Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE

Martin Suchan

Porovnání současných a nových hašovacích funkcí

Katedra Algebry

Vedoucí bakalářské práce: **Doc. RNDr. Jiří Tůma, DrSc.**

Studijní program: **Programování**

2007

Rád bych poděkoval Doc. RNDr. Jiřímu Tůmovi, DrSc. za vedení bakalářské práce, dále Mgr. Danielu Joščákovi za konzultace a cenné rady a také své rodině za stálou podporu.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 28. května 2007 Martin Suchan

Obsah

Název práce: Porovnání současných a nových hašovacích funkcí

Autor: Martin Suchan

Katedra: Katedra Algebry

Vedoucí bakalářské práce: Doc. RNDr. Jiří Tůma, DrSc.

E-mail vedoucího: Jiri.Tuma@mff.cuni.cz

Abstrakt: Obsahem této práce je přinést popis a porovnání nejrozšířenějších kryptografických hašovacích funkcí, které se v současné době používají (květen 2007), a dále je porovnat s návrhy nových hašovacích funkcí vyvíjených pro soutěž Advanced Hash Standard. Součástí této práce je také názorná implementace všech popisovaných funkcí v jazyce C#.

Klíčová slova: hašovací funkce, bezkoliznost, Advanced Hash Standard

Title: Comparative study of current and new hash functions

Author: Martin Suchan

Department: Department of Algebra

Supervisor: Doc. RNDr. Jiří Tůma, DrSc.

Supervisor's e-mail address: Jiri.Tuma@mff.cuni.cz

Abstract: The goal of this study is to present comparison of today's most widely used cryptographic hash functions and compare them with drafts of new hash functions, which are being currently developed for Advanced Hash Standard competition. This study also includes implementation of all described functions in programming language C#.

Keywords: cryptographic hash function, collision-free, Advanced Hash Standard

Úvod

Kryptografické hašovací funkce jsou významnou součástí moderní kryptologie. V porovnání s klasickými hašovacími funkcemi, které se podobně používají k zobrazení velkého definičního oboru na obor menší, kryptografické hašovací funkce (dále jen zkráceně hašovací funkce) se používají především k ověření integrity dat či zajištění autenticity.

Hašovací funkce mají v moderní kryptologii své místo již po několik desítek let, ale teprve s nedávným rozvojem a rozšířením internetu a telekomunikačních služeb získávají hašovací funkce na stále více důležitosti. Mnoho nejen internetových protokolů a aplikací, například ipSEC, SSL, SSH či PGP, staví na specifických vlastnostech těchto funkcí, odhalení významnějších slabin v hašovacích funkcích by tedy mohlo znamenat ohrožení všeobecné bezpečnosti komunikace po síti.

Cílem této bakalářské práce je představit v současné době nejznámější a nejrozšířenější hašovací funkce, popsat jejich základní vlastnosti a implementace, které je též možné si názorně prohlédnut v přiložených zdrojových kódech všech popisovaných funkcí. Dále se tato práce věnuje i popisu tří nových hašovacích funkcí, které byly v nedávné době představeny, a které byly navrhovány s ohledem na robustnost a odolnost proti známým postupům diferenciální kryptoanalýzy. Práce popisuje stávající systémy standardizace hašovacích funkcí NESSIE a CRYPTREC, dále také aktuálně probíhající soutěž *Advanced Hash Standard*, kterou přibližně před rokem vyhlásil NIST a jejímž cílem je v horizontu roku 2011 stanovit podmínky pro hodnocení bezpečnosti hašovacích funkcí a dále také vybrat z připravovaných kandidátů bezpečnou hašovací funkci, která se stane novým standardem FIPS. Tato soutěž je obdobou soutěže *Advanced Encryption Standard*, pomocí níž se před několika lety vybírala bezpečná bloková šifra.

V kapitole 1. je popisována základní terminologie a konstrukce hašovacích funkcí. V kapitole 2. jsou popisované v současné době používané a v kapitole 3. nově navržené hašovací funkce. Kapitola 4. se dále zaměřuje na projekty, jejichž cílem je standardizace hašovacích funkcí.

Jako příloha této práce je názorná implementace všech popisovaných hašovacích funkcí v jazyce C#, dále program pro vizualizaci průběhu vybraných hašovacích funkcí, schopný zobrazovat modulární rozdíl průběhu dvou vstupních bloků, vhodný jako pomůcka pro kryptoanalýzu. Třetí programovou přílohou je jednoduchý test pro porovnání rychlosti výpočtu haše u těchto hašovacích funkcí.

1 Vlastnosti a konstrukce hašovacích funkcí

1.1 Základní teorie a definice

Funkci $f: X \rightarrow Y$ nazveme *jednosměrnou*, pokud je pro libovolné $x \in X$ snadné vypočítat hodnotu $y = f(x)$, ale pro libovolně zvolené $y \in Y$ není možné najít takovou hodnotu $x \in X$, že $y = f(x)$.

Funkci $f: X \rightarrow Y$ nazveme *bezkolizní*, pokud není možné najít dvě hodnoty $x, x' \in X, x \neq x'$ takové, že $f(x) = f(x')$.

Funkci f dále nazýváme *hašovací funkcí*, pokud je jednosměrná a bezkolizní, přijímá na vstupu řetězec binárních dat libovolné délky a jako výstup produkuje řetězec binárních dat pevně dané délky: $f: \{0,1\}^* \rightarrow \{0,1\}^n$. Tento výstupní řetězec, který má délku *n*, označujeme jako *haš*, popřípadě jako *hash*, *digest*, *fingerprint* či *otisk dat*.

Pokud zde mluvíme o tom, že „není možné najít" kolizi či vzor, je tím myšleno, že danou kolizi či vzor neumíme nalézt pomocí současných znalostí a dostupných výpočetních prostředků. Už ze samotné definice funkce plyne, že musí existovat velké množství kolizních vstupních řetězců.

Dále také u moderních hašovacích funkcí platí, že délka vstupního řetězce není naprosto libovolná, obvykle musí být menší než 2^{64} nebo 2^{128} bitů. Toto omezení vyplývá z faktu, že samotná funkce na konci výpočtu zpracovává i tzv. *padding*, neboli zakončení zprávy, do kterého se zapisuje délka vstupního řetězce a tato zapisovaná délka má vždy pevně danou velikost v bitech.

Od kryptografické hašovací funkce se vyžaduje, aby splňovala následující požadavky:

Odolnost proti nalezení vzoru (preimage resistant = jednosměrnost) Pro daný haš **h** není možné najít takovou zprávu **z**, že $\boldsymbol{hash(z) = h}$. Složitost případného hledání vzoru je u dobré hašovací funkce $\mathbf{2^n}$, kde **n** je délka haše v bitech.

Odolnost proti nalezení druhého vzoru (second preimage resistant = slabá bezkoliznost) Pro danou zprávu $\boldsymbol{z_1}$ a její haš **h** není možné najít jinou zprávu $\boldsymbol{z_1 \neq z_2}$ takovou, že $\boldsymbol{hash(z_1) = h = hash(z_2)}$. Složitost případného hledání druhého vzoru je u dobré hašovací funkce také $\mathbf{2^n}$, kde **n** je délka haše v bitech

Bezkoliznost (collision resistant = silná bezkoliznost) Není možné najít dvě různé zprávy $\boldsymbol{z_1 \neq z_2}$ takové, že $\boldsymbol{hash(z_1) = hash(z_2)}$. Složitost hledání kolize je u dobré hašovací funkce díky *narozeninovému paradoxu* rovna $\mathbf{2^{n/2}}$, kde n je délka haše v bitech. Právě poslední podmínka bezkoliznosti hašovací funkce je nejsilnější požadavek. Pokud chceme mít hašovací funkci, jejíž složitost nalezení kolize je aspoň $\mathbf{2^n}$, potom tato funkce musí produkovat haš o délce aspoň **2n**.

Narozeninový paradox

U funkce produkující haš dlouhý 128 bitů nám pro nalezení kolizního páru zpráv s pravděpodobností přibližně 50% stačí vyzkoušet jen $\mathbf{2^{64}}$ zpráv. Může za to tzv. narozeninový paradox, který si můžeme představit takto:

Mějme hašovací funkci, která produkuje haš délky 128 bitů, potom existuje právě $\mathbf{2^{128}}$ různých hašů. Nechť **d** označuje tento počet různých hašů, potom $\bar{\boldsymbol{p}}(\boldsymbol{n}, \boldsymbol{d})$ bude udávat pravděpodobnost $\bar{\boldsymbol{p}}$, že v množině **n** náhodných hašů z množiny **d** nejsou žádné dva stejné.

$$\bar{p}(n,d) = 1 \cdot \left(1 - \frac{1}{d}\right) \cdot \left(1 - \frac{2}{d}\right) \cdots \left(1 - \frac{n-1}{d}\right) =$$

$$= \frac{d \cdot (d-1) \cdots (d-n+1)}{d^n} = \frac{d!}{d^n (d-n)!}$$

Pokud použijeme doplněk $\boldsymbol{p}(\boldsymbol{n}, \boldsymbol{d}) = \mathbf{1} - \bar{\boldsymbol{p}}(\boldsymbol{n}, \boldsymbol{d})$, získáme pravděpodobnost $\boldsymbol{p}$, že mezi **n** haši z **d** možných jsou aspoň dva stejné.

Tento výsledek se dá aproximovat: $\boldsymbol{p}(\boldsymbol{n}, \boldsymbol{d}) \approx \mathbf{1} - \boldsymbol{e}^{\frac{-\boldsymbol{n}(\boldsymbol{n}-\mathbf{1})}{\mathbf{2}\boldsymbol{d}}} \approx \mathbf{1} - (\mathbf{1} - \frac{\boldsymbol{n}}{\mathbf{2}\boldsymbol{d}})^{\boldsymbol{n}-\mathbf{1}}$

Speciální případ narozeninového paradoxu potom říká, že chceme-li s pravděpodobností přibližně 50% najít nějaké dva stejné prvky z celkem $\mathbf{2^n}$ možných, stačí mít množinu $\mathbf{2^{n/2}}$ náhodných prvků. Právě z tohoto důvodu je složitost nalezení kolizního páru zpráv s pravděpodobností 50% u dobré hašovací funkce s hašem délky **n** bajtů jenom $\mathbf{2^{n/2}}$ výpočtů.

1.2 Konstrukce moderních hašovacích funkcí

Vstupní blok dat pro hašovací funkci může být prakticky libovolně dlouhý, od nulové délky, přes pár desítek bajtů, až po několik GB, i mnohem více. Prakticky všechny současné moderní hašovací funkce využívají tzv. Merkle-Demgård konstrukci iterativního výpočtu haše (viz. Kapitola 1.4), který je založen na myšlence, postupného zpracování zprávy po blocích pevně dané délky. Vstupní zpráva se nejdříve doplní o potřebný počet bitů – aby byla celková délka vstupu dělitelná délkou jednoho bloku. Dále se zpráva zpracovává po blocích, výsledkem výpočtu každého bloku je tzv. *kontext* – jakýsi mezivýsledek, který je vstupem do dalšího bloku zpracování, spolu s dalším blokem vstupních dat.

1.3 Zarovnání vstupních dat

Samotný vstupní řetězec dat, který má funkce přijímat, může mít libovolnou velikost. Hašovací funkce ale zpracovává data pouze po blocích určité délky, je tedy nutné zarovnat vstupní řetězec o takové množství bitů, aby byl dělitelný velikostí bloku. Aby se ovšem předešlo možnostem vícenásobných kolizí při použití jednoduchého zakončení, třeba o samé nuly, musí být každá zpráva, včetně zarovnání, jedinečná. Proto se dnes, prakticky u všech moderních hašovacích funkcí, do zarovnání zapisuje i délka původního vstupního řetězce.

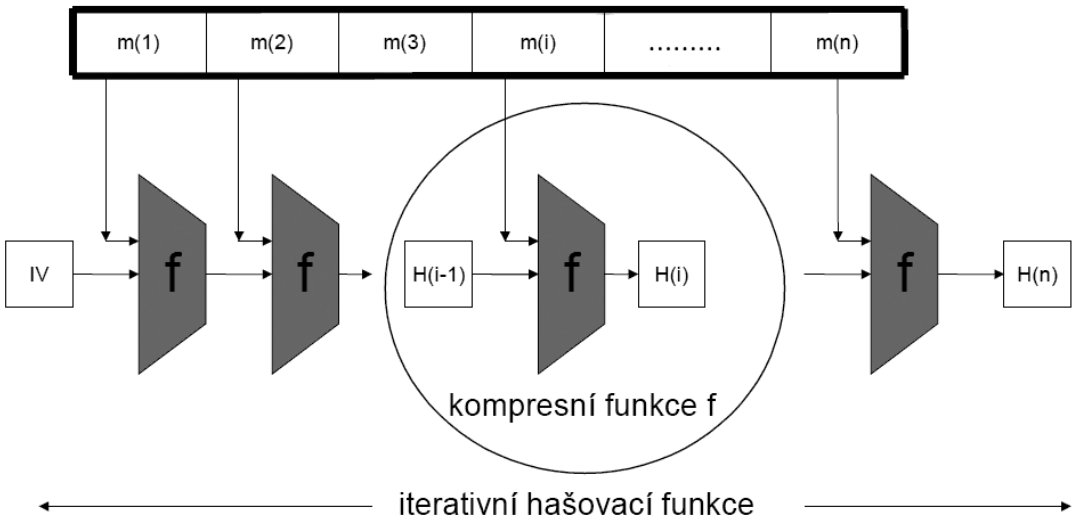
Zarovnání se provádí nejčastěji tak, že se ke vstupnímu řetězci dat doplní binárně jednička, dále potřebný počet nul, a nakonec délka vstupního řetězce v bitech vyjádřená binárně, to vše tak, aby celková konečná délka včetně tohoto zarovnání byla dělitelná velikostí bloku. Pro uložení délky vstupních dat do zarovnání se používá pevně daný počet bitů, nejčastěji 64, například u funkcí MD4, MD5 a SHA1. To také mimo jiné znamená, že vstupní blok dat musí být kratší než, v tomto případě, $\mathbf{2^{64}}$ bitů. To je ovšem velmi velké číslo, přibližně $2 \cdot 10^{18}$ bajtů, což pro současné potřeby bohatě stačí. Doplnění samotné délky vstupu do zpracovávaného řetězce se nazývá tzv. Merkle-Demgård zesílení. Odlišnosti v této implementaci jsou především v rovině použití Little-Endian / Big-Endian.

1.4 Merkle-Demgård konstrukce

Kompresní funkce tvoří samotné jádro hašovací funkce. Kompresní funkce přijímá na vstupu blok vstupních dat m_i a na výstupu produkuje určitý blok dat H_i, který se označuje jako *kontext*. Tento kontext je potom vstupem do kompresní funkce v dalším kroku. Kompresní funkce tedy přijímá na vstupu předchozí kontext H_{i-1} a nový blok dat m_i. Samotná kompresní funkce má tedy dva vstupy a jeden výstup. Počáteční hodnota kontextu se nazývá *Inicializační vektor IV*, který je vždy pro každou hašovací funkci pevně určen.

Kompresní funkci lze popsat jako funkci $f: \{0,1\}^{\mathrm{h}} \times \{0,1\}^{\mathrm{m}} \rightarrow \{0,1\}^{h}$, přičemž $H_0 = IV$ a $H_i = f(H_{i-1}, m_i)$. Tato konstrukce, kterou využívají prakticky všechny moderní hašovací funkce, se nazývá Merkle-Demgård konstrukce, podle dvou autorů, kteří ji nezávisle na sobě představili v roce 1989, viz Obr. 1.

Mimo této konstrukce, využívající jednu kompresní funkci, existují i konstrukce založené na proudových šifrách, které používají dvě, resp. čtyři kompresní funkce v jednom bloku zpracování (MDC-2, resp. MDC-4).



Obr. 1 Merkle-Demgård konstrukce

1.5 Konstrukce kompresní funkce

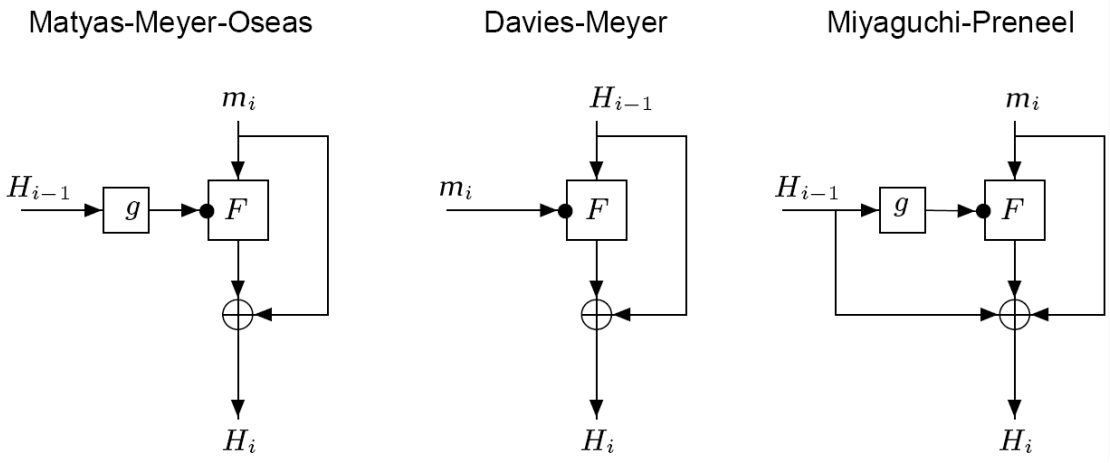
Kvalitní kompresní funkce je základ kvalitní hašovací funkce. Kompresní funkce by měla být velmi robustní, aby zajistila dokonalé promíchání bitů zprávy a jednocestnost. Měla by se chovat co možná nejvíce jako tzv. *náhodné orákulum*. Aby se dosáhlo požadovaných znalostí, vychází se často při konstrukci hašovacích funkcí z vlastností blokových šifer.

Od kvalitní blokové šifry $F_k(x)$ se při pevném klíči k očekává, že se chová také jako náhodné orákulum. Dále zaručuje, že známe-li jakoukoli množinu vstupů-výstupů, tj. otevřených-šifrových textů (x, y), není možné odtud určit (díky složitosti) klíč k. Vzhledem ke klíči je tak bloková šifra jednocestná. Přesněji pro každé x je funkce $k \rightarrow F_k(x)$ jednocestná. Odtud vyplývá možnost konstrukce kompresní funkce takto: $H_i = F_{m_i}(H_{i-1})$, kde F je kvalitní bloková šifra.

U moderních funkcí se používá navíc ještě jedna operace. Jedná se o tzv. Davies-Meyerovu konstrukci kompresní funkce, která zesiluje vlastnost jednocestnosti ještě přičtením předchozího kontextu před výstupem: $H_i = f(H_{i-1}, m_i) = F_{m_i}(H_{i-1})\ xor\ (H_{i-1})$. Výstup je zde tedy navíc maskován vstupem, což ještě více ztěžuje případný zpětný chod. Alternativou je použití Matyas-Meyer-Oseas konstrukce, kde se přičítá vstupní blok dat před výstupem, či Miyaguchi-Preneel konstrukce, kde se přičítá jak předchozí kontext, tak vstupní blok dat viz Obr. 2 (g značí předzpracování vstupního kontextu/klíče).

Náhodné orákulum

Náhodné orákulum je hypotetické zařízení, které na každý vstup odpovídá náhodně vybraným výstupem ze svého oboru hodnot, navíc ale s tou podmínkou, že na stejně zadané vstupy odpovídá vždy stejnými odpověďmi. Cílem návrhu kompresní funkce je, aby se taková funkce chovala podobně jako náhodné orákulum.



Obr. 2 Konstrukce kompresní funkce

1.6 Kryptografické využití hašovacích funkcí

Digitální otisk dat

Hašovací funkce díky své vlastnosti bezkoliznosti slouží jako funkce, která každému souboru či množině dat přiřazuje jednoznačný identifikátor – digitální otisk dat. Jakkoliv dlouhá data lze tedy reprezentovat a identifikovat pomocí otisku majícího jen několik set bitů. O těchto otiscích lze tedy mluvit podobně, jako třeba o otiscích prstů u lidí.

Ověření integrity dat (Modification detection codes, MDC)

Do této skupiny patří například samoopravné kódy. To jsou obvykle jednodušší funkce, jejichž hlavním cílem je výpočet kontrolního součtu pro soubory dat. Jsou používané především na záznamových mediích, pro detekci poškození media či chyby zápisu/čtení, a také v telekomunikaci jako kontrolní součty přenášených dat.

Ověřování autenticity zpráv (Message authentication codes, MAC)

Tyto hašovací funkce přijímají na vstupu navíc i unikátní klíč a jejich cílem je zajištění autenticity posílané zprávy – pro případného útočníka je složité najít takový pár zprávy a klíče, který by dával daný výsledný MAC. Nejčastější je použití tzv. konstrukce HMAC (keyed-hash message authentication code). Zde je vyžadována především odolnost proti nalezení druhého vzoru.

Porovnávání obsahu databází

Hašovací funkci lze jednoduše používat pro porovnání obsahu databází, zda jsou identické. Je jednodušší vypočítat haš u obou databází a poslat přes síť jen haš pro kontrolu, než kontrolovat databáze přímo a zbytečně tak zatěžovat síťové připojení. V tomto případě je důležitá bezkoliznost hašovací funkce.

Ukládání hesel

V přihlašovacích a autentizačních systémech se neukládají přímo hesla, ale jen jejich haše. Případný útočník tedy neví, kdo používá jaké heslo, pokud se mu podaří získat přístup k souboru s haši. Pro ukládání hesel se navíc používá i tzv. *solení*, neboli použití náhodného řetězce dat, který se připojuje k heslu ještě před hašováním. Tímto způsobem lze zabránit použití slovníkového útoku nebo tzv. *time-memory tradeoff* (vykoupení časové složitosti útoku hrubou silou paměťovou složitostí). Příkladem útoků na uložené haše jsou třeba tzv. *rainbow tables* - rozsáhlé, veřejně přístupné databáze hašů ke krátkým heslům [16], či podobně u blokových šifer tzv. *Hellman tables* - předpočítané tabulky klíčů [20]. Při ukládání hašů je důležitá zejména jednosměrnost – dle haše nesmí být možno zjistit heslo.

Pseudonáhodné funkce

Standard PKCS#5 umožňuje využít hašovací funkci k tvorbě „náhodného“ šifrovacího klíče z hesla pomocí pseudonáhodné funkce PRF jako *klíč = PRF(heslo).* Předpis spočívá v hašování hesla a následném mnoho-násobném hašování výsledku. Počet hašování je dán konstantou *c*, jejíž hodnota se doporučuje být minimálně 1000, ale používá se i 2000. Výsledkem je krátký „náhodně vyhlížející“ klíč DK, který je možné využít lépe než původní heslo. Jednak má pevnou délku a také z něho lze využít tolik bitů, kolik potřebujeme. Zde hraje klíčovou roli náhodnost bitů v produkovaných haších.

Pseudonáhodné generátory

Typické použití hašovacích funkcí jako pseudonáhodných generátorů je v případech, kdy máme k dispozici krátký řetězec (náhodných) dat (*seed*), typicky „krátký“ 256bitový náhodný šifrovací klíč. Přitom potřebujeme z tohoto vzorku získat pseudonáhodnou posloupnost o velké délce, například 1 GB apod. U pseudonáhodných generátorů, podobně jako v předchozím příkladě, hraje klíčovou roli náhodnost bitů v produkovaných haších. Například standard PKCS#1 v. 2.1 definuje pseudonáhodný generátor pomocí hašovací funkce *H*, s počátečním nastavením *seed*, takto:

H (seed || 0x00000000), H (seed || 0x00000001), H (seed || 0x00000002)…

Hašovací funkce jako stavební kameny kryptografických protokolů

V praxi se používají hašovací funkce zejména v těchto případech:

Certifikace a digitální podpisy. Hašovacích funkcí se zde používá především pro jejich náhodnost, například jako zakončení v RSA-PSS (Probabilistic Signature Scheme), nebo v tzv. PKI (Public key infrastructure) či pro vytváření tzv. *časových razítek.* Důležitá je zde odolnost proti nalezení druhého vzoru.

Autentifikace, ověřování identity uživatelů. Hašovací funkce se zde používají pro vypočtení tajného klíče, důležitou roli zde hraje jednosměrnost funkce. Příkladem je přihlašovací systém Kerberos, IEEE 802.1X-EAP či APOP (Authenticated Post Office Protocol).

Zabezpečená komunikace. Hašovací funkce se používají v mnoha komunikačních protokolech, jako je třeba sada protokolů IPSec (IP security), kde se hašování používá k zabezpečení procedury pro výměnu klíčů IKE (Internet key exchange) a dále také pro kontrolu integrity přenášených zpráv. Důležitá je zde odolnost proti nalezení druhého vzoru. Dále jsou hašovací funkce klíčové např. v zabezpečeném komunikačním protokolu TLS (Transport Layer Security), SSL (Secure Sockets Layer) či SSH (Secure Shell).

Zabezpečený přenos emailů. Hašovací funkce se zde používají spolu s kryptografickým systémem založeným na veřejném klíči. Příkladem je třeba protokol S/MIME (Secure / Multipurpose Internet Mail Extensions) pro zabezpečení přenosu příloh u emailů, dále třeba program PGP (Pretty Good Privacy) pro šifrování samotných zpráv. Důležitá je zde opět odolnost proti nalezení druhého vzoru.

Dále se hašovací funkce používají třeba i ke vzorkování a filtrování přenášených paketů(PSAMP: IETF), kde se využívá především bezkoliznosti hašovacích funkcí.

Ve většině těchto případů se stále používají starší funkce MD2, MD4, MD5 a SHA1. Důvodem je především to, že jak tyto funkce, tak samotné protokoly vznikly již před delší dobou a nyní jsou v těchto standardech pevně zakotvené. Výjimečně se setkáme s použitím i jiných funkcí, nejčastěji jsou to funkce HAVAL, RIPEMD či RIPMD160.

2 Současné hašovací funkce

2.1 MD2

Historie

Tato funkce byla navržena roku 1989 Ronaldem Rivestem a v květnu roku 1992 uznána jako standard RFC 1319 [1]. MD2 byla původně optimalizována pro osmibitové počítače a podle toho vypadá i její struktura. Existovala mimo to i funkce MD, která byla ovšem použita pouze proprietárně, a dále funkce MD3, která nebyla oficiálně publikována z důvodu blíže nespecifikované bezpečnostní slabiny.

Vlastnosti

Funkce MD2 zpracovává bloky dat o velikosti 128 bitů, velikost kontextu je 128 bitů a výsledný haš je dlouhý 128 bitů. Tato funkce používá na dnešní dobu již poněkud neobvyklou konstrukci. V prvé řadě pracuje pouze s jednotlivými bajty – osmibitové počítače, pro které byla cílena, neuměli pracovat s delšími čísly. Kompresní funkce zpracovává dat celkem v osmnácti rundách, ve kterých se nejdříve inicializuje kontext, který pak prochází dvojnásobným cyklem, ve kterém se provádí substituce pomocí pí-tabulky, v kombinaci s operací xor na výsledných hodnotách.

Bezpečnost funkce

V roce 1997 byly popsány možnosti kolize v samotné kompresní funkci, Rogier [35], avšak pomocí tohoto způsobu se nepodařilo dosáhnout kolize v celé MD2.

V roce 2004 byl publikován preimage attack se složitostí $\mathbf{2^{104}}$, Muller [34]. Závěrem bylo, že funkci MD2 již nelze považovat za jednosměrnou hašovací funkci.

2.2 MD4

Historie

Tato funkce je v řadě již čtvrtou funkcí navrženou Ronaldem Rivestem. Funkce MD4 byla navržena v roce 1990 a uznána v říjnu 1990 jako standard RFC 1186, který byl v květnu 1992 nahrazen standardem RFC 1320 [2]. MD4 využívá oproti MD2 již pokročilejší konstrukce a stala se inspirací pro mnohé pozdější hašovací funkce, jako jsou MD5, SHA či RIPEMD.

Vlastnosti

Funkce MD4 přijímá vstupní blok dat menší než $\mathbf{2^{64}}$ bitů. Vstup je nejdříve doplněn o zarovnání a poté rozdělen na bloky dat o velikosti 512 bitů. Hlavní část algoritmu MD4 pracuje s kontextem o velikosti 128 bitů rozděleným na 4 x 32 bitová slova A, B, C, D. Algoritmus potom v každém cyklu zpracovává 512 bitový vstup, pomocí kterého modifikuje kontext. Každý blok výpočtu se skládá z 3 podobných rund, kde každá runda obsahuje 16 podobných operací založených na nelineární funkci f, bitovém xor a levé bitové rotaci. Výsledný haš je dlouhý 128 bitů.

Bezpečnost funkce

Slabiny funkce MD4 byly demonstrovány již v roce 1991, Boer [19].

V roce 2004 zveřejnil tým Dr. Wangové kolize pro tuto funkci [37] a v roce 2005 také podrobnou kryptoanalýza a postup pro efektivní hledání kolizí se složitostí jen $\mathbf{2^{14}}$ výpočtů funkce [40]. U této funkce lze navíc dosáhnout kolize již v samotné kompresní funkci.

2.3 MD5

Historie

MD5 je nejnovější funkcí navrženou Ronaldem Rivestem. Funkce MD5 byla navržena v roce 1991 a uznána v květnu 1992 jako standard RFC 1321 [3]. MD5 je prakticky optimalizovaná verze funkce MD4, která již neobsahuje kritická místa této předchozí funkce. Používá navíc jedinečnou aditivní konstantu v každém cyklu výpočtu kontextu, oproti třem konstantám v MD4. V současnosti se, i přes její objevené slabiny, jedná spolu s SHA o nejpoužívanější hašovací funkci vůbec.

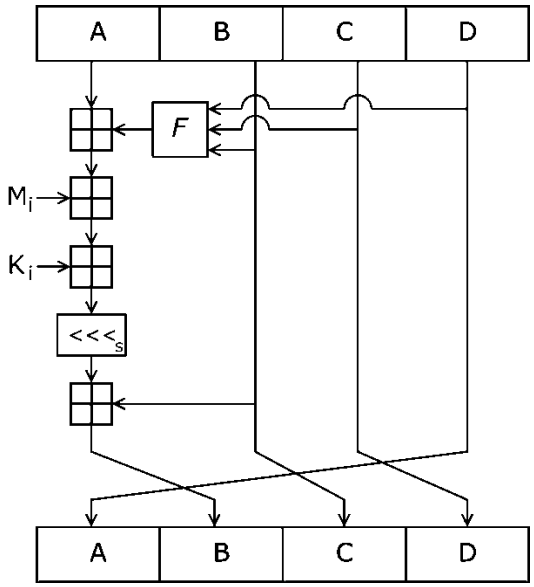
Vlastnosti

Funkce MD5 má velmi podobnou konstrukci, jako MD4. Na vstupu přijímá vstupní blok dat menší než $\mathbf{2^{64}}$ bitů. Vstup je nejdříve doplněn o zarovnání a poté rozdělen na bloky dat o velikosti 512 bitů. Hlavní část algoritmu MD5 pracuje s kontextem o velikosti 128 bitů rozděleným na 4 x 32 bitová slova A, B, C, D. Algoritmus potom v každém cyklu zpracovává 512 bitový vstup, pomocí kterého modifikuje kontext. Každý blok výpočtu se skládá ze 4 podobných rund, kde každá runda obsahuje 16 podobných operací založených na nelineární funkci f, xor a levé bitové rotaci. Výsledný haš je dlouhý 128 bitů. Detail kompresní funkce viz Obr. 4.

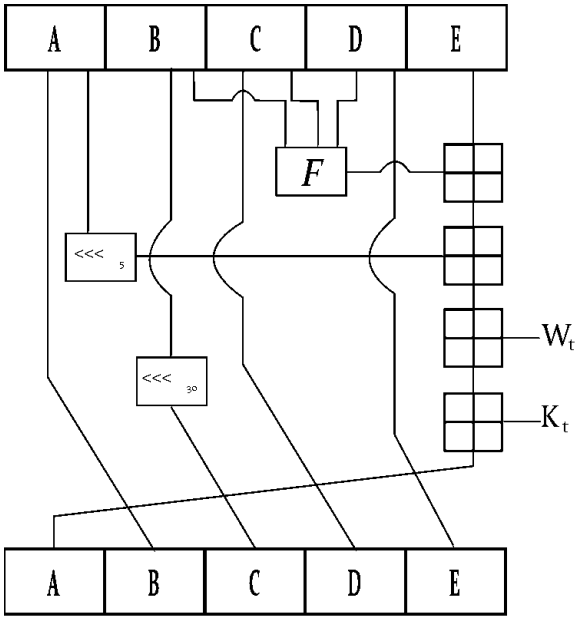
Bezpečnost funkce

První slabiny této funkce byly odhaleny v roce 1996, Dobbertin [23] a i přesto, že se nejednalo a žádnou větší slabinu tak, jak tomu bylo třeba u MD4, bylo již tehdy doporučeno používat funkci jinou, např. SHA-1.

Zásadní zlom v bezpečnosti této funkce přišel v roce 2004, kdy tým dr. Wangové zveřejnil kolize pro tuto funkci [37] a v roce 2005 ukázal možnost hledání kolizí u této funkce v přijatelné výpočetní složitosti [39]. Tento prezentovaný způsob byl poté několikrát zdokonalen, Klíma [28], Stevens [36] a současný stav je takový, že najít kolizi je možno u této funkce již během několika minut na běžném PC, Klíma [31].



Obr. 4 kompresní funkce MD5



Obr. 3 kompresní funkce SHA-1

2.4 SHA-0 a SHA-1

Historie

Jedná se o dvě hašovací funkce, které vychází z konstrukce funkcí MD4 a MD5 a na jejichž návrhu se podílela samotná NSA (National Security Agency). První specifikace algoritmu SHA-0, byla publikovaná roku 1993 úřadem NIST (National Institute of Standards and Technology) jako tzv. Secure Hash Standard, FIPS PUB 180. Tato verze byla ovšem brzy po publikování stažena a nahrazena až v roce 1995, FIPS PUB 180-1, verzí SHA-1 [4]. Ta se od původní verze lišila jen přidáním jedné bitové rotace v expanzní funkci vstupních dat. Tato modifikace byla přidána jako ochrana proti blíže nespecifikované bezpečnostní mezeře v původní funkci.

Vlastnosti

Funkce SHA-0 se od SHA-1 liší jen přidáním jedné bitové rotace v expanzní funkci vstupních dat. SHA-1 na vstupu přijímá vstupní blok dat menší než $\mathbf{2^{64}}$ bitů. Vstup je nejdříve doplněn o zarovnání a poté rozdělen na bloky dat o velikosti 512 bitů. Hlavní část algoritmu SHA-1 pracuje s kontextem o velikosti 160 bitů rozděleným na 5x32 bitová slova A, B, C, D, E. Algoritmus v každém bloku zpracovává 512 bitový vstup, pomocí kterého modifikuje kontext. Vstupní blok dat je v každém cyklu nejdříve rozšířen pomocí expanzní funkce na 80x32 bitových slov. Každý blok výpočtu se skládá ze 4 podobných rund, kde každá runda obsahuje 20 podobných operací založených na nelineární funkci f, bitovém xor a levé bitové rotaci. Výsledný haš je dlouhý 160 bitů. Detaily kompresní funkce viz Obr. 3.

Bezpečnost funkce SHA-0

Již v roce 1998 byl prezentován způsob nalezení kolize se složitostí $\mathbf{2^{61}}$, oproti ideální složitosti $\mathbf{2^{80}}$, Chabaud [25].

V roce 2004 byla nalezena částečná kolize – 142 bitů ze 160. Dále byla také nalezena úplná kolize u algoritmu omezeného na 62 z 80 operací, Biham [24].

12 srpna 2004 byla pomocí superpočítače TERA NOVA nalezena úplná kolize u SHA-0 . Byl použit zobecněný postup se složitostí $\mathbf{2^{51}}$, Joux [27].

17 srpna 2004 byl na konferenci CRYPTO 2004 publikován útok pro nalezení kolize se složitostí $\mathbf{2^{40}}$ operací, Wang [37].

V únoru 2005 byl publikován způsob hledání kolizí u SHA-0 se složitostí jen $\mathbf{2^{39}}$, Wang [38].

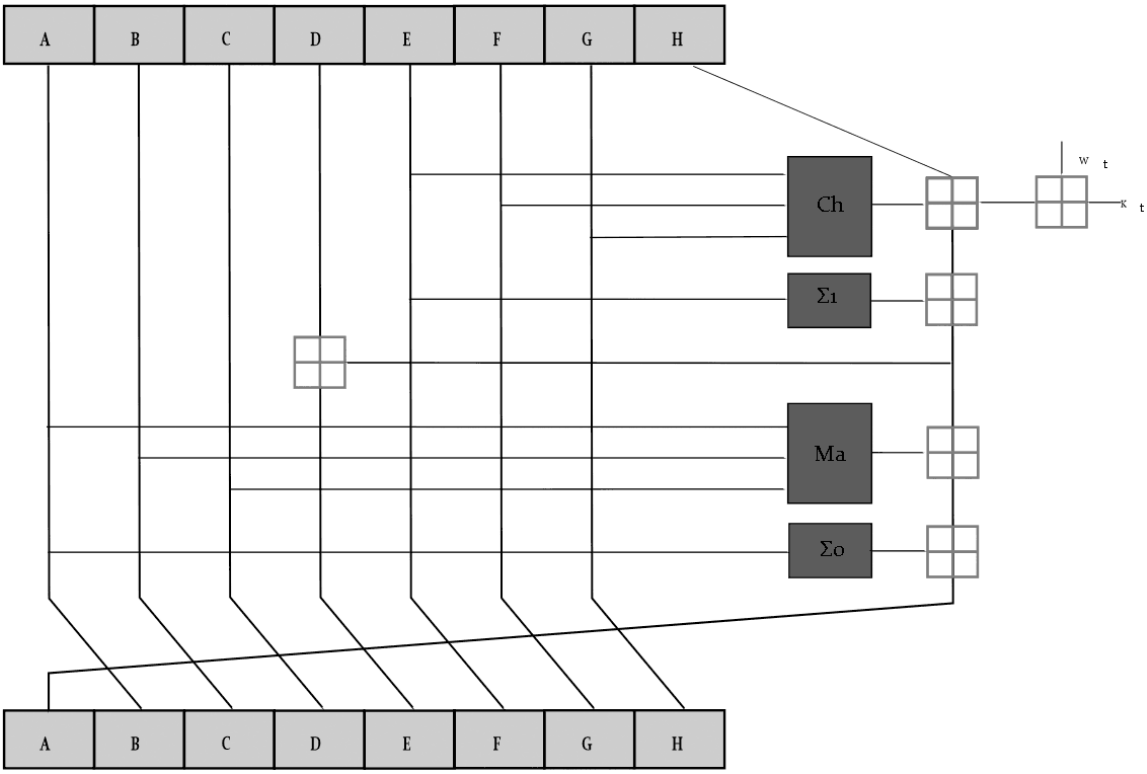
Bezpečnost funkce SHA-1

SHA-1 patří v současné době spolu s funkcí MD5 mezi nejpoužívanější funkce vůbec. Je tedy zřejmé, že právě na tyto dvě funkce se nejvíce soustředí pozornost moderní kryptoanalýzy. Po množství útoků na SHA-0 bylo zřejmé, že ani funkce SHA-1 nebude dlouhodobě nabízet vysokou bezpečnost, následkem čehož bylo po publikování výsledků na CRYPTO 2004 [37] doporučeno přejít na funkce SHA-2 v horizontu roku 2010.

V únoru 2005 byl publikován výsledek při hledání kolizí u SHA-1 se složitostí $\mathbf{2^{69}}$, oproti ideální $\mathbf{2^{80}}$, Wang [41].

V říjnu 2005 byl publikován zlepšený útok na SHA-1 se složitostí nalezení kolize $\mathbf{2^{63}}$, Wang [42].

Obecně dosud nebyla nalezena žádná úplná kolize u SHA-1, ale lze předpokládat, že nalezení kolize je již nyní dosažitelné při použití distribuovaných internetových výpočtů.



Obr. 5 kompresní funkce SHA-2

2.5 SHA-2

Historie

SHA-2 je souhrnné označení 4 funkcí, na jejichž vývoji se podílela NSA a které publikoval NIST: SHA-256, SHA-384, SHA-512 a SHA-224. První tři jmenované funkce byly publikované v roce 2001 jako FIPS PUB 180-2 [5]. Čtvrtá jmenovaná funkce SHA-224 byla specifikovaná dodatečně v roce 2004. Funkce SHA-256, SHA-384 a SHA-512 byly také spolu s funkcí Whirlpool doporučeny projektem NESSIE a byly také uznány jako společný mezinárodní standard ISO/IEC 10118-3 [6].

SHA-256 je funkce pracující s 32bitovými slovy a SHA-224 je odvozená varianta s kratším hašem a jinými inicializačními vektory a konstantami. SHA-512 pracuje s 64 bitovými slovy a podobně funkce SHA-384 je její odvozená varianta s kratším hašem a jinými inicializačními vektory.

Vlastnosti

Funkce SHA-256 a SHA-224 přijímá vstupní blok dat menší než $\mathbf{2^{64}}$ bitů. Vstup je nejdříve doplněn o velikost a poté rozdělen na bloky dat o velikosti 512 bitů. Hlavní část algoritmu pracuje s kontextem o velikosti 256 bitů rozděleným na osm 32 bitových slov A, B, C, D, E, F, G, H. Algoritmus v každém bloku zpracovává 512 bitový vstup, pomocí kterého modifikuje kontext. Vstupní blok dat je v každém cyklu nejdříve rozšířen pomocí expanzní funkce, která je oproti expanzní funkci v SHA1 mnohem robustnější, na 80x32 bitových slov. Každý blok výpočtu se skládá z 80 podobných operací založených na nelineárních funkcích Σ, σ, χ a Maj, viz Obr. 5 kompresní funkce SHA-2. Výsledný haš je dlouhý 256, resp. 224 bitů.

Podobně je tomu u funkcí SHA-512 a SHA-384, které se liší jen tím, že přijímají vstupní blok dat menší než $\mathbf{2^{128}}$ bitů, používají 64 bitová slova, jiné nelineární funkce, konstanty a inicializační vektory. Výsledný haš těchto funkcí je dlouhý 512, resp. 384 bitů.

Bezpečnost funkcí SHA-2

U funkcí SHA-2 dochází k mnohem složitější expanzi vstupního bloku zprávy, než je tomu u funkcí MD5 či SHA-1, což pro kryptoanalýzu představuje zatím nepřekonatelný problém. Dle současných informací nejsou známé žádné slabiny těchto hašovacích funkcí a jsou považované za bezpečné.

2.6 Funkce RIPEMD

Historie:

Původní funkce označovaná jen jako „RIPEMD“ byla navržena v roce 1992 [7]. Její konstrukce byla velmi podobná funkci MD4, prakticky se skládala ze dvou průběhů funkce MD4, které se na konci sečetli podle daného pravidla. S pozdějším objevením slabin u funkce MD4 byly podobné slabiny identifikované i u funkce RIPEMD.

Později, v roce 1996, byla navržena skupina nových funkcí pod jmény RIPEMD-128, RIPEMD-160, RIPEMD-256 a RIPEMD-320 [8]. RIPEMD je zkratkou pro RACE Integrity Primitives Evaluation Message Digest. Autory těchto funkcí jsou Hans Dobbertin, Antoon Bosselaers, Bart Preneel.

Vlastnosti

Funkce přijímá vstupní blok dat menší než $\mathbf{2^{64}}$ bitů. Vstup je nejdříve doplněn o velikost a poté rozdělen na bloky dat o velikosti 512 bitů. Hlavní část algoritmu pracuje s kontextem o velikosti 256 bitů u varianty RIPEMD-128/256 a 320 bitů u variant RIPEMD-160/320. Velikost haše je u původní funkce RIPEMD 128 bitů, u nových funkcí je velikost haše uvedena v názvu.

Všechny funkce zpracovávají paralelně 2 výpočty, které se na konci každého bloku sečtou. Navíc, u funkcí RIPEMD-256 a 320 dochází během paralelního výpočtu ke vzájemné substituci proměnných.

Bezpečnost

U původní funkce RIPEMD byly již v roce 1997 objeveny slabiny, poukazující na to, že kompresní funkce jen se dvěma rundami výpočtu není bezkolizní, Dobbertin [22]. Dále byla v roce 2004 ukázána kolize u celé této funkce, Wang [37] a později v roce 2005 i podrobná kryptoanalýza, Wang [40].

U nových funkcí RIPEMD128/160/256/320 nebyly dosud nalezeny žádné slabiny a spolu s funkcemi SHA-2 jsou považované za bezpečné.

2.7 Funkce HAVAL

Historie:

Jedná se o hašovací funkci, která má volitelnou délku výsledného haše (128, 160, 192, 224, 256 bitů) a také volitelný počet cyklů výpočtu (3, 4, 5), výchozí je haš délky 256 bitů a 5 cyklů výpočtu. Funkce HAVAL byla navržena roku 1992 a jejími autory jsou Yuliang Zheng, Josef Pieprzyk a Jennifer Seberry [9].

Vlastnosti

Funkce přijímá vstupní blok dat menší než $\mathbf{2^{128}}$ bitů. Vstup je nejdříve doplněn o velikost a údaj o tom, jaká verze funkce se používá, poté je rozdělen na bloky dat o velikosti 512 bitů. Hlavní část algoritmu pracuje s kontextem o velikosti 256 bitů. Výpočet probíhá v **n** cyklech, kde **n** je parametr funkce. V každém cyklu se provádí 4 x 8 nelineárních operací se všemi proměnnými + vstupem + jedinečnou konstantou, přičemž výsledek se vždy přiřazuje do jedné z proměnných. Nakonec se z výsledného kontextu složí haš. Způsob složení haše je unikátní pro každou z volitelných velikostí haše.

Bezpečnost

U varianty funkce se třemi cykly a hašem o délce 128 bitů byla roku 2004 nalezena kolize podobným způsobem, jako u algoritmů MD4 a MD5, Wang [37]. Lze tedy předpokládat, že ani jiné varianty funkce HAVAL nebudou mít ideální bezpečnost.

2.8 Funkce Snefru

Historie:

Jedná se o hašovací funkci navrženou v roce 1989 Ralphem Merklem pro firmu Xerox [33], která produkuje 128 bitový, nebo 256 bitový haš. Byla pojmenována podle Egyptského panovníka Sneferu, podobně jako blokové šifry Khufu a Khafre od stejného autora. Existuje více verzí této funkce, 1.0, 2.0, 2.1, 2.2, 2.3, 2.5, lišících se hodnotami v substitučních tabulkách a/nebo délkou výstupu a/nebo počtem průchodů.

Vlastnosti

Funkce přijímá vstupní blok dat menší než $\mathbf{2^{64}}$ bitů. Velikost výstupu může být 128 nebo 256 bitů dlouhý haš. Funkce pracuje buď se 4, nebo 8 průchody hlavním algoritmem. Základní model funkce je založen na práci se substitučními tabulkami, které jsou pevně dané a mají velikost 16 tabulek po 256 x 32 bitových slovech.

Bezpečnost

Původní návrh této hašovací funkce, který používal jen dva průchody hlavní částí algoritmu (v2.0), namísto současných osmi (v2.5), se ukázal jako napadnutelný. Eli Biham a Adi Shamir ukázali, jak lze najít druhý vzor k dané zprávě u této funkce pomocí diferenciální kryptoanalýzy [18]. Jak bylo později ukázáno, i po zvýšení počtu průchodů v této funkci, je možno najít kolize rychleji než hrubou silou, konkrétně se složitostí $\mathbf{2^{88}}$ operací u varianty se 4 průchody a délkou haše 128 bitů – to je ale v současné době nedosažitelná výpočetní složitost.

2.9 Funkce Tiger

Historie:

Jedná se o hašovací funkci navrženou v roce 1996, autory jsou Ross Anderson a Eli Biham [10]. Jedná se funkci vytvořenou cíleně pro 64bitové systémy. Funkce produkuje haš o délce 192 bitů, existuje i modifikace pro haš o velikosti 160/128 bitů, kde se původní haš zkrátí na požadovanou délku. Dále existuje zatím bez oficiální specifikace varianta Tiger2, která se od již existující varianty liší jen způsobem zakončení vstupní zprávy – to je u této funkce řešené stejně, jako je tomu u funkce MD5.

Vlastnosti

Funkce zpracovává vstupní blok dat o délce menší, než $\mathbf{2^{64}}$ bitů. Velikost výstupu je standardně 192 bitů dlouhý haš. Funkce využívá kombinovanou konstrukci založenou na iterativním počítání s nelineárními funkcemi, známou třeba z MD5, a substituci pomocí substitučních tabulek, podobně jako u funkce Snefru.

Bezpečnost

V roce 2006 byl publikován způsob, jak najít kolizi pro funkci Tiger omezenou na 16 průchodů se složitostí $\mathbf{2^{44}}$ výpočtů funkce, později i na 20 průchodů, pro nalezení pseudo-kolize, Kelsey [26]. V současnosti není zatím známá žádná kolize pro tuto funkci, ale s ohledem na nalezení slabin u nejrozšířenějších funkcí, podrobuje se i tato čím dál tím více odborné kryptoanalýze.

2.10 Funkce Whirlpool

Historie:

Jedná se o kryptografickou hašovací funkci navrženou Vincentem Rijmenem (spoluautor Advanced Encryption Standard) a Paulo S. L. M. Barretem [11]. Tato funkce byla spolu s funkcemi SHA-2 doporučena projektem NESSIE a byla také uznána jako společný mezinárodní standard ISO/IEC 10118-3 [6]. Dle slov autorů, Whirlpool nebude nikdy patentovaná a může být používána bez omezení pro jakýkoliv účel.

Vlastnosti

Whirlpool pracuje s čtvercovou blokovou šifrou, vychází z Miyaguchi-Preneel konstrukce a je založená na modifikované šifře AES. Funkce přijímá vstupní blok dat menší než $\mathbf{2^{256}}$ bitů a jejím produktem je haš o délce 512 bitů.

Bezpečnost

Původně existovaly dvě předchozí verze této funkce. První verze Whirlpool-0 byla ta, která byla předložena projektu NESSIE, její upravená verze Whirlpool-T byla vybrána projektem NESSIE mezi doporučené funkce. U této verze byla v roce 2003 nalezena drobná chyba v difuzní vrstvě, která byla opravena a současná verze, jednoduše nazývaná jen Whirlpool, je již považována za bezpečnou.

2.11 Funkce Panama

Historie:

Panama je kryptografické primitivum, které může být použito buď jako hašovací funkce, nebo jako proudová šifra. Byla představena v roce 1998 a jejími autory jsou Joan Daemen a Craig Clapp [21].

Vlastnosti

Panama pracuje se dvěma základními elementy, posuvným registrem se 32 x 32bitové slovy, a s kruhovou mixující funkcí pracující nad 17 x 32 bitovými slovy. Funkce přijímá vstupní blok dat menší než $\mathbf{2^{64}}$ bitů a jejím produktem je haš o délce 256 bitů.

Bezpečnost

U funkce Panama byly objeveny zásadní slabiny v konstrukci, které umožňují najít kolizní pár jen se složitostí $\mathbf{2^{6}}$ ohodnocení stavu funkce, Daemen [13]. Právě z výsledků kryptoanalýzy funkce Panama vycházeli autoři nové funkce RadioGatún, u které byl snížen počet slov používaných v jednotlivých cyklech a přidána zpětná vazba ze stavu na zásobník funkce.

3 Nové hašovací funkce

3.1 Hašovací funkce HDN podle konceptu SNMAC

Hašovací funkce HDN podle koncepce SNMAC, jejímž autorem je Vlastimil Klíma, je velmi mladou funkcí, která byla veřejně představena teprve v lednu roku 2007 [30]. Původně byla vyvíjena jako součást projektu pro Národní Bezpečnostní Úřad ČR pod označením „Speciální bloková šifra" (ST20052006018). Je založena na dvou nových stavebních prvcích, tzv. SNMAC – Speciální hašovací funkci typu NMAC a Speciální blokové šifře, zkráceně SBŠ, Klíma [29].

SBŠ – Speciální bloková šifra

Speciální bloková šifra je nové kryptografické primitivum, které bylo navrženo jako stavební prvek hašovacích funkcí nové generace SNMAC. Na rozdíl od klasické blokové šifry je konstruována s přímým zřetelem na takové vlastnosti, které jsou u klasických šifer očekávány v podstatě jen jako vedlejší efekt. Její návrh striktně předpokládá, že útočník může útočit ze strany klíče, že zná šifrovací klíč a může s ním libovolně manipulovat.

Hašovací funkce typu SNMAC

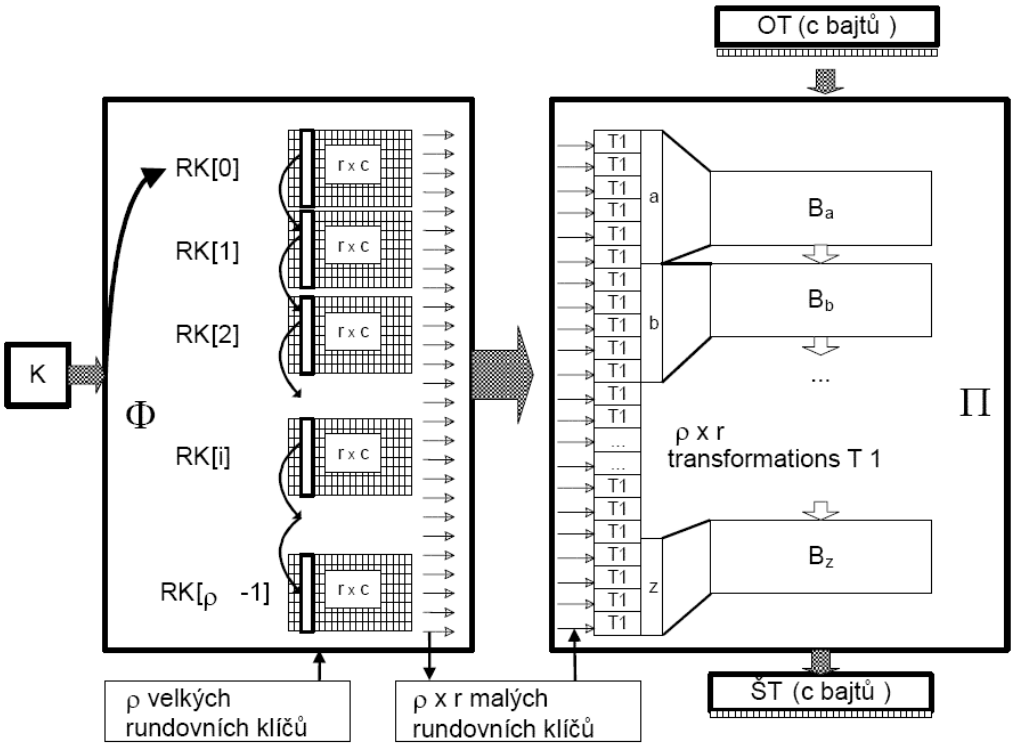
SNMAC je označení pro iterativní hašovací funkce typu NMAC, Bellare [17]. SNMAC využívá speciální blokovou šifru E s n bitovým blokem a K-bitovým klíčem. Má dále kompresní funkci f a závěrečnou úpravu g.

Hašovací funkce SNMAC mají veřejně známá návrhová kritéria, limitně se blíží náhodnému orákulu, jsou výpočetně odolné proti nalezení vzoru a kolize, a umožňují návrh pomocí různých instancí speciálních blokových šifer.

Hašovací funkce HDN využívá ke své konstrukci speciální blokovou šifru z rodiny DN – DoubleNet. Základní myšlenka speciální blokové šifry DN je ta, že oproti klasické blokové šifře věnuje zpracování klíče stejnou pozornost, jako klasická bloková šifra věnuje zpracování otevřeného textu. U DN jedna SP síť zajišťuje míchání klíče a druhá míchání otevřeného textu s klíčem.

DN(n, k)-ρ, je bloková šifra, která má blok o délce n bitů, šifrovací klíč K o délce k bitů a ρ velkých rund, kde ρ je bezpečnostní parametr.

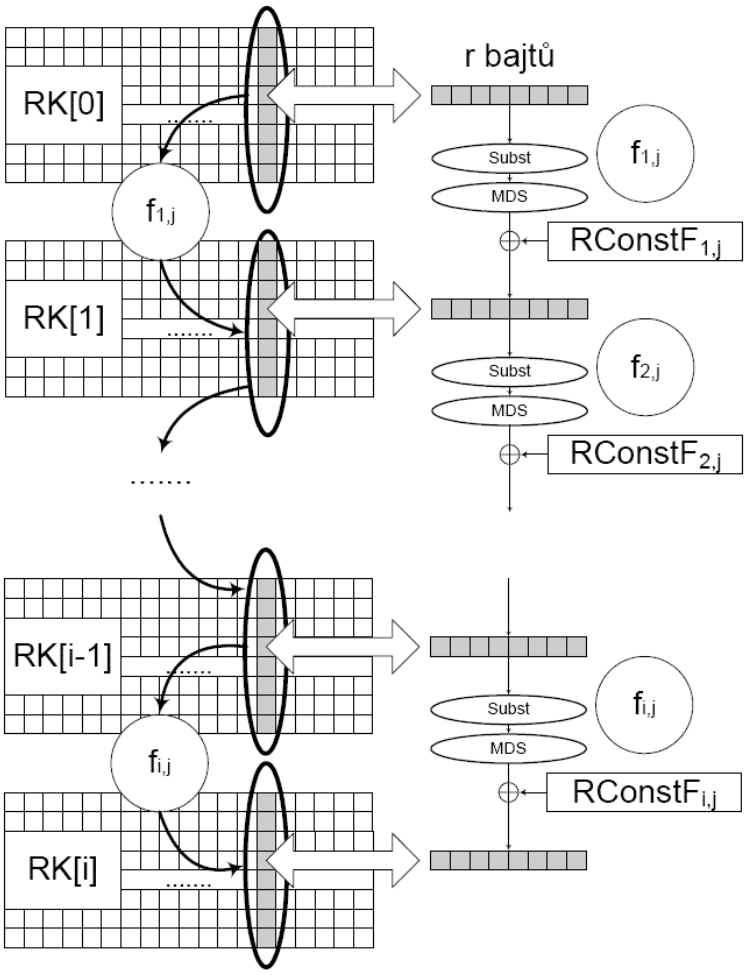
DN se skládá ze dvou funkcí, expanze klíče Φ a součinové šifry Π. Základní myšlenkou dvojité sítě DN je, že klíče *a, b … z* pro dílčí šifry součinové šifry $\Pi = B_z \bullet \ldots \bullet B_b \bullet B_a$ jsou samy vytvářeny kvalitní blokovou šifrou Φ. Se zvyšováním počtu rund se klíče *(a, b, …)* a *(… y, z)* stávají výpočetně neodlišitelnými nezávislými náhodnými veličinami, neboť odpovídají vztahu otevřeného a šifrového textu blokové šifry Φ. Potom i blokové šifry $(B_a, B_b, \ldots)$ $(\ldots, B_y, B_z)$ se stávají výpočetně neodlišitelnými nezávislými náhodnými blokovými šiframi viz Obr. 7. Promíchání sloupců pole RK s otevřeným textem zajišťuje funkce Π. Tento proces je tím efektivnější, čím více je sloupců v poli rundovních klíčů.



Obr. 6 Rodina funkcí DN

Funkce Φ

Jedná se o funkci pro expanzi rundovních klíčů. Vstupem funkce Φ je šifrovací klíč K a výstupem je pole rundovních klíčů RK. Funkce Φ se skládá ze *sloupcové transformace* a *závěrečné klíčové permutace*. Sloupcová transformace naplňuje pole RK a závěrečná klíčová permutace provádí permutaci bajtů v tomto poli. Sloupcová transformace je systém c nezávislých sloupcových transformací F_t, *t = 0, ..., c-1*, které pracují ve sloupcích pole RK. Každá sloupcová transformace je součinovou blokovou šifrou $F_t = f_{\rho-1,t} \bullet \ldots \bullet f_{2,t} \bullet f_{1,t}$ s délkou bloku r bajtů, přičemž její jednotlivé rundy se označují jako dílčí sloupcové transformace ($f_{i,t}$). Sloupec t pole RK se tak postupně naplňuje výsledky dílčích rund blokové šifry F_t. Každá z $(p-1) \times c$ dílčích sloupcových transformací $f_{i,t}, i = 1, \ldots, \rho-1, t = 0, \ldots, c-1$ je elementární transformací, která se skládá ze substituce na úrovni bajtů - SBox *SubsF*, lineární transformace na úrovni bitů – matice typu MDS o rozměru r x r, a přičtení r-bajtové rundovní konstanty *RConstF*. Každá sloupcová transformace F_t je tak ve skutečnosti blokovou šifrou s konstantním klíčem (rundovní klíče jsou konstanty) viz Obr. 7 Diagram funkce Φ, expanze rundovních klíčů.



Obr. 7 Diagram funkce Φ, expanze rundovních klíčů

Funkce Π

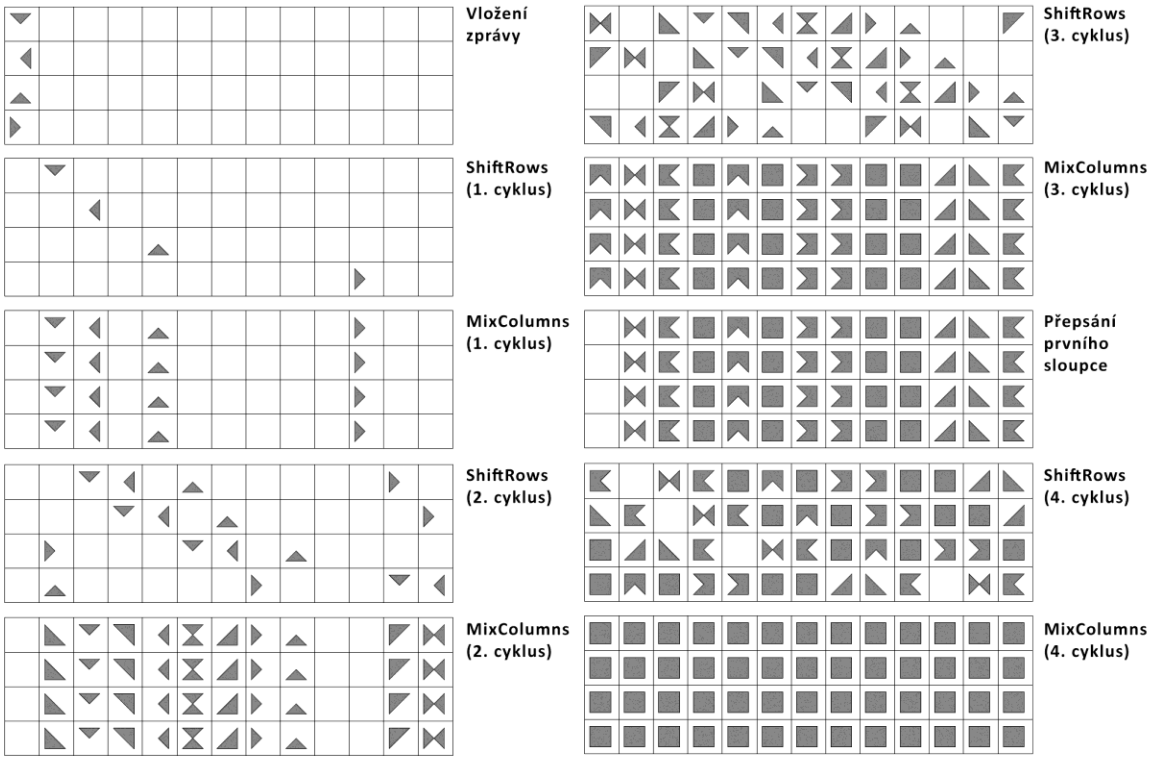
Funkce Π je blokovou šifrou. Otevřený text tvoří *c* bajtů: *indata[0], ..., indata[c – 1]*. Šifrový text tvoří *c* bajtů: *outdata[0], ..., outdata[c – 1]*. Šifrovacím klíčem je pole RK, obsahující $\rho \times r$ malých rundovních klíčů *RK[i][j], i = 0, 1, ..., ρ - 1, j = 0, 1, ..., r - 1*. Primárně je Π součinem $\rho \times r$ elementárních transformací $T1, \Pi_{i=\rho-1,\ldots,0}, \Pi_{j=r-1,\ldots,0}, T1_{i,j}$ kde $T1_{i,j}$ používá malý rundovní klíč *RK[i][j], i = 0, ..., ρ-1, j = 0, ..., r-1*. Výstup z jedné transformace T1 je vstupem do další transformace T1. Vstup do funkce Π je vstupem do první transformace T1, výstup z poslední transformace T1 je výstupem z funkce Π viz Obr. 6 Rodina funkcí DN.

3.2 Funkce Grindahl

Funkce Grindahl byla poprvé představena v březnu 2007 na konferenci Fast Software Encryption 2007 Jejími autory jsou Lars R. Knudsen, Christian Rechberger a Søren S. Thomsen [32]. Tato funkce je založena na jádru blokové šifry Rijndael, avšak využívá jej v poněkud pozměněné formě. Existují v současné době dvě varianty, Grindahl-256 a Grindahl-512, které se liší jen velikostí zásobníku, princip zpracování je stejný. Základem funkce Grindahl-256 je pole 4 × 13 bytů, které představuje stav funkce. S tímto polem se dále manipuluje 4 funkcemi, z nichž dvě, SubBytes a MixColumns, byly převzaty z šifry Rijndael. Další funkcí je ShiftRows , která provádí rotaci řádků o 1, 2, 4 či 10 míst, čtvrtou funkcí je AddConstant, která změní poslední bit posledního bajtu v poli.

Průběh jednoho cyklu začíná vepsáním 4 bajtů ze vstupní zprávy do 4 bajtů prvního sloupce v poli. Následuje operace v pořadí AddConstant, SubBytes, ShiftRows a MixColumns. Ke zprávě se na konec připojí zakončení, které je řešeno tradičně, jako u funkce MD5. Po připojení zakončení se provede 8 prázdných cyklů, které zajistí dostatečnou difuzi všech vstupních bajtů po poli.

Samotná funkce má tu vlastnost, že každý bajt v poli závisí již po 4 cyklech na každém bajtu vstupní zprávy, tím je zajištěna dobrá difuze, viz Obr. 8.



Obr. 8 Vývoj difuze bajtů během 4 cyklů

3.3 RadioGatún

Historie:

RadioGatún [12] je hašovací funkce, která byla představena v roce 2006 a která vychází z konstrukce funkce Panama. Při návrhu funkce RadioGatún se již vycházelo z útoků a slabin, které byly u Panamy známé [13], a tato nová funkce je již vůči nim odolná.

Vlastnosti

Funkce RadioGatún vychází z konstrukce Panamy, obsahuje dva základní prvky, tzv. Belt a Mill (pás a mlýn). Jako operace se používají standardní bitové operace and, xor, not a také cyklické posuny bitů, délka výsledného haše je 256 bitů. Na funkci RadioGatún je také neobvyklé, že velikost každého zpracovávaného slova může být 1 – 64 bitů, lze tedy používat funkci se 64 bitovými slovy na 64 bitových systémech, či funkci se 32 bitovými slovy na 32 bitových systémech. Při volbě velikosti slova v bitech je ale třeba brát v úvahu, že bezpečnost této funkce závisí do jisté míry i na této velikosti zpracovávaných slov.

Bezpečnost

Autoři funkce RadioGatún vycházeli z výsledků kryptoanalýzy funkce Panama [13] – byl snížen počet slov používaných v jednotlivých iteracích z 8 na 3, dále byla přidána zpětná vazba ze stavu na zásobník funkce, která zajistila nelinearitu a odolnost proti útokům použitým na původní funkci Panama.

4 Standardizace hašovacích funkcí

4.1 NESSIE

NESSIE (New European Schemes for Signatures, Integrity and Encryption – nové evropské schema pro podepisování, integritu a šifrování) byl evropský výzkumný projekt, který probíhal v letech 2000 – 2003 a jehož cílem bylo vybrat bezpečné šifrovací primitiva [14]. Tento projekt byl s jistými rozdíly srovnatelný s výběrovým řízením NIST AES či projektem CRYPTREC sponzorovaným japonskou vládou.

Hlavním cílem projektu bylo najít skupinu silných kryptografických primitiv, která bude vybrána pomocí veřejného výběrového procesu. Tento projekt měl také navazovat na v té době končící výběrové řízení AES, jehož cílem bylo vybrat bezpečnou blokovou šifru. Samotný projekt NESSIE vybíral kandidáty hned v několika oblastech – blokové šifry, proudové šifry, hašovací funkce, MAC algoritmy, schémata pro digitální podepisování dat a schémata šifrování s veřejným klíčem. Cílem projektu bylo také vyvinout i způsob odpovídajícího testování a hodnocení těchto funkcí.

Samotný projekt byl zahájen v březnu 2000 výzvou k zasílání návrhů šifrovacích primitiv a také metod, kterými budou tyto návrhy testovány a hodnoceny. Dále následovala první část ohodnocování a testování těchto funkcí, včetně porovnávání rychlostí kritických částí jednotlivých algoritmů v jednotném testovacím softwarovém prostředí. Na základě těchto testů byli v druhé fázi vybráni finalisti, kteří byli podrobeni detailnějšímu zkoumání. Dále proběhlo i zátěžové testování za účelem zjistit, jak si jednotlivé šifrovací systémy vedou v softwarové či hardwarové implementaci, včetně použití v systémech jako jsou čipové Smart karty.

V průběhu celé soutěže se sešlo 42 kandidátů a počátkem roku 2003 bylo vybráno 12 z nich. Dále bylo také „vybráno“ i 5 dalších algoritmů, které sice nebyly zaslané jako kandidáti, ale byly již všeobecně uznávané za bezpečné. Takto vybrané algoritmy byly představeny s vyjádřením, že „u těchto algoritmů nebyly nalezeny žádné slabiny“.

Z hašovacích funkcí, které byly přihlášeny do projektu NESSIE, byly doporučené funkce:

SHA-256, SHA-384, SHA-512 a Whirlpool.

4.2 CRYPTREC

CRYPTREC (CRYPTography Research and Evaluation Committees - komise pro kryptografický výzkum a hodnocení) je zkratkou pro projekt, který byl zorganizován Japonskou vládou, vznikl v dubnu 2000 [15], a jehož předmětem je vyhodnocovat a průběžně sledovat bezpečnost šifrovacích primitiv a dále také vytvořit odpovídající hodnotící kriteria jejich posuzování. Cílem projektu je nalézt a průběžně aktualizovat skupinu bezpečných kryptografických primitiv určených pro vládní a průmyslové použití.

Průběžné výsledky o výzkumu byly vydány v letech 2001, 2002 a 2003, následovala první verze doporučení vydaná v Srpnu 2003. Dále byly také každý rok vydávané aktualizace těchto zpráv v závislosti na stavu výzkumu jednotlivých funkcí.

Z hašovacích funkcí, které byly přihlášeny do projektu CRYPTREC byly doporučené funkce:

RIPEMD-160 , SHA-1, SHA-256, SHA-384 a SHA-512

4.3 AHS

Na rump session konference CRYPTO 2004, která proběhla v srpnu 2004, představil tým Dr. Wangové do té doby unikátní výsledky v kryptoanalýze moderních hašovacích funkcí. Byly zveřejněny kolizní páry a útok pro funkce MD4, MD5, SHA-0, HAVAL(128,3) a RIPEMD. Později také tento tým představil útok na funkci SHA-1, který byl efektivnější než přístup hrubou silou. Tato velká událost vlastně znamenala, že mnoho dosud používaných hašovacích funkcí, které byly považované za bezpečné, není tak bezpečných, jak se předpokládalo.

Po této zprávě přibývalo ten samý rok útoků vůči dalším funkcím, zejména se stále zlepšovaly postupy pro hledání kolizí u funkce MD5. V současné době (polovina roku 2007), je možné najít kolizní pár u funkce MD5 za pár minut i na běžném domácím počítači. Již není potřeba žádných superpočítačů a týdnů počítání. Dále také pokračuje kryptoanalýza druhé v současnosti nejpoužívanější funkce SHA-1. Možnosti nalezení kolizní zprávy u této funkce je v občasné době na hranici našich výpočetních možností – je třeba kolem $\mathbf{2^{63}}$ výpočtu bloku funkce. Teoreticky by bylo možné najít kolizi při použití distribuované výpočetní síly počítačů. Co je ale důležitější, útoky proti hašovacím funkcím se nikdy nehorší, jen se lepší. Právě poslední zmiňovaný útok proti SHA-1 byl představen před více, než rokem a od té doby na jeho zlepšení stále pracuje, dá se tedy předpokládat, že na představení nového, rychlejšího útoku proti SHA-1 nebudeme muset dlouho čekat. Dále také, s ohledem na Moorevy zákony, lze předpokládat, že šance na nalezení kolize i se složitostí $\mathbf{2^{63}}$ bude rok od roku vyšší.

Je jasné, že tato situace začne být již brzy kritická, v současné době nám již nezbylo mnoho funkcí, u kterých dosud nebyl publikován žádný znatelný útok a které lze považovat za zcela bezpečné. Za bezpečné lze v současné době považovat funkce rodiny SHA-2, nové funkce RIPEMD a dále Whirlpool.

Následkem uvedení útoků na funkci SHA-1 v únoru roku 2005, uspořádal NIST koncem roku 2005 kryptografický workshop, jehož cílem bylo rozhodnout, jak vyřešit stávající situaci u hašovacích funkcí. Jeho výsledkem bylo, mimo doporučení přechodu z funkce SHA-1 na SHA-2, také rozhodnutí, že by se měla v dlouhodobém měřítku nalézt i jedna či více hašovacích funkcí podobným způsobem, jakým probíhal výběr šifrovacího standardu AES (Advanced Encryption Standard).

Na kryptografickém workshopu v srpnu 2006 byl představen prozatímní časový plán průběhu výběru takové funkce. Během tohoto workshopu bylo znát, že ač naše znalosti o teorii a o bezpečnosti hašovacích funkcí mají rezervy a nejsou na takové úrovní, jako byly naše znalosti o problematice blokových šifer v době, když NIST vyhlásil soutěž AES, bude stále lepší vyhlásit podobnou soutěž i pro hašovací funkci, než jen zůstat u stálého vylepšování současných metod se znalostmi, které máme.

Předběžná časová osa, která byla navržena pro průběh soutěže, počítala především s následujícími faktory:

- Vývojový proces by měl být podobný tomu, který probíhal u AES.
- Podobně jako u soutěže AES, jednotlivé workshopy by se měly uskutečnit v době jiných workshopů či konferencí, za účelem zajištění co největší účasti zúčastněných stran.
- Standard FIPS 180-2, který obsahuje v současné době funkce SHA-1 a SHA-2, bude podstoupen v roce 2007 a 2012 revizi, bylo by tedy vhodné ukončit výběrový proces nové hašovací funkce tak, aby se mohly jeho výsledky promítnout do aktualizované verze tohoto standardu v roce 2012.

Navržená časová osa soutěže AHS

Následující časová je pouze orientační. Při jejím návrhu se bral v úvahu především časový průběh vývoje u AES.

Rok 2006 – Na druhém kryptografickém workshopu byl zhodnocen současný stav a dohodnut následující časový plán.

Rok 2007 – V únoru byly představeny předběžné minimální akceptační požadavky a hodnotící kriteria pro kandidátské hašovací funkce. Během tohoto roku také bude možnost zasílat komentáře k těmto minimálním požadavkům, přičemž koncem roku bude představena finální verze minimálních akceptačních požadavků a hodnotících kriterií pro kandidátské hašovací funkce. Bude zahájeno přijímání kandidátských návrhů.

Rok 2008 – Ve třetím kvartálu roku 2008 skončí možnost podávat návrhy funkcí do soutěže. Tyto funkce potom budou podrobeny přezkoumání a budou vybrané ty, které splňují základní kladené požadavky. To proběhne na první konferenci během tohoto roku. Dále bude zahájeno období komentářů k vybraným funkcím.

Rok 2009 – V čtvrtém kvartálu bude ukončeno období komentářů k vybraným funkcím. Toto období může být případně rozšířeno v závislosti na množství a kvalitě doručených komentářů. Ke konci roku se uskuteční druhá konference, jejímž cílem bude ohodnotit analýzu vybraných funkcí.

Rok 2010 – V prvém kvartálu roku budou vybráni a představeni finalisti soutěže. Ve druhém kvartálu budou moci autoři finálových návrhů předložit konečné úpravy ke svým funkcím.

Rok 2011 – Začne finálové kolo výběru. Ve druhém kvartálu bude ukončeno období komentářů k finálovým funkcím. Uskuteční se poslední konference, na které budou diskutovány finálové funkce, a bude vybrána vítězná funkce.

Rok 2012 – Bude zveřejněna oficiální specifikace vítězné funkce

5 Závěr

V této práci jsme si ukázali, na jakých konstrukcích jsou založené tradiční hašovací funkce z počátku devadesátých let, jaké měly a mají slabiny a také co tyto slabiny znamenaly pro obecnou bezpečnost těchto funkcí. Dále jsme se také věnovali několika návrhům nových funkcí a právě probíhající soutěži AHS, jejímž cílem je vybrat novou, bezpečnou hašovací funkci.

Jako výsledek této práce bych doporučil v prvé řadě používat jen ty hašovací funkce, u kterých dosud nebyly objeveny žádné bezpečnostní slabiny a které byly též podrobené důkladné kryptoanalýze. Mezi tyto funkce určitě patří zejména rodina funkcí SHA-2 a dále nové funkce z rodiny RIPEMD-160.

Jednu z nejpoužívanějších funkcí současné doby – MD5, u které již dokážeme hledat kolizní páry zpráv během chvilky, již nelze považovat za dostatečně bezpečnou pro určité způsoby použití, kterými jsou například digitální otisky certifikátů či ověřování identity souborů. Stále je však možné používat i tuto funkci na místech, u kterých nám současné problémy nevadí. Tím je např použití v pseudonáhodných funkcích či v pseudonáhodných generátorech čísel. V obecném pojetí bezpečnosti ovšem bude lepší ukončení používání této, a i dalších „prolomených“ funkcí, a nahradit je funkcemi novými. U druhé, taktéž velmi rozšířené funkce SHA-1 nebyl dosud nalezen žádný kolizní pár zpráv, ale s ohledem na probíhající podrobnou kryptoanalýzu této funkce za poslední roky lze předpokládat, že již brzy budeme schopni hledat kolize i pro tuto funkci v rozumném čase.

Práce se dále věnovala i popisu návrhů několika nových hašovacích funkcí, u kterých lze očekávat, že se zúčastní soutěže AHS, byť třeba s jistými změnami ve specifikaci. Tyto popisované funkce používají pokročilé konstrukce, které mají s původní „populární“ konstrukcí založenou na MD4 již jen málo společného.

Do budoucna bude určitě zajímavé a přínosné sledovat průběh soutěže AHS a s tím i související analýzy přihlašovaných funkcí. Lze očekávat, že právě tato soutěž dopomůže najít dosud neobjevené slabiny ve stávajících funkcích a podrobí nově vznikající funkce důsledném rozboru.

6 Bibliografie

1. RFC 1319 – specifikace funkce MD2: http://www.ietf.org/rfc/rfc1319.txt
2. RFC 1320 – specifikace funkce MD4: http://www.ietf.org/rfc/rfc1320.txt
3. RFC 1321 – specifikace funkce MD5: http://www.ietf.org/rfc/rfc1321.txt
4. FIPS 180-1 – specifikace funkce SHA-1: http://www.itl.nist.gov/fipspubs/fip180-1.htm
5. FIPS 180-2 – specifikace funkce SHA-2: http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf
6. NESSIE standard ISO/IEC 10118-3: http://www.incits.org/ref-docs/FDIS_10118-3.pdf
7. "RIPE Integrity Primitives: Final Report of RACE Integriy Primitives Evaluation (R1040)", *Research and Development in Advanced Communication Technologies in Europe, RACE,* June 1992.
8. RIPEMD-160 – specifikace: http://www.esat.kuleuven.be/~bosselae/ripemd160.html
9. HAVAL – specifikace: http://labs.calyptix.com/haval.php
10. Tiger – specifikace: http://www.cs.technion.ac.il/~biham/Reports/Tiger/
11. Whirlpool – specifikace: http://paginas.terra.com.br/informatica/paulobarreto/WhirlpoolPage.html
12. RadioGatún – specifikace: http://radiogatun.noekeon.org/
13. J. Daemen, G. V. Assche "Producing Collisions for Panama, Instantaneously" http://radiogatun.noekeon.org/panama/
14. NESSIE projekt: https://www.cosic.esat.kuleuven.be/nessie/
15. CRYPTREC projekt: http://www.ipa.go.jp/security/enc/CRYPTREC/
16. P. Oechslin, "Password Cracking: Rainbow Tables Explained" https://www.isc2.org/cgi-bin/content.cgi?page=738
17. M. Bellare, R. Canetti, H. Krawczyk (1996). "Keying hash functions for message authentication. Advances in Cryptology", *CRYPTO '96, LNCS 1109* (stránky 1-15)
18. E. Biham, A. Shamir (1991). "Differential Cryptanalysis of Snefru, Khafre, REDOC-II, LOKI and Lucifer", *11th Annual International Cryptology Conference on Advances in Cryptology, LNCS 576* (stránky 156-171).
19. B. Boer, A. B. (1991). "An Attack on the Last Two Rounds of MD4". *In Advances in Cryptology, Proceedings of CRYPTO '91*, (stránky 194-203).
20. S. Bono, M. Green, A. Stubblefield, A. Juels, A.Rubin, M.Szydlo (2005). "Security Analysis of a Cryptographically-Enabled RFID Device", *Security '05*
21. J. Daemen, C. S. K. Clapp (1998). "Fast Hashing and Stream Encryption with PANAMA", *Fast Software Encryption: 5th International Workshop, FSE'98, Paris, France, LNCS 1372*
22. Dobbertin, H. (1997). "RIPEMD with Two-Round Compress Function is Not Collision-Free". *Journal of Cryptology, Vol 10, Number 1, 1997* , 51/70.
23. Dobbertin, H. (1996). "The Status of MD5 after a Recent Attack". *CryptoBytes 2:2 (1996)* , 1--6.

24. E. Biham, R. C. (2004). "Near-Collisions of SHA-0". *Advances in Cryptology - Crypto'2004, LNCS 3152* (stránky 290--305).
25. F. Chabaud, A. J. (1998). "Differential Collisions in SHA-0". *Advances in Cryptology - Crypto'98, LNCS 1462* (stránky 56--71).
26. J. Kelsey, S. Lucks, (2006). "Collisions and Near-Collisions for Reduced-Round Tiger", *Full version: Fast Software Encryption - FSE'2006, LNCS 4047,* (stránky 111-125).
27. Joux, C. L. (2004). "Collision for the full SHA-0 algorithm". *Rump Session Crypto 2004.*
28. Klíma, V. (2005). "Finding MD5 Collisions -- a Toy For a Notebook". *Cryptology ePrint Archive, Report 2005/075.*
29. Klíma, V. (2006). *"Nový koncept hašovacích funkcí SNMAC s využitím speciální blokové šifry a konstrukcí NMAC/HMAC".*
30. Klíma, V. (2007). *"Rodina speciálních blokových šifer DN a hašovacích funkcí nové generace HDN typu SNMAC".*
31. Klíma, V. (2006). "Tunnels in Hash Functions: MD5 Collisions Within a Minute". *Cryptology ePrint Archive, Report 2006/105.*
32. L. R. Knudsen, C. R. (2007). *"Grindahl – a family of hash functions".*
33. R. C. Merkle, (1990), "A Fast Software One-Way Hash Function", *Journal of Cryptology 3:1 pp 43--58.*
34. Muller, F. (2004). "The MD2 Hash Function Is Not One-Way". *Advances in Cryptology - Asiacrypt'2004, LNCS 3329* (stránky 214-229.).
35. N. Rogier, P. C. (1995). "The compression function of MD2 is not collision free". *Selected Areas in Cryptography -- SAC'95.* Ottawa, Canada.
36. Stevens, M. (2006). "Fast Collision Attack on MD5". *Cryptology ePrint Archive, report 2006/104.*
37. X. Wang, D. F. (2004). "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD". *Cryptology ePrint Archive, Report 2004/199.*
38. X. Wang, H. Y. (2005). *"Efficient Collision Search Attacks on SHA-0".*
39. X. Wang, H. Y. (2005). "How to Break MD5 and Other Hash Functions". *Advances in Cryptology - Eurocrypt'2005, LNCS 3494* (stránky 19-35).
40. X. Wang, X. L. (2005). "Cryptanalysis of the Hash Functions MD4 and RIPEMD". *Advances in Cryptology - Eurocrypt'2005, LNCS 3494* (stránky 1-18).
41. X. Wang, Y. L. (2005). *"Collision Search Attacks on SHA1".*
42. X. Wang, Y. L. (2005). *"Finding Collisions in the Full SHA-1".*

7 Přílohy

Tabulka s přehledem hašovacích funkcí:

Jméno	Specifikace Délka haše	Z roku Velikost bloku	Autoři Útoky
MD2	RFC -1319	1989	Rivest
	128	128	slabiny [34], [35]
MD4	RFC-1320	1990	Rivest
	128	512	**kolize** [19], [37], [40]
MD5	RFC-1321	1991	Rivest
	128	512	**kolize** [23], [37], [39], [28], [36], [31]
SHA-0	FIPS 180	1993	NIST/NSA
	160	512	**kolize** [25], [24], [27], [37], [38]
SHA-1	FIPS 180-1	1995	NIST/NSA
	160	512	slabiny [37], [41], [42]
SHA-224	FIPS 180-2	2004	NIST/NSA
	224	512	--
SHA-256	FIPS 180-2	2001	NIST/NSA
	256	512	--
SHA-384	FIPS 180-2	2001	NIST/NSA
	384	1024	--
SHA-512	FIPS 180-2	2001	NIST/NSA
	512	1024	--
RIPEMD	[7]	1990	The RIPE Consortium
	128	512	**kolize** [22], [37]
RIPEMD	[8]	1996	Dobbertin, Bosselaers, Preneel
128, 160, 256, 320	128, 160, 256, 320	512	--
HAVAL	[9]	1992	Zheng, Pieprzyk, Seberry
	128, 160, 192, 224, 256	1024	**kolize** [37]
Snefru	[33]	1990	Merkle
	128, 256	512	slabiny [18]
Tiger	[10]	1996	Anderson, Biham
	128, 160, 192	512	slabiny [26]
Whirlpool	[11]	2000	Barreto, Rijmen
	512	512	--
Panama	[21]	1998	Daemen, Clapp
	256	256	**kolize** [13]
HDN	[30]	2007	Klíma
	512	960	--
Grindahl	[32]	2007	Knudsen, Rechberger, Thomsen
	256, 512	32, 64	--
RadioGatún	[12]	2006	Bertoni, Daemen, Peeters, van Assche
	256	různá	--

Obsah programové přílohy

Programová knihovna SmartHash

Součástí této práce je i názorná implementace popisovaných hašovacích funkcí v jazyce C# pod platformou Microsoft .NET 2.0. Programátorská dokumentace k těmto programům je taktéž součástí přílohy a nachází se v adresáři *doc*.

Program pro vizualizaci hašovacích funkcí VisualHash

Druhou součástí práce je program VisualHash, postavený na platformě .NET 3.0, pro vizualizaci průběhu hašovacích funkcí. Tento program využívá upravených zdrojových kódů z knihovny SmartHash, které obsahují navíc procedury pro ukládání průběžného stavu funkce při hašování za účelem výpisu na obrazovku.

Program obsahuje implementace funkcí MD4, MD5, SHA-0 a SHA-1. Další knihovny je možno jednoduše přidat při zachování struktury programu.

Program VisualHash zobrazuje průběh dvou funkcí paralelně, přičemž ve třetím sloupci zobrazuje jejich modulární rozdíl. Vstup u obou funkcí je možno načíst ze souboru buď binárního, nebo textového, kde jsou hodnoty uležené jako hexadecimální posloupnost znaků. V případě textového souboru lze také určit, zda jsou načítaná data ve formátu Little Endian či Big Endian. Dále je možné si zvolit, kolik bloků výpočtu se bude zobrazovat.

Jako příloha programu VisualHash je i sada referenčních kolizních párů zpráv pro algoritmy MD4 (kolize nastává již v samotné kompresní funkci), MD5 (kolize se dělí do dvou bloků, přičemž po těchto dvou blocích je již kontext obou funkcí totožný) a SHA-0 (kolize probíhá ve čtyřech blocích). Zdrojem všech tří párů je práce [37].

Program pro porovnání rychlosti Hash Benchmark

Třetí programovou přílohou je jednoduchý program, postavený na platformě .NET 3.0, pro testování rychlosti výpočtu vybraných hašovacích funkcí. Program umožňuje zvolit si funkce, které budou testované, dále velikost vstupního bloku dat.

Program po proběhnutí testu zobrazuje rychlost jednotlivých funkcí v *MB/s* nebo doby, kterou potřebovaly dané funkce k provedení výpočtu v *ms*. Je třeba připomenout, že dané implementace funkcí v jazyce C# nejsou ideální a v mnoha případech ani optimalizované pro rychlost. Výsledky testu jsou tedy jen přibližné, ale i tak ukazují to, jak je která funkce řádově rychlá.