Mohutné multikolize a multivzory hašovacích funkcí BLENDER-n

[1)]

Vlastimil Klima [2)]

Abstrakt

Blender-n [1] je jeden z 51 kandidátů na hašovací funkci SHA-3, které postoupily do prvního kola mezinárodní soutěže na nový hašovací standard. V tomto příspěvku prezentujeme multikolizní a multivzorový útok na hašovací funkci Blender-n pro všechny velikosti výstupu n = 224, 256, 384 a 512. Složitost útoku a požadavky na paměť pro nalezení 2^{2n} (multi)vzorů (a multikolizí) algoritmu Blender-n jsou řádově pouze 10 krát větší než nalezení kolize náhodné hašovací funkce s n/2 výstupními bity.

Všechny předchozí útoky na Blender-n byly založeny na triku (Joux, [2]) s využitím mnoha zpráv. Naše útoky využívají jednu zprávu, u níž konstruujeme několik pevných bodů. Stavový registr kompresní funkce Blenderu má osm slov. Vhodnou volbou slov zprávy donutíme polovinu tohoto registru vrátit se do původního stavu. Potom nalezneme kolizi ve zbytku registru se složitostí $2^{n/4}$. Tato kolize vytvoří pevný bod v posloupnosti stavů stavového registru. Využíváme 10 těchto pevných bodů a pomocí nich konstruujeme kolidující zprávy, vedoucí k předepsané hašovací hodnotě.

Dosud známé útoky [4, 5] na Blender-n měly složitost nejméně $2^{n/2}$. Náš 2^{2n}-multikolizní a multivzorový útok má složitost pouze $10*2^{n/4}$.

1 Přehledný popis algoritmu Blender-n

Pro jednoduchost se budeme zabývat pouze algoritmem Blender-256. Útoky na ostatní varianty jsou analogické.

Hašovací funkce Blender je iterovaná hašovací funkce. Používá w-bitová slova (w = 32 pro Blender-256, w = 64 pro Blender-512), stavový registr A o osmi w-bitových slovech, dva bity přenosu (carry c1, c2) a hašovací registr H, který má také osm w-bitových slov. Na počátku jsou stavový registr a carry bity vynulovány a počáteční hodnota registru A je nastavena na konstantu $A^0 = (a0^0, a1^0, a2^0, a3^0, a4^0, a5^0, a6^0, a7^0) = H_{init}$. Registr H obsahuje průběžnou hašovací hodnotu, která je definována jako součet (modulo 2^{32} po slovech) slov stavů stavového registru A, tj. $H^t = \sum_{t=1,\ldots,K} A^t$. Nový stav A je funkcí starého stavu a slova zprávy. Takže máme $(A^{t+1}, c1^{t+1}, c2^{t+1}) = f(A^t, c1^t, c2^t, W^t)$, kde f je kompresní funkce a W^t slovo zprávy.

Před hašováním se zpráva zarovná na bajty a doplní na celistvý počet bloků o 16 slovech. Stručně řečeno "doplněk" je tvořen "výplňovými bajty" (filling), které se skládají z prvních 13 bajtů zprávy, eventuálně opakovaných do potřebné délky, dále z délky zprávy v bitech,

---

[1] Tato zpráva je k dispozici jako IACR eprint Report 2009/006, http://eprint.iacr.org/2009/006.pdf, v češtině na http://cryptography.hyperlink.cz/BMW/BMW_CZ.html

[2)] Nezávislý kryptolog, Praha, http://cryptography.hyperlink.cz, v.klima@volny.cz

přičemž tato délka se kóduje binárně a jen do nezbytného počtu bajtů a dále následuje délka délky zprávy (to je jeden bajt, jež obsahuje počet bajtů, v nichž byla zakódována délka zprávy) a poté nakonec dvě slova kontrolního součtu.

Ta jsou vypočítána takto:
checksum1 = non($\sum_{t=1,...,K} W^t$),
checksum2 = $\sum_{t=1,...,K}$ (nonW^t).

Abychom se vyhnuli technickým detailům, uvažujeme pouze zprávy, které mají celočíselný počet slov, stejných 13 prvních bajtů, stejnou délku a dokonce i stejné kontrolní součty. Je důležité poznamenat, že kontrolní součty i aktualizace registru H a registru A jsou počítány pomocí w-bitových slov, zejména pomocí sčítání modulo 2^w.

Zde uvádíme část původního popisu hašování - odstavec 2.6.2, [1]:

Algoritmus Blender-256 používá 32-bitové proměnné a0, ..., a7, H0, ..., H7, dva bity přenosu c1 a c2. To vytváří stav algoritmu od rundy k rundě. Dále se využívají pomocné 32-bitové proměnné T, T1 a T2 a celé číslo r.

Před hašováním jsou proměnné a naplněny hodnotou (H_{init}): a0 = 6a09e667, ...(cut)..., a7 = 5be0cd19. Když je připravena posloupnost 32-bitových slov word W^t , počítá se:

1. hodnoty T1,T2:
[c1,T1] = (a5 ⊕ W^t) + (a1 ⊕ rotl(a3, 8)) + c1
[c2,T2] = (a0 ⊕ rotr(W^t, 8)) + (a4 ⊕ rotr(a2, 8)) + c2

2. rotační faktor:
r = 8 – (c1 + c2)

3. Rotují se T1,T2:
T1 = rotl(T1, r)
T2 = rotr(T2, r)

4. Vypočte se následující stav:
T = rotr(a0, 7)
a0 = a1 ⊕ T2
a1 = a2 ⊕ T1
a2 = a3 ⊕ T2
a3 = a4 ⊕ T1
a4 = a5 ⊕ T2
a5 = a6 ⊕ T1
a6 = a7 ⊕ T2
a7 = T ⊕ T1

5. Aktualizuje se hašovací hodnota:
H0 = H0 + a0
H1 = H1 + a1
H2 = H2 + a2
H3 = H3 + a3

H4 = H4 + a4
H5 = H5 + a5
H6 = H6 + a6
H7 = H7 + a7

Těchto pět kroků definuje jednu rundu algoritmu. Po zpracování všech slov je hašovací hodnota vyčtena z registru H:
H0 || H1 || H2 || H3 || H4 || H5 || H6 || H7.

2 Stavový registr

Hašování má vnitřní stav, který je dán hodnotami (A, c1, c2, H). Nejprve vytvoříme kolize ve stavovém registru A, potom v registru H. Pro jednoduchost budeme hovořit o registru A, i když budeme mít na mysli i stav bitů c1, c2.

Stavový registr má osm slov, ale vhodnou volbou 256 slov zprávy W^t ho přinutíme, aby ve skutečnosti měl pouze 4 měnící se slova po každých 256 krocích. Označme $A^0 = (a0^0, a1^0, a2^0, a3^0, a4^0, a5^0, a6^0, a7^0) = H_{init}$ počáteční stav registru A.

Základní útok
Runda 0:
Z hodnot A^0 vypočítáme první slovo zprávy W^0 tak, že $\mathbf{T2^0 = 0}$. Tomu odpovídá právě jedna volba W^0, což můžeme vidět z rovnice $T2 = (a0 \oplus rotr(W^t, 8)) + (a4 \oplus rotr(a2, 8)) + c2$. Připomeňme, že registr A rotuje svá slova o jednu pozici doleva a v posledním slově navíc rotuje bity o 7 pozic vpravo:

$A^1 = ($ **$a1^0$** $, a2^0 \oplus T1^0,$ **$a3^0$** $, a4^0 \oplus T1^0,$ **$a5^0$** $, a6^0 \oplus T1^0,$ **$a7^0$** $, rotr(a0^0,7) \oplus T1^0)$.

Runda 1:
Podobně jako v rundě 0 nyní volíme slovo zprávy W^1 tak, že $\mathbf{T1^1 = 0}$. Opět máme právě jedinou volbu, což vidíme z rovnice $T1 = (a5 \oplus W^t) + (a1 \oplus rotl(a3, 8)) + c1$.
Dostáváme
$A^2 = (a2^0 \oplus T1^0 \oplus T2^1,$ **$a3^0$** $, a4^0 \oplus T1^0 \oplus T2^1,$ **$a5^0$** $, a6^0 \oplus T1^0 \oplus T2^1,$ **$a7^0$** $, rotr(a0^0,7) \oplus T1^0 \oplus T2^1,$ **$rotr(a1^0,7)$** $)$.

.....atd....

Runda 256:
V předchozích rundách byly proměnné T2 a T1 voleny tak, aby neměly žádný vliv na hodnoty a1, a3, a5, a7. Proto se stavový registr po 256 rundách vrací na lichých pozicích do původního stavu:
$A^{256} = (a0^{256},$ **$a1^0$** $, a2^{256},$ **$a3^0$** $, a4^{256},$ **$a5^0$** $, a6^{256},$ **$a7^0$** $)$.

3 Kolize ve stavovém registru

Pro jednoduchost "vyplňování" uvažujme, že prvních 13 slov zpráv je konstantních. Dále můžeme také přidat libovolné množství w-bitových slov a řekněme, že tak zvolíme prvních 256 slov. Tuto část zprávy nazýváme první stacionární částí (S1).

Nyní začínáme ze stavu, který vznikne po zpracování S1. Použijeme metodu popsanou výše a konstruujeme posloupnost stavů A^{256*1}, A^{256*2}, A^{256*3}, ..., dokud neobdržíme kolizi v této posloupnosti. Tato kolize vytváří pevný bod (nebo též cyklus), protože z tohoto bodu můžeme jít opět na začátek cyklu (a opakovat ho kolikrát chceme) nebo pokračovat dále. Po prvním cyklu uděláme 256 rund s náhodně volenými slovy zprávy W^t (abychom přeťali determinismus). Potom opět pokračujeme metodou výše a vytváříme posloupnost stavů, dokud neobdržíme druhou kolizi ve stavech A registru.

Složitost nalezení každé uvedené kolize narozeninovým paradoxem je pouze $2^{n/4}$ (přesněji $2^{n/4+1}$ včetně bitů carry).

Označme S1 první stacionární část, C1 část mezi prvními kolidujícími body, S2 druhou stacionární část (256 náhodných kroků) a C2 část mezi druhými kolidujícími body.

Poznamenejme, že cyklus C1 (C2) můžeme procházet kolikrát chceme (budeme to využívat k tomu, abychom kontrolní součty a délku zprávy dotlačili do správných hodnot).

4 Příklad kolize s využitím dvou pevných bodů

Zde popíšeme, jak použít pouze dva pevné body ke konstrukci jednoduché kolize (v další kapitole využijeme 10 pevných bodů ke konstrukci mnoha vzorů, tj. i ke kolizi). Nyní definujeme dvě kolidující zprávy M1 a M2.

První zpráva jde přes S1, poté N1 krát přes cyklus C1, jednou přes S2 a jednou přes cyklus C2.

Druhá zpráva jde jednou přes S1, jednou přes cyklus C1, jednou přes S2 a N2 krát přes cyklus C2.

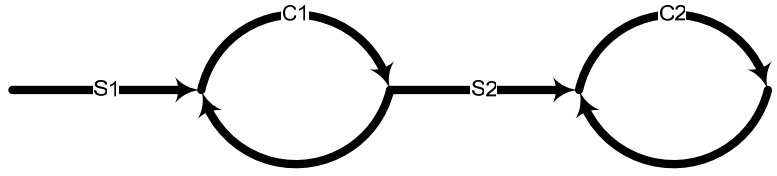
Označme L(S1), L(C1), L(S2), L(C2) počty rund, které odpovídají uvedeným částem S1, C1, S2, C2 posloupnosti stavů. Dále nechť L(M1), L(M2) je celkový počet rund při zpracování zpráv M1 a M2.

Hašovací hodnota je definována jako součet (zvlášť po slovech modulo 2^{32}) odpovídajících stavů A, obdržených při zpracování slov zprávy.

Označme S(S1), S(C1), S(S2) a S(C2) příspěvky A-stavů částí S1, C1, S2, C2 do celkové hašovací sumy. Nechť S(M1), S(M2) je celkový součet stavů, když se zpracuje celá zpráva M1 a M2.

Označme s(S1), s(C1), s(S2) a s(C2) součty slov zprávy v odpovídajících částech stavové posloupnosti.

Definujeme $N1 = 2^w * L(C2) + 1$, $N2 = 2^w * L(C1) + 1$. Potom M1 a M2 budou mít stejnou délku, kontrolní součty a hašovací hodnoty.



Délky

Délky zpráv ve slovech jsou

$L(M1) = L(S1) + Nl*L(C1) + L(S2) + 1*L(C2) = L(S1) + (2^w * L(C2) + 1)*L(C1) + L(S2) + L(C2) = L(S1) + L(C1) + L(S2) + L(C2) + 2^w * L(C2)*L(C1)$,

$L(M2) = L(S1) + 1*L(C1) + L(S2) + N2*L(C2) = L(S1) + L(C1) + L(S2) + (2^w * L(C1) + 1)* L(C2) = L(S1) + L(C1) + L(S2) + L(C2) + 2^w * L(C2)*L(C1)$,

tedy stejné $L = L(M1) = L(M2)$.

Kontrolní součty
Označme K délku zprávy ve slovech a X součet slov zprávy, $X = \sum_{t=1,...,K} W^t$. Potom máme (modulo 2^w)
$checksum1 = non(\sum_{t=1,...,K} W^t) = non\ X = 0xFF...FF - X = 1 - X$,
$checksum2 = \sum_{t=1,...,K} (nonW^t) = \sum_{t=1,...,K} (1 - W^t) = K - \sum_{t=1,...,K} W^t = K - X$.

Když zprávy mají stejné délky L(M1) a L(M2) a stejné součty všech slov X(M), pak také mají stejné checksum1 a checksum2.

Protože součet je počítán modulo 2^w, máme
$X(M1) = s(S1) + Nl*s(C1) + s(S2) + 1*s(C2) = s(S1) + (2^w * L(C2) + 1)*s(C1) + s(S2) + s(C2) = s(S1) + s(C1) + s(S2) + s(C2)$,

$X(M2) = s(S1) + 1*s(C1) + s(S2) + N2*s(C2) = s(S1) + s(C1) + s(S2) + (2^w * L(C1) + 1)*s(C2) = s(S1) + s(C1) + s(S2) + s(C2)$,

takže kontrolní součty zpráv M1 a M2 jsou stejné.

Průběžné hašovací hodnoty
M1 i M2 končí ve stejném posledním stavu - je to poslední stav cyklu C2. Vypočítejme h(M1) a h(M2).

Protože součty jsou počítány ze slov modulo 2^w, máme
$h(M1) = S(S1) + Nl*S(C1) + S(S2) + 1*S(C2) = S(S1) + (2^w * L(C2) + 1)*S(C1) + S(S2) + S(C2) = S(S1) + S(C1) + S(S2) + S(C2)$,

$h(M2) = S(S1) + 1*S(C1) + S(S2) + N2*S(C2) = S(S1) + S(C1) + S(S2) + (2^w * L(C1) + 1)*S(C2) = S(S1) + S(C1) + S(S2) + S(C2)$,

takže i tyto hodnoty jsou stejné.

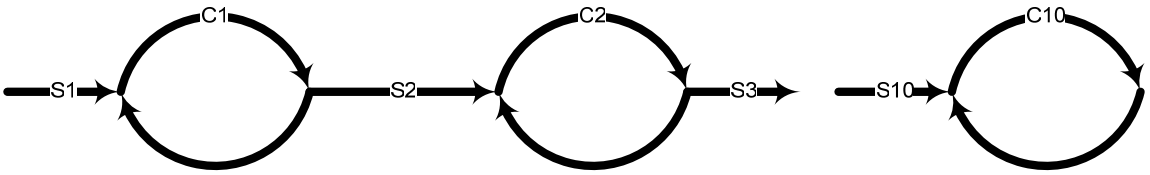
Hašovací hodnoty
K oběma zprávám můžeme nyní přidat jakýkoliv suffix. Potom dokončíme hašování zpracováním společné části: tj. část "vyplnění", délka, délka délky a kontrolní součty.

Složitost útoku a paměťové požadavky
Jak jsme viděli výše, všechny požadavky uvedeného útoku na Blender-n jsou řádově pouze $2^{n/4}$.

5 Multivzory

V této části popíšeme jinou metodu, která poskytuje velké množství multivzorů (tj. vytváří také multikolize).
Zvolme libovolnou hašovací hodnotu H. Vytvoříme zprávu M (velké množství zpráv M) tak, že h(M) = H, a to v následujících krocích:

1. Nastav první stacionární část na náhodnou hodnotu větší než 13. Použij proceduru z kapitoly 3 a najdi první kolizní cyklus C1. Následuje náhodná stacionární část S2 (mající malou náhodnou velikost) a druhý kolizní cyklus C2, stacionární část S3 (malá), ..., a končíme s S10 a cyklem C10. Označme A^{fin} finální stav stavového registru.
2. Nechť A^{fin} je finální stav všech uvažovaných zpráv. Poznamenejme, že A^{fin} je stav po zpracování všech slov zprávy (před zpracováním "doplňku").
3. Zvolme L jako bitovou délku zprávy, například kolem hodnoty $w*2^{n/2 + 8+w+\log(10)}$ (tj. $2^{n/2 + 8+w+\log(10)}$ w-bitových slov). Všechny uvažované zprávy budou mít tuto předepsanou délku.
4. Zvolme libovolnou hodnotu X jako budoucí součet všech slov zprávy. Všechny uvažované zprávy budou mít tento součet slov. Všechny zprávy budou mít také stejné "dokončení" (filling, délka, délka délky, kontrolní součty).
5. Nyní známe finální stav A^{fin} a hodnoty "dokončení", proto můžeme všechny tyto hodnoty zpracovat a obdržet hodnotu dH jejich příspěvku do hašovacího registru.
6. Nyní můžeme odstranit tento příspěvek z cílové hašovací hodnoty a příspěvek součtu slov z cílového součtu slov zprávy. Obdržíme "opravené" hodnoty H, L a X. Dále uvažujme, že L je vyjádřena přímo jako počet w-bitových slov zprávy.
7. Úlohou je najít zprávu (zprávy) s počtem slov L, součtem slov X, finálním stavem A^{fin} a hašovací hodnotou H.



Konstrukce multivzorů
Pro každé i = 1, ..., 10 označme

L(Ci) - délku cyklu Ci (ve w-bitových slovech),
s(Ci) - součet slov W^t, odpovídajících cyklu Ci
S(Ci) - součet stavů A^t, odpovídajících cyklu Ci

L(Si) - délku cyklu Si ve slovech,
s(Si) - součet slov W^t, odpovídajících stacionární části Si
S(Si) - součet stavů A^t, odpovídajících stacionární části Si

Vytvoříme velké množství zpráv M tak, že projdeme jednou stacionárními částmi S1, ..., S10 a mnohokrát a různě přes cykly C1, ..., C10. Pouze potřebujeme nastavit počty průchodů tak, aby součty slov, součty stavů a součty dílčích délek byly stejné a rovné předepsaným hodnotám. Pro každé i = 1, ..., 10 označme k_i počet průchodů cyklem Ci.

Potřebujeme řešit soustavu rovnic:

(H): $H = \sum_{i=1,...,10} k_i * S(C_i) \bmod 2^w$
(X): $X = \sum_{i=1,...,10} k_i * s(C_i) \bmod 2^w$
(L): $L = \sum_{i=1,...,10} k_i * L(C_i)$

Poznamenejme, že rovnice (H) je soustava osmi rovnic, protože X a $S(C_i)$ jsou vektory slov, (H) je jedna rovnice a (L) také. Máme 10 rovnic s 10 neznámými k_i. Kdyby (L) byla také modulární (mod 2^w), řešili bychom ji jednoduše Gaussovou eliminační metodou. Ale poslední rovnice obsahuje absolutní hodnoty, takže se jí budeme věnovat více.

Poznámka. Když zvolíme více cyklů (pevných bodů), budeme mít více stupňů volnosti v systému H-X-L (více rovnic než neznámých) a proto obdržíme mnohem více řešení. Přitom složitost příliš nenaroste - z násobícího koeficientu 10 bude 11. Zvýšení počtu cyklů můžeme také použít, pokud obdržený systém rovnic by byl lineárně závislý, a to nahrazením toho cyklu, který lineární závislost způsobuje, cyklem novým.

Řešení systému H-X-L

Označme dolní a horní část proměnné V jako $V^L = V \bmod 2^w$, $V^H = (V - V^L)/2^w = V >> w$.
Poznamenejme, že
$S(C_i) = S(C_i)^L$ a $s(C_i) = s(C_i)^L$, zatímco $L(C_i) = L(C_i)^H * 2^w + L(C_i)^L$ a $k_i = k_i^H * 2^w + k_i^L$.

Přepišme H-X-L:

(H): $H = \sum_{i=1,...,10} k_i^L * S(C_i)^L \mod 2^w$
(X): $X = \sum_{i=1,...,10} k_i^L * s(C_i)^L \mod 2^w$
(LL): $L^L = \sum_{i=1,...,10} (k_i^H * 2^w + k_i^L) * (L(C_i)^H * 2^w + L(C_i)^L) \mod 2^w$
(LH): $L^H = (\sum_{i=1,...,10} (k_i^H * 2^w + k_i^L) * (L(C_i)^H * 2^w + L(C_i)^L)) >> w$

Poslední dvě rovnice jsou

(LL): $L^L = \sum_{i=1,...,10} k_i^L * L(C_i)^L \mod 2^w$
(LH): $L^H = (\sum_{i=1,...,10} (k_i^H * 2^w * L(C_i) + k_i^L * L(C_i)^H * 2^w + k_i^L * L(C_i)^L)) >> w$
$= c$ [3] $+ \sum_{i=1,...,10} (k_i^H * L(C_i) + k_i^L * L(C_i)^H)$

Nyní můžeme najít řešení systému H-X-LL 10 lineárních rovnic s 10 neznámými k_i^L (mod 2^w):

(H): $H = \sum_{i=1,...,10} k_i^L * S(C_i)^L \mod 2^w$
(X): $X = \sum_{i=1,...,10} k_i^L * s(C_i)^L \mod 2^w$
(LL): $L^L = \sum_{i=1,...,10} k_i^L * L(C_i)^L \mod 2^w$

Potom nahradíme k_i^L ve zbývající rovnici (LH) a máme

(LH): $L^H = \sum_{i=1,...,10} (k_i^H * L(C_i) + k_i^L * L(C_i)^H) = \sum_{i=1,...,10} (k_i^H * L(C_i)) + CC$
kde CC je konstanta s hodnotou $\sum_{i=1,...,10} k_i^L * L(C_i)^H$.

Zbývá řešit jednu rovnici s 10 neznámými proměnnými k_i^H. Připomeňme, že se jedná o diofantickou rovnici s tím rozdílem, že hledáme její nezáporná řešení. Budeme jí řešit hrubou silou[4].

Cykly C_i budou obsahovat kolem $2^{n/4}$ bodů. Body jsou stavy po zpracování 256 slov, takže konstanty $L(C_i)$ budou kolem $2^{n/4} * 2^8$ a CC kolem $10 * 2^w * 2^{n/4} * 2^8 \leq 2^{\log(10)+w+n/4+8}$.

---

[3] c je přenos z dolní části, může nabývat maximálně 10 hodnot, takže v dalším můžeme uvažovat, že c = 0.
[4] jistě existují efektivnější metody

Hodnota L^H je kolem $2^{n/2+8+w+\log(10)}/2^w = 2^{n/2+8+\log(10)}$, takže hodnota $L^{rest} = L^H$ - CC bude také kolem $2^{n/2+8+\log(10)}$. Máme
(LH): $L^{rest} = \sum_{i=1,...,10} k_i^H * L(C_i)$.
Řekněme, že cyklus C_1 je nejmenší z cyklů. Nyní můžeme nastavit 9 proměnných k_i^H pro i = 2, ..., 10 libovolně a pak vypočítat k_1^H z rovnice (LH). Jestliže výraz $L^{rest} - \sum_{i=2,...,10} k_i^H * L(C_i)$ bude dělitelný $L(C_1)$, dostaneme jedno řešení. To se stane s pravděpodobností $1/L(C_1)$. Protože rozdíl mezi L^{rest} a $L(C_i)$ je ohromný, můžeme očekávat ohromný počet řešení, více než zhruba $((2^{n/2+8+\log(10)}/10) / 2^{n/4} * 2^8)^9 / L(C_1) \geq 2^{2n}$.

Každé řešení reprezentuje cestu z počátečního stavu, přes různý počet průchodů 10 pevnými body a končí v posledním známém jedinečném stavu. Proto všechny tyto zprávy mají stejnou předepsanou hašovací hodnotu.

Složitost nalezení 2^{2n} multivzorů (multikolizí) funkce Blender-n je zhruba 10 krát větší, než nalezení kolize náhodné hašovací funkce, ale s n/2-bitovým výstupem.

6 Závěr

Ukázali jsme 2^{2n}-multikolize a 2^{2n}-multivzory pro Blender-n pro všechny výstupní délky se složitostí zhruba $10*2^{n/4}$.

7 Literatura

[1] Colin Bradbury: BLENDER, A Proposed New Family of Cryptographic Hash Algorithms, http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/documents/Blender.zip

[2] Antoine Joux: Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions, CRYPTO 2004, LNCS, Vol. 3152, pp. 306-316. Springer, 2004

[3] Vlastimil Klima: A near-collision attack on BLENDER, http://cryptography.hyperlink.cz/BMW/near_collision_blender.pdf

[4] Florian Mendel: Preimage Attack on Blender, http://ehash.iaik.tugraz.at/uploads/4/48/Blender-preimage.pdf

[5] Craig Newbold: Observations and Attacks On The SHA-3 Candidate Blender, http://ehash.iaik.tugraz.at/uploads/4/48/Blender-preimage.pdf

[6] Liangyu Xu: Semi-free start collision attack on Blender, http://ehash.iaik.tugraz.at/uploads/4/48/Blender-preimage.pdf