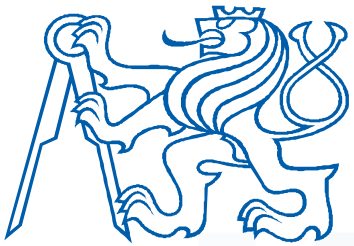
České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra telekomunikační techniky

Hashovací funkce

Ing. Tomáš Vaněk, Ph.D.



Obsah

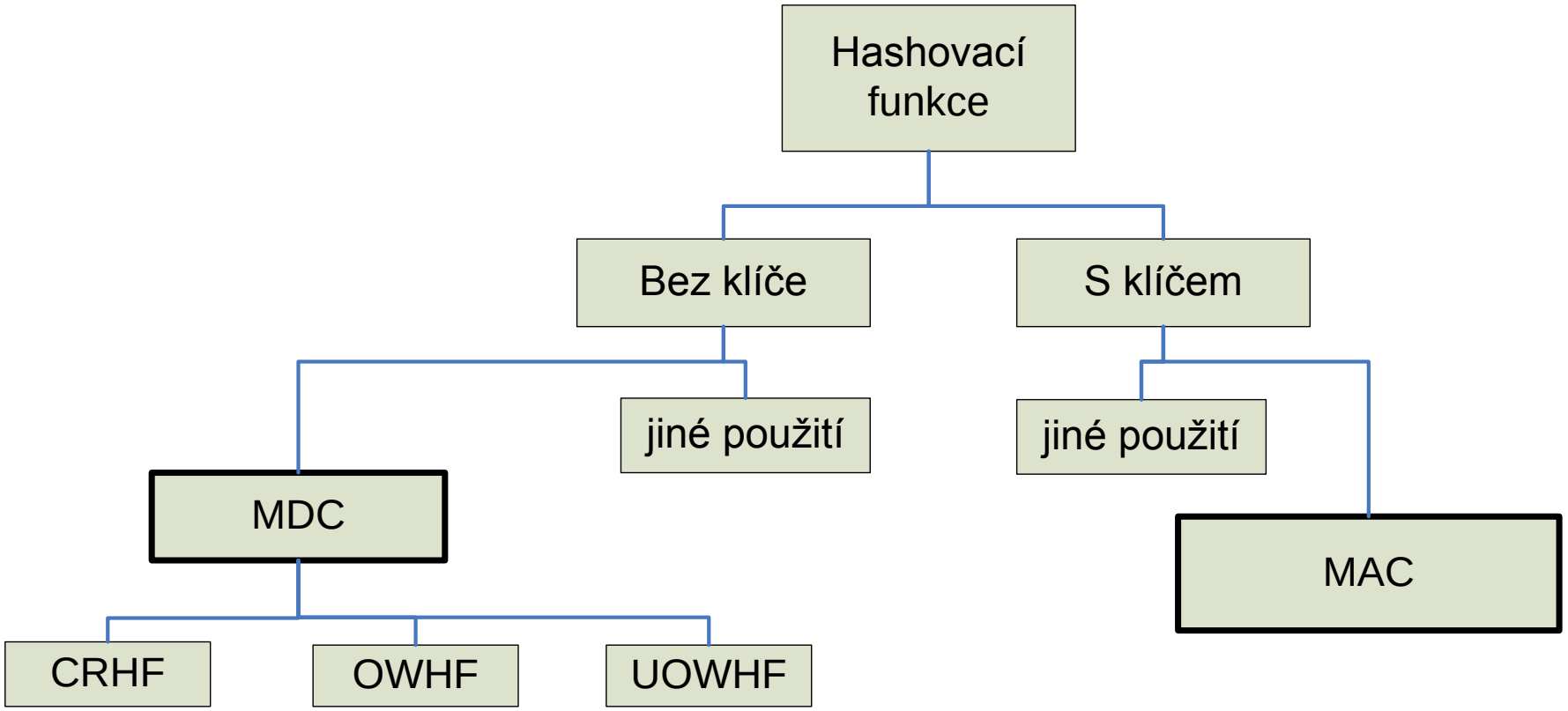
- MD-5
- SHA-1,2
- RIPEMD-160
- Whirlpool
- útoky na hashovací funkce
- rozdíl MAC a HMAC
- SHA-3

Hashovací funkce

- matematická funkce
- vstup: posloupnost téměř libovolné konečné délky (text, hudba, obrázky, video,…)
- výstup: posloupnost konstantní délky
  - typicky stovky bitů
  - haš, hašé, hash, hashový kód, otisk zprávy

$$h=H(M)$$

Rozdělení hashovacích funkcí



Rozdělení hashovacích funkcí

Obecně lze hashovací funkce rozdělit na dvě třídy:

- hashovací funkce bez klíče – mají jediný vstupní parametr – zprávu *M*
- hashovací funkce s klíčem – dva nezávislé vstupy – zprávu *M* a klíč *K*

Hashovací funkce je funkce *h*, která musí splňovat **minimálně** tyto požadavky:

1. **komprese** — funkce *H* mapuje libovolně konečně velký vstup *x*, na výstup *h(x)* pevné délky *n*
2. **jednoduchost výpočtu** - pro danou hashovací funkci *H* a vstup *x*, je jednoduché spočítat *h(x)*

Jiné hashovací funkce

Jiné bez klíče

- cyklické kontrolní součty (např. CRC-32)
- realizuje kompresi + je jednoduše vypočitatelné

Jiné s klíčem

- identifikace pomocí challenge-response mechanismu nebo potvrzování při výměnách klíčů
- odpověď (response) obsahuje nějakým způsobem zpracované heslo i výzvu (challenge)

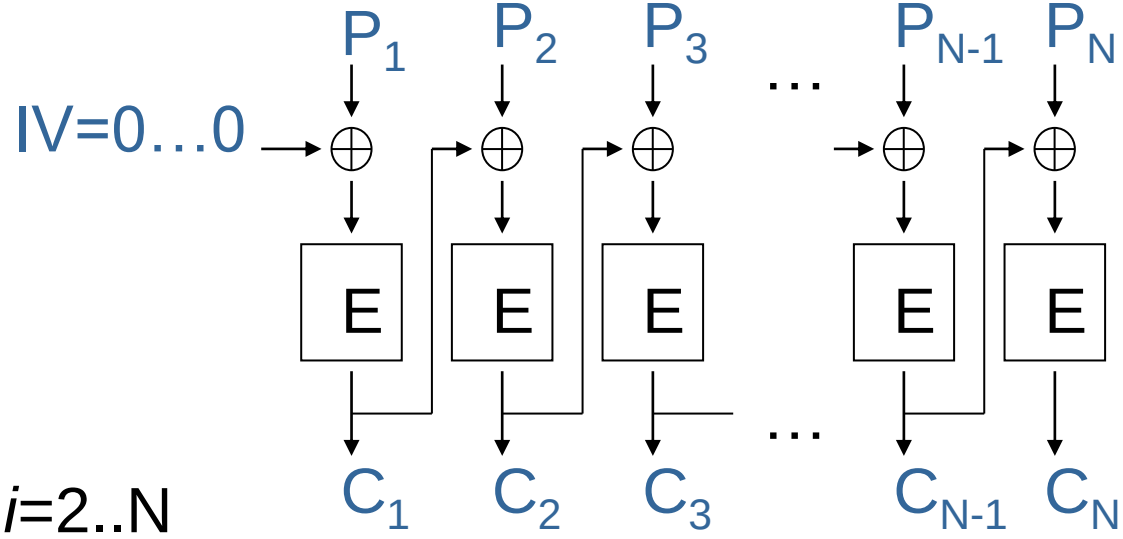
MAC - Message Authentication Code

- skupina funkcí parametrizovaných (tajným) klíčem *k*
- zajišťuje integritu zprávy a zároveň zaručuje i zdroj zprávy (autentizace)
- MAC musí mít následující vlastnosti:

1. **komprese** — funkce h_k mapuje libovolně velký konečný vstup *x*, na výstup $h_k(x)$ pevné délky *n*
2. **jednoduchost výpočtu** - pro danou hashovací funkci *h*, klíč *k* a vstup *x*, je jednoduché spočítat $h(x)$. Výsledek se nazývá MAC nebo MAC-hodnota
3. **výpočetní odolnost** – pro jeden nebo více párů $(x_i; h_k(x_i))$ je výpočetně nemožné spočítat jinou dvojici $(x; h_k(x))$ pro $x \neq x_i$ (zahrnující i možnost $h_k(x)=h_k(x_i)$ pro některé *i*) – tzn. z $h_k(x)$ není možné získat k (označuje se jako „**key non-recovery**“)

Autentizace zprávy pomocí CMAC (bez utajení)

- CBC-MAC; DES–CBC, IV= 000…000, tajný klíč K
- ze zprávy { m_1, …, m_n } se vygeneruje ŠT { c_1, …, c_n }
- Poslední blok c_n se prohlásí za MAC, ostatní se zahodí
- Alice pošle Bobovi { m_1, …, m_n, c_n }
- Bob zrekonstruuje z přijatých bloků m1, …, m_n hodnotu c_n
- pokud nebyla zpráva při přenosu změněna, pak se musí $c_n = c_n$



$C_1 = E(P_1 \oplus IV)$

$C_i = E(P_i \oplus C_{i-1})$ pro i=2..N

MDC – Modification Detection Code

- někdy též MIC (Message Integrity Check)
- účelem je poskytnout reprezentativní vzorek (hash) zprávy, který budu vyhovovat dalším dodatečným podmínkám - viz. dále
- zajišťuje integritu dat

OWHF – One-way Hash Function

- je obtížné najít vstup jež bude mít nějaký předem daný otisk (hash)

CRHF – Collision Resistant Hash Function

- je obtížné nalezení dvou různých vstupních textů se stejným otiskem (hashem)

UOWHF – Universal One-way Hash Function

OWHF, CRHF, UOWHF – kryptografické hashovací funkce

Základní vlastnosti dobré hashovací funkce

1. Odolnost vůči získání předlohy (Preimage resistance)
   Pro všechny výstupy z hashovací funkce je výpočetně nemožné získat vstup, kterému odpovídá daný otisk. Neboli nelze najít předobraz *x'* takový, že h(*x'*)=y, když známe pouze *h* a *y* a neznáme *x'* .

Základní vlastnosti dobré hashovací funkce

2. Odolnost vůči získání jiné předlohy (2^{nd} preimage resistance) – je výpočetně nemožné nalézt ke vstupu *x* druhý vstup x' takový, že x'≠ x a zároveň h(*x*')=h(*x*).

3. Odolnost vůči nalezení kolize (Collision resistance) – je výpočetně nemožné systematicky vytvářet dva libovolné různé vstupní texty *x* a *x'* pro které platí, že x' ≠ x a zároveň h(x)=h(y)

Slova „jednoduché“ a „výpočetně nemožné“ v předchozích definicích nemají přesnou a striktní definici.

Narozeninový paradox

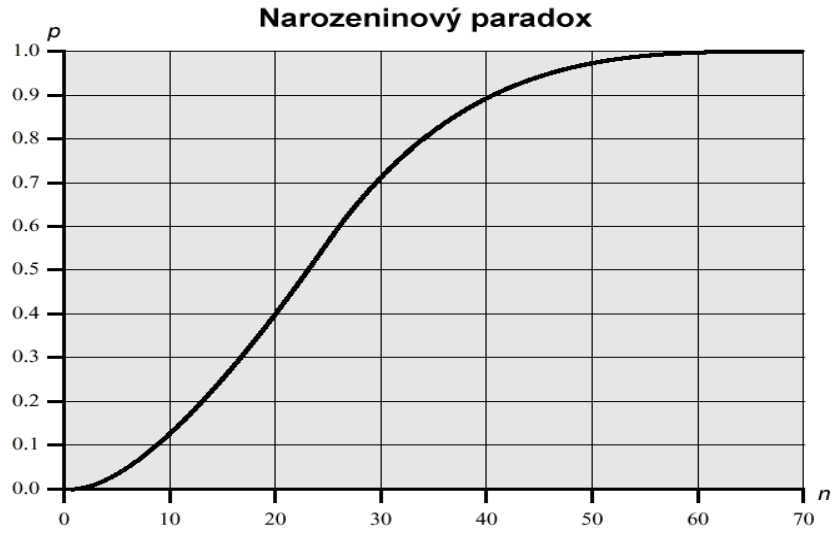
Předpoklad: Nechť M je množina o *n* různých prvcích

Vybírejme náhodně *k* prvků z množiny M.
Po každé volbě zaznamenejme výsledek a vraťme prvek zpět.

Tvrzení: Po výběru $k=n^{1/2}$ prvků je pravděpodobnost výběru již jednou vybraného prvku cca 50%.

P(365,23)=0,507

P(365,30)=0,706



Terminologie - alternativní pojmy

odolnost vůči získání předlohy / preimage resistance

=

jednocestná / jednosměrná / one-way

odolnost vůči získání jiné předlohy / 2nd preimage resistance

=

slabá odolnost proti kolizím / odolnost proti kolizím 2. řádu

odolnost vůči nalezení kolize / collision resistance

=

silná odolnost proti kolizím / odolnost proti kolizím 1. řádu

Jednocestná funkce (One-Way Function)

- funkce *f* splňují tyto vlastnosti:
  1) pro všechna *x* je f(x) „snadno“ spočitatelná
  2) z pouhé znalosti *f(x)* je „výpočetně nemožné“ najít *x*

Rozdíly OWF a OWHF:

- Není požadavek na 2^{nd}-preimage resistance
- Není požadavek na kompresi vstupu
- Není požadavek na konečnou délku vstupní posloupnosti

Pozn.: Dnes není známa žádná dokazatelně jednocestná funkce. Existují pouze kandidáti OWF v tom smyslu, že jejich vyřešení vede na NP-úplný problém.
Důkaz existence OWF by byl zároveň důkazem, že P ≠ NP.

MDC – OWHF (One-way Hash Function)

Podmínky:

1. komprese
2. jednoduchost výpočtu
3. odolnost vůči získání předlohy
4. odolnost vůči získání jiné předlohy

Lze nalézt kolizi po provedení $2^{n/2}$ operací , kde n je délka výstupu (paradox společných narozenin)

MDC – CRHF (Collision Resistant Hash Function)

Podmínky:

1. komprese
2. jednoduchost výpočtu
3. odolnost vůči získání předlohy
4. odolnost vůči získání jiné předlohy
5. odolnost vůči nalezení kolize

MDC – UOWHF (Universal One-way Hash Function)

Podmínky:

1. **komprese**
2. **jednoduchost výpočtu**
3. **odolnost vůči získání předlohy**
4. **odolnost vůči získání jiné předlohy**
5. **univerzálně jednosměrná** - pro vstupní text *x* a náhodný klíč *k* je pro útočníka obtížné najít druhý vstupní text *x'* ($x \neq x'$) takový, že $H_k(x)=H_k(x')$

- Náhodný klíč *k* je index, kterým je zvolena konkrétní hashovací funkce H_k z rodiny hashovacích funkcí H
- požadavek na „univerzální jednosměrnost“ je výrazně slabší než požadavek na „odolnost proti kolizím“
- neefektivní v porovnání s CRHF
- nestandardizované -> nepoužívá se
- algoritmus ACE-sign od IBM
  http://www.zurich.ibm.com/security/ace/sig.pdf

Další obvyklé požadavky na HF zahrnují:

- nekorelovatelnost vstupních a výstupních bitů
  - znemožní statistickou kryptoanalýzu

- odolnost vůči skoro-kolizím (*Near-collision resistance*)
  - je obtížné nalézt *x* a *y* taková, že x≠y a zároveň *H*(*x*) a *H*(*y*) se liší jen v malém počtu bitů.

- lokální odolnost vůči získání předlohy
  - je obtížné najít i jen část vstupu *x* ze znalosti *H*(*x*)

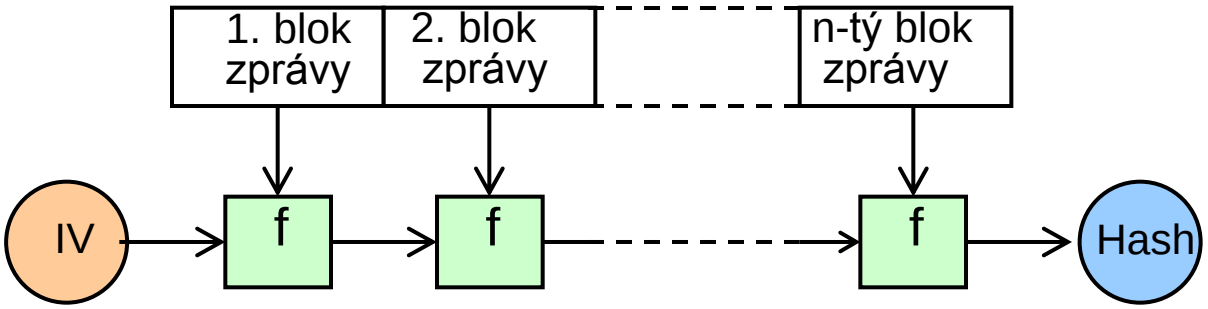
Použití hashovacích funkcí

- **Digitální / elektronický podpis** – zajištění integrity
- **PKI** - integrita X.509 certifikátů a seznamů CRL
- **Časové značky –** integrita časové značky
- **Kerberos -** generování klíčů
- **IEEE 802.1X EAP :** EAP-FAST, EAP-TLS, EAP-TTLS… používají TLS protokol, který používá hashovací funkce
- **APOP -** autentizace pomocí MD5
- **RADIUS -** integrita dat
- **IPsec (IKE, AH, ESP)** – integrita zpráv, generování pseudonáhodných posloupností.
- **SSL/TLS -** handshake protokol používá hashovací funkce kvůli tvorbě HMAC a při generování klíčů a IV
- **SSH** – HMAC a integrita přenášených dat
- **S/MIME** - použití hashovacích funkcí v digitálním podpise

Použití hashovacích funkcí - obecně

- zajištění integrity dat
- digitální podpis
- ukládání hesel
- porovnání obsahu dvou kopií dat
- generování pseudonáhodných posloupností (PRNG)

Merkle-Damgårdova struktura hashovací funkce



f – kompresní funkce

IV – inicializační vektor

Hash – výsledný otisk zprávy

- bloky *f* provádějí kompresní funkci
- toto schéma využívá naprostá **většina** moderních hashovacích funkcí – MD-5, SHA-1, RIPEMD-160
- vstup musí být doplněn na celistvý násobek délky bloků
- musí být jednoznačně určitelné kolik se doplnilo (jinak by jednoduše vznikala řada kolizí)
- Merkle-Damgårdovo zesílení (doplnění výplně posledního bloku o délku zprávy)

Merkle-Damgårdova metoda

- v roce 1989 ji nezávisle na sobě navrhli Merkl a Damgård
- nazývá se Merklova meta metoda nebo Merkle-Damgårdova
- Je matematicky dokázáno, že pokud je kompresní funkce bezkolizní, bude bezkolizní i iterovaná hashovací funkce realizovaná pomocí Merkle-Damgårdovy metody* -> stačí najít vhodnou funkci *f*

** Toto neplatí obecně!!!*

Merkle-Damgårdova meta-metoda pro tvorbu HF

Vstup: funkce *f* odolná vůči kolizím.
Výstup: iterovaná CRHF h odolná vůči kolizím

Vstup délky *x* se rozdělí na *n* bloků x_1 ,... ,x_n o délce *m* bitů. Poslední blok x_n se zleva doplní nulami na délku *m* bitů. Volitelně lze provést Merklovo-Dammgårdovao posílení.

Výpočet s-bitového výstupu zprávy *x:*

$h(x) = H_{t+1} = f(H_t \,||\, x_{t+1});$

$H_0 = 0$

$H_i = f(H_{i-1} \,||\, x_i).$

Požadavky na kompresní fci

Realizace kompresní funkce v ideálním světě

- **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.

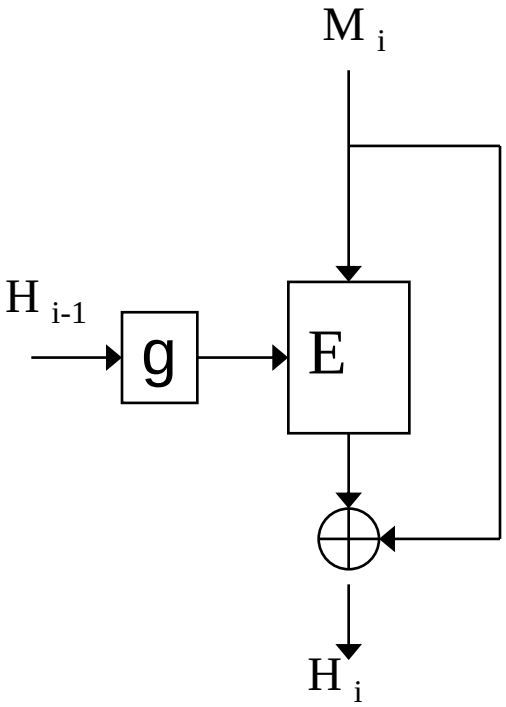
Realizace kompresní funkce v reálném světě

- vhodnou blokovou šifrou
- nelineární funkcí

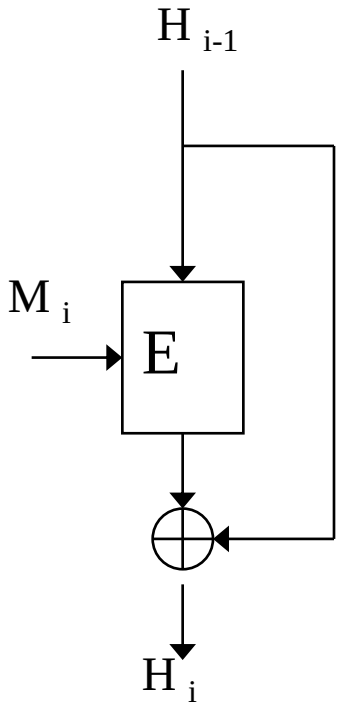
Cílem návrhu kompresní funkce je, aby se taková funkce chovala podobně jako náhodné orákulum.

Realizace kompresní funkce hashovací funkce pomocí blokové šifry

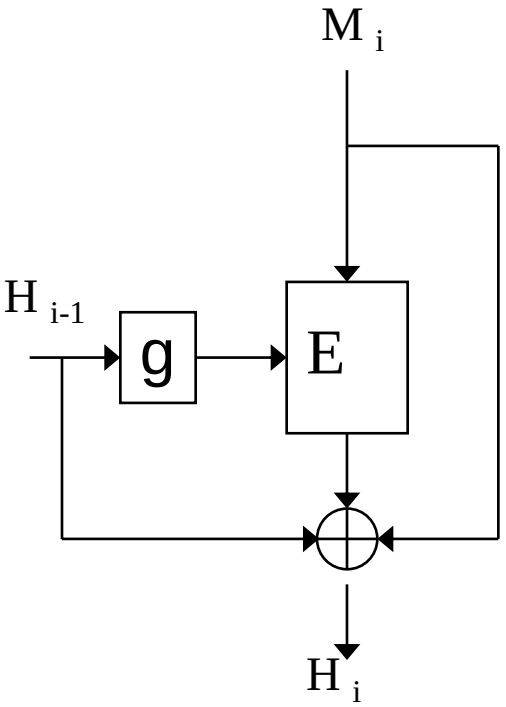
Matyas - Meyer - Oseas



Davies - Mayer



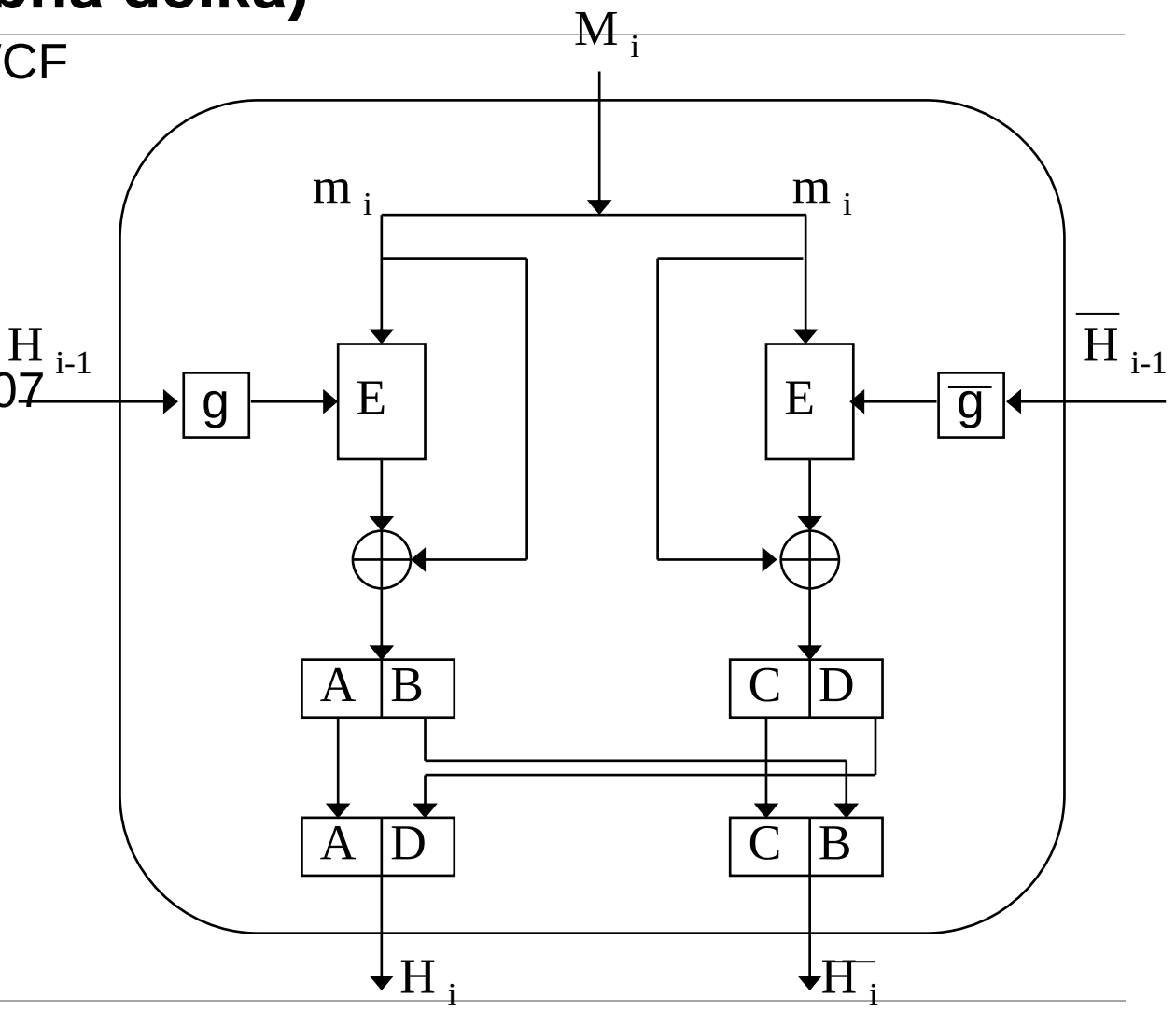
Miyaguchi - Preneel



g – funkce, která upraví hash z předchozího kroku tak, aby ho bylo možné použít jako heslo pro blokovou šifru E (doplnění, vhodná konverze)

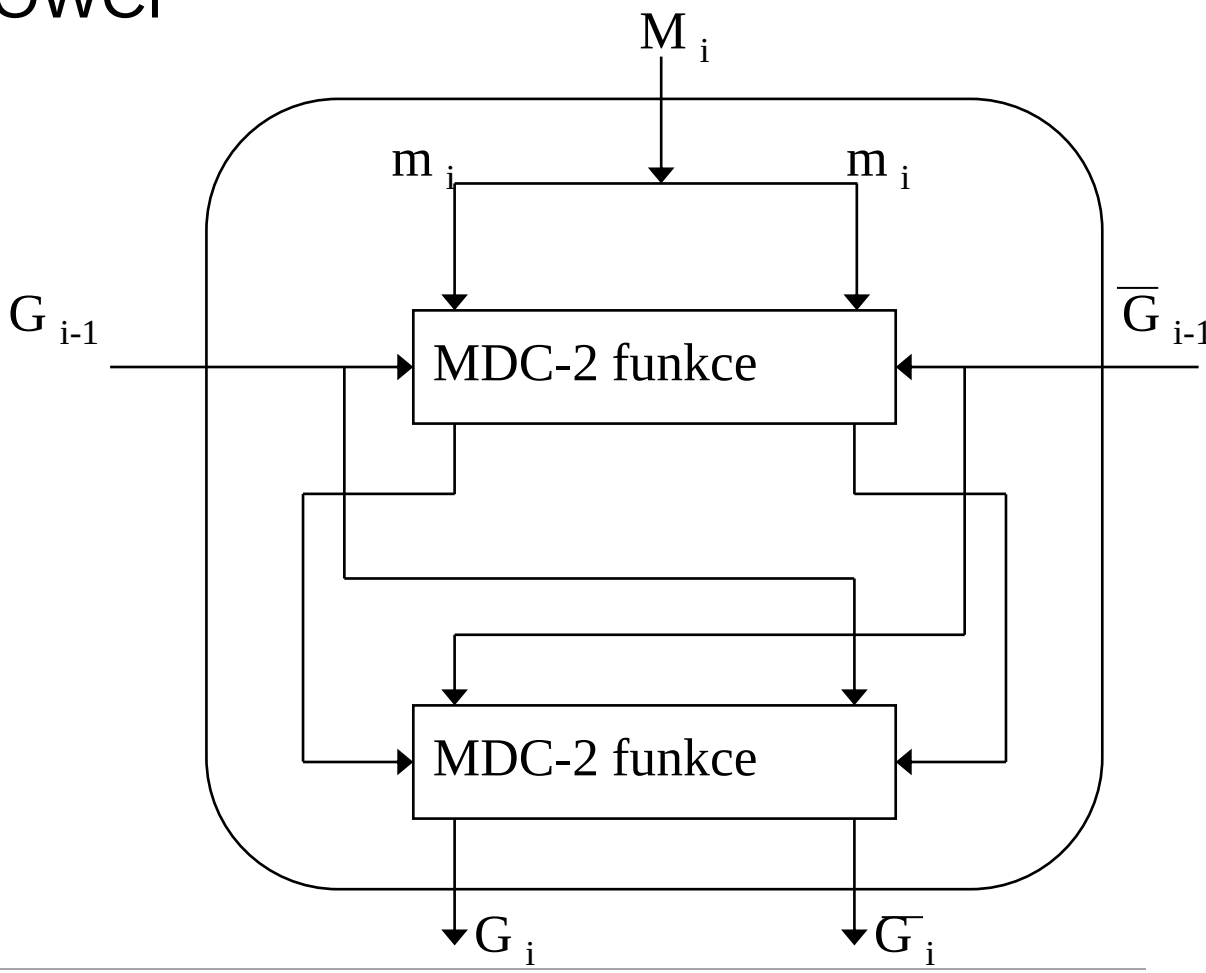
MDC-2 (dvojnásobná délka)

- single-block-length OWCF
  - vstup 128b
  - výstup 128b
- blok „E“ – DES
- patentováno IBM
- patent vypršel 28.8. 2007
- lze volně používat
- nepoužívá se



MDC-4 (dvojnásobná délka)

- double-block-length OWCF
- vstup délky *n*
- výstup délky *2n*



Hirose

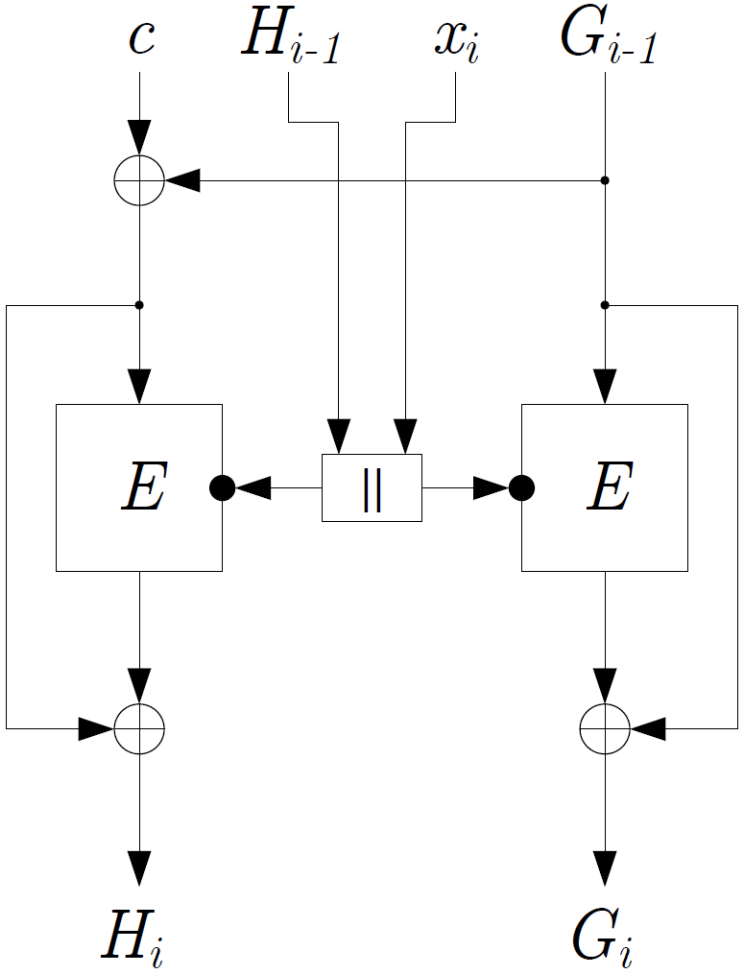
- 2006 - double-block-length OWCF
- vstup délky *n*
- výstup délky *2n*
- bloková šifra + permutace
- klíč *k* > blok *n*
- AES-192, AES-256 (s blokem 128b)

$G_i = E_{Ki}(G_{i-1}) \oplus G_{i-1}$

$H_i = E_{Ki}(p(G_{i-1})) \oplus p(G_{i-1})$.

$p(G_{i-1})$ je libovolná permutace s nějakým celým číslem na *n*-bitovém řetězci, typicky definované: $p(x) = x \oplus c$ pro libovolnou nenulovou konstantu *c*.

Šifrování je implementováno ve standardní Davies-Meyerově zapojení.



Hashovací funkce - přehled

MD2 – kompromitovaná, nepoužívá se

MD4 – kompromitovaná, nepoužívá se

MD5 – oblíbená, ale kompromitovaná funkce.

Od srpna 2004 je veřejně znám postup nalezení kolizí a to i pro málo odlišná vstupní data.

RIPEMD-128 – kompromitovaná, nepoužívá se

RIPEMD-160 – může být kompromitována

WHIRLPOOL – považuje se za bezpečnou

TIGER – považuje se za bezpečnou

GRINDAHL – mladá (03/2007), jádro z AES

Hashovací funkce - přehled

SHA-0 – kompromitována, nepoužívá se

SHA-1 – oblíbená, ale již kompromitovaná funkce. V únoru 2005 byl zveřejněn objev algoritmu, který umožňuje nalézt kolizi podstatně rychleji než hrubou silou. Prakticky zatím neprovedeno.

SHA-2 – dosud považována za spolehlivou
– není to jedna hashovací funkce, ale více variant souhrnně označovaných jako SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512)

MD-4

- Většina dnes používaných kryptografických hashovacích funkcí vychází z algoritmu MD-4 (Message Digest)
- MD4 byla navržena s ohledem na efektivní zpracování na existujících 32bitových procesorech.
- teoretická odolnost algoritmu MD-4 proti kolizím je 2^{64} pro 128-bitový výstup
- v praxi byly nalezeny kolize v prostoru hash kódů 2^{20} ke kompresní funkci
- není považována za bezpečnou
- 3 kola po 16 krocích ..celkem 48 rund

MD-5

Vstup: řetězec proměnné délky (neomezeně dlouhé)
Výstup: pevná délka128 bitů

Počet rund: 4

Počet kroků v rundě: 16

Vstup je zpravován po úsecích délky 512 bitů

Zpráva je zarovnána tak, aby byla dělitelná 512.

Zarovnání:
- zpráva se doplní zprava o jeden bit s hodnotou „1“
- zpráva se doplní zprava bity s hodnotou „0“ až do délky o 64 bitů menší než je číslo dělitelné 512
- posledních 64 bitů obsahuje číslo reprezentující původní délku zprávy mod 2^{64}

MD-5 - změny vůči MD-4

- konzervativní varianta MD4 (pomalejší)
- přidáno 4. kolo o 16 krocích
- celkem 64 rund
- změna logické funkce v druhé rundě
- jiné bitové posuny v jednotlivých krocích
- jiné aditivní konstanty (jedinečné v každé rundě)
- v každém kroku se připočítá výsledek z minulé rundy
  - urychlení lavinového efektu
- čísla se zapisují podle konvence „little-endian“

MD-5

M_i – vstupní zpráva 32bitů

K_i – konstanta 32bitů

– mění se pro každou operaci

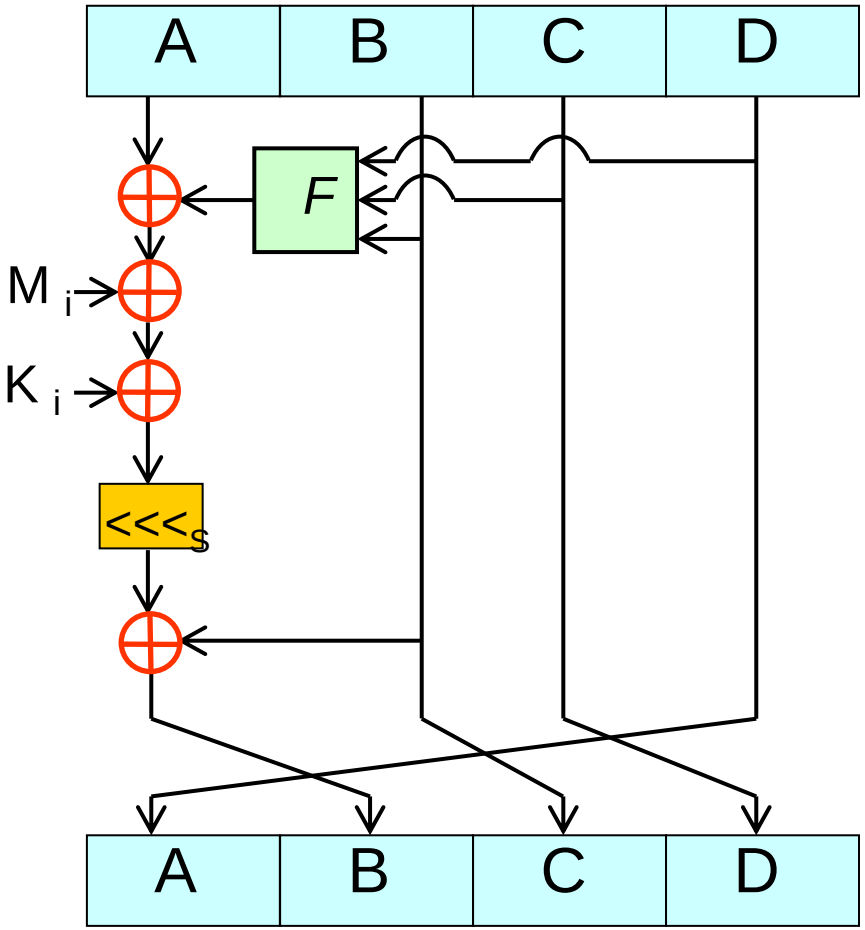
F – nelineární funkce

$<<<_s$ – rotace o *s* bitů doleva, hodnota *s* se mění v každé operaci

⊕ – operace sčítání mod 2^{32}

A,B,C,D – pomocné registry
– na začátku konstanty

– po poslední rundě obsahují hash
– velikost 128 bitů (4x32)



Jeden krok algoritmu MD-5

MD-5 expanze bloku

- vstupní blok M_i má délku 512 bitů
- před vstupem do vlastního algoritmu je rozdělen na 32 bitové části (16 různých bloků)
- každý blok vstupuje do kompresní funkce čtyřikrát, pokaždé s jinou nelineární funkcí *f*
- celkem tedy 4 kola *16 kroků = 64 rund
- konstanta K_i je spočítána jako celá část z 4294967296 násobku absolutní hodnoty funkce sin(i), kde i je úhel v radiánech

MD-5 - nelineární funkce „F“

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z)$$

$$G(X,Y,Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$$

$$H(X,Y,Z) = X \oplus Y \oplus Z$$

$$I(X,Y,Z) = Y \oplus (X \vee \neg Z)$$

$\oplus$ XOR

$\vee$ OR

$\wedge$ AND

$\neg$ NOT

- v každém kole probíhá jiná nelineární operace v bloku F

MD5 - počáteční hodnoty IV

- registr A: 0x01234567
- registr B: 0x89abcdef
- registr C: 0xfedcba98
- registr D: 0x76543210

SHA-0 …SHA-1

- SHA-0 představena NISTem v roce 1993 jako SHS (Secure Hash Standard)
- standard NIST PUB-180
- těsně před schválením v roce 1995 stažena
  - na pokyn NSA
- mírně modifikována a schválena jako SHA-1 (PUB 180-1)
- byla přidána dodatečná rotace vlevo do každého z prováděných kol kompresní funkce
- v srpnu 1998 byl odhalen pravděpodobný důvod této změny – (Differential Collisions in SHA-0)

  www.springerlink.com/index/P795V6NJ1VJ525KP.pdf

- změna v SHA-1 ničí zarovnání bitů vstupu x po průchodu kompresní funkcí

SHA-0

- SHA – Secure Hash Algorithm
- odvozeno od MD4
- počet výstupních bitů rozšířen na 160
- kompresní funkce má o 1 kolo navíc
- každé kolo má 20 kroků místo původních 16
- celkem 80 rund (4x20)
- jiné hodnoty IV
- pět počátečních nenulových aditivních konstant
- konvence je big-endian (na rozdíl od algoritmů MD)

SHA-1

Vstup: řetězec proměnné délky (max. 2^{64} -1 bitů)
Výstup: pevná délka160 bitů

Počet rund: 4

Počet kroků v rundě: 20

Vstup je zpravován po úsecích délky 512 bitů

Zpráva je zarovnána tak, aby byla dělitelná 512.

Zarovnání:
- zpráva se doplní zprava o jeden bit s hodnotou „1“
- zpráva se doplní zprava bity s hodnotou „0“ až do délky o 64 bitů menší než je číslo dělitelné 512
- posledních 64 bitů obsahuje číslo reprezentující původní délku zprávy

SHA-1 expanze bloku

- vstupní blok M_i má délu 512 bitů
- před vstupem do vlastního algoritmu je rozdělen a expandován:
  1) rozdělení na 32 bitové části (16 různých bloků) označených $W_0 \ldots W_{15}$
  2) expanzní funkce $E\{0,1\}^{512} \rightarrow E\{0,1\}^{2560}$ těchto 16 částí rozšíří na 80 podle schématu

$$W_t = (W_{t-3} \quad W_{t-8} \quad W_{t-14} \quad W_{t-16)})<<<1$$

$$t = 16..79$$

SHA-1

$<<<_5$ – rotace o 5

$<<<_{30}$ – rotace o 30

F – nelineární funkce mění se v každé rundě

$\oplus$ – operace sčítání mod 2^{32}

W_t – vstupní zpráva, 32b

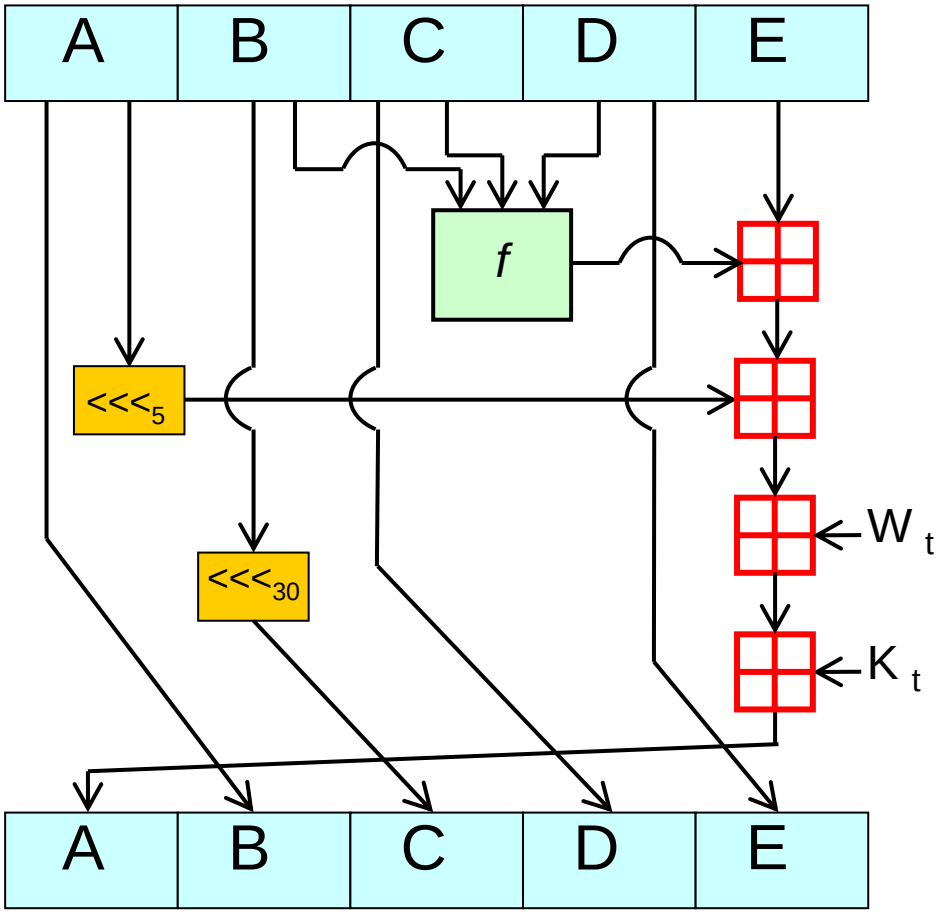
K_t – konstanta, 32b

A,B,C,D,E – vnitřní stavy - 32bitů (každý)

- celkem 160 bitů

- na konci procesu tam je hash

- na počátku konstanty



SHA-1

Funkce *F* v jednotlivých rundách algoritmu SHA-1

$$f(t,B,C,D) = (B \wedge C) \vee (\neg B \wedge D) \qquad (0 \le t \le 19)$$

$$f(t,B,C,D) = B \oplus C \oplus D \qquad (20 \le t \le 39)$$

$$f(t,B,C,D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D) \qquad (40 \le t \le 59)$$

$$f(t,B,C,D) = B \oplus C \oplus D \qquad (60 \le t \le 79)$$

Konstanta K_t

K_t = 0x5A827999 (0 ≤ t ≤ 19)
K_t = 0x6ED9EBA1 (20 ≤ t ≤ 39)
K_t = 0x8F1BBCDC (40 ≤ t ≤ 59)
K_t = 0xCA62C1D6 (60 ≤ t ≤ 79)

Počáteční hodnoty registrů A..E

A = 0x67452301 B = 0xEFCDAB89
C = 0x98BADCFE D = 0x10325476
E = 0xC3D2E1F0

bezpečnost SHA-1

- není garantována bezpečnost po roce 2010 (tzn. vhodné pouze pro krátkodobou bezpečnost)
- NIST doporučuje ukončit používání SHA-1, nejpozději do konce r. 2010
- než bude standardizována SHA-3, používat SHA-2
- 2005 - navržen hardwarový SHA-1 Cracker (podobně jako u DESu)
  - 303 PC
  - v každém PC 16 desek
  - na každé desce 32 jader
  - cena: 1.000.000 $
  - doba prolomení SHA-1 cca 2 dny

http://csrc.nist.gov/CryptoToolkit/tkhash.htm

SHA-256

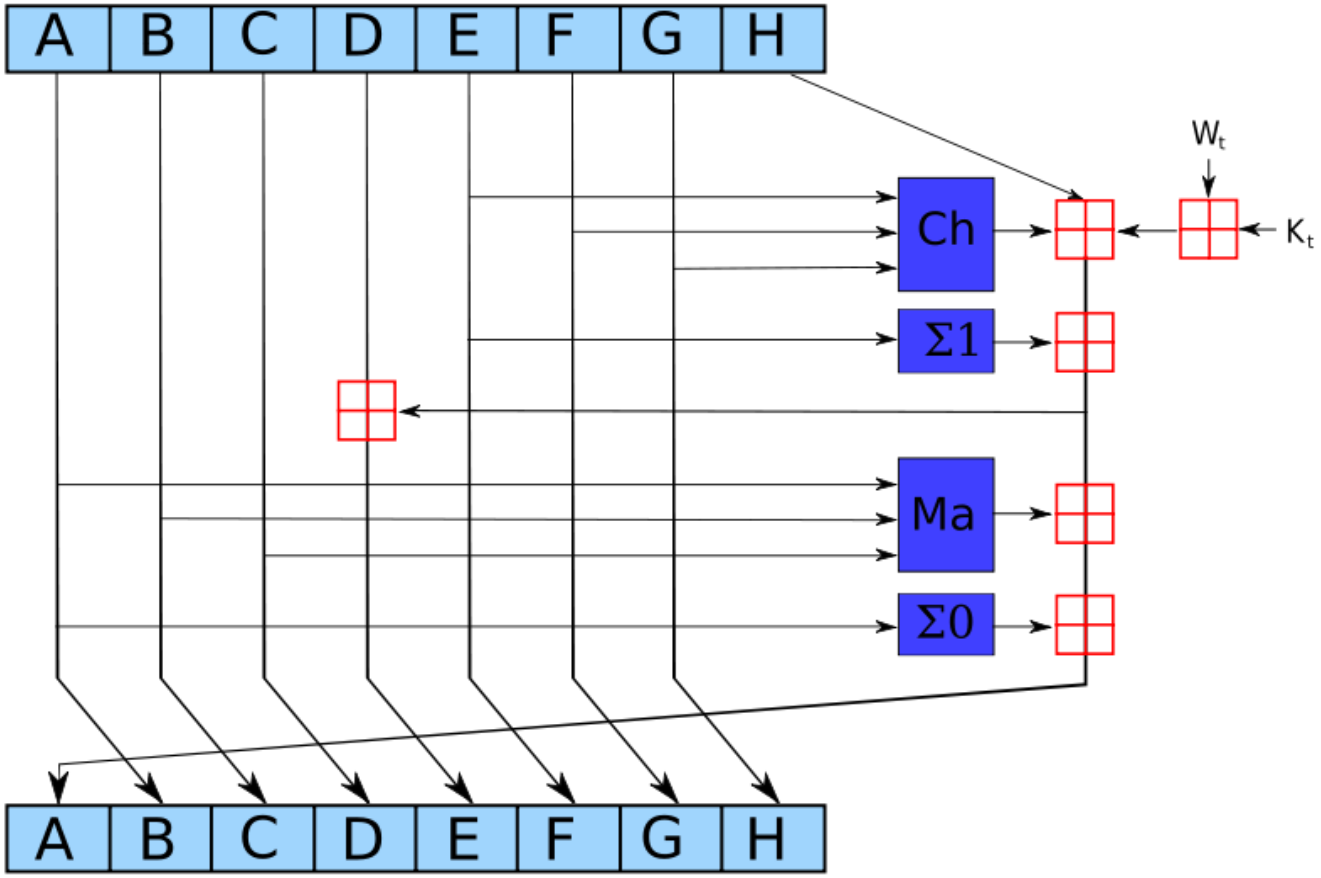
- po přijetí standardu AES vznikly další modifikace SHA-1
- označují se jako SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512)
- nové vstupní hodnoty (IV) - odpovídají desetinným částem druhých odmocnin prvních osmi prvočísel
- v každém kole se používají odlišné aditivní konstanty – odpovídají desetinným částem třetích odmocnin prvních 64 prvočísel
- opět konvence big-endian
- SHA-224 je realizována jako SHA-256 s jinými počátečními hodnotami a příslušně oříznutým výstupem

SHA-512

- pracuje s 64bitovými slovy (tzn. registry A,B,C,D,E, konstanta K_t i zpráva W_t má délku 64b)
- vstupní text se zpracovává po 1024bitových částech
- počáteční hodnoty registrů odpovídají desetinným částem druhých odmocnin 9. - 16. prvočísel
- délka výstupu 512 bitů

- SHA-384 vychází ze SHA-512
  - jiné aditivní konstanty
  - jiné počáteční hodnoty registrů
  - výsledný hash je zleva zkrácen z 512 bitů na 384 bitů

SHA-512



Srovnání hashovacích funkcí SHA-x

	SHA-0	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Délka hashe	128	160	224	256	384	512
Délka zprávy	$<2^{64}$	$<2^{64}$	$<2^{64}$	$<2^{64}$	$<2^{128}$	$<2^{128}$
Velikost bloku	512	512	512	512	1024	1024
Velikost slova	32	32	32	32	64	64
Počet kroků algoritmu	80	80	64	64	80	80
Ekvivalentní Bezpečnost[b]	64	80	112	128	192	256
Nalezení Kolize	Ano	Teoret. 2^{63}	Ne	Ne	Ne	Ne

RIPEMD-128

- **RIPEMD - RACE Integrity Primitives Evaluation Message Digest**
- narozdíl od SHA-x vznikajících u NSA, RIPEMD vznikala v akademickém prostředí
- zakódována pomocí little-endian
- oproti MD4 o jedno kolo navíc
- jiné logické funkce v kompresní funkci
- stejné IV jako SHA-1, ale zapsané v little-endian
- operace na 16 vstupních slovech probíhají paralelně ve dvou větvích; na konci kompresní funkce jsou spojeny
- 2004 – nalezena kolize

RIPEMD-160

- navržena v rámci projektu RIPE (Race Integrity Primitive Evaluation), v rámci kterého byly úspěšně provedeny útoky na MD4 a MD5
- původní verze RIPEMD s hashem délky 128 byla nahrazena bezpečnější verzí RIPEMD-160 s délkou hashe 160 bitů
- aditivní konstanty – druhé a třetí odmocniny čísel 2,3,5,7
- téměř shodná s RIPEMD-128, ale navíc:
  - přidává pátou 32-bitovou zřetězenou proměnnou
  - přidává další kolo
  - výstupní hash je rozšířený na 160 bitů
- není chráněna patenty
- zatím bezpečná

RIPEMD-160

Vstup: maximální délka 2^{64}-1 bitů

Výstup: 160 bitů

Počet rund: 5

Počet operací v rundě: 16

Vstup je zpravován po úsecích délky 512 bitů

Zpráva je zarovnána tak, aby byla dělitelná 512.

Zarovnání:
- zpráva se doplní zprava o jeden bit s hodnotou „1“
- zpráva se doplní zprava bity s hodnotou „0“ až do délky o 64 bitů menší než je číslo dělitelné 512
- posledních 64 bitů obsahuje číslo reprezentující původní délku zprávy

RIPEMD-160

A..E 16bitové registry

F nelineární funkce

<<< rotace doleva o *s*

Funkce *F* v jednotlivých rundách algoritmu RIPEMD-160

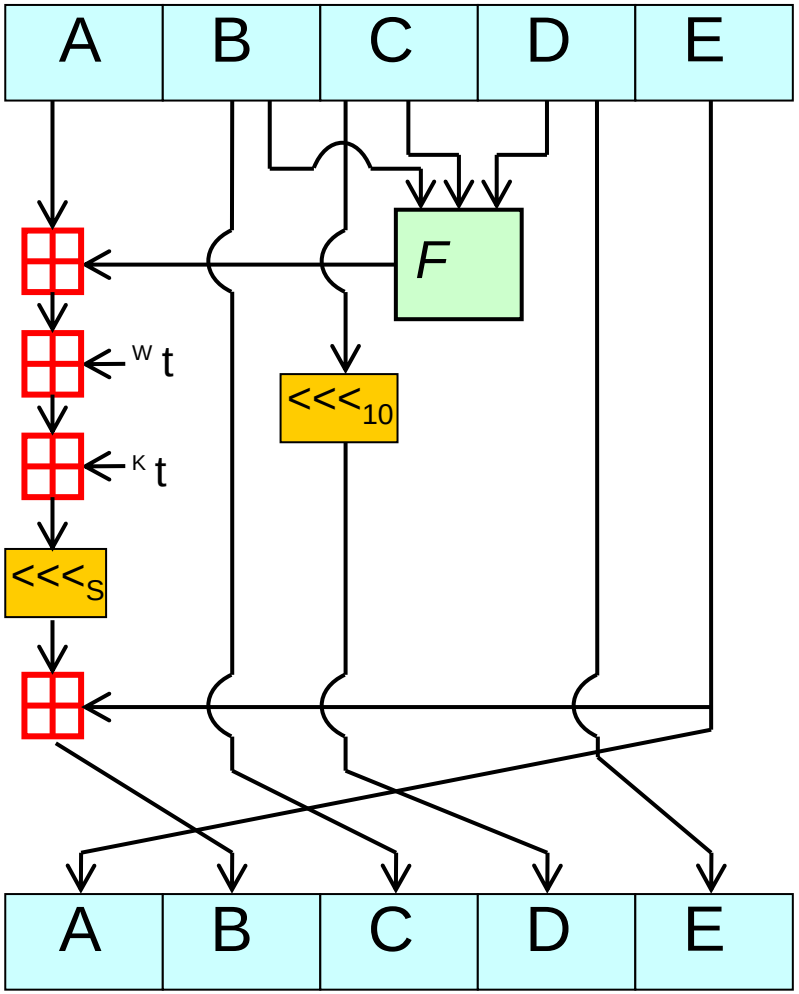
$$f(j,x,y,z) = x \oplus y \oplus z \qquad (0 \le j \le 15)$$

$$f(j,x,y,z) = (x \wedge y) \vee (\neg x \wedge z) \qquad (16 \le j \le 31)$$

$$f(j,x,y,z) = (x \vee \neg y) \oplus z \qquad (32 \le j \le 47)$$

$$f(j,x,y,z) = (x \wedge z) \vee (y \wedge \neg z) \qquad (48 \le j \le 63)$$

$$f(j,x,y,z) = x \oplus (y \vee \neg z) \qquad (64 \le j \le 79)$$



RIPEMD-160

Na rozdíl od MD5 a SHA-1 se data zpracovávají po 2x16 bitech vždy ve dvou paralelních větvích.

Konstanta K_t

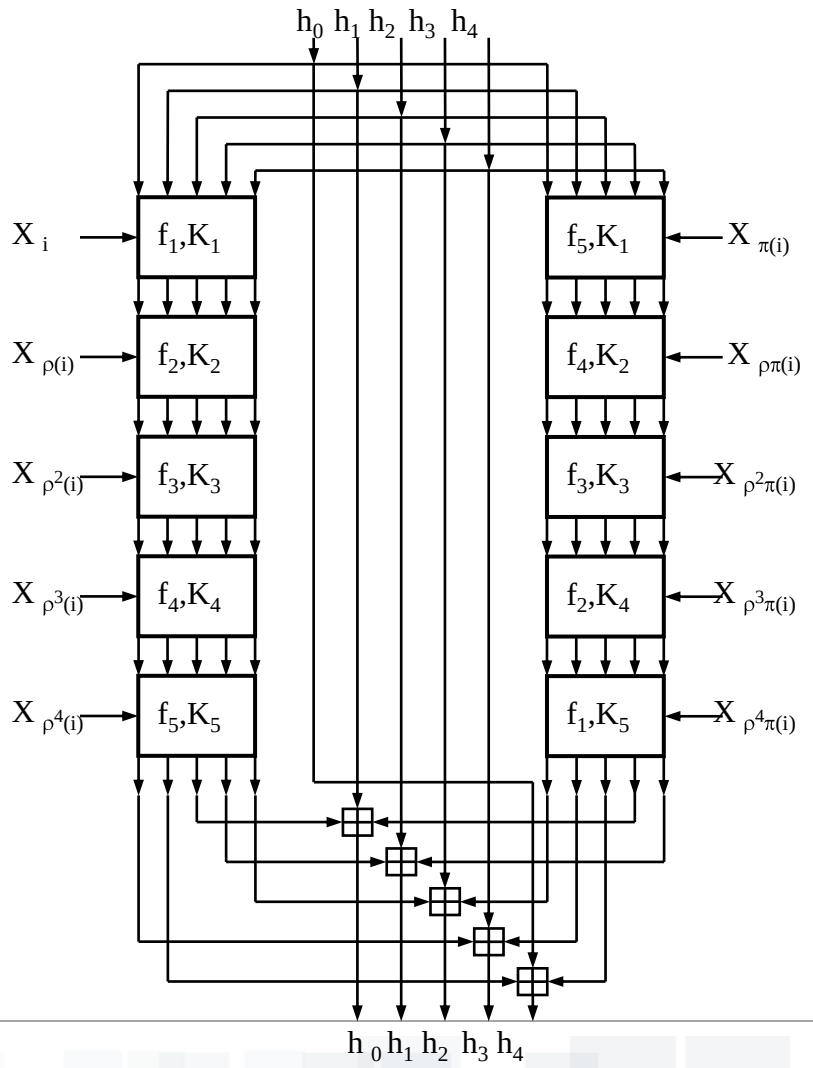
$$K_1 = 0 \qquad K_1' = 2^{30} \cdot \sqrt[3]{2}$$

$$K_2 = 2^{30} \cdot \sqrt{2} \qquad K_2' = 2^{30} \cdot \sqrt[3]{3}$$

$$K_3 = 2^{30} \cdot \sqrt{3} \qquad K_3' = 2^{30} \cdot \sqrt[3]{5}$$

$$K_4 = 2^{30} \cdot \sqrt{5} \qquad K_4' = 2^{30} \cdot \sqrt[3]{7}$$

$$K_5 = 2^{30} \cdot \sqrt{7} \qquad K_5' = 0$$



Srovnání MD-5, SHA-1, RIPEMD-160

Hashovací funkce	**MD5**	**SHA1**	**RIPEMD-160**
Délka hashe [b]	128	160	160
Velikost zpracovávaného bloku [b]	512	512	512
Počet rund	4	4	5 párů
Počet kroků v rundě	15	20	16
Maximální velikost vstupu [b]	∞	$2^{64}-1$	$2^{64}-1$
Počet logických funkcí	4	4	5
Počet konstant	64	4	10

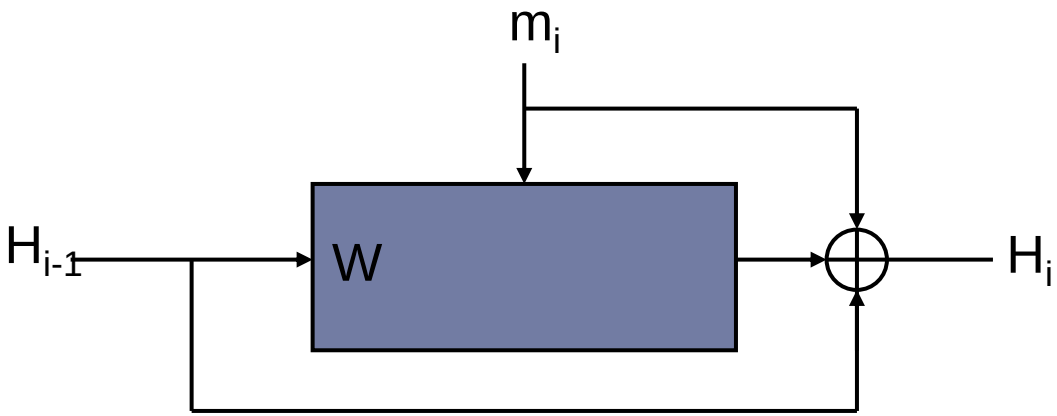
Srovnání rychlostí algoritmů

Algortimus	Délka hashe [b]	Počet rund	Rychlost vzhledem k MD-4
MD-4	128	3 x 16	1,00
MD-5	128	4 x 16	0,68
RIPEMD-128	128	2x (4x16)	0,39
RIPEMD-160	160	2x (5x16)	0,24
SHA-1	160	4 x 20	0,28
SHA-256	256	4 x 20	0,12
SHA-512	512	4 x 20	0,03

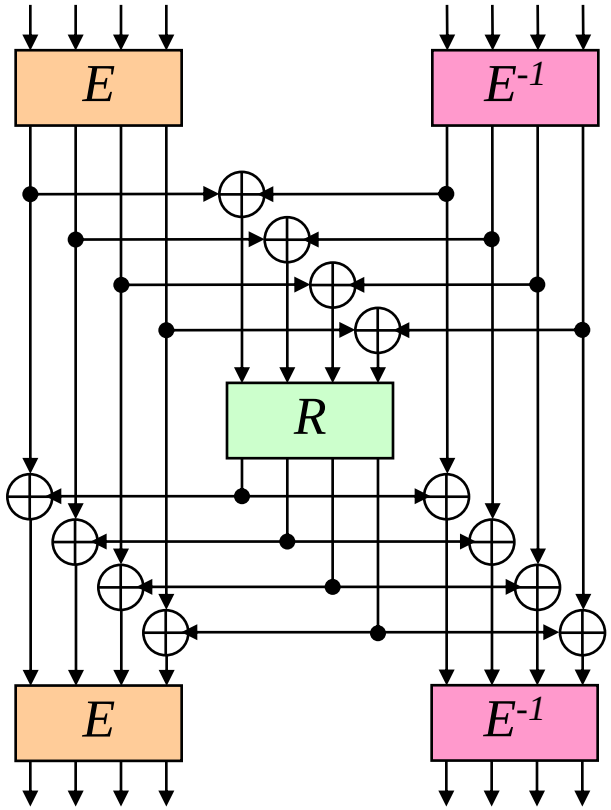
Whirlpool

- není patentovaná
- standardizována v ISO/IEC 10118-3
- kompresní funkce je tvořena speciální blokovou šifrou
  - upravený AES
  - 10 rund
  - jeden speciální S-box
- vstup $<2^{256}$ bitů
- výstup 512 bitů
- vnitřní struktura WHIRPOOLu má Miyaguchi-Preneelovo schéma
- pojmenována po spirálové galaxii v souhvězdí Honících se psů
- bezpečná
- používá se např. v FreeOTFE, Truecrypt

Whirlpool



Miyaguchi-Preneelova kompresní funkce



S-box

Whirlpool

E mini S-box:

u	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
E(*u*)	1	B	9	C	D	6	F	3	E	8	7	4	A	2	5	0

E^{-1} mini S-box:

u	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$E^{-1}(u)$	F	0	D	7	B	E	5	A	9	2	C	1	3	4	8	6

R mini S-box:

u	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
R(*u*)	7	C	B	D	E	4	9	F	6	3	8	A	2	5	1	0

Další známé hashovací funkce

- HAVAL
- Tiger

Všechny současné hashovací funkce jsou iterované a víceméně napadnutelné podobnými útoky jako MD5.

Z dlouhodobé perspektivy bude nutné vymyslet zcela nový typ hashovacích funkcí založených na jiném principu než jsou stávající.‘

Do té doby používat funkce z rodiny SHA-2.

HMAC – základní údaje

- keyed-hash MAC, kryptografický kontrolní součet, kryptografická hašovací funkce, klíčovaná haš,
- klíčované hašové autentizační kódy zpráv

- označení podle toho, jakou hašovací funkci
- používá, např. HMAC-SHA-1, HMAC-MD-5

- zpracovává nejen data, ale i klíč
- výsledek je závislý na klíči a tudíž "nepadělatelný„
- ověřuje integritu dat a zdroj dat
- odlišuje se od MAC, funkčně podobný používá jiné stavební prvky je kryptograficky silnější

HMAC – základní údaje

- doporučení RFC 2104, ANSI X9.71, FIPS 198
- nejpoužívanější HMAC jsou HMAC-SHA-1 a HMAC-MD5

HMAC-SHA-1:
blok B=64 bajtů, klíč *K*, H = SHA-1

ipad = řetězec B bajtů „0x36“

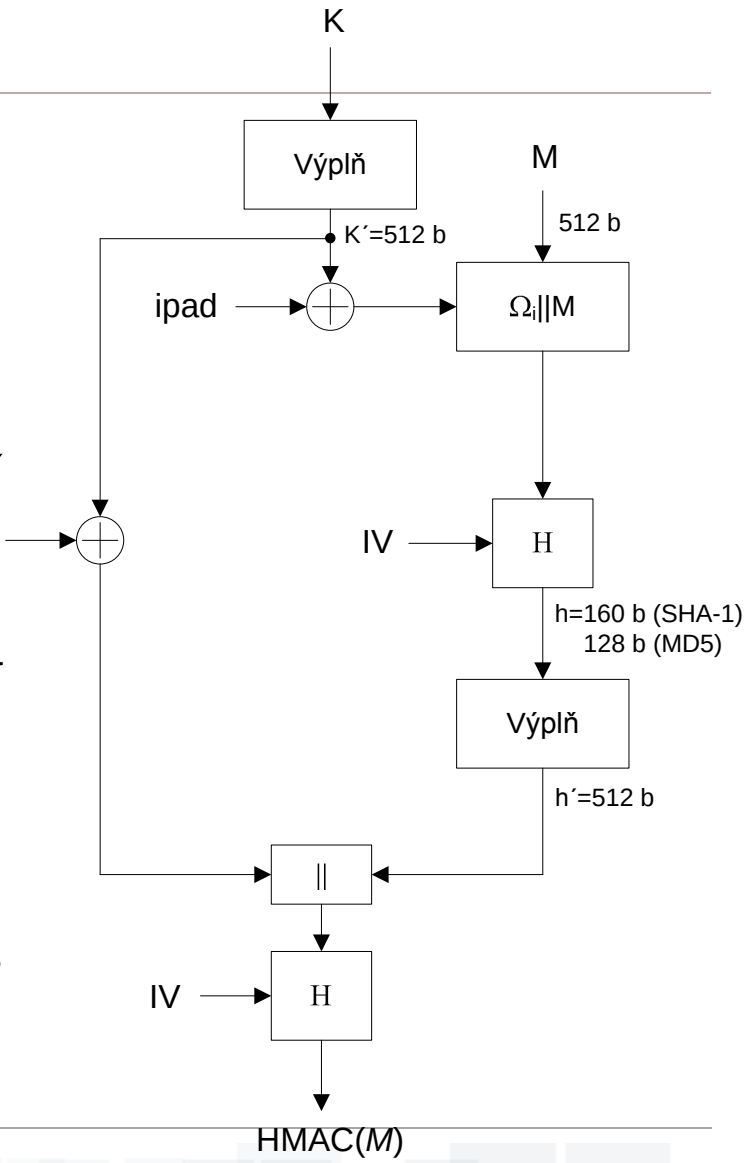
opad = řetězec B bajtů „0x5C“

klíč K se doplní nulovými bajty do plného bloku délky B

$$HMAC(M) = H((K \oplus opad) \,||\, H((K \oplus ipad) \,||\, M))$$

HMAC

- Klíč *K* se doplní bity s hodnotou log.0 tak, aby měl délku 512 bitů (pokud má klíč délku např. 160 bitů, připojí se na konec 352 bitů s hodnotou log.0, tím vznikne klíč *K*).
- Provede se operace XOR klíče *K* s konstantou *ipad*.
- K výsledku předchozího kroku se připojí 512-bitová zpráva *M*.
- Na blok vytvořený v kroku 3 se aplikuje hash funkce *H*, výsledek *h* je doplněn nulami na délku 512 bitů (vznikne *h*).
- Provede se operace XOR klíče *K* s konstantou *opad*.
- K výsledku bodu 5 se připojí *h* .
- Na výsledek bodu 6 se aplikuje hash funkce *H*.



NMAC

$NMAC(K_1, K_2, M) = H(K_2 \,||\, H(K_1 \,||\, M))$

K_1 – vnitřní klíč
K_2 – vnější klíč
$H(K_1 \,||\, M)$ – vnitřní hash
$H(K_1 \,||\, M)$ – vnější hash

- jednodušší verze HMAC
- příliš se nepoužívá
- stejné značení jako u HMAC (NMAC-MD5, NMAC-SHA1)

Útoky na hashovací funkce

Historie prolomení MD-5

- v červenci 2004 neexistoval úspěšný útok na MD5
- MD5 byla považována za bezpečnou
- nebyla známa žádná metoda rychlejší než útok hrubou silou se složitostí 2^{64}
- projekt „Rainbow table“

Červenec 2004

- čínský kolektiv oznámil nalezení kolize

Duben 2006

- nalezení kolize na počítači s procesorem Pentium-M 1,6GHz trvá průměrně 31 sekund.
- zrychlení možné díky „tunelům“
  Program na hledání kolizí najdete na
  http://cryptography.hyperlink.cz/2004/kolize_hash.htm

Čínský útok

- V srpnu 2004 byly na konferenci Crypto 2004 prezentovány kolize hashovacích funkcí MD4, MD5, HAVAL-128 a RIPEMD
- demonstrována kolize dvou různých 1024bitových zpráv

Postup vygenerování kolize:

1) naleznou se dvě různé zprávy délky 512 bitů M1, M2 (Číňanům to trvalo cca 60min na multiprocesorovém serveru IBM p690 (32procesorů))

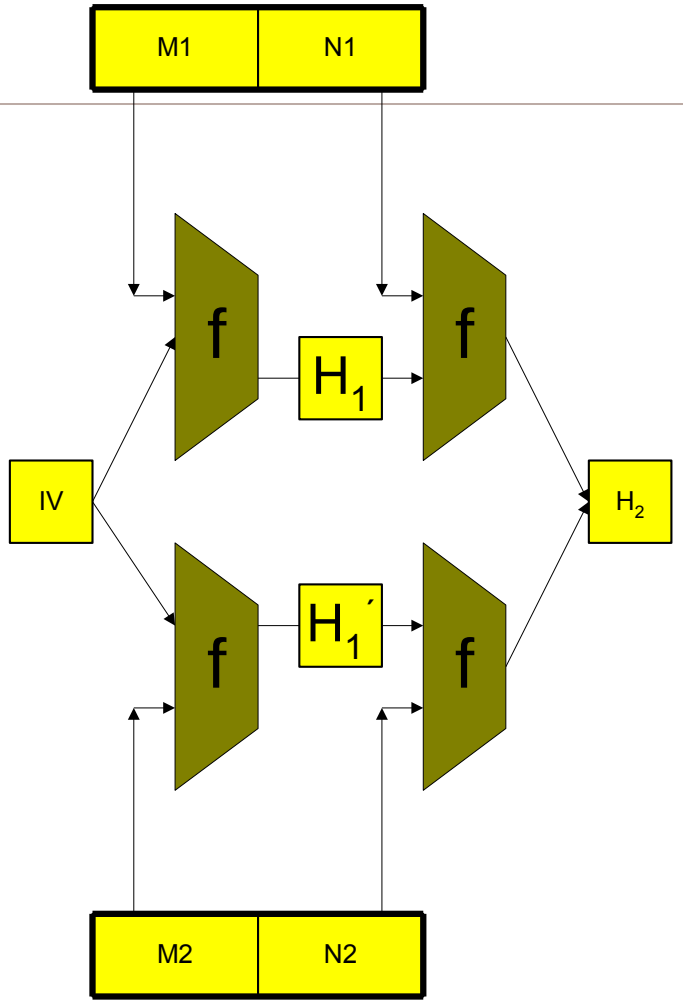
2) Nalezení dvou různých 512bitových půlzpráv N1, N2 (několik sekund) pro které platí, že že složené zprávy (M1, N1) a (M2, N2) mají stejnou hash .

H(M1||N1)=H(M1||N2)

Čínský útok

- M_1 512 bitů
- N_1 512 bitů
- M_2 512 bitů
- N_1 512 bitů

Zprávy M_1 M_2 a N= N= se lišili jen velmi málo. $M_2 = M_1 + C$, $N_2 = N_1 - C$, kde C je 512 bitů velká vhodně zvolená konstanta která má pouze 3 bity nenulové.
Informace o konstrukci konstanty C bylo to nejdůležitější na co Číňané přišli.



Obrázek z prezentace V. Klímy – Hashovaci funkce a čínský útok

Čínský útok

- detaily o útoku Číňané tají
- později zveřejnily samotné diferenční schéma, ale ne to jak na to přišli
- v březnu 2005 přišel V.Klíma na možnost rychlejšího generování kolizí
  - 1. fáze 1000-2000x rychlejší než u Číňanů
  - 2. fáze 2-80x pomalejší než u Číňanů
  - celkově umožňuje generovat kolizi v řádu několika hodin na běžném domácím PC

Příklad kolize

M1 = 2dd31d1 c4eee6c5 69a3d69 5cf9af98 87b5ca2f ab7e4612 3e580440 897ffbb8 634ad55 2b3f409 8388e483 5a417125 e8255108 9fc9cdf7 f2bd1dd9 5b3c3780

N1 = d11d0b96 9c7b41dc f497d8e4 d555655a c79a7335 cfdebf0 66f12930 8fb109d1 797f2775 eb5cd530 baade822 5c15cc79 ddcb74ed 6dd3c55f d80a9bb1 e3a7cc35

M2 = 2dd31d1 c4eee6c5 69a3d69 5cf9af98 7b5ca2f ab7e4612 3e580440 897ffbb8 634ad55 2b3f409 8388e483 5a41f125 e8255108 9fc9cdf7 72bd1dd9 5b3c3780

N2 = d11d0b96 9c7b41dc f497d8e4 d555655a 479a7335 cfdebf0 66f12930 8fb109d1 797f2775 eb5cd530 baade822 5c154c79 ddcb74ed 6dd3c55f 580a9bb1 e3a7cc35

Hash: 9603161f a30f9dbf 9f65ffbc f41fc7ef

Příklad kolize

- http://www.win.tue.nl/hashclash/SoftIntCodeSign/
- dva různé spustitelné soubory se stejným hashem
  - 1. program vytiskl „Hello world“
  - 2. program ve smyčce tiskl „Good bye world“
- každý program měl původně jiný hash
- nalezl se „ocásek“ (832B) takový, že po připojení k oběma programům mají tyto nyní stejný hash

Co je zatím bezpečné

- zatím nelze k dané zprávě vytvořit jinou, se stejnou haší (neumíme kolizi druhého řádu)
- zatím nejsou ohroženy minulé digitální podpisy pomocí MD5. Jsou ohroženy budoucí všude tam, kde útočník vytváří soubor k digitálnímu podpisu (nebezpečí vytvoření druhého falešného souboru).
- zatím nejsou ohroženy funkce HMAC a PRNG využívající slabé hašovací funkce

Co teď

- použití slabé hashovací funkce je možné vylepšit jejím vícenásobným použitím (tj. zvýšený výpočetní výkon).
- průběžně počítat souběžně několik instancí hašovací funkce s různými inicializačními vektory IV_1, IV_2, IV_3, …IV_L
- výsledným hashem je spojení jednotlivých hashů

  $MD5_{IV1}(M) \,||\, MD5_{IV2}(M) \,||\, MD5_{IV3}(M) \,||\, MD5_{IV4}(M) \,||\, \ldots\ldots \,||\, MD5_{IVL}(M)$
- Složitost prolomení dnešními metodami je větší než $2^{6(L-2)+64}$
- Pro L = 14:
  - délka kódu je 14*128 = 1792 bitů
  - složitost 2^{136}

Nevýhody: délka hashe, čas výpočtu

Co teď

- Průběžně počítat souběžně několik instancí hashovací funkce s různými inicializačními vektory IV_1, IV_2, IV_3, …IV_L
- výsledným hashem je řetezec:
  MD5 ($MD5_{IV1}(M)$ || $MD5_{IV2}(M)$ || $MD5_{IV3}(M)$ || ... || $MD5_{IVL}(M)$ **)**
- řeší problém s velikostí výsledku
  - délka kódu 128 bitů
  - složitost prolomení současnými metodami je 2^{64}

Současné použití SHA a MD-5

- k nalezení kolize funkce MD5 || SHA1 potřebujeme pouze 2^{80} zpráv, které mají stejnou MD5 hash
- Mezi nimi nalezneme pak dvě, které mají i stejnou SHA1, čili kolizi celé MD5 ||SHA1
- obtížnost tohoto útoku je cca 2^{80} (2^{63} kolize u SHA, 2^{20} kolize u MD-5)

Útoky na SHA-2

Aktuální stav útoků na SHA-x

- 2008 úspěšný útok vůči podmínce 1 (odolnost vůči získání předlohy - preimage resistance) pro
  - oslabená SHA-512 se 46 rundami z 80
  - oslabená SHA-256 se 41 rundami z 64

Hledání SHA-3

Hledání nové hashovací funkce

Stávající stav: zveřejněny kolizní páry a útoky na algoritmy MD4,MD5,SHA-0, HAVAL(128,3), RIPEMD.

Byl zveřejněn útok na SHA-1, který je efektivnější než brute force. Zatím je za hranicí výpočetních možností (2^{63} výpočtů bloku funkce).
Bezpečné funkce: SHA-2, RIPEMD-160, Whirlpool

02/2005 – NIST vyhlašuje workshop na téma „Jak dál“

06/2006 – představen prozatímní časový plán výběru nové hashovací funkce SHA-3 neboli
AHS – Advanced Hash Standard

Hledání SHA-3

- analogický proces s hledáním AES
- do roku 2012 by měl být nalezen kandidát aby mohl být obsažen v aktualizace standardu FIPS 180-2 (plánované aktualizace jsou v 2007 a 2012)

Navržená časová osa

08/2006 – zhodnocení stavu, dohodnutí čas. plánu

02/2007 – představeny minimální akceptační požadavky kandidátů a hodnotící kritéria. Do konce roku přijímány komentáře k nim.

12/2007 – finální verze požadavků a kritérií. Začíná přijímání kandidátských návrhů.

Hledání SHA-3

3Q/2008 – Konec přihlašování nových funkcí do soutěže. Začátek procesu zkoumání kandidátů.

4Q/2009 – Ukončení komentářů k vybraným kandidátům. Toto období může být dle potřeby prodlouženo. Koncem roku konference s cílem hodnocení analýzy vybraných funkcí.

1Q/2010 – Výběr finalistů. Poslední úpravy funkcí, které postoupili do finále.

2Q/2011 – Ukončení finálového kola výběru. Poslední konference – výběr vítěze.

2012 – Zveřejnění oficiální specifikace AHS.

SHA-3

Přehled algoritmů kandidujících na SHA-3:
http://ehash.iaik.tugraz.at/wiki/The_SHA-3_Zoo

- 64 přihlášených algoritmů
- 56 je dnes veřejně dostupných

Z nich:

- 15 algoritmů vyřazeno ještě před prvním kolem
  - stáhli sami autoři nebo
  - nalezena kolize nebo
  - 2nd-preimage kolize
- 27 algoritmů vyřazeno před druhým kolem
  - včetně MD-6

Hledání SHA-3

- 14 algoritmů postoupilo do druhého kola
  - BMW – Blue Midnight Wish
  - spoluautor V. Klíma
  - jeden z nejrychlejších kandidátů
- pokračování až konec srpna 2010
- do té doby „public-review“ jendnotlivých kandidátů

- 17.9.2010 – NIST dosud neví kolik ani jak podle jakých pravidel vybrat finalisty !
  - výběr 4-6 algoritmů snad do konce roku 2010
  - různé typy kontrukcí (wide-pipe, narrow-pipe, sponge, AES-based)
- 9.12.2010 – NIST oznámil 5 finalistů

SHA-3 finále

- 5 algoritmů, které postoupilo do finále
  - BLAKE
  - Grøstl
  - JH
  - Keccak
  - **Skein (B. Schneier) - kandidát na vítěze ?**
- překvapivě vyřazeny „silní“ kandidáti
  - BMW
  - SIMD
  - Shabal

SHA-3 - atuální stav

- další záměr NIStu SHA256= trunc_{256}(SHA-512)
  - proč ?
  - protože SHA512 je na 64bitových CPU výrazně rychlejší než SHA256
  - velké posílení bezpečnosti

Předpoklady

- než se rozšíří SHA3 bude naprostá většina počítačů 64 bitových
- většina počítačů bude mít HW instrukce pro AES / vektorové výpočty

Odkazy

- Xiaoyun Wang, Dengguo Feng , Xuejia Lai, Hongbo Yu: Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD, rump session, CRYPTO 2004, *Cryptology ePrint Archive*, Report 2004/199,http://eprint.iacr.org/2004/199.pdf

- http://crypto-world.info/klima/2005/cryptofest_2005.htm

- http://cryptography.hyperlink.cz/md5/Vlastimil_Klima_MD5_kolize.pdf

- http://cryptography.hyperlink.cz/2006/tunely.pdf

- **RFC 1321 – The MD5 Message-Digest Algorithm**

- **RFC 3174 – US Secure Hash Algorithm 1 (SHA1)**

Dotazy