Analýza Blue Midnight Wish – srovnání složitosti (bezpečnosti) BMW a dalších kandidátů SHA-3

RNDr. Vlastimil Klíma, kryptolog konzultant, Praha
(http://cryptography.hyperlink.cz, v.klima@volny.cz)
Prof. Danilo Gligoroski, Norwegian University of Science and Technology, Norway (danilog@item.ntnu.no , http://www.item.ntnu.no/people/personalpages/fac/danilog/start)

Článek navazuje na příspěvky v číslech 1 - 3 Crypto-Worldu 2010, s nímž má společnou skoro celou úvodní stranu a několik obrázků. Volně také navazuje na články o BMW v 12/2009, 3/2009 a 7-8/2009. V čísle 1 jsme se zabývali hledáním vzoru (úloha první), v čísle 2 hledáním kolize (úloha druhá), v čísle 3 různými bloky BMW.

Chceme stimulovat analýzy a útoky na BMW a prezentovat otevřené problémy. Ty by se mohly stát předmětem studentských prací. Proč? Velkou výhodou oproti jiným tématům je, že tyto rozbory jsou nyní velmi žádané, ať s negativním nebo pozitivním výsledkem. Když bude problém vyřešen nebo naopak bude ukázáno, že je složitý, je to v obou případech žádaný a velmi dobře publikovatelný výsledek.

Označení
Článek bude využívat označení zavedené v Crypto-Worldu 12/2009. Připomeňme jen šířku slova w = 32 nebo 64 bitů, délku bloku zprávy a průběžné haše n = 16*w (mají 16 slov) a výpočet haše:

1. Předzpracování
(a) Doplň zprávu M jednoznačným definovaným způsobem o délku zprávy v bitech a doplněk
(b) Rozděl zprávu na celistvý násobek (N) m-bitových bloků $M^{(1)}$, ..., $M^{(N)}$.
(c) Nastav počáteční hodnotu průběžné haše $H^{(0)}$ na konstantu ($CONST^0$).
2. Výpočet haše
For i = 1 to N: $H^{(i)} = f(M^{(i)}, H^{(i-1)})$.
3. Finalizace
$H^{final} = f(H^{(N)}, CONST^{final})$, kde $CONST^{final}$ je konstanta.
4. Závěr
H(M) = dolních n bitů z hodnoty H^{final}.

O míře složitosti
V následujícím se pokusíme najít přijatelnou a prakticky využitelnou míru složitosti kandidátů na SHA-3. Víme, že pro každého kandidáta je úloha nalezení kolize nebo vzoru převeditelná na řešení systému Booleovských rovnic. Proto se pokusíme porovnat jejich složitost. Navrhneme míru, která sice není perfektní, ale uvidíme, že složitost těchto kandidátů dobře odráží. V každém případě získáme horní odhad složitosti.

Definice složitosti
Budeme porovnávat složitost zápisu Booleovských rovnic v algebraické normální formě. Tato forma zápisu využívá operace XOR a AND. Pochopitelně, že více nám vadí operace AND, neboť při absenci operací AND by se jednalo o snadno řešitelný lineární systém. Avšak i operace XOR budeme počítat a budeme též počítat počet všech meziproměnných. Jedinými proměnnými jsou sice jen bity zprávy, avšak než se vypočítá hašovací hodnota, vznikne řada

meziproměnných, což je nejlépe vidět z programové realizace dané funkce. Každou dílčí funkci hašovací funkce můžeme napsat jako polynom a každý polynom můžeme zapsat postupně pomocí meziproměnných tak, že vznikají pouze elementární rovnice typu
$a = b \oplus c$,
$a = b*c$,
kde * označuje AND a kde proměnné a, b, c jsou Booleovské proměnné.

Takto můžeme například zapsat aritmetické sčítání w-bitových slov a = b + c modulo 2^w, a to následovně. Označíme bity i = 0 ,..., w-1 indexem i, tj. $a = (a_{w-1}, ..., a_0)$, kde a_{w-1} je nejvyšší bit a a_0 nejnižší. Bity přenosu označíme $carry_i$ pro i = 1, ..., w-1, což jsou právě vznikající meziproměnné (kromě $a_{w-1}, ..., a_0$).

Máme
pro bit i = 0:
$a_0 = b_0 \oplus c_0$, $carry_1 = b_0 * c_0$,
pro bit i = 1, ..., w - 2:
$a_i = b_i \oplus c_i \oplus carry_i$, $carry_{i+1} = b_i*c_i \oplus b_i*carry_i \oplus c_i*carry_i$,
pro bit i = w - 1:
$a_{w-1} = b_{w-1} \oplus c_{w-1} \oplus carry_{w-1}$.

Funkci $carry_{i+1} = b_i*c_i \oplus b_i*carry_i \oplus c_i*carry_i$ rozložíme na elementární operace, ale rovnou uděláme drobnou optimalizaci, když b_i vytkneme z prvních dvou členů a součet $c_i \oplus carry_i$ použijeme také k výpočtu a_i.
Máme:
$s_i = c_i \oplus carry_i$,
$t_i = b_i * s_i$,
$u_i = c_i * carry_i$,
$carry_{i+1} = t_i \oplus u_i$.

Tedy sčítání slov a = b + c můžeme zapsat normalizovaně elementárními operacemi takto:
pro bit i = 0:
$a_0 = b_0 \oplus c_0$, $carry_1 = b_0 * c_0$,
pro bit i = 1, ..., w - 2:
$s_i = c_i \oplus carry_i$, $t_i = b_i * s_i$, $u_i = c_i * carry_i$, $a_i = b_i \oplus s_i$, $carry_{i+1} = t_i \oplus u_i$,
pro bit i = w - 1:
$v = b_{w-1} \oplus c_{w-1}$, $a_{w-1} = v \oplus carry_{w-1}$.

Máme zde
(1 + (w - 2)* 3 + 2) operací ⊕
a
(1 + (w - 2)*2 + 0) operací *
s
2 + (w - 2)*(1 + 1 + 1 + 1 + 1) + 1*(1 + 1) novými meziproměnnými (a_i, $carry_i$, s_i, t_i, u_i, v), tj. máme 3w - 3 operací ⊕, 2w – 3 operací * a 5w - 6 nových meziproměnných.

Další optimalizace
Existuje ještě lepší optimalizace [P]. Výraz $carry_{i+1} = b_i*c_i \oplus b_i*carry_i \oplus c_i*carry_i$ můžeme zapsat pomocí jedné operace AND, a to takto:
$carry_{i+1} = (b_i \oplus carry_i) * (c_i \oplus carry_i) \oplus carry_i$.

Nyní máme o jednu operaci ⊕ navíc, ale ušetřili jsme jednu meziproměnnou (u) a jednu AND operaci, což je ještě lepší optimalizace:
$s_i = c_i \oplus carry_i$,
$t_i = b_i \oplus carry_i$,
$carry_{i+1} = s_i * t_i \oplus carry_i$,

Sčítání slov a = b + c můžeme zapsat normalizovaně elementárními operacemi takto:
pro bit i = 0:
$a_0 = b_0 \oplus c_0$, $carry_1 = b_0 * c_0$,
pro bit i = 1, ..., w - 2:
$\mathbf{s_i = c_i \oplus carry_i, t_i = b_i \oplus carry_i, carry_{i+1} = s_i * t_i \oplus carry_i, a_i = b_i \oplus s_i,}$
pro bit i = w - 1:
$v = b_{w-1} \oplus c_{w-1}$, $a_{w-1} = v \oplus carry_{w-1}$.

Máme zde
(1 + (w - 2)* 4 + 2) operací ⊕
a
(1 + (w - 2)*1 + 0) operací *
s
2 + (w - 2)*(1 + 1 + 1 + 1) + 1*(1 + 1) novými meziproměnnými (a_i, $carry_i$, s_i, t_i, v), tj. máme 4w - 5 operací ⊕, w – 1 operací * a 4w - 4 nových meziproměnných.

Pokud bychom při zápisu řádku
$a_i = b_i \oplus c_i \oplus carry_i$, $carry_{i+1} = b_i * c_i \oplus b_i * carry_i \oplus c_i * carry_i$
do elementárních operací postupovali bez optimalizace, zápis by vyšel místo pěti na sedm elementárních operací. Při zápisu všech funkcí v každém schématu záleží velmi na tom, zda podobnou možnost optimalizace objevíme. Pokud ne, a optimalizace bude přesto existovat, dostaneme horní odhad skutečné reálné složitosti zápisu. Pokud ano, přiblížíme se více reálné složitosti. Avšak i horní odhad složitosti je dobrým výsledkem, neboť pokud vyjde malý, je jisté, že příslušná funkce je jednoduchá.

Normalizace úlohy
Nechceme poškodit žádného z kandidátů na SHA-3, avšak žádná míra není dokonalá, proto námi navržená míra pravděpodobně některého kandidáta upřednostní a jiného poškodí. Cílem je ale spíše uvést tuto míru obecně, nikoli konkrétní variantu.

Také při rozhodování, jakou úlohu měřit, jestli úlohu nalezení kolize nebo vzoru, narazíme na různá dilemata. Tyto úlohy mohou jako řešení mít zprávy o milionech bloků, musíme tedy jak úlohu samu, tak délku zprávy nějak normalizovat. Proto nebudeme formulovat ani úlohu získání kolize ani úlohu získání vzoru, ale srovnáme složitost soustavy Booleovských rovnic, které popisují vznik hašovacího kódu co nejmenším počtem rovnic, proměnných a operací. Tomu přizpůsobíme i délku zprávy. Volíme ji takovou, aby měla maximální délku, ale takovou, aby vyžadovala co nejméně operací, což je tolik, tolik je potřeba na zpracování zprávy prázdné. Konkrétně u SHA-1 by to byla zpráva délky 512 – 65 bitů, vyžadující jedno volání kompresní funkce. U SHA-256 by to také byla zpráva délky 512 – 65 bitů, u BMW256 by to shodou okolností také byla zpráva délky 512 - 65 bitů. U BMW512 by to byla zpráva délky 1024 – 65 bitů, u SHA-512 také 1024 – 65 bitů. Všechny tyto zprávy vyžadují stejné volání kompresní funkce jako zpráva o délce 0 bitů (prázdný řetězec). Pochopitelně budeme porov-

návat složitost kandidátů pro stejný hašový výstup, tedy například pro varianty s 256 bitovým hašovým kódem.

Hašovací funkce typu XAR, XAS
Tito kandidáti na SHA-3, kam patří například BMW a Skein, používají pouze operace XOR, ADD a ROT (SHIFT). U těchto funkcí musíme vypočítat pouze složitost sčítání, operace rotací nebo posunů nevnáší do soustavy rovnic žádné nové meziproměnné ani operace (shift se neprovede, pouze se v dané rovnici, která se shiftovanými proměnnými pracuje, použije proměnná, odpovídající danému posunu).

Ostatní funkce
Všechny ostatní funkce, které se v kandidátských algoritmech objevují, rozložíme na jednodušší operace, až dospějeme k základním bitovým funkcím typu $y_i = f(a_i, b_i,...)$, které vyjádříme v algebraické normální formě. Měli bychom se rozhodně zabývat tím, jak tyto funkce optimalizovat. Protože to může být značně náročná úloha, pro první přiblížení můžeme optimalizovat jen zjevné výpočty.

Přesnost naší míry
Domníváme se, že toto je velmi dobrá míra složitosti (nikoli bezpečnosti), protože když je schéma uděláno dobře, není v něm žádná zkratka, nevíme, jak řešit soustavu rovnic, neznáme slabiny schématu, a proto řešení námi zapsané soustavy rovnic je jediná cesta, jak prolomit schéma. Pokud předpokládáme, že všichni kandidáti na SHA-3 jsou uděláni dobře, je řešení systému normalizovaných rovnic u každého odrazem jeho skutečné bezpečnosti. A protože tento systém je normalizovaný pro všechna schémata stejně, lze jeho složitostí porovnávat bezpečnost schémat.

Závěr
Záměrně zde nevypočítáváme složitost kandidátů podle navržené metody a ponecháváme to na zájemcích, studentech. Porovnání je velmi potřebné a tato míra je pro některé kandidáty docela jednoduše spočitatelná, pro jiné postačí k velmi potřebnému výsledku velmi málo.

[P] Michael Fischer, Rene Peralta: Counting Predicates of Conjunctive Complexity One YALEU/DCS/TR-1222, December 2001 (Revised February 2002), Yale University Department of Computer Science