Crypto-World

Informační sešit GCUCMP

ISSN 1801-2140

Ročník 11, číslo 3/2009 **15.březen 2009**

3/2009

Připravil: Mgr. Pavel Vondruška

Sešit je přednostně distribuován registrovaným čtenářům.
Starší sešity jsou dostupné na adrese
http://crypto-world.info
(1310 registrovaných odběratelů)

Obsah :

Příloha: ---

A. Prvá konferencia SHA-3 kandidátov

Michal Hojsík, MFF UK, hojsik@matfyz.cz

Koncom februára som mal tú možnosť zúčastniť sa **"The First SHA-3 Candidate Conference"** usporiadanej americkým úradom pre štandardy a technológie NIST. Konferencia sa konala v belgickom Leuven hneď po workshope Fast Software Encryption 2009 (ďalej FSE09). V tomto príspevku sa pokúsim popísať moje postrehy z tejto konferencie, a to technické aj netechnické. Nebudem sa venovať obsahovej stránke jednotlivých prezentácii. Tie sú voľne dostupné na stránkach NISTu [1] a tí, ktorí sa o súťaž o nový štandard zaujímajú, si ich pravdepodobne už dávno prešli.

Ako už samotný program prezrádzal, veľká časť konferencie pozostávala z prezentácii jednotlivých funkcií. Pre každý návrh bolo vyhradených 18 minút, počas ktorých rečník popísal danú funkciu, ale hlavne sa snažil poukázať na prednosti svojho návrhu. Vyzdvihovali sa (preukázateľná) bezpečnosť, rýchlosť, novosť, ale napríklad aj skúsenosti návrhárov v oblasti hešovacích funkcií.

Pred samotným konaním konferencie bolo v súťaži **42 návrhov**, pričom 5 funkcií v programe chýbalo, a teda nemalo byť na konferencii prezentovaných. Nakoniec sa ale z pôvodne plánovaných 37 prednášok uskutočnilo iba 36. Dôvod bol ten, že autor funkcie SHAMATA, turecký kryptológ Orhun Kara, svoj návrh stiahol. Stalo sa tak potom, ako bol na rump session workshopu FSE09 odprezentovaný príspevok popisujúci praktické kolízie funkcie SHAMATA-256 [2].

Ďalší zaujímaný príspevok na rump session FSE09 týkajúci sa súťaže o SHA-3 mal Dmitry Khovratovich [3]. Ten vo svojej prezentácii uviedol, že pomocou fixovania bitov vstupu a výstupu dokáže odlíšiť kompresnú funkciu MD6 obmedzenú na 33 rúnd od náhodnej funkcie. Ako som sa neskôr dozvedel Dmitry k tomuto výsledku dospel pomocou novej (zatiaľ nepublikovanej) kryptoanalytickej techniky. V najbližšej dobe teda môžeme očakávať ďalšie zaujímavé výsledky od skupiny kryptológov na Luxemburskej univerzite a je pravdepodobné, že sa nebudú týkať iba hešovacích funkcií.

Pretože 36 prezentácii nových hešovacích funkcií je na 3 dni vcelku veľa, bolo každé spestrenie vítané. O prvé sa postaral Jongsung Kim hneď v prvej sekcii. Na začiatku

prezentácie funkcie ARIRANG vysvetlil, že ARIRANG je meno najpopulárnejšej kórejskej ľudovej piesne a následne z nej spustil asi minútovú ukážku. Ďalší autori ale taktiež nezaháľali a väčšina z nich nezabudla objasniť pôvod mena ich funkcie. Keď potom poobede počas druhého dňa konferencie prezentoval Ron Rivest funkciu MD6, už len s úsmevom podotkol že MD6 nie je označenie žiadnej piesne, jedla ani zeleniny, ale iba prosté „message-digest“.

Pre mnohých ale zaujímavejšiu časť konferencie tvorili diskusie a prezentácie NISTu. Hneď prvý deň sa diskutovalo o cieľových platformách pre ktoré by mala budúca funkcia byť vhodná. Ako NIST uviedol vo svojich požiadavkách [4], v prvom kole bude dôležitá hlavne výkonnosť jednotlivých kandidátov pri softwarových implementáciách, a to predovšetkým na 32 bitovom Intel procesore a 64 bitovom procesore od AMD. Mimo záujem ale nie sú ani 8 bitové procesory a NIST uviedol, že uvíta každé informácie o implementáciách na takýchto platformách. Rýchlosť hardwarových implementácii jednotlivých kandidátov sa ale bude detailne posudzovať až v druhom kole, a to pravdepodobne aj kvôli náročnosti takého posudzovania. Väčšina tímov sa teda sústredila a svoje návrhy optimalizovala hlavne pre 32 a 64 bitovú architektúru. Boli ale aj takí, ktorí navrhli hešovaciu funkciu cielene vhodnú aj pre

mikroprocesory. Takýmito návrhmi sú napríklad CubeHash Dana Bernsteina alebo EnRUPT z dielne Sean O'Neila.

Jeden z príspevkov v diskusii o platformách bol od zamestnankyne firmy Motorola. Tá vyjadrila názor, že z hľadiska hardwarových implementácií by bolo veľmi vhodné, aby nová hešovacia funkcia pozostávala výhradne z „bežných" stavebných prvkov. Nové prvky by podľa nej s veľkou pravdepodobnosťou znamenali oveľa zložitejšie, priestorovo väčšie, ale hlavne drahšie implementácie. Ako neskôr podotkol Niels Ferguson z Microsoftu, istá nemenovaná spoločnosť vydávajúca 500 miliónov platobných kariet každý rok sa už v minulosti rozhodla neinvestovať do bezpečnejších kariet len z dôvodu ich vyššej ceny. Pri tomto množstve kariet totiž nárast ceny o 10 centov na kartu znamená 50 miliónov ročne, čo je čiastka presahujúca prípadné straty. Ďalších príspevkov v diskusii o platformách volal po vhodnosti SHA-3 pre mikroprocesory používané v RFID, resp. v senzorových sieťach. Nato reagoval Adi Shamir prehlásením, že trh sa bude v budúcnosti určite rozrastať o ďalšie a ďalšie procesory, takže by bolo vhodné aby NIST stanovil konkrétne záujmové platformy, pre ktoré sa bude testovať a optimalizovať. NIST následne prehlásil, že by uvítal návrhy na cieľové platformy z komerčnej sféry.

Ďalšia diskutovaná téma boli parametre navrhnutých funkcií. NIST vo svojom „request for candidate algorithm" [4] uviedol, že bude uprednostňovať flexibilné algoritmy. Taktiež upresnil, že pod flexibilitou sa môžu (nie výhradne) chápať napríklad zmeny algoritmu pomocou zmeny vybraných parametrov (napr. počet rúnd, dĺžka bloku) umožňujúce security/performance tradeoffs. To spôsobilo, že veľké množstvo návrhov obsahuje takéto meniteľné parametre. Jeden z príspevkov z publika počas diskusie ale proti prehnanej voľnosti namietal. Argumentoval tým, že ak bude štandard obsahovať väčšie množstvo parametrov je takmer isté že viacero jeho implementácii bude obsahovať ich (z bezpečnostného hľadiska) najhoršie možné kombinácie. K tomu sa pridal aj Adi Shamir a doporučil NISTu aby vyzval autorov jednotlivých návrhov na zviazanie parametrov do jediného bezpečnostného parametra, čím by sa predišlo problémom so zlými kombináciami parametrov.

Ďalšia téma, ktorá publikum veľmi zaujímala bola, či NIST dovolí zmeny v algoritmoch. V už spomínanom dokumente [4] NIST uvádza, že počas prvého kola nepovolí v návrhoch žiadne zmeny. Otázka teda znela, ako to bude s kolom druhým. Je jasné, že zmeny v návrhoch

môžu byť problematické. Často znehodnotia už prevedenú analýzu, ale hlavne (narýchlo urobené) zmeny môžu zaniesť do navrhovaných algoritmov nové zraniteľnosti (prípad samosynchrónnej prúdovej šifry Mosquito a jej zmenenej verzie Moustique). NIST nakoniec v jednom zo svojich príspevkov [5] prehlásil, že dovolí návrhárom funkcií vybraných do druhého kola zmeniť odporúčané parametre a rozsah parametrov. Dovolí ale taktiež „relatively minor algorithm changes“, pokiaľ ich návrhári dostatočne odôvodnia. NIST vyzval autorov funkcií, aby začali čo najskôr pracovať na potenciálnych zmenách, pretože po vyhlásení funkcií postupujúcich do druhého kola na to nebudú mať dostatok času.

NIST taktiež počas konferencie prehlásil, že relatívne pre výber kandidátov do druhého kola budú iba výsledky oznámené do prvého júna, pričom funkcie ktoré postúpia do druhého kola oznámi najneskôr počas konferencie Crypto 2009 ktorá sa koná v auguste. K tomu zaznela z hľadiska požiadavka, aby NIST publikoval tzv. „focus functions“ teda funkcie, ktoré majú veľkú šancu dostať sa do druhého kola. To ale William Burr ako zástupca NISTu kategoricky odmietol. Dotyčný divák hneď reagoval požiadavkou, nech teda NIST aspoň zverejní „neperspektívne“ funkcie načo William Burr opäť povedal, že NIST nebude robiť žiadny predbežný výber. Následne reagoval aj Bruce Schneier prehlásením, že je dosť jasné, ktoré funkcie sú perspektívne a ktoré nie, a to aj bez prehlásenia NISTu. Po ukončení konferencie som sa zúčastnil diskusie na túto tému so zástupcami NISTu. Počas nej Eli Biham povedal že on, a pravdepodobne aj viacero ďalších kryptológov, začnú pracovať na analýze kandidátov až v druhom kole, aby náhodou nestrácali čas na potenciálne neperspektívnych, aj keď možno zaujímavých funkciách. NIST by im teda oznámením „focus functions“ dal viac času a prispel by tak k lepšej analýze. Zástupcovia NISTu povedali, že to zvážia.

Posledná téma ktorú by som chcel spomenúť, a ktorá bola predmetom početných diskusií, sú návrhy používajúce (časti) AES alebo všeobecne S-boxy ako stavebný prvok. Z pôvodných 51 kandidátov prvého kola až 33 používa v kompresnej funkcii S-box(y) a nich veľká časť používa (časti) AES. Pre dizajnérov bolo použitie AES vďaka jeho vlastnostiam asi jasná voľba. NIST ale jednoznačne povedal, že v druhom kole chce mať veľkú rôznorodosť, takže väčšina AES používajúcich funkcií vypadne. Viacero tímov taktiež asi vsádzalo na to, že potom, ako Intel uvedie procesory s AES inštrukciami získajú ich funkcie na Intel procesoroch rýchlostnú prevahu nad ostatnými návrhmi. Pre ilustráciu, v súčasnosti dosahuje najrýchlejšia uverejnená softwarová implementácia AES rýchlosť približne 10.5 cpb (cycles per byte) na procesoroch Intel Core 2 Quad Q9550, pričom v blízkej dobe bude uverejnená

implementácia dosahujúca na rovnakých procesoroch rýchlosť až 8.1 cpb. No a v rámci svojej pozvanej prednášky na FSE09 predstavil Shay Gueron z Intelu nové AES inštrukcie, ktoré budú obsahovať procesory Intel dostupné na trhu už tento rok, a ktoré umožnia implementácie AES s rýchlosťou až 0.75 cpb. Čiastočne ale asi sklamal viacero návrhárov tým, že nové inštrukcie nebudú obsahovať samotné dielčie transformácie AES (SubBytes, ShiftRows, MixColumns), ale celú AES rundu ako celok. K jednotlivým transformáciám sa bude dať dostať až kombináciou viacerých procesorových inštrukcií. V každom prípade zrýchlenie týchto kandidátov sa bude týkať zatiaľ iba nových procesorov Intel aj keď sa už potichu hovorí o tom, že AMD asi taktiež príde s AES inštrukciami.

Viackrát počas konferencie zaznela námietka proti návrhom používajúcim AES (alebo obecne S-boxy) týkajúca sa ich bezpečnosti voči útokom pomocou postranných kanálov. Rýchle implementácie totiž používajú table lookups, ktoré viedli k útokom pomocou postranných kanálov na AES (napr. cache-timing). Proti table lookups v budúcej SHA-3 sa počas konferencie vyjadrilo viacero známych osobností ako napríklad Ron Rivest, Dan Bernstein alebo Niels Ferguson. Návrhári funkcií používajúcich AES oponovali tým, že nové bit-sliced implementácie AES sú odolné voči cache-timing útokov, ale napríklad modulárne sčítanie použité v iných návrhoch je zraniteľné voči differential-power-analysis útokom, proti čomu ale nikto nevystupuje. Podľa nich použitie AES znamená hlavne známe vlastnosti, dôveru v jeho bezpečnosť a taktiež prípadné využitie už existujúcej implementácie AES.

Na záver len poznamenám, že ďalšia SHA-3 konferencie sa bude konať tesne po konferencii Crypto 2010 v americkej Santa Barbare.

[1] http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html

[2] http://fse2009rump.cr.yp.to/3afe8f8890d89a83ff3564a6ffc3eddd.pdf

[3] http://fse2009rump.cr.yp.to/fe1a0e11287a9864c1d897a3110ebaa2.pdf

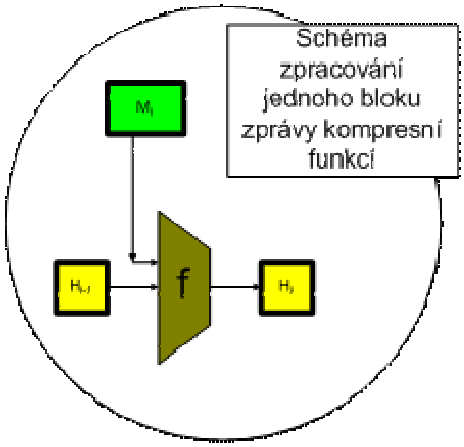
[4] http://csrc.nist.gov/groups/ST/hash/federal_register.html

[5]http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/Feb2009/documents/Soura_TunableParameters.pdf

B. Blue Midnight Wish, popis a principy

Vlastimil Klíma, nezávislý konzultant - kryptolog, Praha,
(http://cryptography.hyperlink.cz, v.klima@volny.cz)

Připomeňme si v krátkosti, co mají BMW a SHA-2 společné. Jsou založeny na iterativním principu a kompresní funkci. Zpráva, která se má hašovat, se doplní definovaným způsobem tzv. paddingem a počtem zpracovávaných bitů původní zprávy a zarovná se na nejbližší násobek délky bloku buď 512/1024 bitů podle toho, zda se jedná o BMW256/BMW512 nebo SHA256/SHA512.



Obr.: Iterativní princip hašovací funkce

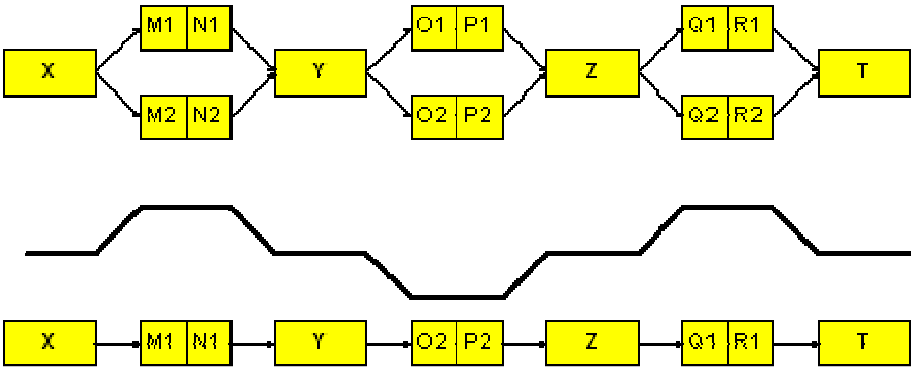
Potom se tyto funkce shodují v tom, že používají tzv. iterativní výpočet s použitím tzv. kompresní funkce a průběžné hašovací hodnoty. Průběžná hašovací hodnota se nastaví na počátku na hodnotu tzv. inicializačního vektoru. Potom se v N krocích vždy ze staré průběžné hašovací hodnoty a daného bloku zprávy pomocí kompresní funkce vytvoří nová hodnota průběžné haše. Poslední průběžná hodnota haše (nebo její část) je pak prohlášena za skutečnou hodnotu haše. Hašování tedy probíhá u BMW i SHA podle stejného následujícího scénáře.

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 $H_0 = IV$.

2. Výpočet haše
For i = 1 to N
{
$H_i = f(M_i , H_{i-1})$
}
3. Závěr
H (M) = definovaných n bitů z hodnoty H_N.

Multikolize
Co NISTu vadilo na SHA-2? Především to byla Jouxova práce [17], která ukázala, že funkce typu SHA-2 a mnoho jiných umožňují nalézt multikolize rychleji, než u náhodného orákula. A to i přesto, že samotné SHA-2 byly odolné proti (jednoduché) kolizi! To znamená, že nalézt jednoduchou kolizi k SHA2-n (n je počet bitů výstupní haše, pro jednoduchost uvažujme n =

256/512) vyžadovalo $2^{n/2}$ výpočtů pro narozeninový paradox. V tomto směru byla SHA2-n velmi kvalitní, stejně kvalitní jako náhodné orákulum (náhodná funkce). To, co vadilo, byl právě objev Jouxe [17], který ukázal, že konstrukce iterativního typu umožňují nalézt multikolize poměrně snadno, a to pomocí kolizí jednoduchých, pokud je výsledná haš stejně široká, jako průběžná haš (tj. n-bitová). U náhodného orákula je k dosažení $r = 2^k$ multikolizí nutno provést zhruba $2^{n*(r - 1) /r}$ operací, tedy počet operací blízký 2^n, zatímco Joux navrhl metodu, kde postačilo pouze $k * 2^{n/2}$ operací. Také byla nalezena možnost konstrukce multivzorů pomocí pevných bodů [18], využívajících stejných slabin iterativní konstrukce. Muselo se tedy něco změnit na vlastní konstrukci hašovacích funkcí.



Obr.: Multikolize pomocí jednoduchých kolizí

Útok prodloužením zprávy

NISTu také vadil fakt, že funkce typu SHA-2 jsou náchylné na útok prodloužením zprávy. Pokud útočník zná hodnotu H(M), může spočítat libovolnou hodnotu H(extM), kde extM je rozšíření zprávy M na zprávu M_1, ..., M_N, E pro libovolný řetězec E. Útočník prostě vezme hašovací hodnotu H(M). Ví, že to je průběžná hašovací hodnota po zpracování bloku M_N, a proto jen pokračuje ve zpracování řetězce E, aniž by musel znát zprávu M. Samozřejmě opět za podmínky, že výsledná haš je stejně široká, jako průběžná haš. Útok vadí například při tzv. naivním použití haše při autentizaci zprávy M klíčem K, pomocí hodnoty H(K || M). Útočník může ke zprávě M přidat svůj libovolný doplněk E a dopočítat výše uvedeným způsobem H(K || M || E), což vypadá, že znal hodnotu K. Proto byla požadována odolnost proti tomuto útoku.

Dvojnásobná pumpa

K řešení těchto problémů byly v roce 2005 a 2006 konány dvě speciální konference NISTu [19], [20] ještě před vypsáním soutěže o nový standard SHA-3. Na nich bylo prezentováno několik myšlenek, jak zabránit multikolizím a multivzorům novými konstrukcemi. Problém je v tom, že přirozené a prakticky použitelné hašovací funkce, tj. rychlé a bez zvláštních nároků na paměť, mají všechny konstrukci iterativního typu, ale právě ony mají uvedené generické neduhy. Zvláštností jsou například konstrukce typu stromové struktury, vyžadující více paměti apod. Jednu z elementárních myšlenek navrhl Lucks [16], a to zdvojnásobit šířku průběžné haše a jako výslednou haš brát pouze polovinu výsledné průběžné haše. V takové konstrukci n-bitové hašovací funkce (má n bitů výstupu) by bylo potřeba k nalezení *k* vnitřních kolizí pro Jouxův útok celkem $k * 2^{2n/2} = k * 2^n$ operací, což je nyní velmi dostačující obrana. Dvojnásobná vnitřní šířka však při výpočtu haše trvá zhruba 4x déle, protože kvalitní 2n bitový (průběžný) výstup lze obvykle zajistit přibližně se čtyřnásobnou složitostí. To byla zase rána pro návrháře! Avšak část kandidátů, přihlášených do soutěže o standard SHA-3 této myšlenky využila. Začalo se jí říkat dvojnásobná pumpa (double pipe). Dvojnásobná pumpa zabraňuje všem výše uvedeným útokům.

Konstrukce BMW využívá také dvojnásobné pumpy, ale uvidíme, že BMW má uvnitř a *přechodně* dokonce čtyřnásobnou pumpu. Dobře je čtyřnásobná pumpa vidět z obrázku z originální dokumentace. Je to hodnota (Q_a, Q_b). Přitom Q_a, Q_b a H_i jsou dvojnásobně široké oproti klasické jednoduché pumpě (výsledná haš je polovina hodnoty H_N), takže vnitřní stav BMW je v jednom okamžiku - (Q_a, Q_b), tj. dokonce čtyřnásobně široký oproti výsledné haši.

Algorithm: BLUE MIDNIGHT WISH

Input: Message M of length l bits, and the message digest size n.

Output: A message digest *Hash*, that is n bits long.

1. Preprocessing
   (a) Pad the message M.
   (b) Parse the padded message into N, m-bit message blocks, $M^{(1)}, M^{(2)}, \ldots, M^{(N)}$.
   (c) Set the initial value of the double pipe $H^{(0)}$.
2. Hash computation

   For $i = 1$ to N

   {

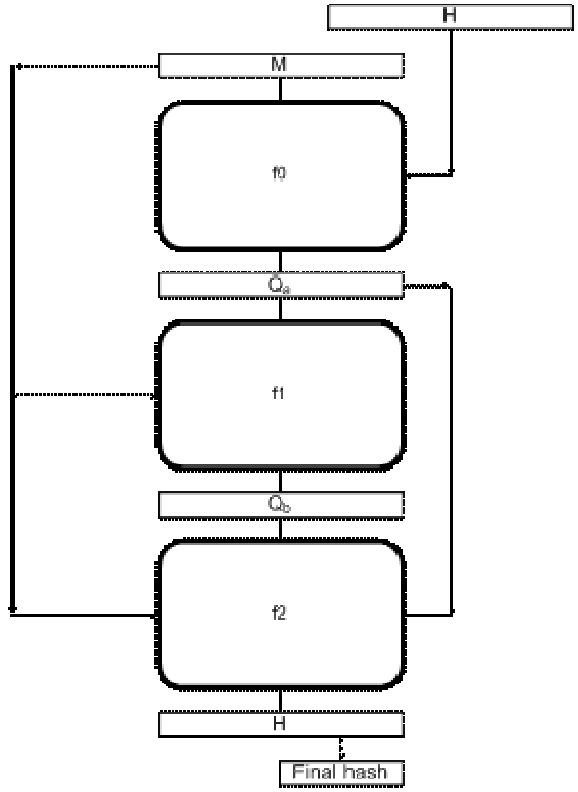
   $Q_a^{(i)} = f_0(M^{(i)}, H^{(i-1)})$;

   $Q_b^{(i)} = f_1(M^{(i)}, Q_a^{(i)})$;

   $H^{(i)} = f_2(M^{(i)}, Q_a^{(i)}, Q_b^{(i)})$;

   }
3. *Hash* =Take_n_Least_Significant_Bits$(H^{(N)})$.

Obr.: Dvojnásobná (a lokálně dočasně čtyřnásobná) pumpa u BMW



Obr.: Základní schéma BMW

Základní struktura

BMW lze představit na různých úrovních podrobnosti. Základem jsou funkce f_0, f_1, f_2. Stará hodnota průběžné haše H a blok zprávy M vstupuje do funkce f_0 a vytváří proměnnou Q_a. Q_a a M vstupuje do funkce f_1 a vytváří proměnnou Q_b. A konečně nyní všechny tři Q_b a Q_a a M vstupují do funkce f_2 a vytváří novou hodnotu průběžné haše H. Připomeňme, že hodnoty M, Q_a, Q_b, H jsou dvojnásobně široké než je finální hašovací hodnota.

Základní princip - míchání

Funkce f_0

Funkce f_0 má ve skutečnosti jakoby jeden vstup. Na počátku totiž dojde ihned k binárnímu součtu vstupních hodnot ($M \oplus H$) a poté se už pracuje jen s touto hodnotou (ale jen v f_0!, M samotné ještě vstoupí do hry v f_2). f_0 je vůči tomuto vstupu bijektivní transformace, která má za cíl rozprostřít malou změnu vstupu do co nejvíce bitů výstupu, hodnoty Q_a.

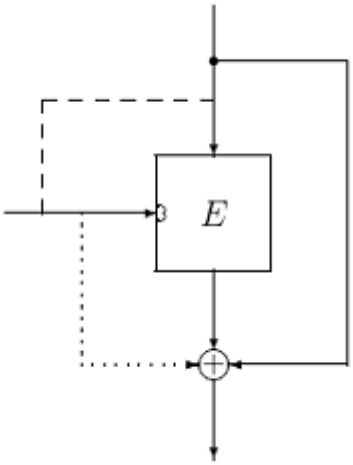
Funkce f_1

Funkce f_1 je jiná, má dva vstupy a může být chápána jako slabá bloková šifra, u níž je klíč tvořen vstupem M a otevřený text je tvořen vstupem Q_a. Výstupem je "šifrový text" Q_b. Tato funkce sice není konstruována jako bloková šifra, ale míchá bity "otevřeného textu (Q_a)" a "klíče (M)".

Funkce f_2

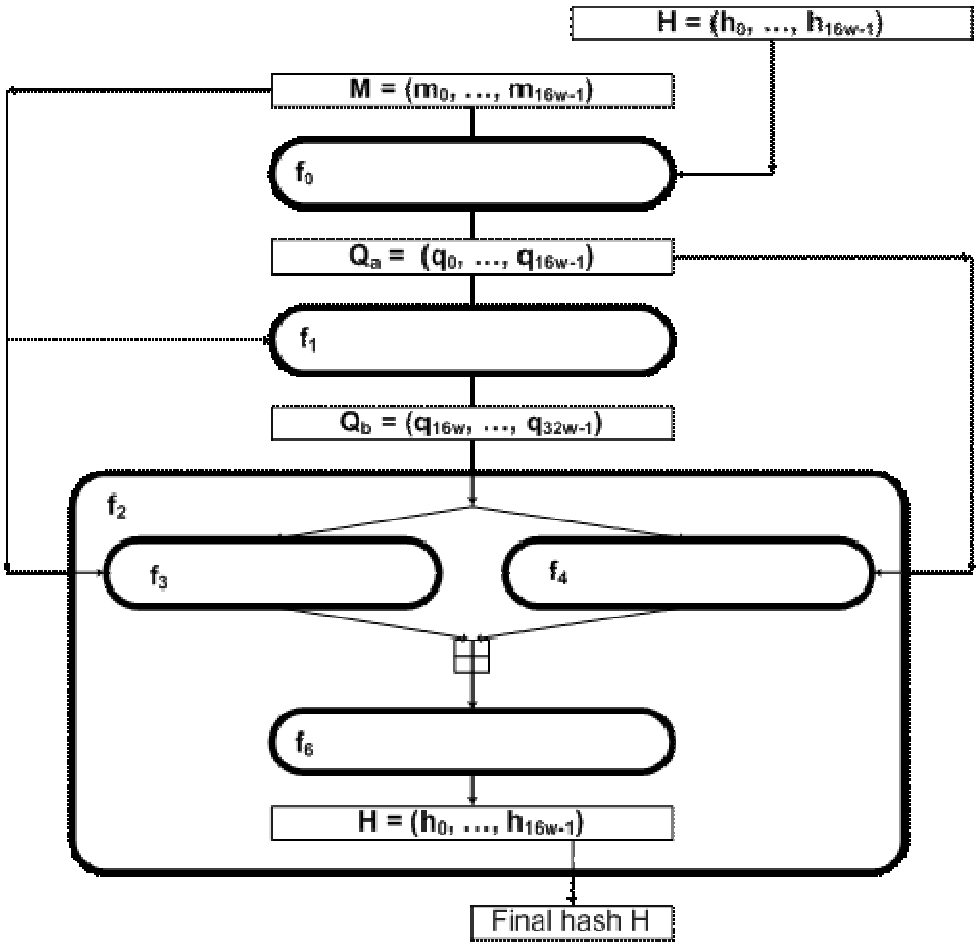
Konečně funkce f_2 je opět trochu jiná než předchozí dvě, neboť má tři vstupy - M, Q_a a Q_b a jediný výstup - novou hodnotu H. Tato funkce míchá a komprimuje všechny tři vstupy do výstupu. K tomu využívá určitým způsobem bijekce a různé další principy.

no.	function expression
1	$E(H_{i-1}, X_i) \oplus X_i$
2	$E(H_{i-1}, X_i \oplus H_{i-1}) \oplus X_i \oplus H_{i-1}$
3	$E(H_{i-1}, X_i) \oplus X_i \oplus H_{i-1}$
4	$E(H_{i-1}, X_i \oplus H_{i-1}) \oplus X_i$
5	$E(X_i, H_{i-1}) \oplus H_{i-1}$
6	$E(X_i, X_i \oplus H_{i-1}) \oplus X_i \oplus H_{i-1}$
7	$E(X_i, H_{i-1}) \oplus X_i \oplus H_{i-1}$
8	$E(X_i, X_i \oplus H_{i-1}) \oplus H_{i-1}$
9	$E(X_i \oplus H_{i-1}, X_i) \oplus X_i$
10	$E(X_i \oplus H_{i-1}, H_{i-1}) \oplus H_{i-1}$
11	$E(X_i \oplus H_{i-1}, X_i) \oplus H_{i-1}$
12	$E(X_i \oplus H_{i-1}, H_{i-1}) \oplus X_i$

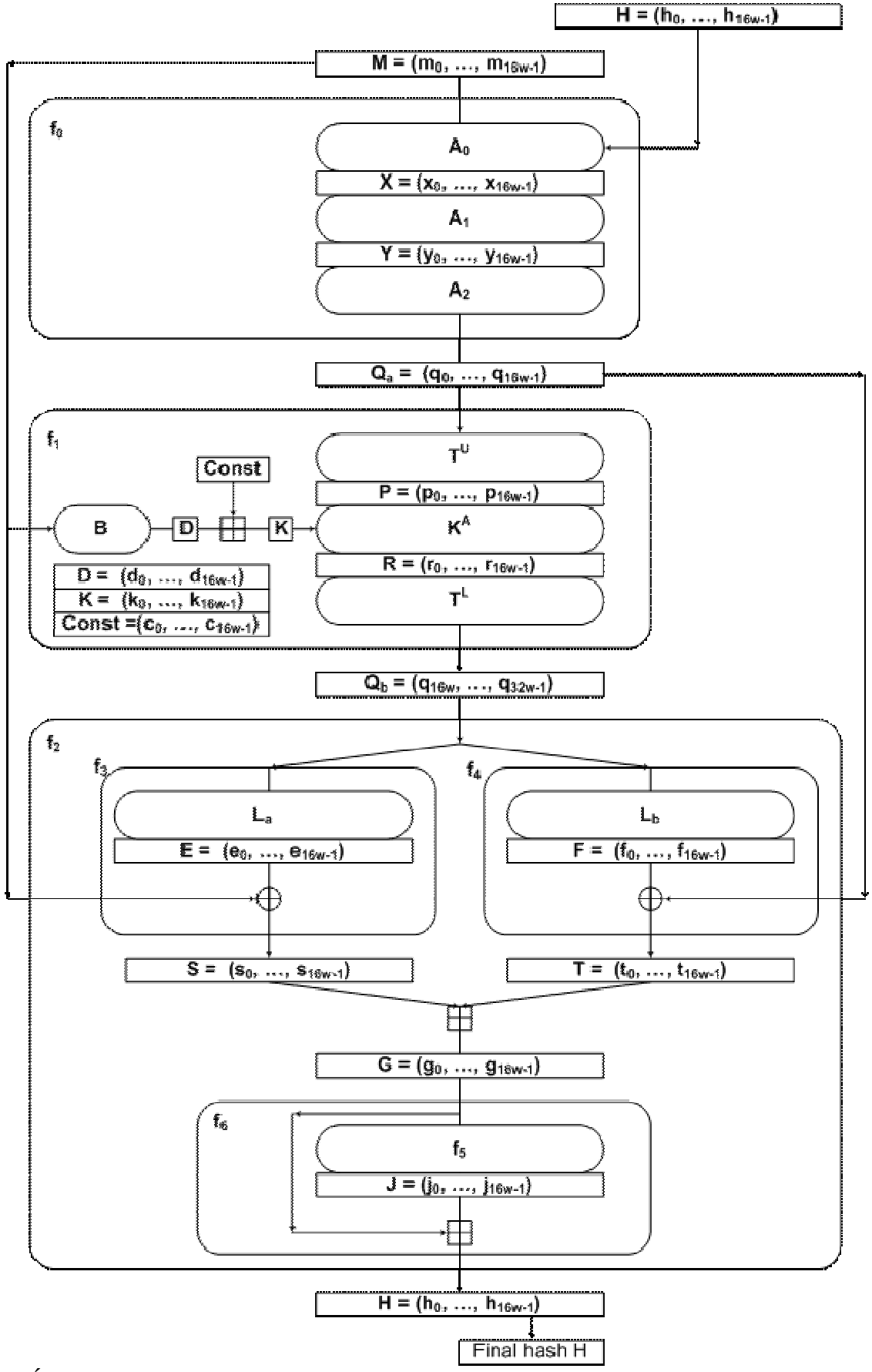


Obr.: Bezpečná schémata typu PGV [21], konstruovaná pomocí blokové šifry

Z hlediska blokové šifry se ve funkci f_2 míchá dohromady otevřený text, šifrový text i klíč. Proto se také v dokumentaci BMW hovoří o tom, že toto je určité zobecnění všech 12 "bezpečných" schémat z práce PGV [21], zahrnujících klasickou Davies-Meyerovu konstrukci (schéma PGV číslo 5) nebo Myiaguchi-Preneelovu konstrukci (schéma PGV číslo 7) nebo zbylých 10 z tzv. bezpečných schémat typu PGV. Všechna tato schémata totiž také míchají otevřený text, klíč a šifrový text blokové šifry nějakým způsobem dohromady. Protože f_2 je složitější funkcí, hovoříme o zobecnění, i když je to pouze v intuitivní rovině.



Obr.: Podrobnější dekompozice BMW



Obr.: Úplná dekompozice BMW (w je šířka slova, pro BMW256/512 je w = 32/64)

Podrobnější popis funkcí

Nyní se podíváme na funkce f_0, f_1, f_2 podrobněji. Obrázky platí pro obě základní verze BMW256/512, které se jen liší počtem bitů slova w = 32/64.

Téměř všechny transformace na obrázku jsou bijekcí, buď jednoho vstupu nebo obou, když jeden je fixován. Výjimkou je bijektivní lineární matice L, která je rozdělena na dvě části L_a a L_b, které nejsou bijekcí. Obě části L_a a L_b se však (aritmeticky) sčítají, a proto při zanedbání bitů přenosu dochází při jejich sečtení ke sloučení binárnímu: $L = L_a \oplus L_b$. Tím vzniká bijektivní lineární obraz $L(Q_b)$ vstupu Q_b ve dvou částech. Pokud uvažujeme bity přenosu, tyto bity ovlivňují bity vyšší, ale jen se snižující se pravděpodobností, a proto základní zobrazení $L(Q_b)$ zůstává z pravděpodobnostního hlediska dominantní částí součtu funkcí f_3 a f_4.

V celém schématu jsou použité funkce kompozicí dílčích funkcí, přičemž je (až na 1 výjimku) použito pravidlo, že se střídají aritmetické a binární operace. Tím je výsledná kompozice nelineární jak vzhledem k binárnímu vyjádření, tak vzhledem k aritmetickému (číselnému, mod 2^{32} nebo 2^{64}) vyjádření.

$$f_0 : \{0,1\}^{2m} \rightarrow \{0,1\}^m$$

Input: Message block $M^{(i)} = (M_0^{(i)}, M_1^{(i)}, \ldots, M_{15}^{(i)})$, and the previous double pipe $H^{(i-1)} = (H_0^{(i-1)}, H_1^{(i-1)}, \ldots, H_{15}^{(i-1)})$.
Output: First part of the quadruple pipe $Q_a^{(i)} = (Q_0^{(i)}, Q_1^{(i)}, \ldots, Q_{15}^{(i)})$.

1. Bijective transform of $M^{(i)} \oplus H^{(i-1)}$:

$$\begin{aligned}
W_0^{(i)} &= (M_5^{(i)} \oplus H_5^{(i-1)}) - (M_7^{(i)} \oplus H_7^{(i-1)}) + (M_{10}^{(i)} \oplus H_{10}^{(i-1)}) + (M_{13}^{(i)} \oplus H_{13}^{(i-1)}) + (M_{14}^{(i)} \oplus H_{14}^{(i-1)})\\
W_1^{(i)} &= (M_6^{(i)} \oplus H_6^{(i-1)}) - (M_8^{(i)} \oplus H_8^{(i-1)}) + (M_{11}^{(i)} \oplus H_{11}^{(i-1)}) + (M_{14}^{(i)} \oplus H_{14}^{(i-1)}) - (M_{15}^{(i)} \oplus H_{15}^{(i-1)})\\
W_2^{(i)} &= (M_0^{(i)} \oplus H_0^{(i-1)}) + (M_7^{(i)} \oplus H_7^{(i-1)}) + (M_9^{(i)} \oplus H_9^{(i-1)}) - (M_{12}^{(i)} \oplus H_{12}^{(i-1)}) + (M_{15}^{(i)} \oplus H_{15}^{(i-1)})\\
W_3^{(i)} &= (M_0^{(i)} \oplus H_0^{(i-1)}) - (M_1^{(i)} \oplus H_1^{(i-1)}) + (M_8^{(i)} \oplus H_8^{(i-1)}) - (M_{10}^{(i)} \oplus H_{10}^{(i-1)}) + (M_{13}^{(i)} \oplus H_{13}^{(i-1)})\\
W_4^{(i)} &= (M_1^{(i)} \oplus H_1^{(i-1)}) + (M_2^{(i)} \oplus H_2^{(i-1)}) + (M_9^{(i)} \oplus H_9^{(i-1)}) - (M_{11}^{(i)} \oplus H_{11}^{(i-1)}) - (M_{14}^{(i)} \oplus H_{14}^{(i-1)})\\
W_5^{(i)} &= (M_3^{(i)} \oplus H_3^{(i-1)}) - (M_2^{(i)} \oplus H_2^{(i-1)}) + (M_{10}^{(i)} \oplus H_{10}^{(i-1)}) - (M_{12}^{(i)} \oplus H_{12}^{(i-1)}) + (M_{15}^{(i)} \oplus H_{15}^{(i-1)})\\
W_6^{(i)} &= (M_4^{(i)} \oplus H_4^{(i-1)}) - (M_0^{(i)} \oplus H_0^{(i-1)}) - (M_3^{(i)} \oplus H_3^{(i-1)}) - (M_{11}^{(i)} \oplus H_{11}^{(i-1)}) + (M_{12}^{(i)} \oplus H_{12}^{(i-1)})\\
W_7^{(i)} &= (M_1^{(i)} \oplus H_1^{(i-1)}) - (M_4^{(i)} \oplus H_4^{(i-1)}) - (M_5^{(i)} \oplus H_5^{(i-1)}) - (M_{12}^{(i)} \oplus H_{12}^{(i-1)}) - (M_{13}^{(i)} \oplus H_{13}^{(i-1)})\\
W_8^{(i)} &= (M_2^{(i)} \oplus H_2^{(i-1)}) - (M_5^{(i)} \oplus H_5^{(i-1)}) - (M_6^{(i)} \oplus H_6^{(i-1)}) + (M_{13}^{(i)} \oplus H_{13}^{(i-1)}) - (M_{15}^{(i)} \oplus H_{15}^{(i-1)})\\
W_9^{(i)} &= (M_0^{(i)} \oplus H_0^{(i-1)}) - (M_3^{(i)} \oplus H_3^{(i-1)}) + (M_6^{(i)} \oplus H_6^{(i-1)}) - (M_7^{(i)} \oplus H_7^{(i-1)}) + (M_{14}^{(i)} \oplus H_{14}^{(i-1)})\\
W_{10}^{(i)} &= (M_8^{(i)} \oplus H_8^{(i-1)}) - (M_1^{(i)} \oplus H_1^{(i-1)}) - (M_4^{(i)} \oplus H_4^{(i-1)}) - (M_7^{(i)} \oplus H_7^{(i-1)}) + (M_{15}^{(i)} \oplus H_{15}^{(i-1)})\\
W_{11}^{(i)} &= (M_8^{(i)} \oplus H_8^{(i-1)}) - (M_0^{(i)} \oplus H_0^{(i-1)}) - (M_2^{(i)} \oplus H_2^{(i-1)}) - (M_5^{(i)} \oplus H_5^{(i-1)}) + (M_9^{(i)} \oplus H_9^{(i-1)})\\
W_{12}^{(i)} &= (M_1^{(i)} \oplus H_1^{(i-1)}) + (M_3^{(i)} \oplus H_3^{(i-1)}) - (M_6^{(i)} \oplus H_6^{(i-1)}) - (M_9^{(i)} \oplus H_9^{(i-1)}) + (M_{10}^{(i)} \oplus H_{10}^{(i-1)})\\
W_{13}^{(i)} &= (M_2^{(i)} \oplus H_2^{(i-1)}) + (M_4^{(i)} \oplus H_4^{(i-1)}) + (M_7^{(i)} \oplus H_7^{(i-1)}) + (M_{10}^{(i)} \oplus H_{10}^{(i-1)}) + (M_{11}^{(i)} \oplus H_{11}^{(i-1)})\\
W_{14}^{(i)} &= (M_3^{(i)} \oplus H_3^{(i-1)}) - (M_5^{(i)} \oplus H_5^{(i-1)}) + (M_8^{(i)} \oplus H_8^{(i-1)}) - (M_{11}^{(i)} \oplus H_{11}^{(i-1)}) - (M_{12}^{(i)} \oplus H_{12}^{(i-1)})\\
W_{15}^{(i)} &= (M_{12}^{(i)} \oplus H_{12}^{(i-1)}) - (M_4^{(i)} \oplus H_4^{(i-1)}) - (M_6^{(i)} \oplus H_6^{(i-1)}) - (M_9^{(i)} \oplus H_9^{(i-1)}) + (M_{13}^{(i)} \oplus H_{13}^{(i-1)})
\end{aligned}$$

2. Further bijective transform of $W_j^{(i)}, j = 0, \ldots, 15$:

$$\begin{aligned}
Q_0^{(i)} &= s_0(W_0^{(i)}); & Q_1^{(i)} &= s_1(W_1^{(i)}); & Q_2^{(i)} &= s_2(W_2^{(i)}); & Q_3^{(i)} &= s_3(W_3^{(i)});\\
Q_4^{(i)} &= s_4(W_4^{(i)}); & Q_5^{(i)} &= s_0(W_5^{(i)}); & Q_6^{(i)} &= s_1(W_6^{(i)}); & Q_7^{(i)} &= s_2(W_7^{(i)});\\
Q_8^{(i)} &= s_3(W_8^{(i)}); & Q_9^{(i)} &= s_4(W_9^{(i)}); & Q_{10}^{(i)} &= s_0(W_{10}^{(i)}); & Q_{11}^{(i)} &= s_1(W_{11}^{(i)});\\
Q_{12}^{(i)} &= s_2(W_{12}^{(i)}); & Q_{13}^{(i)} &= s_3(W_{13}^{(i)}); & Q_{14}^{(i)} &= s_4(W_{14}^{(i)}); & Q_{15}^{(i)} &= s_0(W_{15}^{(i)});
\end{aligned}$$

Table 2.2: Definition of the function f_0 of BLUE MIDNIGHT WISH

Funkce f_0

Dva vstupy jsou na počátku binárně sečteny $A_0(M, H) = M \oplus H$. Tato hodnota je pak bijektivně transformována nejprve čistou aritmetickou kombinací - bijekcí A_1 za použití operací ADD a její výsledek je poté transformován čistou binární lineární kombinací - bijekcí A_2 za použití operací $\oplus$ na výstup Q_a. Hodnota Q_a je tak nelineární bijektivní transformací hodnoty $M \oplus H$.

Máme $f_0(M, H) = A_2(A_1(A_0(M, H)))$, tj. $f_0 = A_2 \bullet A_1 \bullet A_0$ je kompozicí bijektivních transformací, kde se střídají binární a aritmetické operace. A_0 je podle tabulky 2.2. z originálu definována jako $A_0(H, M) = X = M \oplus H$ (po slovech), A_1 je tamtéž definována jako A_1: X→ W, kde

```
W[ 0] = X[5]-X[7]+X[10]+X[13]+X[14]
W[ 1] = X[6]-X[8]+X[11]+X[14]-X[15]
W[ 2] = X[0]+X[7]+X[ 9]-X[12]+X[15]
W[ 3] = X[0]-X[1]+X[ 8]-X[10]+X[13]
W[ 4] = X[1]+X[2]+X[ 9]-X[11]-X[14]
W[ 5] = X[3]-X[2]+X[10]-X[12]+X[15]
W[ 6] = X[4]-X[0]-X[ 3]-X[11]+X[13]
W[ 7] = X[1]-X[4]-X[ 5]-X[12]-X[14]
W[ 8] = X[2]-X[5]-X[ 6]+X[13]-X[15]
W[ 9] = X[0]-X[3]+X[ 6]-X[ 7]+X[14]
W[10] = X[8]-X[1]-X[ 4]-X[ 7]+X[15]
W[11] = X[8]-X[0]-X[ 2]-X[ 5]+X[ 9]
W[12] = X[1]+X[3]-X[ 6]-X[ 9]+X[10]
W[13] = X[2]+X[4]+X[ 7]+X[10]+X[11]
W[14] = X[3]-X[5]+X[ 8]-X[11]-X[12]
W[15] = X[12]-X[4]-X[6]-X[ 9]+X[13]
```

A_2 je definována jako sada lineárních bijekcí, aplikovaných na jednotlivá slova W, viz tabulka 2.2., přičemž tyto lineární bijekce (tzv. xorshifty) jsou definovány (uvádíme je pro jednoduchost jen pro 32bitovou verzi) takto:

$$s_0(x) = SHR^1(x) \oplus SHL^3(x) \oplus ROTL^4(x) \oplus ROTL^{19}(x)$$
$$s_1(x) = SHR^1(x) \oplus SHL^2(x) \oplus ROTL^8(x) \oplus ROTL^{23}(x)$$
$$s_2(x) = SHR^2(x) \oplus SHL^1(x) \oplus ROTL^{12}(x) \oplus ROTL^{25}(x)$$
$$s_3(x) = SHR^2(x) \oplus SHL^2(x) \oplus ROTL^{15}(x) \oplus ROTL^{29}(x)$$
$$s_4(x) = SHR^1(x) \oplus x$$
$$s_5(x) = SHR^2(x) \oplus x$$
$$r_1(x) = ROTL^3(x)$$
$$r_2(x) = ROTL^7(x)$$
$$r_3(x) = ROTL^{13}(x)$$
$$r_4(x) = ROTL^{16}(x)$$
$$r_5(x) = ROTL^{19}(x)$$
$$r_6(x) = ROTL^{23}(x)$$
$$r_7(x) = ROTL^{27}(x)$$

("xorshifty" typu r jsou použity později ve funkci f_1).

Funkce f_1

U funkce f_1 hovoříme o slabé šifře, protože oba typy míchání nejsou v žádném případě dokonalé. Zde je také jediné místo, kde se používá princip měnitelného (laditelného, tunable) parametru pro možné zesílení nebo urychlení hašovací funkce. Označíme-li E onu blokovou šifru, můžeme f_1 vyjádřit jako $f_1(M, Q_a) = E_{B(M)}(Q_a)$. Klíčem je M. Klíč je expandován bijektivní transformací B na 16 rundovních klíčů B(M). V každé rundě je použit jeden rundovní klíč. Rundy jsou dvojího typu, první typ je složitější a je použit na počátku. Počet těchto rund se označuje expandrounds1 a defaultní hodnota je expandrounds1 = 2 rundy. Poté

následuje expandrounds2 rund druhého typu, které jsou jednodušší. Je jich expandrounds2 = 16 - expandrounds1, tj. defaultně 14. Máme

$f_1 : \{0,1\}^{2m} \rightarrow \{0,1\}^m$

Input: Message block $M^{(i)} = (M_0^{(i)}, M_1^{(i)}, \ldots, M_{15}^{(i)})$, and the first part of the quadruple pipe $Q_a^{(i)} = (Q_0^{(i)}, Q_1^{(i)}, \ldots, Q_{15}^{(i)})$.
Output: Second part of the quadruple pipe $Q_b^{(i)} = (Q_{16}^{(i)}, Q_{17}^{(i)}, \ldots, Q_{31}^{(i)})$.

1. Double pipe expansion according to the tunable parameters $ExpandRounds_1$ and $ExpandRounds_2$.

   1.1 For $ii = 0$ to $ExpandRounds_1 - 1$
   $Q_{ii+16}^{(i)} = expand_1(ii + 16)$

   1.2 For $ii = ExpandRounds_1$ to $ExpandRounds_1 + ExpandRounds_2 - 1$
   $Q_{ii+16}^{(i)} = expand_2(ii + 16)$

Table 2.3: Definition of the function f_1 of BLUE MIDNIGHT WISH

$$\begin{aligned} expand_1(j) = \; & s_1(Q_{j-16}^{(i)}) + s_2(Q_{j-15}^{(i)}) + s_3(Q_{j-14}^{(i)}) + s_0(Q_{j-13}^{(i)}) \\ & + s_1(Q_{j-12}^{(i)}) + s_2(Q_{j-11}^{(i)}) + s_3(Q_{j-10}^{(i)}) + s_0(Q_{j-9}^{(i)}) \\ & + s_1(Q_{j-8}^{(i)}) + s_2(Q_{j-7}^{(i)}) + s_3(Q_{j-6}^{(i)}) + s_0(Q_{j-5}^{(i)}) \\ & + s_1(Q_{j-4}^{(i)}) + s_2(Q_{j-3}^{(i)}) + s_3(Q_{j-2}^{(i)}) + s_0(Q_{j-1}^{(i)}) \\ & + M_{(j-16) \bmod 16}^{(i)} + M_{(j-13) \bmod 16}^{(i)} - M_{(j-6) \bmod 16}^{(i)} + K_j \end{aligned}$$

$$\begin{aligned} expand_2(j) = \; & Q_{j-16}^{(i)} + r_1(Q_{j-15}^{(i)}) + Q_{j-14}^{(i)} + r_2(Q_{j-13}^{(i)}) \\ & + Q_{j-12}^{(i)} + r_3(Q_{j-11}^{(i)}) + Q_{j-10}^{(i)} + r_4(Q_{j-9}^{(i)}) \\ & + Q_{j-8}^{(i)} + r_5(Q_{j-7}^{(i)}) + Q_{j-6}^{(i)} + r_6(Q_{j-5}^{(i)}) \\ & + Q_{j-4}^{(i)} + r_7(Q_{j-3}^{(i)}) + s_5(Q_{j-2}^{(i)}) + s_4(Q_{j-1}^{(i)}) \\ & + M_{(j-16) \bmod 16}^{(i)} + M_{(j-13) \bmod 16}^{(i)} - M_{(j-6) \bmod 16}^{(i)} + K_j \end{aligned}$$

Zvláštnosti funkce f_1

Funkci f_1 lze rozložit na tzv. horní a dolní trojúhelníkové bijekce T^U (upper triangle) a T^L (lower triangle), mezi nimiž je vrstva přičtení klíče K^A (key addition): $f_1 = T^L \bullet K^A \bullet T^U$. Přitom T^U a T^L jsou samy o sobě kombinací binárních a aritmetických kombinací, takže vytváří nelineární blok. Tento rozklad není přímo vidět v následujících definicích funkcí expand1 a expand2, ale lze to tak rozložit, viz dodatek.

Zajímavé je, že f_1 je bijekce jak při pevném M, tak při pevném Q_a. Pro blokové šifry je obvyklé, že pro pevný klíč je šifrový text bijektivním zobrazením textu otevřeného. Není ale obvyklé, aby při pevném otevřeném textu byla bloková šifra také bijektivnín zobrazením klíče na šifrový text. Zde to má velký význam v tom, že oba dva vstupy jsou chápány jako rovnocenné, a to i jako klíč, i jako otevřený text. V závěrečném zpracování se také oba dva vstupy kombinují ve funkci f_2, dohromady s jejich společně dosaženým výstupem - šifrovým textem. Další vlastností je, že vstup Q_a je bijektivním obrazem $M \oplus H$, zatímco vstup klíče je bijektivním obrazem M. Klíčová myšlenka je, že při analýze nebo útoku na hašovací funkci se nutně musí jedna z těchto hodnot (M, $M \oplus H$) změnit, jinak celá kompresní funkce v daném kroku proběhne stejně. Při útoku se musí uvažovat různé zprávy, jinak bychom nic nevyzkoumali. To vede k tomu, že se v některém kroku kompresní funkce liší buď vstup M nebo vstup průběžné haše H nebo oba dva. Přitom $M \oplus H$ může být i stejné i různé. V

každém případě se ale dvojice vstupů (M ⊕ H, M) musí v tomto okamžiku lišit. Důsledkem je, že jeden nebo oba vstupy funkce f_1 se musí změnit (a jejich změny jsou eventuelně ještě rozprostřeny bijekcemi f_0 a B). Funkce f_1 pak tyto změny propaguje na svůj výstup. Protože výstup Q_b je funkcí dvou vstupů M a H, nemůže být jejich bijektivním obrazem a nutně existuje velká množina vnitřních kolizí při zobrazení (M, H) na Q_b. Proto se vstupy (M, M ⊕ H) nezapomínají a vstupují do zpracování f_2, prostřednictvím svých bijekcí. Proto je také garantováno, že tyto vstupy se ve funkci f_2 mění, i kdyby Q_b zůstalo konstantní.

Funkce f_2

Funkce f_2 používá lineární (binární) bijektivní matici L, která je rozdělena na dvě části L_a a L_b, L = L_a ⊕ L_b, přičemž i L_a a L_b mají vysokou hodnost (jsou téměř bijektivní). Účelem f_2 je kombinovat vstupy M, a Q_a s Q_b, což se děje sloučením Q_b s M a s (M ⊕ H), přesněji téměř bijektivních obrazů Q_b s bijektivními obrazy M a M ⊕ H. Zde je opět velmi důležité, že alespoň jeden z výrazů Q_b, M a M ⊕ H se musí v daném kroku (viz výše), lišit. Jestliže se totiž neodlišuje ani M ani Q_b, musí se odlišovat H, tedy odlišuje se výraz M ⊕ H, kde M je sice konstantní, ale H se mění.

Folding $f_2 : \{0,1\}^{3m} \rightarrow \{0,1\}^m$

Input: Message block $M^{(i)} = (M_0^{(i)}, M_1^{(i)}, \ldots, M_{15}^{(i)})$,
quadruple pipe $Q^{(i)} = (Q_0^{(i)}, Q_1^{(i)}, \ldots, Q_{15}^{(i)}, Q_{16}^{(i)}, \ldots, Q_{31}^{(i)})$.
Output: New double pipe $H^{(i)} = (H_0^{(i)}, H_1^{(i)}, \ldots, H_{15}^{(i)})$.

1. Compute the cumulative temporary variables XL and XH.

$$XL = Q_{16}^{(i)} \oplus Q_{17}^{(i)} \oplus \ldots \oplus Q_{23}^{(i)}$$
$$XH = XL \oplus Q_{24}^{(i)} \oplus Q_{25}^{(i)} \oplus \ldots \oplus Q_{31}^{(i)}$$

2. Compute the new double pipe $H^{(i)}$:

f5 La Lb

$$\begin{aligned}
H_0^{(i)} &= \left(SHL^5(XH) \oplus SHR^7(Q_{16}^{(i)}) \oplus M_0^{(i)}\right) + \left(XL \oplus Q_{24}^{(i)} \oplus Q_0^{(i)}\right)\\
H_1^{(i)} &= \left(SHR^7(XH) \oplus SHL^8(Q_{17}^{(i)}) \oplus M_1^{(i)}\right) + \left(XL \oplus Q_{25}^{(i)} \oplus Q_1^{(i)}\right)\\
H_2^{(i)} &= \left(SHR^5(XH) \oplus SHL^5(Q_{18}^{(i)}) \oplus M_2^{(i)}\right) + \left(XL \oplus Q_{26}^{(i)} \oplus Q_2^{(i)}\right)\\
H_3^{(i)} &= \left(SHR^1(XH) \oplus SHL^5(Q_{19}^{(i)}) \oplus M_3^{(i)}\right) + \left(XL \oplus Q_{27}^{(i)} \oplus Q_3^{(i)}\right)\\
H_4^{(i)} &= \left(SHR^3(XH) \oplus Q_{20}^{(i)} \oplus M_4^{(i)}\right) + \left(XL \oplus Q_{28}^{(i)} \oplus Q_4^{(i)}\right)\\
H_5^{(i)} &= \left(SHL^6(XH) \oplus SHR^6(Q_{21}^{(i)}) \oplus M_5^{(i)}\right) + \left(XL \oplus Q_{29}^{(i)} \oplus Q_5^{(i)}\right)\\
H_6^{(i)} &= \left(SHR^4(XH) \oplus SHL^6(Q_{22}^{(i)}) \oplus M_6^{(i)}\right) + \left(XL \oplus Q_{30}^{(i)} \oplus Q_6^{(i)}\right)\\
H_7^{(i)} &= \left(SHR^{11}(XH) \oplus SHL^2(Q_{23}^{(i)}) \oplus M_7^{(i)}\right) + \left(XL \oplus Q_{31}^{(i)} \oplus Q_7^{(i)}\right)\\
H_8^{(i)} &= ROTL^9(H_4^{(i)}) + \left(XH \oplus Q_{24}^{(i)} \oplus M_8^{(i)}\right) + \left(SHL^8(XL) \oplus Q_{23}^{(i)} \oplus Q_8^{(i)}\right)\\
H_9^{(i)} &= ROTL^{10}(H_5^{(i)}) + \left(XH \oplus Q_{25}^{(i)} \oplus M_9^{(i)}\right) + \left(SHR^6(XL) \oplus Q_{16}^{(i)} \oplus Q_9^{(i)}\right)\\
H_{10}^{(i)} &= ROTL^{11}(H_6^{(i)}) + \left(XH \oplus Q_{26}^{(i)} \oplus M_{10}^{(i)}\right) + \left(SHL^6(XL) \oplus Q_{17}^{(i)} \oplus Q_{10}^{(i)}\right)\\
H_{11}^{(i)} &= ROTL^{12}(H_7^{(i)}) + \left(XH \oplus Q_{27}^{(i)} \oplus M_{11}^{(i)}\right) + \left(SHL^4(XL) \oplus Q_{18}^{(i)} \oplus Q_{11}^{(i)}\right)\\
H_{12}^{(i)} &= ROTL^{13}(H_0^{(i)}) + \left(XH \oplus Q_{28}^{(i)} \oplus M_{12}^{(i)}\right) + \left(SHR^3(XL) \oplus Q_{19}^{(i)} \oplus Q_{12}^{(i)}\right)\\
H_{13}^{(i)} &= ROTL^{14}(H_1^{(i)}) + \left(XH \oplus Q_{29}^{(i)} \oplus M_{13}^{(i)}\right) + \left(SHR^4(XL) \oplus Q_{20}^{(i)} \oplus Q_{13}^{(i)}\right)\\
H_{14}^{(i)} &= ROTL^{15}(H_2^{(i)}) + \left(XH \oplus Q_{30}^{(i)} \oplus M_{14}^{(i)}\right) + \left(SHR^7(XL) \oplus Q_{21}^{(i)} \oplus Q_{14}^{(i)}\right)\\
H_{15}^{(i)} &= ROTL^{16}(H_3^{(i)}) + \left(XH \oplus Q_{31}^{(i)} \oplus M_{15}^{(i)}\right) + \left(SHR^2(XL) \oplus Q_{22}^{(i)} \oplus Q_{15}^{(i)}\right)
\end{aligned}$$

Table 2.4: Definition of the folding function f_2 of BLUE MIDNIGHT WISH

$$L_a(Qb) = \begin{pmatrix} SHL^5(XH) \oplus & SHR^5(Q_{16}^{(i)}) \\ SHR^7(XH) \oplus & SHL^8(Q_{17}^{(i)}) \\ SHR^5(XH) \oplus & SHL^5(Q_{18}^{(i)}) \\ SHR^1(XH) \oplus & SHL^5(Q_{19}^{(i)}) \\ SHR^3(XH) \oplus & Q_{20}^{(i)} \\ SHL^6(XH) \oplus & SHR^6(Q_{21}^{(i)}) \\ SHR^4(XH) \oplus & SHL^6(Q_{22}^{(i)}) \\ SHR^{11}(XH) \oplus & SHL^2(Q_{23}^{(i)}) \\ XH \oplus & Q_{24}^{(i)} \\ XH \oplus & Q_{25}^{(i)} \\ XH \oplus & Q_{26}^{(i)} \\ XH \oplus & Q_{27}^{(i)} \\ XH \oplus & Q_{28}^{(i)} \\ XH \oplus & Q_{29}^{(i)} \\ XH \oplus & Q_{30}^{(i)} \\ XH \oplus & Q_{31}^{(i)} \end{pmatrix} \quad L_b(Qb) = \begin{pmatrix} XL \oplus & Q_{24}^{(i)} \\ XL \oplus & Q_{25}^{(i)} \\ XL \oplus & Q_{26}^{(i)} \\ XL \oplus & Q_{27}^{(i)} \\ XL \oplus & Q_{28}^{(i)} \\ XL \oplus & Q_{29}^{(i)} \\ XL \oplus & Q_{30}^{(i)} \\ XL \oplus & Q_{31}^{(i)} \\ SHL^8(XL) \oplus & Q_{23}^{(i)} \\ SHR^6(XL) \oplus & Q_{16}^{(i)} \\ SHL^6(XL) \oplus & Q_{17}^{(i)} \\ SHL^4(XL) \oplus & Q_{18}^{(i)} \\ SHR^3(XL) \oplus & Q_{19}^{(i)} \\ SHR^4(XL) \oplus & Q_{20}^{(i)} \\ SHR^7(XL) \oplus & Q_{21}^{(i)} \\ SHR^2(XL) \oplus & Q_{22}^{(i)} \end{pmatrix}$$

Závěr

V tomto článku jsme uvedli základní popis a některé vlastnosti hašovací funkce Blue Midnight Wish, kandidáta, který je z hlediska rychlosti na špičce ze všech 51 kandidátů na funkci SHA-3. Viděli jsme také velký význam bijekcí při jeho návrhu. Z toho důvodu můžeme nově BMW chápat jako zkratku principu "Bijections Mounted Widely". Čtenářům se omlouváme, nemohli jsme uvést všechny vlastnosti a principy, spíše jen ukázat, co se skrývá za nic neříkajícími rovnicemi nebo pár instrukcemi zdrojového kódu. Podrobnější popis, analýzu a průběžné novinky je možné sledovat na internetu (např. [2]).

Literatura

[1] (nezávislá) oficiální domácí stránka NIST k projektu SHA-3: http://csrc.nist.gov/groups/ST/hash/index.html
[2] stránka autora s novinkami k projektu SHA-3 a algoritmům BMW a EDON-R: http://cryptography.hyperlink.cz/BMW/BMW_CZ.html (naleznete tam rozcestník a všechny zde uvedené linky)
[3] Stránka kandidátů, kteří postoupili do prvního kola (NIST): http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/submissions_rnd1.html
[4] Stránka SHA-3 na wiki: http://en.wikipedia.org/wiki/SHA-3
[5] Stránka SHA-3 projektu ECRYPT: http://ehash.iaik.tugraz.at/index.php/The_SHA-3_Zoo
[6] Seznam všech autorů všech kandidátů: http://ehash.iaik.tugraz.at/wiki/SHA-3_submitters
[7] Stránka NIST, věnovaná první konferenci kandidátů SHA-3: http://csrc.nist.gov/groups/ST/hash/sha-3/Round1/Feb2009/program.html
[8] Stránka o SW výkonnosti algoritmů eBash: http://bench.cr.yp.to/results-hash.html
[9] Stránka o HW výkonnosti algoritmů (ECRYPT): http://ehash.iaik.tugraz.at/wiki/SHA-3_Hardware_Implementations
[10] Srovnávací stránka (Fleischmann-Forler-Gorski): http://www.uni-weimar.de/cms/fileadmin/medien/medsicherheit/Research/SHA3/Classification_of_the_SHA-3_Candidates.pdf
[11] Srovnávací stránka (Niels Ferguson): http://www.skein-hash.info/sha3-engineering
[12-14] Domovská stránka DN, HDN a SNMAC, v češtině, http://cryptography.hyperlink.cz/SNMAC/SNMAC_CZ.html ,

[12] V. Klima: O návrhu speciálních blokových šifer a speciálních hašovacích funkcí, MKB 2007, Praha, Hotel Olympik, December, 6. – 7., 2007, http://cryptography.hyperlink.cz/2007/Klima_MKB_2007_sbornik.pdf
[13] V. Klima: Special block cipher family DN and new generation SNMAC-type hash function family HDN, IACR ePrint archive Report 2007/050, February, 2007, IACR ePrint archive Report 2007/050, February, 2007, http://eprint.iacr.org/2007/050.pdf, v češtině na http://cryptography.hyperlink.cz/SNMAC/DN_HDN_CZ.pdf
[14] V. Klima: A New Concept of Hash Functions SNMAC Using a Special Block Cipher and NMAC/HMAC Constructions, IACR ePrint archive Report 2006/376, http://eprint.iacr.org/2006/376.pdf, October, 2006, v češtině na http://cryptography.hyperlink.cz/SNMAC/SNMAC_CZ.pdf
[15] Vlastimil Klíma: O kolizích hašovacích funkcí Turbo SHA-2, IACR ePrint archive Report 2008/003, January, 2008, http://eprint.iacr.org/2008/003.pdf v češtině na http://cryptography.hyperlink.cz/2008/Klima_TurboSHA_CZ.pdf
[16] Stefan Lucks. Design principles for iterated hash functions. Cryptology ePrint Archive, Report 2004/253, 2004. http://eprint.iacr.org/
[17] A. Joux: Multicollisions in iterated hash functions. Application to cascaded constructions. Proceedings of Crypto 2004, LNCS 3152, pages 306-316.
[18] John Kelsey, Bruce Schneier: Second Preimages on n-bit Hash Functions for Much Less than 2^n Work, http://eprint.iacr.org/2004/304/, November 15, 2004
[19] CRYPTOGRAPHIC HASH WORKSHOP, NIST, October 31-November 1, 2005 http://csrc.nist.gov/groups/ST/hash/first_workshop.html
[20] SECOND CRYPTOGRAPHIC HASH WORKSHOP, USA, August 24-25, 2006, http://csrc.nist.gov/groups/ST/hash/second_workshop.html
[21] R. Govaerts B. Preneel and J. Vandewalle. Hash functions based on block ciphers: A synthetic approach. In Proceedings of CRYPTO 1993, volume 773 of LNCS, pages 368–378, 1994.

Dodatek: Rozklad funkce f_1 na T^U, K^A a T^L.

Původní definici funkce f_1 lze vyjádřit následujícími rovnicemi. V nich si už připravíme půdu pro rozklad, přičemž žlutě vyznačíme výrazy patřící do transformace T^U, zelené tvoří transformaci T^L a konstanty společně se slovy M tvoří klíč K^A.

Připomeňme, že první dvě rovnice jsou jiné (složitější funkce expand1) než zbylé (expand2).

```
Q[16] =
s1(Q[ 0])+s2(Q[ 1])+s3(Q[ 2])+s0(Q[ 3])+s1(Q[ 4])+s2(Q[ 5])+s3(Q[ 6])+
s0(Q[ 7])+s1(Q[ 8])+s2(Q[ 9])+s3(Q[10])+s0(Q[11])+s1(Q[12])+s2(Q[13])+
s3(Q[14])+s0(Q[15])+K[0]

Q[17]=
s1(Q[ 1])+s2(Q[ 2])+s3(Q[ 3])+s0(Q[ 4])+s1(Q[ 5])+s2(Q[ 6])+s3(Q[ 7])+
s0(Q[ 8])+s1(Q[ 9])+s2(Q[10])+s3(Q[11])+s0(Q[12])+s1(Q[13])+s2(Q[14])+
s3(Q[15])+ K[1] + s0(Q[16])

Q[18]=
Q[ 2]+r1(Q[ 3])+Q[ 4]+r2(Q[ 5])+Q[ 6]+r3(Q[ 7])+Q[ 8]+r4(Q[ 9])+
Q[10]+r5(Q[11])+Q[12]+r6(Q[13])+Q[14]+r7(Q[15])+ K[2] +s5(Q[16])+s4(Q[15])

Q[19]=
Q[ 3]+r1(Q[ 4])+Q[ 5]+r2(Q[ 6])+Q[ 7]+r3(Q[ 8])+Q[ 9]+r4(Q[10])+
Q[11]+r5(Q[12])+Q[13]+r6(Q[14])+Q[15] + K[3] +r7(Q[16])+s5(Q[17])+s4(Q[18])

Q[20]=
```

```
Q[ 4]+r1(Q[ 5])+Q[ 6]+r2(Q[ 7])+Q[ 8]+r3(Q[ 9])+Q[10]+r4(Q[11])+
Q[12]+r5(Q[13])+Q[14]+r6(Q[15])+ K[4] +Q[14]+r7(Q[13])+s5(Q[12])+s4(Q[11])

Q[21]=
Q[ 5]+r1(Q[ 6])+Q[ 7]+r2(Q[ 8])+Q[ 9]+r3(Q[10])+Q[11]+r4(Q[12)+
Q[13]+r5(Q[14])+Q[15] + K[5] +r6(Q[16])+Q[17]+r7(Q[18])+s5(Q[19])+s4(Q[20])

Q[22]=
Q[ 6]+r1(Q[ 7])+Q[ 8]+r2(Q[ 9])+Q[10]+r3(Q[11])+Q[12]+r4(Q[13])+
Q[14]+r5(Q[15]) + K[6] +Q[16]+r6(Q[17])+Q[18]+r7(Q[19])+s5(Q[20])+s4(Q[21])

Q[23]=
Q[ 7]+r1(Q[ 8])+Q[ 9]+r2(Q[10])+Q[11]+r3(Q[12])+Q[13]+r4(Q[14])+
Q[15] + K[7] +r5(Q[16])+Q[17]+r6(Q[18])+Q[19]+r7(Q[20])+s5(Q[21])+s4(Q[22])

Q[24]=
Q[ 8]+r1(Q[ 9])+Q[10]+r2(Q[11])+Q[12]+r3(Q[13])+Q[14]+r4(Q[15]) + K[8]+
Q[16]+r5(Q[17])+Q[18]+r6(Q[19])+Q[20]+r7(Q[21])+s5(Q[22])+s4(Q[23])

Q[25]=
Q[ 9]+r1(Q[10])+Q[11]+r2(Q[12])+Q[13]+r3(Q[14])+Q[15] + K[9] +r4(Q[16])+
Q[17]+r5(Q[18])+Q[19]+r6(Q[20])+Q[21]+r7(Q[22])+s5(Q[23])+s4(Q[24])

Q[26]=
Q[10]+r1(Q[11])+Q[12]+r2(Q[13])+Q[14]+r3(Q[15]) + K[10]+Q[16]+r4(Q[17])+
Q[18]+r5(Q[19])+Q[20]+r6(Q[21])+Q[22]+r7(Q[23])+s5(Q[24])+s4(Q[25])

Q[27]=
Q[11]+r1(Q[12])+Q[13]+r2(Q[14])+Q[15] + K[11]+r3(Q[16])+Q[17]+r4(Q[18])+
Q[19]+r5(Q[20])+Q[21]+r6(Q[22])+Q[23]+r7(Q[24])+s5(Q[25])+s4(Q[26])

Q[28]=
Q[12]+r1(Q[13])+Q[14]+r2(Q[15]) + K[12]+Q[16]+r3(Q[17])+Q[18]+r4(Q[19])+
Q[20]+r5(Q[21])+Q[22]+r6(Q[23])+Q[24]+r7(Q[25])+s5(Q[26])+s4(Q[27])

Q[29]=
Q[13]+r1(Q[14])+Q[15] + K[13]+r2(Q[16])+Q[17]+r3(Q[18])+Q[19]+r4(Q[20])+
Q[21]+r5(Q[22])+Q[23]+r6(Q[24])+Q[25]+r7(Q[26])+s5(Q[27])+s4(Q[28])

Q[30]=
Q[14]+r1(Q[15]) + K[14]+Q[16]+r2(Q[17])+Q[18]+r3(Q[19])+Q[20]+r4(Q[21])+
Q[22]+r5(Q[23])+Q[24]+r6(Q[25])+Q[26]+r7(Q[27])+s5(Q[28])+s4(Q[29])

Q[31]=
Q[15] + K[15]+r1(Q[16])+Q[17]+r2(Q[18])+Q[19]+r3(Q[20])+Q[21]+r4(Q[22])+
Q[23]+r5(Q[24])+Q[25]+r6(Q[26])+Q[27]+r7(Q[28])+s5(Q[29])+s4(Q[30]),
```

kde "klíč" K je definován jako
K = (K0], ..., K[15]) = B(M) + C, kde C je tvořena násobením const5 (pro m = 512 je const5 = 0x05555555 a pro m = 512 je const5 = 0x0555555555555555) a
K[i] = (i+16)*const5 + M[i mod 16]+M[(i + 3)mod 16]-M[(i + 10)mod 16] pro i = 0, ..., 15.

Nyní definujeme **horní trojúhelníkovou transformaci** $P = T^U(Q_a)$:
$P = (P[0], ..., P[15]) = T^U(Q_a) = T^U (Q[0], ..., Q[15])$, kde

```
P[ 0] =
s1(Q[ 0])+s2(Q[ 1])+s3(Q[ 2])+s0(Q[ 3])+s1(Q[ 4])+s2(Q[ 5])+s3(Q[ 6])+
s0(Q[ 7])+s1(Q[ 8])+s2(Q[ 9])+s3(Q[10])+s0(Q[11])+s1(Q[12])+s2(Q[13])+
s3(Q[14])+s0(Q[15])
```

```
P[ 1]=
s1(Q[ 1])+s2(Q[ 2])+s3(Q[ 3])+s0(Q[ 4])+s1(Q[ 5])+s2(Q[ 6])+s3(Q[ 7])+
s0(Q[ 8])+s1(Q[ 9])+s2(Q[10])+s3(Q[11])+s0(Q[12])+s1(Q[13])+s2(Q[14])+
s3(Q[15])

P[02]=
Q[ 2]+r1(Q[ 3])+Q[ 4]+r2(Q[ 5])+Q[ 6]+r3(Q[ 7])+Q[ 8]+r4(Q[ 9])+
Q[10]+r5(Q[11])+Q[12]+r6(Q[13])+Q[14]+r7(Q[15])

P[03]=
Q[ 3]+r1(Q[ 4])+Q[ 5]+r2(Q[ 6])+Q[ 7]+r3(Q[ 8])+Q[ 9]+r4(Q[10])+
Q[11]+r5(Q[12])+Q[13]+r6(Q[14])+Q[15]

P[ 4]=
Q[ 4]+r1(Q[ 5])+Q[ 6]+r2(Q[ 7])+Q[ 8]+r3(Q[ 9])+Q[10]+r4(Q[11])+
Q[12]+r5(Q[13])+Q[14]+r6(Q[15])

P[ 5]=
Q[ 5]+r1(Q[ 6])+Q[ 7]+r2(Q[ 8])+Q[ 9]+r3(Q[10])+Q[11]+r4(Q[12)+
Q[13]+r5(Q[14])+Q[15]

P[ 6]=
Q[ 6]+r1(Q[ 7])+Q[ 8]+r2(Q[ 9])+Q[10]+r3(Q[11])+Q[12]+r4(Q[13])+
Q[14]+r5(Q[15])

P[ 7]=
Q[ 7]+r1(Q[ 8])+Q[ 9]+r2(Q[10])+Q[11]+r3(Q[12])+Q[13]+r4(Q[14])+
Q[15]

P[ 8]=
Q[ 8]+r1(Q[ 9])+Q[10]+r2(Q[11])+Q[12]+r3(Q[13])+Q[14]+r4(Q[15])

P[ 9]=
Q[ 9]+r1(Q[10])+Q[11]+r2(Q[12])+Q[13]+r3(Q[14])+Q[15]

P[10]=
Q[10]+r1(Q[11])+Q[12]+r2(Q[13])+Q[14]+r3(Q[15])

P[11]=
Q[11]+r1(Q[12])+Q[13]+r2(Q[14])+Q[15]

P[12]=
Q[12]+r1(Q[13])+Q[14]+r2(Q[15])

P[13]=
Q[13]+r1(Q[14])+Q[15]

P[14]=
Q[14]+r1(Q[15])

P[15]=
Q[15]
```

Definujeme **přičtení klíče jako transformaci K^A**:
$R = K^A(P, K) = P + K = (R[0], ..., R[15])$, kde $R[i] = K[i] + P[i]$, $i = 0,...,15$

Definujeme **dolní trojúhelníkovou transformaci** $Q_b = (Q[16], ..., Q[31]) = Q_b = T^L(R)$.
Poznamenejme, že tento výpočet je na rozdíl od předchozího "zpětnovazební", tj. nově počítané Q se používají ihned dále v následujícím vztahu.

```
Q[16]=
R[0]

Q[17]=
R[1] + s0(Q[16])

Q[18]=
R[2] +s5(Q[16])+s4(Q[15])

Q[19]=
R[3] +r7(Q[16])+s5(Q[17])+s4(Q[18])

Q[20]=
R[4] +Q[14]+r7(Q[13])+s5(Q[12])+s4(Q[11])

Q[21]=
R[5] +r6(Q[16])+Q[17]+r7(Q[18])+s5(Q[19])+s4(Q[20])

Q[22]=
R[6] +Q[16]+r6(Q[17])+Q[18]+r7(Q[19])+s5(Q[20])+s4(Q[21])

Q[23]=
R[7] +r5(Q[16])+Q[17]+r6(Q[18])+Q[19]+r7(Q[20])+s5(Q[21])+s4(Q[22])

Q[24]=
R[8]+
Q[16]+r5(Q[17])+Q[18]+r6(Q[19])+Q[20]+r7(Q[21])+s5(Q[22])+s4(Q[23])

Q[25]=
R[9]+r4(Q[16])+Q[17]+r5(Q[18])+Q[19]+r6(Q[20])+Q[21]+r7(Q[22])+s5(Q[23])+s4
(Q[24])

Q[26]=
R[10]+Q[16]+r4(Q[17])+Q[18]+r5(Q[19])+Q[20]+r6(Q[21])+Q[22]+r7(Q[23])+s5(Q[
24])+s4(Q[25])

Q[27]=
R[11]+r3(Q[16])+Q[17]+r4(Q[18])+Q[19]+r5(Q[20])+Q[21]+r6(Q[22])+Q[23]+r7(Q[
24])+s5(Q[25])+s4(Q[26])

Q[28]=
R[12]+Q[16]+r3(Q[17])+Q[18]+r4(Q[19])+Q[20]+r5(Q[21])+Q[22]+r6(Q[23])+Q[24]
+r7(Q[25])+s5(Q[26])+s4(Q[27])

Q[29]=
R[13]+r2(Q[16])+Q[17]+r3(Q[18])+Q[19]+r4(Q[20])+Q[21]+r5(Q[22])+Q[23]+r6(Q[
24])+Q[25]+r7(Q[26])+s5(Q[27])+s4(Q[28])

Q[30]=
R[14]+Q[16]+r2(Q[17])+Q[18]+r3(Q[19])+Q[20]+r4(Q[21])+Q[22]+r5(Q[23])+Q[24]
+r6(Q[25])+Q[26]+r7(Q[27])+s5(Q[28])+s4(Q[29])

Q[31]=
R[15]+r1(Q[16])+Q[17]+r2(Q[18])+Q[19]+r3(Q[20])+Q[21]+r4(Q[22])+Q[23]+r5(Q[
24])+Q[25]+r6(Q[26])+Q[27]+r7(Q[28])+s5(Q[29])+s4(Q[30])
```

Vidíme, že
$f_1 = T^L \bullet K^A \bullet T^U$, kde T^U, K^A a T^L jsou bijektivní transformace.Tím máme popsán rozklad funkce f_1: **$Q_b = f_1(M, Q_a) = T^L(T^U(Q_a) + B(M) + C)$.**

C. SmartCard Forum 2009

Zveme Vás na druhý ročník velmi úspěšné odborné konference věnované tématu využití čipových technologií v oblasti bezpečnosti, která proběhne dne: 21.05.2009 v Praze, v konferenčním sále společnosti OKsystem s.r.o.

Konference je určena pro širokou odbornou veřejnost a manažery v oblastech informačních a komunikačních technologií, se zaměřením na využití čipových karet.

Konferenční poplatek je 0,- Kč, avšak je nutná registrace předem:

http://www.oksystem.cz/akce/registrace_smartcardforum

Na konferenci zazní tyto prezentace:

František Maleč (Státní tiskárna cenin) - Identifikační doklady v ČR

Ivo Rosol (OKsystem s.r.o.) - Elektronické občanské průkazy v EU

Tomáš Rosa (Raiffeisenbank, a.s.) - Penetrační testy RFID čipů aneb když pravda je horší než lež

Tomáš Trpišovský (IMA s.r.o.) - Moderní technologie pro identifikace replikačních robotů

Václav Lín (OKsystem s.r.o). - SIM karty a SIM aplikace

Vlastimil Klíma (nezávislý kryptolog) - Současná kryptologie a hashovací funkce v praxi

Moderátorem akce je: pan **Petr Koubský**

D. O čem jsme psali v březnu 2000 – 2008

Crypto-World 3/2000

Crypto-World 3/2001

Crypto-World 3/2002

Crypto-World 3/2003

Příloha : crypto_p3.pdf

E. Závěrečné informace

1. Sešit

Crypto-World je oficiální informační sešit "Kryptologické sekce Jednoty českých matematiků a fyziků" (GCUCMP). Obsahuje články podepsané autory. Případné chyby a nepřesnosti jsou dílem P.Vondrušky a autorů jednotlivých podepsaných článků, GCUCMP za ně nemá odbornou ani jinou zodpovědnost.
Adresa URL, na níž můžete najít tento sešit (zpravidla 3 týdny po jeho rozeslání) a předchozí sešity GCUCMP, denně aktualizované novinky z kryptologie a informační bezpečnosti, normy, standardy, stránky některých členů a další související materiály:
http://crypto-world.info

2. Registrace / zrušení registrace

Zájemci o e-zin se mohou zaregistrovat pomocí e-mailu na adrese pavel.vondruska@crypto-world.info (předmět: Crypto-World) nebo použít k odeslání žádosti o registraci elektronický formulář na http://crypto-world.info. Při registraci vyžadujeme pouze jméno a příjmení, titul, pracoviště (není podmínkou) a e-mail adresu určenou k zasílání kódů ke stažení sešitu.
Ke zrušení registrace stačí zaslat krátkou zprávu na e-mail pavel.vondruska@crypto-world.info (předmět: ruším odběr Crypto-Worldu!) nebo opět použít formulář na http://crypto-world.info. Ve zprávě prosím uveďte jméno a příjmení a e-mail adresu, na kterou byly kódy zasílány.

3. Redakce

E-zin Crypto-World

Redakční práce:	Pavel Vondruška
Stálí přispěvatelé:	Pavel Vondruška
	Jaroslav Pinkava
Jazyková úprava:	Jakub Vrána
Přehled autorů:	http://crypto-world.info/obsah/autori.pdf
NEWS	Vlastimil Klíma
(výběr příspěvků,	Jaroslav Pinkava
komentáře a	Tomáš Rosa
vkládání na web)	Pavel Vondruška
Webmaster	Pavel Vondruška, jr.

4. Spojení (abecedně)

redakce e-zinu	ezin@crypto-world.info ,	http://crypto-world.info
Vlastimil Klíma	v.klima@volny.cz ,	http://cryptography.hyperlink.cz/
Jaroslav Pinkava	Jaroslav.Pinkava@zoner.cz ,	http://crypto-world.info/pinkava/
Tomáš Rosa	t_rosa@volny.cz ,	http://crypto.hyperlink.cz/
Pavel Vondruška	pavel.vondruska@crypto-world.info ,	http://crypto-world.info/vondruska/index.php
Pavel Vondruška,jr.	pavel@crypto-world.info ,	http://webdesign.crypto-world.info
Jakub Vrána	jakub@vrana.cz ,	http://www.vrana.cz/