD (SUB)
- permutace obsahuje 3*(16*(U a V a jednu 32bitovou operaci AND))+ 3*12 ADD
- operace U = 3*x modulo 2^{32}, složitost je 30 AND (sčítá se x + 2x, druhý argument má o 1 bit méně, tj. jeden bit carry se ušetří)
- operace V = 5*x modulo 2^{32}, složitost je 29 AND (sčítá se x + 4x, druhý argument má o 2 bity méně, tj. dva bity carry se ušetří)

- permutace = 3*16*(U + V + AND32) + 36*ADD = 48*(30+29+32)+36*31 = 48*91+36*31=5484 AND
- počáteční a závěrečné +/- = 32 ADD = 32*31 = 992 AND
- složitost = 992 + 5484 = 6476 AND
- složitost na 1 bit = 6476/512=**13**

SIMD

- dost modifikovaná D-M konstrukce: (h, m) --> P(h, E_m(h xor m))
- blok má 512/1024 bitů podle délky haše 256/512
- vnitřní stav má 16 slov (32/64b)
- nejprve expanduje blok zprávy 512/1024 na osminásobek (4096/8192) a poté následuje tzv. Feistelovo schéma
- expanze:
- první úroveň: NTT je lineární MDS kód nad 64/128 "bajty" (prvky tělesa), vytvoří 128/256 bajtů
- druhá úroveň: vnitřní kódy, z bajtu udělá dva, tj. 256/512 bajtů
- třetí úroveň (permutace): také zdvojnásobí šířku, nakonec máme 128/256 slov
- složitost expanze odhadnuta = 7360 operací AND (tento výsledek není zatím příliš ověřen)
- Feistelovo schéma:
- slova se zpracovávají po čtyřech (krok), celkem 36 kroků
- krok = 3 ADD + AND32 = 125 AND
- Feistelovo schéma celkem = 36*125 AND = 4500 AND
- složitost = 4500 + 7360 = 11860 AND operací na 512 bitů zprávy
- složitost na 1 bit = 11860/512 = **23**

Literatura

Kandidáti druhého kola SHA-3: http://csrc.nist.gov/groups/ST/hash/sha-3/Round2/index.html