Katedra informatiky
Fakulta matematiky, fyziky a informatiky
Univerzity Komenského


# Kryptograficky silné hašovacie funkcie

Michal Mikuš

**Diplomový vedúci:**
Doc. RNDr. Daniel Olejár, PhD.

Bratislava, 2007

Univerzita Komenského
Fakulta matematiky, fyziky a informatiky
Katedra informatiky
odbor informatika


# Kryptograficky silné hašovacie funkcie

(diplomová práca)

Michal Mikuš

**Diplomový vedúci:**
Doc. RNDr. Daniel Olejár, PhD.

Bratislava, 2007

# Zadanie

Úlohou diplomovej práce je prezentovať súčasný stav kryptograficky silných hašovacích funkcií, konkrétne popísať:

1. Merkle-Damgårdov spôsob konštrukcie a jej nedávno objavené nedostatky

2. prehľad a analýzu navrhovaných spôsobov vylepšení, prípadne nových konštrukcií

Prehlasujem, že som diplomovú prácu vypracoval samostatne, iba s pomocou literatúry uvedenej v zozname a konzultácií s vedúcim diplomovej práce.

.....

Chcel by som sa poďakovať Doc. RNDr. Danielovi Olejárovi, PhD. za odbornú pomoc, množstvo cenných pripomienok a rád, ako aj za prejavenú ochotu pri vedení práce.
Moja vďaka patrí aj M.Stanekovi, PhD., J.Szolgayovej a mojim rodičom, ktorí mi veľmi pomohli.

# Abstrakt

Kryptografia je kľúčovým odvetvím súčasného digitálneho veku a hašovacie funkcie sú jedným zo základných stavebných kameňov kryptografie. Pred troma rokmi zaznamenala konštrukcia, na ktorej sú postavené všetky súčasné hašovacie funkcie, odhalenie teoretickej slabiny, ktorá spôsobila nájdenie ďalších dvoch. V tejto práci sa venujeme analýze objavených útokov a ich aplikáciám.

V druhej časti popisujeme navrhnuté riešenia na vylepšenie tejto konštrukcie, ktoré vznikli ako reakcia na objavené nedostatky. V závere ponúkame porovnanie ich bezpečnosti a efektívnosti.

**Kľúčové slová:** hašovacia funkcia, Merkle-Damgårdova konštrukcia, multikolízia, expandovateľná správa, Nostradamov útok

# Obsah

# Zoznam obrázkov



# Zoznam tabuliek

# 1 Úvod

> *"A spectre is haunting IT security – the spectre of hash function cryptanalysis. Hash functions, the cryptographic workhorses for data authenticity and integrity, are broken one by one."*
>
> R.Weiss, S.Lucks

Tieto dve vety obrazne, ale pravdivo vystihujú súčasnú situáciu v kryptografii, hlavne v oblasti ochrany autentickosti a integrity údajov. V uplynulých troch rokoch boli totiž odhalené rôzne nedostatky v kryptografických hašovacích funkciách (ďalej len hašovacích funkciách). Tieto nedostatky môžeme rozdeliť do dvoch kategórií:

- chyby *v štruktúre* hašovacích funkcií,
- chyby konkrétnych funkcií, napr. MD5.

Vzhľadom na zhošujúcu sa situáciu hašovacích funkcií, *NIST* (National Institute for Standards and Technology) koncom januára 2007 ohlásil usporiadanie verejnej súťaže – podobne ako tomu bolo pri AES – v ktorej sa v priebehu nasledujúcich šiestich rokov (2007–2012) rozhodne o novom štandarde pre hašovaciu funkciu s menom *AHS* (Advanced Hash Standard).

Cieľom tejto práce je podrobnejšie preskúmať vnútornú štruktúru všetkých súčasných hašovacích funkcií, poskytnúť ucelený prehľad objavených *štrukturálnych nedostatkov* a prezentovať návrhy na vylepšenie štruktúry, ktorá by mohla byť základom pre AHS.

Táto práca je určená čitateľovi, ktorý absolvoval aspoň úvod do kryptológie a má základnú predstavu o kryptografických hašovacích funkciách. Taktiež predpokladáme zakladné znalosti z teórie pravdepodobnosti a výpočtovej zložitosti.

Väčšina pojmov tejto oblasti pochádza z anglickej literatúry a niektoré nemajú ešte zaužívaný slovenský ekvivalent. Aby sa predišlo zbytočným nedorozumeniam a mätúcim názvom, ponechávame tieto pojmy v anglickom jazyku.

Diplomová práca je rozdelená do nasledujúcich častí:

V druhej časti uvádzame potrebné definície a označenia, zhrnieme základné vlastnosti hašovacích funkcií a útoky na ne. Ďalej definujeme bezpečnosť ideálnej hašovacej funkcie, náhodné orákulá a ich vlastnosti.

Tretia časť sa zaoberá iterovanou štruktúrou všetkých súčasných hašovacích funkcií, rozdelením podľa kompresnej funkcie a Merkle-Damgårdovým zosilnením.

Vo štvrtej časti analyzujeme objavené chyby uvedenej konštrukcie podľa [Jo04], [KS04], [KK05] a dopĺňame algoritmy týchto útokov.

Na záver uvádzame niektoré dostupné návrhy riešenia týchto nedostatkov, skúmame ich odolnosť voči popísaným útokom (v časti 5.2.3 prezentujeme vlastnú modifikáciu útoku na druhý vzor na jednu z konštrukcií) a porovnávame ich z hľadiska bezpečnosti a efektívnosti.

# 2 Základné pojmy a označenia

## 2.1 Úvodné definície

Hašovacie funkcie vo všeobecnosti zobrazujú väčšie množiny prvkov na menšie. Sú to teda funkcie $h : D \to R$, kde $|D| > |R|$. Prvok definičného oboru $D$ je obyčajne nejaká *správa*, *súbor* alebo *dokument*. Prvok oboru hodnôt $R$ voláme *výsledok*, *odtlačok*, *digitálny odtlačok*, alebo jednoducho *haš*. Definičný obor hašovacích funkcií definujeme ako množinu bitových reťazcov ľubovoľnej, ale konečnej dĺžky. Oborom hodnôt sú bitové reťazce konštantnej dĺžky – $n$ bitov.

Existuje aj druhá kategória hašovacích funkcií, ktorých vstupným parametrom je navyše tajný kľúč, nazývaná autentizačné kódy (MAC). Používajú sa na zabezpečenie autentickosti údajov a bývajú skonštruované buď pomocou nejakej hašovacej funkcie bez kľúča, alebo blokovej symetrickej šifry. V tejto práci sa budeme zaoberať iba hašovacími funkciami bez tajného kľúča.

**Definícia 2.1.1.** *Nech $D = \{0,1\}^*$ a $R = \{0,1\}^n$. Funkciu $h : D \to R$, budeme nazývať* hašovacou*, ak pre dané $M \in D$ je ľahké vypočítať $h(M)$.*

**Poznámka 2.1.1.** Pre konkrétne hašovacie funkcie je dĺžka vstupu ohraničená dostatočne veľkou konštantou $d = 2^{64}$. Teda $D = \{0,1\}^{2^{64}}$. Veľkosť výstupu $n$ sa v súčasnosti rovná 128 až 512 bitov[1].

Hlavná motivácia používania hašovacích funkcií je vytvorenie unikátneho odtlačku k danej správe, ktorý ju môže reprezentovať, pretože je s ňou jednoznačne identifikovateľný. Malá veľkosť odtlačku uľahčuje manipuláciu a umožňuje zatajiť obsah samotnej správy, čo sú dôležité predpoklady pre použitie v mnohých protokoloch a kryptografických systémoch. Keďže však $|D| > |R|$, tak nevyhnutne musí existovať aspoň jedna dvojica správ s tým istým odtlačkom – kolízia. Presnejšie vzťahy medzi prvkami $D$ a $R$ hašovacej funkcie $h : D \to R$ určujú nasledujúce definície.

**Definícia 2.1.2.** *Množina prvkov $\{M_1, \ldots, M_k\}$ z $D$ sa nazýva $k$-násobný vzor prvku $y \in R$, ak platí $h(M_1) = \ldots = h(M_k) = y$.*

**Definícia 2.1.3.** *Množina prvkov $\{M_1, \ldots, M_k\}$ z $D$ sa nazýva $k$-násobný druhý vzor prvku $M \in D$, ak platí $h(M_1) = \ldots = h(M_k) = h(M)$ a zároveň $M_i \neq M$ pre každé $i = 1, \ldots, k$.*

[1] Závisí to od konkrétnej hašovacej funkcie.

**Označenie 2.1.1.** 1-násobný vzor prvku $y$ voláme jednoducho vzor, podobne 1-násobný druhý vzor prvku $M$ budeme volať druhý vzor prvku $M$. Kvôli stručnosti budeme $k$-násobný vzor (resp.$k$ násobný druhý vzor) nazývať skrátene $k$-vzor (resp. druhý $k$-vzor).

**Definícia 2.1.4.** *Množinu správ rovnakej dĺžky $M_1, \ldots, M_k \in D$ nazývame $k$-kolízia funkcie $h$, ak platí: $h(M_1) = h(M_2) = \ldots = h(M_k)$.*

**Označenie 2.1.2.** 2-kolíziu budeme jednoducho volať kolízia funkcie $h$. Ak nebude záležať na počte správ $k$-kolízie, budeme hovoriť o multikolízii.

Ak bude z kontextu jasné, o ktorú hašovaciu funkciu sa jedná (prípadne to bude jedno), budeme hovoriť len o multikolíziách, resp.$k$-kolíziách.

Pre bezpečné použitie hašovacích funkcií sa kolíziám potrebujeme čo najviac vyhnúť, kladieme preto na danú funkciu dodatočné požiadavky. Tieto požiadavky sa spravidla líšia v rôznych aplikáciách, definujeme preto viacero vlastností, ktoré hašovacia funkcia môže spĺňať.

**Definícia 2.1.5.** *Hašovaciu funkciu $h : D \to R$ budeme považovať za jednosmernú, ak $\forall y \in R$ je* ťažké *vypočítať $M \in D$ tak, aby platilo $h(M) = y$.*

Táto vlastnosť sa niekedy nazýva aj odolnosť voči nájdeniu vzoru (*preimage resistance*). Znamená, že z daného digitálneho odtlačku nejakej správy nevieme spätne zistiť, ako správa vyzerala. Jednosmernosť zaručuje utajenie obsahu správy.

Pojem "je *ťažké* vypočítať" nebudeme formálne definovať. Pre konečné množiny $D$ a $R$ je totiž ľahké zostrojiť algoritmus, ktorý vzor nájde v čase $O(1)$. Prakticky nám ale stačí zabezpečiť, aby počet operácií (napríklad volaní hašovacej funkcie) nevyhnutný na vypočítanie vzoru presiahol zvolenú rozumnú hodnotu ($2^{64}$ alebo $2^{80}$ – to závisí od momentálne dostupnej výpočtovej sily).

**Definícia 2.1.6.** *Hašovacia funkcia $h : D \to R$ je slabo odolná voči kolíziám, ak pre dané $M \in D$ je ťažké nájsť $M' \in D$ tak, aby platilo $M' \neq M$ a zároveň $h(M') = h(M)$, teda je ťažké nájsť druhý vzor k ľubovoľnej správe $M \in D$.*

Ekvivalentný názov je aj odolnosť voči nájdeniu druhého vzoru (*2nd-preimage resistance*). Táto vlastnosť nám v určitom prípade zaručuje unikátnosť digitálneho odtlačku, pretože k danej správe efektívne nevieme nájsť inú, s tým istým odtlačkom. Samozrejme, platí to len v prípade, ak nemáme žiadnu kontrolu nad pôvodnou správou a nemohli sme si cielene zvoliť takú, ku ktorej už poznáme druhý vzor.

Pre prípad, že prichádzame do kontaktu s nedôveryhodnými účastníkmi protokolu a potrebujeme mať istotu, že nejaký digitálny odtlačok naozaj

zodpovedá jedinej správe, potrebujeme aby použitá hašovacia funkcia spĺňala silnejšiu vlastnosť:

**Definícia 2.1.7.** *Hašovacia funkcia $h : D \to R$ je silne odolná voči kolíziám, ak je ťažké nájsť dvojicu $M, M' \in D$ takú, že $M \neq M'$ a $h(M') = h(M)$.*

Silnú odolnosť voči kolíziám (*collision resistance*) väčšinou skracujeme na odolnosť voči kolíziám. Z definície je vidieť, že silná odolnosť implikuje slabú, naopak to však neplatí.

Aj keď to z definície nie je jasne vidieť, tak odolnosť voči kolíziám znamená v praxi aj jednosmernosť. Ak by totiž existoval spôsob, ako "invertovať" funkciu $h$, tak existuje jednoduchý algoritmus [MS04], ktorý na $k$ pokusov (volaní inverznej funkcie $h^{-1}$) nájde kolíziu funkcie $h$ s pravdepodobnosťou $P_{kol} = 1 - (|R|/|D|)^k$.

Vlastností, ktoré môžeme stanoviť pre hašovaciu funkciu je oveľa viac, pre ďalšie definície odkážeme čitateľa na [Pre03], [MOV97] alebo [Sti02]. Väčšina praktických aplikácií však vyžaduje niektoré z uvedených. Napríklad ukladanie odtlačkov prístupových hesiel vyžaduje iba jednosmernosť použitej hašovacej funkcie, schéma na digitálne podpisovanie vyžaduje odolnosť voči kolíziám (a teda aj slabú odolnosť aj jednosmernosť).

Útokom na hašovaciu funkciu rozumieme akúkoľvek snahu o napadnutie niektorej z jej vlastností. Útoky rozdeľujeme podľa ich cieľa:

**hľadanie vzoru správy:** cieľom je pre danú funkciu $h$ a $y \in R$ nájsť $M \in D$, pre ktoré $h(M) = y$.

**útok na druhý vzor:** útočník sa ku $M, h(M)$ snaží nájsť $M' \neq M$, aby $h(M') = h(M)$.

**hľadanie kolízie:** hľadá sa ľubovoľné $M_1, M_2 \in D$, aby $h(M_1) = h(M_2)$.

**útok na $k$-vzor:** pre dané $h$ a $y \in R$ hľadáme rôzne $M_1, \ldots, M_k$ pre ktoré $h(M_1) = \ldots = h(M_k) = y$.

**hľadanie $k$-kolízie:** cieľom je pre funkciu $h$ nájsť $k$-ticu $M_1, \ldots, M_k$ pre ktorú $h(M_1) = \ldots = h(M_k)$.

Pri akomkoľvek útoku predpokladáme – podľa Kerckhoffovho zákona – že útočník pozná všetky konštanty, zdrojové kódy a postupy, ktoré nie sú špecifikované ako utajené. V našom prípade teda pozná presný popis funkcie a postup spracovania správy.

Úspešnosť útoku môžeme hodnotiť viacerými spôsobmi. Aby sme predišli rozoberaniu množstva (často implementačných) možností, budeme útok

hodnotiť podľa času spotrebovaného na dosiahnutie zvoleného cieľa s určitou pravdepodobnosťou[2]. Pri určovaní času zanedbáme premenlivé faktory, ako sú napríklad konkrétna implementácia, či výpočtová sila útočníka a vezmeme do úvahy len počet volaní hašovacej (resp. kompresnej[3]) funkcie.

## 2.2 Všeobecné útoky

Sú to útoky nezávislé na konkrétnom predpise funkcie, predpokladajú, že funkcia je čierna skrinka. Jediným dôležitým parametrom je dĺžka výstupu $n$. Vďaka tomu sú aplikovateľné na akúkoľvek hašovaciu funkciu. Pri odhade časovej zložitosti týchto útokov nás zaujíma len počet volaní hašovacej funkcie.

Do tejto kategórie spadajú napríklad náhodné skúšanie, úplné prehľadávanie a narodeninový útok (*birthday attack*).

### 2.2.1 Narodeninový útok

Je to najlepší známy všeobecný útok na hľadanie kolízií v hašovacích funkciách. Postup je veľmi jednoduchý:

1. zvolíme náhodne $q$ rôznych prvkov $m_1, \ldots, m_q \in D$
2. vypočítame ich odtlačky $h(m_1), \ldots, h(m_q) \in R$
3. ak sú medzi odtlačkami nejaké dva rovnaké, vrátime kolíziu

Na odhad časovej zložitosti potrebujeme vedieť počet volaní hašovacej funkcie, v tomto prípade je to $q$. Pravdepodobnosť úspechu – vrátenia kolízie – vyžaduje informáciu o distribúcii obrazov v obore hodnôt danej funkcie $h$. Predpokladajme, že obrazy sú distribuované rovnomerne, teda na každý obraz $y \in R$ sa zobrazí rovnaký počet vzorov $M \in D$. Inak povedané, pre ľubovoľný $y \in R$ je pravdepodobnosť udalosti, že obraz náhodne zvoleného prvku $M \in D$ je práve $y$, rovná $1/|R|$. Ak si označíme pravdepodobnosť úspešného vrátenia kolízie ako $P(q)$, tak na základe [MS04] platí: $P(q) \geq 1 - e^{-\frac{(q-1)q}{2r}}$, kde $r = |R|$.

Dolný odhad pravdepodobnosti úspechu je uvedený v [Wie05], predpokladá $r > 1000$ a $0 \leq q \leq 2\sqrt{r \ln r}$. Potom platí

$$P(q) \leq 1 - e^{-\frac{q^2}{2r} - \frac{q^3}{6r^2}}$$

[2]Pravdepodobnosť však nemôže byť ľubovoľne malá, musí presahovať určitú hranicu, napr. 1/2.

[3]Bude definovaná neskôr.

Uvedené odhady znamenajú, že ak napríklad chceme, aby útok na $n$-bitovú hašovaciu funkciu ($|R| = 2^n$) uspel s pravdepodobnosťou aspoň jedna polovica, tak potrebujeme zvoliť $q$ aspoň $\sqrt{2^n} \cdot \sqrt{2 \ln 2} \approx 1.177 \cdot 2^{n/2}$. Ak by sme chceli dosiahnuť pravdepodobnosť úspechu $P = 0.99$, potrebujeme zvoliť $q \approx 3.03 \times 2^{n/2}$. Najmenší počet pokusov $q$ na dosiahnutie pravdepodobnosti $P = 0.01$ je $q \approx 0.142 \times 2^{n/2}$.
Vidíme, že tieto hodnoty $q$ sa od seba asymptoticky nelíšia, môžeme teda predpokladať, že časová náročnosť nájdenia kolízie pomocou narodeninového útoku s pevnou pravdepodobnosťou $0 < P < 1$ bude zhruba $2^{n/2}$ volaní hašovacej funkcie $h$.

Ak predpokladáme, že prvky $m \in D$ nie sú zobrazované do $R$ rovnomerne, potrebujeme si vypočítať pravdepodobnosť, že náhodné dva prvky z $D$ tvoria kolíziu. Túto pravdepodobnosť si označíme $1/\beta(h)$ (v súlade so značením v [Wie05], odkiaľ je tento odhad).

Nech $h : D \to R$ je hašovacia funkcia. Označme $|R| = r$ a $R = \{y_1, \ldots, y_r\}$. Označme $d_i$ počet vzorov z $D$, ktoré sa zobrazia na dané $y_i$. Teda $d_i = |h^{-1}(y_i)|$ pre $i = 1, \ldots, r$. Potom

$$\beta(h) = \frac{|D|(|D|-1)}{d_1(d_1-1)+\ldots+d_r(d_r-1)}.$$

Pre pravdepodobnosť úspechu narodeninového útoku v tomto prípade platí:

$$P(q) \geq 1 - e^{-\frac{q-1}{\sqrt{\beta(h)}}}\left(1 + \frac{q-1}{\sqrt{\beta(h)}}\right).$$

Poznamenáme len, že pravdepodobnosť je v tomto prípade vyššia, ako bola pri rovnomernej distribúcii obrazov, útok bude teda efektívnejší. Pri vytváraní hašovacej funkcie je snaha urobiť ju čo najviac náhodnou, teda dosiahnuť čo najrovnomernejšiu distribúciu obrazov v $R$. Z výsledkov M.Bellara a T.Kohna [BK04] vyplýva, že momentálne používané funkcie (SHA-1, MD5) sa svojim rozdelením obrazov blížia rovnomernému.

### 2.2.2 Zovšeobecnený narodeninový útok

Jednoduchým pozmenením tretieho kroku narodeninového útoku ho môžeme použiť na hľadanie multikolízií. Namiesto hľadania dvojice rovnakých odtlačkov budeme hľadať rovnakú $k$-ticu ($k$ bude tretím vstupným parametrom útoku):

**ALGORITMUS:** VseobNarodUtok$(h, q, k)$

**Premenné:**

1. $h$ = hašovacia funkcia, pre ktorú chceme nájsť multikolíziu
2. $k$ = veľkosť multikolízie

**Postup:**

1. vyber náhodne $q$ prvkov $m_1, \ldots, m_q \in D$
2. vypočítaj odtlačky $h(m_1), \ldots, h(m_q) \in R$
3. ak existuje množina $K \subset \{h(m_1), \ldots, h(m_q)\}$ rovnakých výsledkov mohutnosti $k$, tak vráť $k$-kolíziu $K$, inak vráť neúspech

**Časová zložitosť:**

$q$ volaní hašovacej funkcie

**Poznámka 2.2.1.** Dodáme len, že najefektívnejší spôsob ako hľadať množinu $K$ v treťom kroku algoritmu, je utriediť si odtlačky – pevná dĺžka umožňuje použitie radix sortu – a potom jednoduchým prejdením nájsť $k$-ticu rovnakých.

**Zložitosť všeobecného narodeninového útoku**

Na určenie efektívnosti popísaného útoku využijeme výsledky [NS06] a [DM89]. M.Nandi a D.Stinson totiž v [NS06] dokázali, že náhodne zvolená $q$-tica správ (s rovnomerným rozdelením) obsahuje $k$-kolíziu s pravdepodobnosťou $0 < p < 1$, ak $q \approx (k!)^{1/k} 2^{n(k-1)/k} (ln(1/(1-p)))^{1/k}$. Pre ľubovoľné $0 < p < 1$ konštantné platí $q = \Theta(k2^{n(k-1)/k})$, za predpokladu rovnomernej distribúcie obrazov hašovacej funkcie $h$.

Pre dopredu zvolené $k$ môžeme napísať $q = \Theta(2^{n(k-1)/k})$, čo je zároveň odhad časovej zložitosti narodeninového útoku na nájdenie $k$-kolízie, ktorý budeme používať.

## 2.3 Bezpečnosť hašovacích funkcií

Bezpečnosť konkrétnej vlastnosti hašovacej funkcie sa zvykne odhadovať časovou zložitosťou najlepšieho útoku na zvolenú vlastnosť. Budeme hovoriť, že hašovacia funkcia je ideálne bezpečná, ak najlepší útok na $k$-vzor a druhý $k$-vzor je úplné preberanie (zložitosť $k2^n$) a najlepší útok na $k$-kolízie je narodeninový útok. Časová náročnosť najlepšieho útoku na ideálne bezpečnú $n$-bitovú funkciu je pre zvolené vlastnosti uvedená v tab.1.

Na záver definujeme dôležitý pojem, pomocou ktorého budeme hodnotiť bezpečnosť hašovacích funkcií. Je to abstraktná funkcia, ktorú reálne nevieme zostrojiť, môžeme ju však využiť na dokázanie niektorých vlastností, napríklad Merkle-Damgårdovej konštrukcie.

| Cieľ útoku | Ideálna bezpečnosť |
|---|---|
| nájdenie vzoru | $2^n$ |
| nájdenie druhého vzoru | $2^n$ |
| nájdenie kolízie | $2^{n/2}$ |
| nájdenie $k$-kolízie | $2^{n(k-1)/k}$ |
| nájdenie $k$-vzoru | $k2^n$ |
| nájdenie druhého $k$-vzoru | $k2^n$ |

Tabuľka 1: Časová náročnosť útokov na ideálnu hašovaciu funkciu.

Náhodné orákulum je ideálne kryptografické primitívum, ktoré za každý nový vstup vráti náhodný výsledok s rovnomerným rozdelením zo svojho oboru hodnôt. Pre rovnaké vstupy vráti rovnaký výsledok. Formálne ho môžeme definovať takto:

**Definícia 2.3.1.** *Funkciu $f : \{0,1\}^a \to \{0,1\}^b$, vybranú rovnomerne náhodne z množiny všetkých funkcií z $\{0,1\}^a$ do $\{0,1\}^b$, nazývame náhodným orákulom pevnej dĺžky.*

Pre zaujímavé parametre $a$ a $b$ je nemožné implementovať takúto funkciu, budeme vždy predpokladať nejaké verejne prístupné orákulum, ktoré pre dané $x \in \{0,1\}^a$ vráti $y = f(x) \in \{0,1\}^b$.

**Definícia 2.3.2.** *Funkciu $g : \{0,1\}^* \to \{0,1\}^b$, vybranú náhodne z množiny všetkých funkcií s daným definičným oborom a oborom hodnôt nazývame náhodným orákulom premenlivej dĺžky.*

Na takéto orákulum sa môžeme pozerať ako na nekonečne veľa orákul pevnej dĺžky: pre každé $a \in \mathbb{N}_0$ jedno orákulum $g_a : \{0,1\}^a \to \{0,1\}^b$.

Miera bezpečnosti, ktorú poskytujú náhodné orákulá ako hašovacie funkcie, je daná nasledovnými faktami:

**Fakt 1** Nech $F : \{0,1\}^* \to \{0,1\}^n$ je náhodné orákulum pevnej dĺžky. Potom nájdenie $k$-kolízie vyžaduje čas $\Omega(2^{n(k-1)/k})$ a nájdenie $k$-násobného vzoru a $k$-násobného druhého vzoru trvá $\Omega(k2^n)$.

Nasledujúci využijeme, ked budeme analyzovať bezpečnosť hašovacích funkcií pomocou ich kompresných funkcií, resp. budeme modelovať kompresnú funkciu ako náhodné orákulum pevnej dĺžky.

**Fakt 2** Nech $F : \{0,1\}^{m+n} \to \{0,1\}^n$ je náhodné orákulum pevnej dĺžky. Potom nájdenie $k$-kolízie vyžaduje čas $\Omega(2^{n(k-1)/k})$ a nájdenie $k$-násobného vzoru a $k$-násobného druhého vzoru trvá $\Omega(k2^n)$.

Na základe definícií náhodných orákul môžeme formulovať ideálnu bezpečnosť hašovacej funkcie aj takto:

**Definícia 2.3.3.** *Hašovacia funkcia $h$ je ideálne bezpečná, ak na ňu neexistuje úspešný útok, s nižšou časovou náročnosťou, ako na náhodné orákulum premenlivej dĺžky s rovnako dlhým výstupom.*

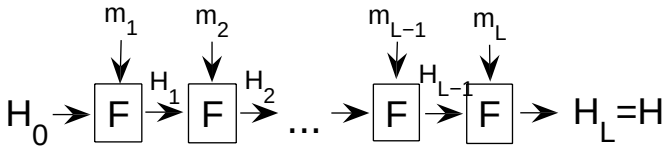
# 3 Spôsob konštrukcie hašovacích funkcií

Najvýhodnejší spôsob, ako vytvoriť hašovaciu funkciu $h$ schopnú spracovávať vstupné správy takmer ľubovoľnej dĺžky, je založený na rozdelení vstupnej správy na bloky fixnej dĺžky a ich postupnom spracovávaní. Ak by sme totiž vstupnú správu chceli spracovať naraz, musíme predpokladať maximálnu možnú dĺžku ($2^{64}$ bitov), čo je príliš neefektívne, nakoľko väčšina vstupov bude rádovo menšej dĺžky.

## 3.1 Iterovaná konštrukcia

Tento spôsob vytvárania hašovacích funkcií sa ukázal taký jednoduchý, ľahko použiteľný, robustný a efektívny, že sú na jeho princípe[4] založené všetky súčasné hašovacie funkcie.

Iterovaná hašovacia funkcia spracováva vstupnú správu po blokoch pevnej dĺžky. Označme si dĺžku jednotlivých blokov $b$ a predpokladajme, že správa $M$ sa dá rozdeliť presne na $L$ blokov $m_1, \ldots, m_L$. Potom spracovanie správy $M$ možno realizovať pomocou funkcie $F : \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$, ktorá už spracováva vstup konštantnej dĺžky. Jej vstupnými hodnotami v $i$-tom kroku sú vždy nový blok správy $m_i$ a výstup $F$ z predchádzajúceho kroku.




Obr. 1: Iterovaná hašovacia funkcia.

Funkcia $F$ určitým spôsobom komprimuje $n+b$ bitové reťazce na $n$ bitové, dostala preto názov *kompresná funkcia*. Výstup kompresnej funkcie voláme *priebežný odtlačok* správy $M$ alebo *medzivýsledok*. Po spracovaní posledného bloku $m_L$ sa posledný medzivýsledok kompresnej funkcie $F$ prehlási za výsledný odtlačok správy $M$.

[4]Spolu s Merkle-Damgårdovým zosilnením, popísaným v ďalšej časti.

Na to, aby tento spôsob fungoval pre ľubovoľnú správu $M$, bolo potrebné dodefinovať ešte dve veci. Spôsob, ako "zaokrúhliť" dĺžku správy $M$ na násobok $b$ bitov – tento spôsob budeme volať *spôsob výplne* (*padding rule*) – a prvý krok kompresnej funkcie, pretože v prvom kroku nie je k dispozícii výsledok z predchádzajúceho kroku.

Najjednoduchšie riešenie problému prvého kroku kompresnej funkcie je definovanie nultého výsledku ako nejakú pevnú konštantu, ktorá je rovnako verejná a prístupná ako hašovacia funkcia. Túto konštantu budeme nazývať inicializačný vektor a označovať $H_{IV}$ alebo $H_0$.

Spôsob výplne musí musí byť jednoznačný a invertovateľný, aby sme z dvoch rôznych správ nedostali rovnakú postupnosť blokov. Dobrým spôsobom sa ukázalo za správu $M$ pridať jeden jednotkový bit a za ním toľko nulových bitov, aby dĺžka správy vzrástla na najbližší možný násobok $b$. Počet pridaných nulových bitov je teda z intervalu $[0, b-1]$.

Formálne môžeme definovať iterovanú hašovaciu funkciu $h$ pomocou dvoch komponentov: predpisom kompresnej funkcie $F : \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ a spôsobom jej iterácie, ktorá zahŕňa konštantu $H_0$, spôsob výplne a v tomto prípade predpis $F(H_{i-1}, m_i) = H_i$. Symbolom $H_i$ označujeme $i$-ty medzivýsledok kompresnej funkcie $F$.

## 3.2 Merkle-Damgårdove zosilnenie iterovanej koštrukcie

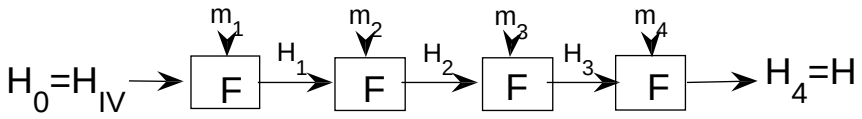
Iterovaná konštrukcia mala napriek všetkým svojim výhodám rôzne bezpečnostné slabiny[5]. V roku 1989 však I.Damgård [Dam89] a nezávisle od neho aj R.Merkle [Mer89] prezentovali vylepšenie spôsobu výplne správy, ktoré riešilo tieto problémy. Toto vylepšenie voláme Merkle-Damgårdove a hašovacie funkcie vytvorené pomocou tohto vylepšenia patria do Merkle-Damgårdovej triedy.

Začiatok predspracovania správy ostáva taký istý, ako sme už popísali: správa $M$ sa rozdelí na niekoľko blokov $m_1, \ldots, m_{L-1}$. Podstata vylepšenia spočíva v pripojení nového bloku $m_L$, ktorý bude obsahovať binárny zápis dĺžky správy $M$ a pripojí sa na koniec.

Vďaka poslednému bloku je možné teraz dokázať, že ak použitá kompresná funkcia $F$ je silne odolná voči kolíziám, tak aj výsledná iterovaná hašovacia funkcia $h$, skonštruovaná pomocou Merkle-Damgårdovho zosilnenia, bude silne odolná voči kolíziám.

**Označenie 3.2.1.** Ak nebude vyslovene uvedené inak, malým $h$ budeme

[5]V tejto práci sa im nebudeme venovať, sú dostupné v [Pre03], [MOV97]




Obr. 2: Priebeh výpočtu funkcie Merkle-Damgårdovej triedy pre trojblokovú správu $M$, $m_4$ obsahuje dĺžku správy $M$.

vždy označovať hašovaciu funkciu Merkle-Damgårdtriedy a veľkým $F$ jej kompresnú funkciu. Symboly $H_i$ budú označovať medzivýsledky kompresnej funkcie a $m_i$ bloky správ.

**Tvrdenie 3.2.1.** *Nech $F : \{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n$ je odolná voči kolíziám. Potom aj $h : \{0,1\}^* \to \{0,1\}^n$ definovaná ako iterovaná funkcia s kompresnou funkciou $F$ je odolná voči kolíziám.*

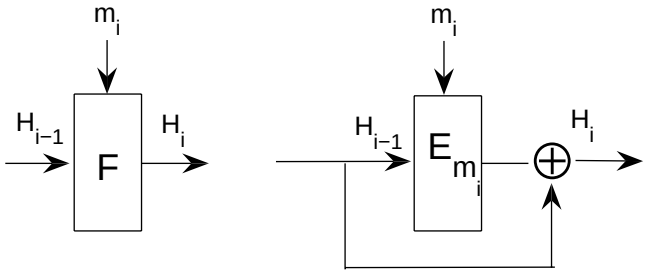
*Dôkaz.* Tvrdenie dokážeme sporom. Nech vieme nájsť dve správy $M \neq M'$, ktoré tvoria kolíziu hašovacej funkcie $h$, teda $h(M) = h(M')$. Označíme $L$ a $L'$ dĺžky správ $M$ a $M'$ a rozlíšime dva prípady:

V prípade, že $L \neq L'$, potom sa posledné bloky $m_L$ a $m'_{L'}$ nemôžu rovnať, ale $F(H_{L-1}, m_L) = h(M) = h(M') = F(H'_{L'-1}, m'_{L'})$. Dvojice $(H_{L-1}, m_L)$ a $(H'_{L'-1}, m'_{L'})$ teda tvoria kolíziu $F$, čo je spor s predpokladom.

Druhý prípad: $L = L'$. To znamená, že $m_L = m'_{L'}$. Keďže $M$ a $M'$ tvoria kolíziu, tak $F(H_{L-1}, m_L) = F(H'_{L'-1}, m'_{L'})$. Ak by sa $H_{L-1} \neq H'_{L-1}$[6], tak by sme dostali kolíziu kompresnej funkcie $F$. Ak $H_{L-1} = H'_{L-1}$, tak rozoberieme zase predchádzajúci krok kompresnej funkcie. Platí $F(H_{L-2}, m_{L-1}) = F(H'_{L-2}, m'_{L-1})$. Máme dve možnosti, buď sa dvojica $(H_{L-2}, m_{L-1})$ nerovná dvojici $(H'_{L-2}, m_{L-1})$ a dostávame kolíziu kompresnej funkcie $F$ a teda spor s predpokladom, alebo sa dvojice rovnajú a potom rozoberieme predchádzajúci krok výpočtu funkcie $F$.
Takto sa "vrátime" až po prvý krok výpočtu $F$ s tým, že platí $F(H_0, m_1) = F(H'_0, m'_1)$ a všetky predchádzajúce dvojice $(H_{i-1}, m_i)$ a $(H'_{i-1}, m'_i)$ boli rovnaké. V tomto prípade sa už ale $m_1$ nemôže rovnať $m'_1$, pretože $M$ a $M'$ sú rôzne. To znamená, že sme schopní nájsť kolíziu v kompresnej funkcii a to je spor. □

[6] $L = L'$.

Merkle-Damgårdova konštrukcia sa ukázala ako silný nástroj na vytváranie hašovacích funkcií, pretože dokázala zredukovať problém nájdenia odolnej funkcie s neobmedzeným definičným oborom $\{0,1\}^*$ na problém nájdenia odolnej funkcie s konečným definičným oborom $\{0,1\}^{n+b}$.




Obr. 3: Davies-Meyerov spôsob ako využiť blokovú šifru na konštrukciu kompresnej funkcie. Blok správy $m_i$ vstupuje ako tajný kľúč, $H_i$ ako blok správy pre šifrovaciu transformáciu $E$.

## 3.3 Kompresné funkcie hašovacích funkcií

Možností, ako skonštruovať kompresnú funkciu je viacero. Často používané konštrukcie sú založené na:

**blokových šifrách** : v kompresnej funkcii sa využije šifrovacia funkcia $E_k(m)$ nejakej blokovej šifry. Existuje viacero spôsobov, ako pomocou šifrovacej funkcie definovať kompresnú, najznámejšie sú Davies-Meyerov (obr.3), Matyas-Meyer-Oseasov, či Miyaguchi-Preneelov. Viac detailov, ako aj analýza bezpečnosti je uvedená v [Pre03].

**ťažkých problémoch** : príkladom je Chaum-van Heijst-Pfitzmannova hašovacia funkcia [Sti02], o ktorej je dokázané, že nájdenie kolízie vedie k nájdeniu diskrétneho logaritmu $\log_\alpha(\beta)$, pre pevné $\alpha, \beta \in \mathbb{Z}_p$. Táto funkcia je však príliš pomalá na to, aby našla praktické využitie.

**špeciálnej konštrukcii** : tieto funkcie sú navrhnuté "od základu", so zreteľom na rýchlosť. Ich bezpečnosť je založená na dostatočnej náhodnosti a väčšinou sa nedá dokázať. Príkladom sú funkcie MD4, MD5, SHA-0, SHA-1, RIPEMD, RIPEMD-160 a mnohé iné.

Pretože táto práca sa zaoberá *štrukturálnymi* nedostatkami hašovacích funkcií, kompresnú funkciu budeme modelovať ako čiernu skrinku a budeme o nej vždy predpokladať maximálnu bezpečnosť. To znamená, že nájdenie kolízie v kompresnej funkcii s dĺžkou výstupu $n$ vyžaduje $\Theta(2^{n/2})$ operácií, nájdenie vzoru, alebo druhého vzoru $\Theta(2^n)$ operácií. Stručne povedané, kompresná funkcia je náhodné orákulum pevnej dĺžky (s $n$-bitovým výstupom).

# 4 Útoky na Merkle-Damgårdovu triedu HF

## 4.1 Útok predlžovaním správy

Útok predlžovaním správy – the length extension attack – je známou nevýhodou Merkle-Damgårdovej iterovanej konštrukcie. Umožňuje ho absencia výstupnej transformácie – po spracovaní posledného bloku sa získaný medzivýsledok prehlási za výstupný odtlačok.

Táto jednoduchá konštrukcia umožňuje útočníkovi na základe digitálneho odtlačku nejakej správy $M$ vytvoriť nový platný odtlačok správy $M||y$, pre skoro ľubovoľné $y$, za predpokladu, že pozná dĺžku správy $M$. Ak totiž prvý blok časti $y$ bude totožný s blokom $m_L$ – čo útočník vie dosiahnuť, ak pozná dĺžku $M$, tak $h(M||y) = H_{IV}(m_1||\ldots||m_{L-1}||m_L||y_2||\ldots||y_{L'}) = h_{h(M)}(y_2||\ldots||y_{L'})$. Kde $h_{h(M)}$ označuje výpočet funkcie $h$ s inicializačným vektorom $h(M)$ namiesto pôvodne stanoveného $H_{IV}$.

Tento útok predlžovaním správy znemožnil jednu jednoduchú konštrukciu MAC kódov pomocou hašovacích funkcií, pri ktorej sa výsledný autentizačný kód rovnal odtlačku tajného kľúča zreťazeného so správou: $MAC(k, M) = h(k||M)$. V tomto prípade by útočník vedel skonštruovať MAC pre nejaké predĺženie správy $M$ bez toho, aby vedel tajný kľúč $k$.

## 4.2 Predlžovanie kolízií

Ak sa nám podarilo nájsť kolíziu dvoch správ $M_1, M_2$, ktoré majú navyše rovnakú dĺžku, je jednoduché vytvárať ďalšie kolízie. Potom totiž platí:

$$h(M_1||S) = h(M_2||S)$$

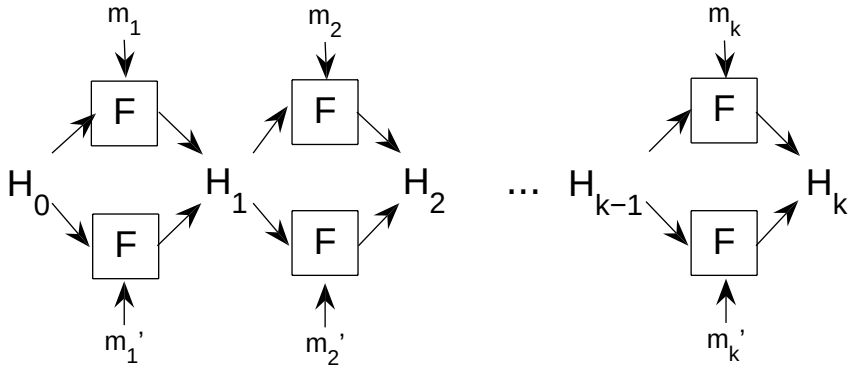
pre ľubovoľný suffix $S$ a nejakú hašovaciu funkciu $h$ z Merkle-Damgårdovej triedy.

## 4.3 Jouxov útok

Pojem multikolízie prvýkrát publikoval v roku 2004 Antoine Joux [Jo04]. Takisto prezentoval útok, ktorým sa dajú hľadať rádovo rýchlejšie ako pomocou narodeninového útoku a ako sa dajú využiť na hľadanie kolízií tzv. kaskádových hašovacích funkcií (*cascaded hash functions*, alebo na hľadanie $k$-vzorov.

### 4.3.1 Hľadanie multikolízií

V časti 2.1 sme uviedli definíciu a popísali, ako sa dajú hľadať $k$-kolízie pomocou zovšeobecneného narodeninového útoku. Zložitosť tohto útoku je




Obr. 4: Nádväznosť jednotlivých kolidujúcich dvojíc multikolízie.

$\Theta(k2^{n(k-1)/k})$. Narodeninový útok však predpokladal celú hašovaciu funkciu ako čiernu skrinku (resp.náhodné orákulum).

A.Joux využíva iterovanú štruktúru hašovacích funkcií na to, aby zostrojil $2^k$-kolíziu v čase $O(k \cdot 2^{n/2})$. Hlavná myšlienka útoku (obr. 4) spočíva v nájdení $k$ dvojíc blokov $(m_1, m_1'), \ldots, (m_k, m_k')$ (alebo častí správ, ktoré majú rovnakú dĺžku), pre ktoré platí:

$$F(H_{zac}, m_1) = F(H_{zac}, m_1') = H_1$$

$$F(H_{i-1}, m_i) = F(H_{i-1}, m_i') = H_i$$

$$F(H_{k-1}, m_k) = F(H_{k-1}, m_k') = H_k$$

Dvojice $(a_i, b_i)$ nájdeme pomocou narodeninového útoku, každú v čase $O(2^{n/2})$. Množina

$$\left\{ c_1||c_2||\ldots||c_k \;\middle|\; c_i = a_i \text{ alebo } b_i, \text{ pre } i = 1, \ldots, k \right\}$$

obsahuje $2^k$ správ rovnakej dĺžky, ktoré sa hašujú na rovnakú hodnotu. Celý postup vyžaduje $O(k \cdot 2^{n/2})$ volaní kompresnej funkcie $F$.

**ALGORITMUS:** Multikolízia$(k, h, H_{zac})$

**Premenné:**

1. $k$ = parameter $2^k$-kolízie
2. $h$ = hašovacia funkcia, pre ktorú chceme nájsť multikolíziu
3. $H_{zac}$ = začiatočný medzivýsledok multikolízie (napr. inicializačný vektor $H_{IV}$)

4. $H_{tmp}$ = pomocný medzivýsledok
5. NájdiKolíziu$(H, h, L)$ = funkcia, ktorá na základe narodeninového útoku vráti dve správy dĺžky $L$ blokov s rovnakým odtlačkom, $H$ je počiatočný medzivýsledok tejto kolízie
6. $A$ = dvojrozmerné pole $k \times 2$ kolidujúcich blokov

**Postup:**

1. $H_{tmp} = H_{zac}$
2. for $i = 1$ to $k$:
   - $(m_1, m_2)$ = NájdiKolíziu$(H_{zac}, h, 1)$
   - $A[i][0] = m_1$
   - $A[i][1] = m_2$
   - $H_{tmp} = F(H_{tmp}, m_1)$
3. Vráť multikolíziu $A$

**Časová zložitosť:**

$O(k \cdot 2^{n/2})$, kde $n$ je dĺžka výstupu hašovacej funkcie $h$.

### 4.3.2 Význam multikolízií

Najpodstatnejším dôsledkom objavenia multikolízií bolo odkrytie teoretickej slabiny v Merkle-Damgårdovej konštrukcii po 15 rokoch jej používania. Okrem toho, že sme schopní generovať určitý druh kolízií lepšie ako hrubou silou, to znamená, že konštrukcia nie je taká dokonalá, ako možno doteraz vyzerala a napriek tomu, že sa pozeráme na kompresnú funkciu ako na náhodné orákulum, celá hašovacia funkcia náhodným orákulom rozhodne nie je.

**Aplikácia multikolízií v praxi**

Oveľa hmatateľnejším, aj keď menej podstatným dôsledkom multikolízií je znemožnenie nasledovnej konštrukcie. Vieme, že narodeninový útok je veľmi účinný na hašovacie funkcie s malou výstupnou dĺžkou. Na zníženie efektívnosti narodeninového útoku nám ale stačí jednoducho vytvoriť funkciu, ktorá ma dostatočne dlhý výstup. Jednoduché a efektívne riešenie by bolo zreťazenie niekoľkých kratších výstupných hodnôt. Napríklad, pre dané dve hašovacie funkcie $h_1$ a $h_2$, definujeme kaskádovú hašovaciu funkciu $G(m) = h_1(m)||h_2(m)$, ktorá má dĺžku výstupu danú súčtom výstupov

funkcií $h_1$ a $h_2$. Uvedený postup tvorby nových funkcií sa často používal, pretože mohol využiť už implementované hašovacie funkcie, ktoré zastarali len preto, že mali príliš krátky výstup a nie preto, že by sa v nich našla nejaká bezpečnostná slabina.

Rýchly spôsob konštrukcie multikolízií však umožňuje hľadať kolízie takto vytvorenej hašovacej funkcie rýchlejšie, ako narodeninový útok pre funkciu vytvorenej dĺžky. Ak si totiž zostrojíme $2^{N_2/2}$-kolíziu pre funkciu $h_1$ ($N_2$ je dĺžka výstupu funkcie $h_2$), tak podľa narodeninového paradoxu máme pravdepodobnosť väčšiu ako $1/2$, že týchto $2^{N_2/2}$ správ obsahuje aj kolíziu funkcie $h_2$. Nájdená dvojica správ potom tvorí kolíziu funkcie $G(m) = h_1(m)||h_2(m)$.

**ALGORITMUS:** NájdiKolíziu$(h_1, h_2, H_{zac})$

**Premenné:**

1. $N_2$ = dĺžka výstupu funkcie $h_2$
2. $A$ = dvojrozmerné pole $N_2/2 \times 2$ blokov, ktoré tvoria multikolíziu funkcie $h_1$
3. $B$ = zoznam $2^{N_2/2}$ odtlačkov funkcie $h_2$ zo správ v $A$

**Postup:**

1. $A$ = Multikolízia$(N_2/2, h_1, H_{zac})$
2. Pomocou $A$ vytvor zoznam $B$ všetkých odtlačkov správ
   for $i = 1$ to $2^{N_2/2}$:
   - $m = \emptyset$
   - for $j = 1$ to $N_2/2$:
     - nech $b$ je $j$-ty bit čísla $i$
       $m = m||A[j][b]$
   - ulož do $B$ dvojicu $(m, h_2(m))$
3. Nájdi v $B$ dva zhodné odtlačky $H_i, H_j$
4. Vráť zodpovedajúcu kolíziu $m_i, m_j$

**Časová zložitosť:**

$O(N_2 \cdot 2^{N_1/2} + 2^{N_2/2})$ volaní kompresnej funkcie

**Hľadanie $k$-násobného (druhého) vzoru**

Hľadanie $k$-kolízií možno využiť ako nástroj aj na efektívne hľadanie $k$-vzorov daného digitálneho odtlačku $y$. Postup má dve fázy: v prvej útočník vygeneruje $k$-kolíziu $m_1, \ldots, m_k$ a v druhej hľadá správu $m_{k+1}$ pre ktorú platí: $F(h(m_1), m_{k+1}) = y$.

Prvá fáza zaberie približne $lg(k)2^{n/2}$ volaní kompresnej funkcie, druhá však $2^n$, čo je priemerný počet pokusov na nájdenie vzoru k danému obrazu $y$ pre $n$-bitovú funkciu. Celkový čas potrebný na tento útok je $O(2^n)$ volaní.

Pri hľadaní druhého $k$-vzoru danej správy $M$ je postup veľmi podobný. Budeme hľadať $k$-vzor obrazu $y = h(M)$. Aby sme predišli tomu, že nejaké $m_i$ bude totožné s $M$, tak pri hľadaní $k$-kolízie zvolíme dĺžku jednotlivých kolízií tak, aby celková dĺžka výslednej multikolízie bola rôzna od dĺžky $M$. Časová náročnosť tohto modifikovaného postupu bude rovnako $O(2^n)$ operácií.

## 4.4 Expandovateľné správy

V tejto časti popíšeme čo sú to *expandovateľné správy* a ako sa dajú hľadať pre ľubovoľnú hašovaciu funkciu Merkle-Damgårdovej triedy. Potom uvedieme útok podľa J.Kelseyho a B.Schneiera [KS04], ktorý je vďaka nim schopný nájsť pre každé $k \in \mathbb{N}$ druhý vzor ľubovoľnej správy dlhej $2^k + k$ blokov v čase $k \cdot 2^{n/2+1} + 2^{n-k+1}$, čo je predsa len menej než $2^n$, ako sa očakáva od odolnej hašovacej funkcie.

### 4.4.1 Hľadanie expandovateľných správ

Expandovateľná správa je vlastne druh multikolízie, pri ktorej majú jednotlivé správy rôzne dĺžky, ale hašujú sa na rovnaký medzivýsledok. Keďže pri Merkle-Damgårdovom zosilnení sa za správu pripojí blok obsahujúci dĺžku správy, tak výsledný odtlačok je pre tieto jednotlivé správy vo všeobecnosti rôzny, no nám bude stačiť rovnaký medzivýsledok. Expandovateľnú správu, ktorá môže mať všetky dĺžky od $a$ blokov po $b$ blokov[7] budeme nazývať $(a, b)$-expandovateľná správa.

Hľadanie expandovateľných správ sa veľmi nelíši od hľadania multikolízií. Pri multikolíziách sme hľadali dvojice jednoblokových správ, ktoré kolidovali a ich pospájaním vzniklo viacero správ rovnakej dĺžky, ktoré sa hašovali na rovnaký výsledok. V prípade konštrukcie expandovateľnej správy budeme hľadať dvojice čiastkových správ, ktoré budú tvoriť kolíziu, ale budú mať *rôznu* dĺžku a ich pospájaním dostaneme správy rôznych dĺžok, ktoré budú mať rovnaký medzivýsledok.

[7] vrátane dĺžok $a$ a $b$.

**Definícia 4.4.1.** *Množinu dvojíc správ* $(m_1, m'_1),\ldots,(m_k, m'_k)$*, kde* $m_i$ *a* $m'_i$ *majú vo všeobecnosti rôznu dĺžku, budeme nazývať expandovateľná správa, ak platí:*

$$F(H_{i-1}, m_i) = F(H_{i-1}, m'_i) = H_i, \text{ pre } i = 1, \ldots, k$$

*Hodnota* $H_0$ *predstavuje vstupný a* $H_k$ *výstupný medzivýsledok kompresnej funkcie* $F$*.*

**Príklad 4.4.1.** Majme správy $m_1, m_2, m_3, m_4$, ktorých dĺžky sú v poradí $1, 3, 5, 9$ blokov. Ak platí $F(H_0, m_1) = F(H_0, m_2)$ pre nejaké $H_0$ a $F(H_1, m_3) = F(H_1, m_4)$ pre $H_1 = F(H_0, m_1)$, potom $m_1, m_2, m_3, m_4$ tvoria expandovateľnú správu, ktorá môže nadobúdať dĺžky $6, 8, 10, 12$ blokov.

### Hľadanie kolízie z dvoch správ rôznej dĺžky

Na to, aby sme vytvorili expandovateľnú správu, potrebujeme veľa dvojíc správ, ktoré majú rôznu dĺžku a pri rovnakom počiatočnom medzivýsledku vrátia rovnaký koncový medzivýsledok. Tieto dvojice sa nehľadajú o nič ťažšie ako samotné kolízie. Útočník, ktorý chce nájsť kolíziu medzi správami dĺžky 1 a $p$ blokov, potrebuje zostrojiť $2^{n/2}$ rôznych blokov a $2^{n/2}$ rôznych správ dĺžky $p$ a nájsť medzi nimi dvojicu, ktorá dáva rovnaký medzivýsledok. Pre efektívnosť si útočník môže dokonca zvoliť takú množinu správ dĺžky $p$, ktoré budú mať prvých $p-1$ blokov rovnakých a budú sa líšiť len v poslednom bloku. Vďaka tomu sa táto množina bude dať generovať rovnako rýchlo ako množina pozostávajúca z jednoblokových správ. Uvedieme jednoduchý algoritmus, ktorý nájde kolidujúcu dvojicu správ dĺžky 1 a $p$ začínajúcu z medzivýsledku $H_{zac}$.

**ALGORITMUS:** NájdiKolíziu$(p, H_{zac})$

**Premenné:**

1. $n$ = dĺžka medzivýsledkov kompresnej funkcie
2. $p$ = požadovaná dĺžka druhej správy
3. $A, B$ = zoznamy hašovacích medzivýsledkov
4. $m$ = fixovaný blok opakujúci sa $p-1$-krát v každej správe dĺžky $p$
5. $H_{zac}$ = vstupný medzivýsledok tejto kolízie
6. $H_{tmp}$ = pomocný medzivýsledok kompresnej funkcie
7. $M(i)$ = $i$-ty blok použitý na hľadanie kolízie

**Postup:**

1. Vypočítaj pomocný medzivýsledok $H_{tmp}$ pre $p$-blokovú správu spracovaním prvých $p-1$ blokov:

- $H_{tmp} = H_{zac}$
- for $i = 1$ to $p - 1$:
  - $H_{tmp} = F(H_{tmp}, m)$

2. Skonštruuj zoznamy $A, B$:
   - for $i = 1$ to $2^{n/2}$:
     - $A[i] = F(H_{zac}, M(i))$
     - $B[i] = F(H_{tmp}, M(i))$
3. Nájdi $i, j$ také, že $A[i] = B[j]$
4. Výstupná dvojica správ je $(M(i), m||m|| \ldots ||m||M(j))$ a výstupný medzivýsledok kompresnej funkcie je $F(H_{zac}, M(i))$

**Časová zložitosť:**

$p - 1 + 2^{n/2+1}$ volaní kompresnej funkcie

**Hľadanie $(k, k + 2^k - 1)$-expandovateľnej správy**

Tento algoritmus sa dá celkom ľahko použiť na hľadanie expandovateľných správ, ktoré majú široký rozsah dĺžok. Na nájdenie $(k, k + 2^k - 1)$-expandovateľnej správy budeme potrebovať dvojicu správ dĺžky jeden blok a $2^{k-1} + 1$ blokov. Ďalej dvojice s dĺžkami 1 a $2^{k-2} + 1$, 1 a $2^{k-3} + 1$, a tak ďalej až po dvojicu 1 a 2.[8] Tento postup je zhrnutý v nasledujúcom algoritme, výsledné dvojice správ ukladáme do poľa $E$.

**ALGORITMUS:** NájdiExpSprávu$(k, H_{zac})$

**Premenné:**

1. $H_{zac}$ = vstupný medzivýsledok expandovateľnej správy E
2. $H_{tmp}$ = pomocný medzivýsledok kompresnej funkcie
3. $E$ = dvojrozmerné pole kolidujúcich párov, $E[i][1]$ je prvá správa $i$-tej dvojice, $E[i][2]$ je druhá

**Postup:**

1. $H_{tmp} = H_{zac}$
2. for $i = 0$ to $k - 1$:
   - $(m_1, m_2, h_{tmp}) =$ NájdiKolíziu$(2^i + 1, H_{tmp})$
   - $E[k - i - 1][1] = m_1$
   - $E[k - i - 1][2] = m_2$
3. Vráť výsledné pole $E$

**Časová zložitosť:**

$k \cdot 2^{n/2+1} + 2^k \approx k \cdot 2^{n/2+1}$ volaní kompresnej funkcie

[8] S parametrom $k$ klesáme až po nulu: $2 = 2^0 + 1$.

Na konci tohto procesu máme pole $k \times 2$ správ, z ktorých vieme jednoducho zostrojiť ľubovoľne dlhú správu (z intervalu $(k, k + 2^k - 1)$) bez toho, aby to ovplyvnilo koncový medzivýsledok kompresnej funkcie po spracovaní tejto správy. Pre danú dĺžku $L \in (k, k + 2^k - 1)$, zostrojíme správu dlhú $L$ blokov týmto spôsobom. Všimnime si, že $(k - i - 1)$-vá dvojica správ sa líši presne o $2^i$ blokov. Keď si napíšeme číslo $|L| - k$ v dvojkovej sústave, tak jednotka na $i$-tom mieste[9] určí, že máme použiť dlhšiu správu z $(k - i)$-tej dvojice a naopak nula na $i$-tom mieste čísla $|L| - k$ určí kratšiu správu z $(k - i)$-tej dvojice.

**ALGORITMUS:** VytvorSprávu$(E, k, L)$

**Premenné:**

1. $L$ = požadovaná dĺžka výslednej správy v blokoch
2. $k$ = parameter udávajúci, že $E$ obsahuje $(k, k+2^k-1)$-expandovateľnú správu
3. $E$ = dvojrozmerné pole kolidujúcich párov rôznych dĺžok
4. $M$ = výsledná správa
5. $T$ = pomocná premenná udávajúca koľko blokov treba ešte pridať správe $M$

**Postup:**

1. Začneme s prázdnou správou $M = \emptyset$
2. if $(L < k)$ or $(L > k + 2^{k-1} - 1)$ then Chybná dĺžka správy
3. Nech $T = L - k$
4. Do $M$ pripájame niektoré časti správy uložené v $E$ tak, aby sme dostali požadovanú dĺžku správy
   - i=0
   - while T>0:
     - if $T > 2^{k-i-1}$ then:
       - $M = M||E[i][2]$
       - $T = T - 2^{k-i-1}$
     - else:
       - $M = M||E[i][1]$
     - i=i+1
5. Vráť správu $M$

**Časová zložitosť:**

$L$ operácií z reťazcami znakov – zanedbateľná v porovnaní s ostatnými postupmi

[9] bity číslujeme od konca, teda prvý bit má hodnotu $2^0$.

**Expandovateľné správy pomocou pevných bodov**

Existuje aj efektívnejšia metóda vytvárania expandovateľných správ v prípade, že hašovacia funkcia umožňuje jednoduché hľadanie pevných bodov. Pevný bod hašovacej funkcie je dvojica $(H_{i-1}, m_i)$ taká, že $H_{i-1} = F(H_{i-1}, m_i)$, teda medzivýsledok kompresnej funkcie sa po spracovaní bloku $m_i$ zopakuje. Výhodou pevného bodu je, že viacnásobným opakovaním bloku $m_i$ môžme predlžovať správu a nemeniť pritom medzivýsledok. Toto nám umožní ešte jednoduchšiu konštrukciu expandovateľných správ, pretože odpadá nutnosť generovania kolidujúcich dvojíc správ rôznej dĺžky.

Všetky kompresné funkcie založené na Davies-Meyerovej konštrukcii, to je napríklad celá trieda SHA funkcií, MD4, MD5 a Tiger, takéto pevné body majú a dajú sa ľahko nájsť – presný algoritmus na hľadanie pevných bodov je v dodatku A. Podstatné je, že tento algoritmus nájde v konštantnom čase dvojicu $(H_{i-1}, m_i)$, kde si môžeme zvoliť blok $m_i$, ale neumožňuje nám nijakú kontrolu nad $H_{i-1}$. Na to, aby sme získali kontrolu nad začiatočným medzivýsledkom expandovateľnej správy, potrebujeme nájsť jeden blok $m_{pref}$ taký, že kompresná funkcia po jeho spracovaní skončí v medzivýsledku $H_{i-1}$ – aby sme mohli nadviazať pevným bodom $(H_{i-1}, m_i)$. Expandovateľná správa pomocou pevného bodu je teda dvojica $(m_{pref}, m_{pb})$.

**ALGORITMUS:** NájdiExpSprávuCezPevnéBody($H_{zac}$)

**Premenné:**

1. $H_{zac}$ = vstupný medzivýsledok expandovateľnej správy
2. $H_{tmp}$ = pomocný medzivýsledok
3. NájdiNáhodnýPevnýBod() = algoritmus, ktorý vráti dvojicu $(H, m)$, ktorá je pevným bodom
4. $A, C$ = zoznamy medzivýsledkov kompresnej funkcie
5. $B, D$ = zoznamy blokov správ
6. $M(i)$ = funkcia, ktorá vráti vždy nový blok správy

**Postup:**

1. Vytvor zoznam $2^{n/2}$ pevných bodov
   - for $i = 1$ to $2^{n/2}$:
     - $(H, m)$ = NájdiNáhodnýPevnýBod()
     - $A[i] = H$
     - $B[i] = m$
2. Vytvor zoznam $2^{n/2}$ medzivýsledkov, ktoré vieme dosiahnuť z $H_{zac}$
   - for $i = 1$ to $2^{n/2}$:

- $H_{tmp} = F(H_{zac}, M(i))$
- $C[i] = H_{tmp}$
- $D[i] = M(i)$

3. Nájdi zhodu medzi zoznamami $A$ a $C$: $i, j$ také, že $A[i] = C[j]$
4. Vráť expandovateľnú správu $(D[j], B[i])$

**Časová zložitosť:**

$2^{n/2+1}$ volaní kompresnej funkcie

Pre $n$-bitovú hašovaciu funkciu teda stačí $2^{n/2+1}$ volaní kompresnej funkcie na to, aby sme týmto postupom zostrojili $(1, 2^k)$-expandovateľnú správu, kde $2^k$ je maximálna dĺžka správy, ktorú vie funkcia spracovať. Vytvorenie správy požadovanej dĺžky je už jednoduchá záležitosť: nakopírujeme blok pevného bodu toľkokrát, koľko treba.

**ALGORITMUS:** VytvorSprávuPB$(L, m_{pref}, m_{pb})$

**Premenné:**

1. $L$ = požadovaná dĺžka výslednej správy v blokoch
2. $m_{pref}$ = prvý blok expandovateľnej správy
3. $m_{pb}$ = druhý (opakovateľný) blok expandovateľnej správy

**Postup:**

1. $M = m_{pref}$
2. for $i = 1$ to $L$:
   - $M = M||m_{pb}$
3. Vráť $M$

**Práca:**

$L$ operácií s reťazcami znakov – zanedbateľná

### 4.4.2 Využitie expandovateľných správ na hľadanie druhého vzoru

Najprv si ukážeme jednoduchý útok na iterovanú triedu hašovacích funkcií bez Merkle-Damgårdovho zosilnenia a potom popíšeme jeho vylepšenú verziu, ktorá pomocou expandovateľných správ dokáže "oklamať" kontrolu o dĺžke správy. Označme si $h$ ľubovoľnú iterovanú hašovaciu funkciu, jej kompresnú funkciu $F$ a veľkosť výsledného odtlačku nech je $n$ bitov.

**Útok na druhý vzor dlhej správy**

Tento útok je schopný nájsť druhý vzor dlhej správy $M$ (dĺžky $L$ blokov[10]) pomocou $2^n/L$ volaní kompresnej funkcie. Postup je jednoduchý: útočník spracuje správu $M$ a uloží si všetky jej medzivýsledky $H_1, \ldots, H_L$. Potom postupným skúšaním hľadá takú správu $M'$, ktorá spĺňa $h(M') = H_i$ pre niektoré $i = 1, \ldots, L$. Teda spracovanie správy $M'$ skončí v tom istom medzivýsledku, ako spracovanie prvých $i$ blokov pôvodnej správy $M$. Potom druhý vzor pre správu $M$ je takisto aj $M'||m_{i+1}|| \ldots ||m_L$. Otázkou je, koľko správ $M'$ musí útočník vyskúšať. Keďže pravdepodobnosť, že výsledok $h(M')$ sa bude zhodovať s niektorým $h_i$ je rovná $2^k/2^n$, stredná hodnota počtu pokusov bude $2^{n-k}$ volaní kompresnej funkcie[11]. Takto nájdený druhý vzor má však takmer určite rôznu dĺžku od pôvodnej správy. Z toho vyplýva, že Merkle-Damgårdove zosilnenie tento útok znemožní, pretože na koniec správy pripojí iný blok pre rôzne dlhé správy a zmení tým výsledný odtlačok.

**Útok pomocou expandovateľných správ**

Expandovateľné správy nám umožnia úspešne previesť popísaný útok na celú triedu iterovaných hašovacích funkcií aj s Merkle-Damgårdovým zosilnením. Vďaka nim dokážeme totiž prispôsobiť dĺžku nájdenej správy tak, aby sa zhodovala s dĺžkou pôvodnej správy a potom sa aj kontrolné bloky aj výsledné odtlačky budú rovnať.

Dlhú správu, ku ktorej chceme nájsť druhý vzor, si označíme $M$. Nech $M$ je dlhá $2^k + k$ blokov, jej bloky označme $m_i$ a medzivýsledky kompresnej funkcie $H_i$. Vytvoríme si expandovateľnú správu $E$. Medzivýsledok kompresnej funkcie po spracovaní $E$ označme $H_{kon}$. Teraz budeme hľadať blok $m_{link}$ taký, ktorý nadpojí správu $E$ na niektorý medzivýsledok pôvodnej správy $M$. Formálne musí pre $m_{link}$ platiť $F(H_{kon}, m_{link}) = H_i$ pre niektoré $i = k+1, \ldots, 2^k + k$ pre $(k, k + 2^k - 1)$-expandovateľnú správu bez pevného bodu a $F(H_{kon}, m_{link}) = H_i$ pre niektoré $i = 1, \ldots, 2^k + k$ pre $(1, k + 2^k - 1)$-expandovateľnú správu s pevným bodom. Keď taký nájdeme, tak pospájame výslednú správu a našli sme druhý vzor ku $M$. Nasledujúci algoritmus nepredpokladá existenciu pevných bodov v hašovacej funkcii.

**ALGORITMUS:** ÚtokNaDlhúSprávu($M_{ciel}$)

**Premenné:**

1. $M_{ciel}$ = správa, ku ktorej treba nájsť druhý vzor
2. $m_{link}$ = blok správy použitý na nadpojenie expandovateľnej správy

[10] pre zjednodušenie, nech $L = 2^k$ pre nejaké $k$.

[11] Každý pokus stojí toľko volaní kompresnej funkcie, aká je dĺžka správy $M'$, ale pre $M'$ dlhú jeden blok prebehne útok v želanom čase $2^{n-k}$.

3. $A$ = zoznam medzivýsledkov kompresnej funkcie
4. $H_{kon}$ = koncový medzivýsledok expandovateľnej správy
5. $H_{IV}$ = inicializačný vektor hašovacej funkcie

**Postup:**

1. $E$ =NájdiExpSprávu$(k, H_{IV})$
2. $H_{kon}$ = medzivýsledok po spracovaní expandovateľnej správy v $E$
3. Vytvor zoznam $A$ pozostávajúci z $2^k + k + 1$ medzivýsledkov, ktoré postupne dostávame pri správe $M_{ciel}$
   - $H_0 = H_{IV}$
   - $m_i = i$-ty blok správy $M_{ciel}$
   - $H_i = F(H_{i-1}, m_i)$
4. Nájdi blok správy $M_{link}$ taký, že $F(H_{kon}, m_{link}) = H_i$ pre nejakú hodnotu $i = k + 1, \ldots, 2^k + k$
5. $E'$ =VytvorSprávu$(E, k, i)$
6. Vráť druhý vzor $E'||m_{link}||m_{i+1}|| \ldots ||m_{2^k+k}$

**Časová zložitosť:**

Je daná sučtom práce na nájdenie expandovateľnej správy a práce na nájdenie bloku $m_{link}$. Vo všeobecnom prípade to je $(k \cdot 2^{n/2+1} + 2^k) + (2^{n-k+1})$ volaní kompresnej funkcie a v prípade, že funkcia obsahuje pevné body je to $3 \cdot 2^{n/2+1} + 2^{n-k+1}$ volaní funkcie $F$.

**Zhrnutie útoku**

Z výslednej zložitosti útoku si môžeme všimnúť, že pre správy kratšie ako $2^{n/2}$ – teda pre $k < n/2$ – leží ťažisko časovej zložitosti v nájdení spájajúceho bloku $m_{link}$, pretože vytvorenie expandovateľnej správy je oveľa rýchlejšie [12]. Z toho môžeme usúdiť, že hašovacie funkcie, ktoré umožňujú rýchle hľadanie pevných bodov sú rovnako odolné voči tomuto útoku na druhý vzor, ako ostatné funkcie

Odhad časovej zložistosti útoku si teda môžeme zjednodušiť na $2^{n-k+1}$ – nájdenie bloku $m_{link}$ – z čoho je vidieť, že útok je tým rýchlejší, čím je pôvodná správa dlhšia. Z praktického hľadiska je zrejmé, že pri bežných správach, alebo dokumentoch o dĺžke rádovo stovky blokov – povedzme 256 blokov – je zrýchlenie tohto útoku oproti úplnému preberaniu pomerne malé – "iba" 128-násobné. Pri ochrane integrity dát, povedzme pevného disku o

[12] Toto je zároveň prípad väčšiny terajších hašovacích funkcií.

veľkosti 256 GB, je veľkosť vstupnej správy $2^{8+38-9} = 2^{37}$ blokov (veľkosť jedného bloku je 512 bitov [13]). V tomto prípade bude nájdenie spájajúceho bloku vyžadovať $2^{n-36}$ volaní kompresnej funkcie, čo je už dosť výrazné zrýchlenie oproti želaným $2^n$ a znamená teoretické zlomenie slabej odolnosti hašovacej funkcie. V praxi to však nemusí byť také vážne, pri dostatočne veľkom výstupe hašovacej funkcie (napr.128 bitov) je to stále za hranicami výpočtovej dosiahnuteľnosti ($2^{92}$ operácií).

Otvorenou otázkou ostáva, či sa náhodou nedajú expandovateľné správy využiť na iný útok, ako na druhý vzor.

## 4.5 Nostradamov útok

Hašovacie funkcie nepotrebujú v každej aplikácii spĺňať všetky základné vlastnosti, ako jednosmernosť, slabú a silnú odolnosť voči kolíziám. Pri každom ich uplatnení je vždy špecifikované, ktoré vlastnosti má použitá funkcia splniť. Autori tohoto útoku J.Kelsey a T.Kohno [KK05] upozorňujú, že tieto vlastnosti spolu prekvapivo súvisia viac, ako sme doteraz videli a bezpečnosť hašovacej funkcie by sme mali prísne posudzovať podľa jej najľahšie porušiteľnej vlastnosti – silnej odolnosti voči kolíziám.

**Príklad: Dokazovanie vedomosti pomocou hašovacej funkcie**

Predstavme si nasledujúci príklad: Na začiatku roka 2006 sa objaví v novinách nasledovný odsek:

> Ja, Nostradamus, zverejňujem MD5 odtlačok $H$ správy, ktorá obsahuje mnohé dôležité predpovede budúcnosti, okrem iného aj ceny akcií niekoľkých najväčších spoločností ku poslednému dňu roku 2006.

Niekedy v januári roku 2007 Nostradamus zverejní správu. Na jej začiatku sú presné ceny akcií spomínaných podnikov a potom pokračuje množstvo hmlistých a ešte nesplnených predpovedí týkajúcich sa ďalekej budúcnosti. Celá správa sa hašuje na $H$.

Hlavná otázka je, či je táto situácia dôkazom toho, že Nostradamus vie predpovedať budúcnosť (resp. ceny akcií na rok dopredu). Na prvý pohľad sa zdá, že áno[14]. Vieme síce, že nájdenie kolízie v MD5 je otázkou pár sekúnd na osobnom počítači [Kli06], ale slabá odolnosť voči kolíziám nebola napadnutá a teda nájdenie druhého vzoru – čo vyzerá, že je nutnou podmienkou na úspech takéhoto útoku – je stále nedosiahnuteľné.

[13] Tak je to pri väčšine súčasných hašovacích funkcií.

[14] A aj na druhý.

Odpoveďou na túto otázku bude nasledujúci odsek, v ktorom objasníme dátovú štruktúru a postup, vďaka ktorým je možné falšovať znalosť nejakej informácie pomocou hašovacích funkcií, v ktorých je možné generovať kolízie.

### 4.5.1 Priebeh Nostradamovho útoku

Nostradamov útok umožňuje útočníkovi vytvoriť odtlačok správy, ktorú celkom nepozná. Tento útok je úzko spätý s multikolíznym útokom [Jo04] a útokom na druhý vzor dlhej správy [KS04]. Najprv popíšeme jeho priebeh a následne vysvetlíme jednotlivé kroky.

1. *Predvýpočet:* Útočník vytvorí tzv. *diamantovú štruktúru*, ktorá začína ľubovoľne zvolenými medzivýsledkami kompresnej funkcie a umožňuje zostrojiť správu, ktorá z hociktorého z týchto medzivýsledkov skončí vo výslednom odtlačku $H$. V tomto okamihu môže útočník zverejniť výsledok $H$ a tvrdiť, že vlastní správu, ktorá sa naňho zobrazí.

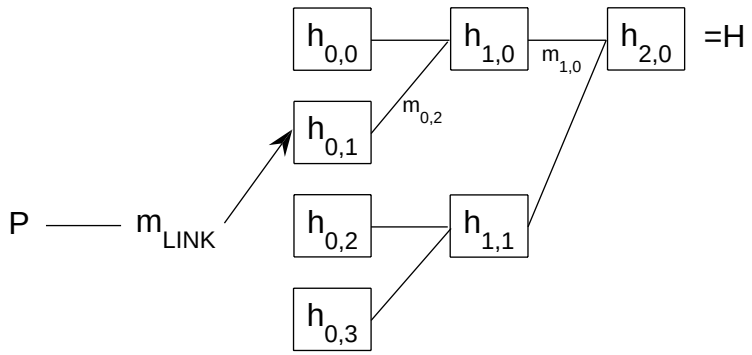
2. *Určenie prefixu:* Útočník sa dozvie prefix $P$ správy, ku ktorej sa zaviazal svojim odtlačkom v prvom kroku.

3. *Nájdenie spájajúcej správy:* Útočník nájde blok $m_{link}$, ktorý môže pripojiť ku $P$, aby kompresná funkcia skončila po spracovaní $P||m_{link}$ v niektorom z medzivýsledkov, ktorými začína predvytvorená diamantová štruktúra.

4. *Vytvorenie správy:* Na záver vytvorí útočník z daného medzivýsledku a diamantovej štruktúry suffix $S$, ktorý pripojí za spájajúci blok a vytvorí správu $P||m_{link}||S$, ktorá sa hašuje na $H$.

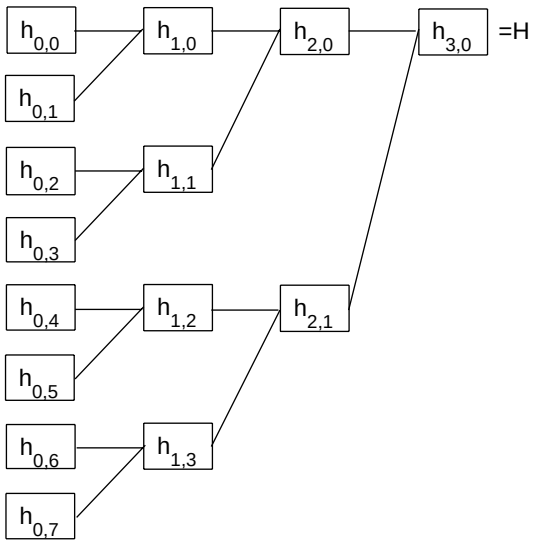
### 4.5.2 Diamantová štruktúra

Táto štruktúra je vlastne multikolízia, ale celkom odlišného tvaru ako sme definovali v časti 4.3. Jej základom je binárny strom, ktorého vrcholy reprezentujú medzivýsledky kompresnej funkcie a hrany reprezentujú časti správ (väčšinou pozostávajú z blokov). Štruktúra je parametrizovaná ľubovoľným prirodzeným číslom $k$ – predstavuje šírku štruktúry. Listy tohto stromu tvorí práve $2^k$ počiatočných medzivýsledkov $h[0,0],\ldots,h[0,2^k-1]$, ktoré si môžeme zvoliť úplne ľubovoľne. Hrana $m[0,i]$ spája vrcholy $h[0,i]$ a $h[1,i\ div\ 2]$ práve vtedy, ak $F(h[0,i],m[0,i]) = h[1,i\ div\ 2]$. Na vytvorenie ďalšieho "poschodia" tohto stromu útočník teda potrebuje nájsť $2^k$ správ, ktoré z nich budú viesť do $2^{k-1}$ medzivýsledkov. Rovnakým postupom potom redukuje $2^{k-1}$ vrcholov na $2^{k-2}$ a takto ďalej až na jeden spoločný medzivýsledok $H$.




Obr. 5: Nostradamov útok. Na obrázku sú zobrazené všetky časti výslednej správy. V tomto prípade sa $P||m_{link}$ zobrazilo na $h_{0,1}$. Výsledná správa po štvrtom kroku útoku je $P||mlink||m_{0,1}||m_{1,0}$.

Výsledkom je multikolízia, ktorá môže začínať v ľubovoľnom z $2^k$ počiatočných medzivýsledkov. Jej vytvorenie trvá o niečo dlhšie: $2^{n/2+k/2+2}$ volaní kompresnej funkcie. Príklad diamantovej štruktúry pre $k = 3$ je na obr.6.




Obr. 6: Diamantová štruktúra.

**Vytváranie diamantovej štruktúry**

Najjednoduchší algoritmus na vytvorenie tejto štruktúry je založený na narodeninovom útoku: vytvoríme ku každému z prvých dvoch vrcholov $h[0,0]$ a $h[0,1]$ $2^{n/2}$ správ a vypočítame $2^{n/2+1}$ medzivýsledkov, v ktorých skončí kompresná funkcia po ich spracovaní z daného počiatku $h[0,0]$ alebo $h[0,1]$. Podľa narodeninového paradoxu sa medzi nimi nachádza kolízia s pravdepodonosťou väčšou ako $1/2$ a teda sme našli vrchol $h[1,0]$ [15]. Takto by sme na zredukovanie $2^k$ na $2^{k-1}$ medzivýsledkov potrebovali priemerne $2^{k+n/2}$ volaní kompresnej funkcie.

Ako lepší spôsob sa ukázalo nefixovať poradie medzivýsledkov a priebežne vytvárať strom počas hľadania kolízií. Na zobrazenie $2^k$ medzivýsledkov na $2^{k-1}$vytvoríme $2^{n/2-k/2+1/2}$ správ pre každý medzivýsledok – spolu teda $2^{n/2+k/2+1/2}$ správ – a vypočítame zodpovedajúce medzivýsledky. Pravdepodobnosť, že sa dva riadky v nejakom medzivýsledku zhodujú je rovná výrazu $(2^{n/2-k/2+1/2})^2 \cdot 2^{-n}$. Po úprave $2^{n-k+1} \cdot 2^{-n} = 2^{-k+1}$. Stredná hodnota počtu riadkov kolidujúcich s nejakým konkrétnym riadkom je teda $2^{-k+1} \cdot 2^k = 2^1 = 2$. Celková práca vykonaná na zredukovanie $2^k$ vrcholov stromu je priemerne $2^{n/2+k/2+3/2}$ a prácu na vytvorenie celej diamantovej štruktúry už dostaneme jednoducho:

$$\begin{aligned}
\sum_{i=1}^{k} 2^{n/2+i/2+3/2} &= 2^{n/2+3/2}\Big(\sum_{i=1}^{k} 2^i\Big)^{1/2} \\
&= 2^{n/2+3/2}\big(2^{k+1}-2\big)^{1/2} \\
&\approx 2^{n/2+3/2}\big(2^{k+1}\big)^{1/2} \\
&= 2^{n/2+k/2+2}
\end{aligned}$$

Je to teda $2^{n/2+k/2+2}$ volaní kompresnej funkcie.

**ALGORITMUS:** VytvorDiamantovúŠtruktúru$(k, A, h)$

**Premenné:**

1. $k$ = parameter štruktúry
2. $A$ = zoznam $2^k$ medzivýsledkov najširšieho "poschodia" diamantovej štruktúry
3. $h$ = hašovacia funkcia ($F$ je jej kompresná funkcia)
4. $n$ = dĺžka medzivýsledkov kompresnej funkcie
5. $B$ = pole $2^k \times (2^{n/2-k/2+1}+1)$ medzivýsledkov hašovacej funkcie

[15] Ak sa kolízia nenašla, zopakujeme postup.

6. $D$ = pole $2^k \times (k+1)$ reprezentujúce vrcholy diamantovej štruktúry – v prvom stĺpci bude $2^k$ medzivýsledkov, v druhom $2^{k-1}$, treťom $2^{k-2}$, ..., v poslednom stĺpci výsledný medzivýsledok $H$
7. $S$ = pole $2^k \times k$, v ktorom sú zapamätané hrany (hrana je dvojica $(m, i)$) diamantovej štruktúry – v prvom stĺpci bude $2^k$ správ a indexov, v druhom $2^{k-1}$, treťom $2^{k-2}$, ..., v poslednom stĺpci dve správy, ktoré zjednotia medzivýsledok na $H$. Index danej správy ukazuje na nasledujúci vrchol štruktúry, pre správu v prvom stĺpci to bude číslo od 1 po $2^{k-1}$, posledné dve správy budú mať index 1. V každom stĺpci budú dva indexy správ rovnaké, to sú tie, ktoré tvoria kolíziu.
8. Správa$(l, j)$ = funkcia, ktorá vráti vždy iný blok správy (prípadne viac blokov, ak tolerujeme dlhší suffix)

**Postup:**

1. Definujeme prvé poschodie (prvý stĺpec) štruktúry:
   $D[i][1] = A[i]$
2. for $i = k$ to 1:
   Zredukuj $2^i$ medzivýsledkov na $2^{i-1}$:
   - for $l = 1$ to $2^i$:
     - for $j = 1$ to $2^{n/2-k/2+1}$:
       * vypočítaj $B[l, j] = F(D[l, i]$,Správa$(l, j))$

   Nájdi kolízie v poli $B$ a doplň príslušné údaje do $D$ a $S$:
   - for $l = 2^{i-1}$ to 1:
     - $(l_1, j_1, l_2, j_2) =$ NájdiKolidujúceRiadky$(B, 2l)$
     - $S[2l, i] = (B[l_1][j_1], 2^{i-1} - l + 1)$
     - $h_{kol} = F(D[l_1][i], B[l_1][j_1])$
     - $D[2l, i+1] = h_{kol}$
     - $S[2l-1, i] = (B[l_2][j_2], 2^{i-1} - l + 1)$
     - $D[2l-1, i+1] = h_{kol}$
3. Vráť štuktúru $(D, S)$

**Časová zložitosť:**

$2^{n/2+k/2+2}$ volaní kompresnej funkcie

Správu, ktorú treba spracovať, aby sa ľubovoľne vybraný medzivýsledok (v nejakom vrchole stromu) zobrazil na výsledný odtlačok $H$, dostaneme zreťazením správ prislúchajúcich ceste od daného vrchola ku koreňu. Keďže

Merkle-Damgårdove zosilnenie pripája na koniec dĺžku správy, tak potrebujeme len, aby boli všetky správy priradené hranám tejto štruktúry rovnako dlhé. Ako sme už spomenuli, používajú sa jednoducho bloky. Potom takto vytvorená diamantová štruktúra vždy vyprodukuje správu dĺžky $k$ blokov.

Ak by sme sa v treťom kroku útoku snažili nadpojiť blok $m_{link}$ na ľubovoľný vrchol diamantovej štruktúry, tak na koniec tejto štruktúry (resp. za koreň stromu) musíme pripojiť $(1, k+1)$-expandovateľnú správu[16], aby sme zabezpečili jednotnú dĺžku celej správy ($k+1$ blokov).

**Aplikácia iných kryptoanalytických útokov**

Popísaný spôsob vytvorenia diamantovej štruktúry sa opieral iba o narodeninový paradox, čo je útok pomocou hrubej sily. Na nájdenie kolízií možno samozrejme využiť aj iné výsledky kryptoanalýzy, konkrétna aplikácia závisí od predpokladov a časovej náročnosti daných útokov:

1. Algoritmus na nájdenie kolízií začínajúcich z rovnakého medzivýsledku sa na vytváranie diamantovej štruktúry nehodí, pretože pre každá dvojica správ, ktorú potrebujeme nájsť, začína v rozných medzivýsledkoch.

2. Algoritmus, ktorý nájde kolíziu pre ľubovoľnú diferenciu inicializačného vektora, sa dá priamo využiť, avšak treba vopred zafixovať pozície $2^k$-tice medzivýsledkov. Ak algoritmus pracuje v čase $2^w$, tak urýchli redukovanie $2^k$ medzivýsledkov na $2^{k-1}$ len v prípade, že $w + k - 1 < n/2 + k/2 + 1/2$, po úprave $w < n/2 - k/2 + 3/2$.

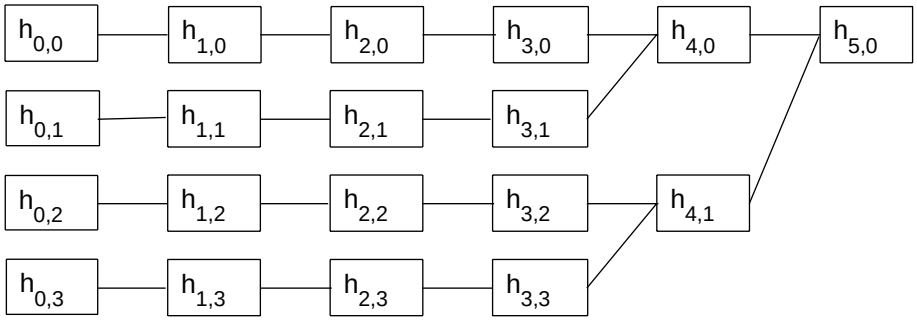
**Predĺžená diamantová štruktúra**

Hlbšie využitie expandovateľných správ nám otvára možnosť zväčšiť počet medzivýsledkov, pred ktoré budeme pripájať blok $m_{link}$ a zmenšiť si tým prácu nutnú v treťom kroku Nostradamovho útoku. Práca na vytvorenie diamantovej štruktúry sa tým výrazne nezmení, zvýši sa ale dĺžka správy vytvorenej diamantovou štruktúrou.

Obrázok 7 zobrazuje predĺženú diamantovú štruktúru. Za každý z $2^k$ medzivýsledkov sme pridali $2^r - 1$ blokov a po každom pridanom bloku vypočítali medzivýsledok. Dostávame takto $2^{k+r}$ medzivýsledkov, do ktorých sa budeme snažiť nadpojiť spájajúci blok $m_{link}$. Zvyšok štruktúry dorobíme, ako v prechádzajúcom prípade. Keďže je malá pravdepodobnosť, že $m_{link}$ trafíme, do $2^k$ prvých medzivýsledkov, tak na záver štruktúry musíme pridať $(1, 2^r+1)$-expandovateľnú správu[17], ktorá zjednotí dĺžku vyprodukovanej

[16] alebo $(lg(k), lg(k) + k - 1)$-expandovateľnú správu, podľa toho, či funkcia umožňuje jednoduché hľadanie pevných bodov.

[17] alebo $(r, 2^r + r - 1)$-expandovateľnú správu, ak nepoužívame pevné body.




Obr. 7: Predĺžená diamantová štruktúra pre $k = 2$, $r = 2$.

správy na $2^r + k + 1$ blokov.

Výsledná práca na zostrojenie takejto $2^r$-predĺženej štruktúry je približne rovná $2^{k+r} + 2^{n/2+k/2+2}$ a štruktúra vždy vytvorí správu dĺžky $2^r + k + 1$.

### 4.5.3 Hľadanie spájajúcej správy

Útočník už teda vytvoril diamantovú štruktúru, zverejnil odtlačok $H$ a dozvedel sa prefix $P$, ktorým má začínať správa hašujúca sa do $H$. Označme $H_{pref}$ medzivýsledok kompresnej funkcie $F$ po spracovaní $P$. Je veľká pravdepodobnosť $(1-2^{k-n})$, že $H_{pref}$ sa nenachádza medzi hodnotami $h[0,0], \ldots, h[0,2^k-1]$. V tomto prípade musí útočník nájsť nejakú časť správy (najlepšie blok[18]) $m_{link}$, pre ktorý platí $F(H_{pref}, m_{link}) = h[0,i]$ pre nejaké $0 \leq i \leq 2^k - 1$. Vzhľadom na pravdepodobnosť úspechu rovnú $2^{k-n}$ je očakávaný počet pokusov na nájdenie vyhovujúceho bloku $2^{n-k}$.

### 4.5.4 Časová zložitosť útoku

V predchádzajúcich častiach sme popísali viacero možností vytvorenia diamantovej štruktúry, teraz rozoberieme časovú zložitosť osobitne pre každú z nich. V základnom prípade má štruktúra $2^k$ medzivýsledkov v najširšom "poschodí" a $m_{link}$ sa snažíme nadpojiť len na týchto $2^k$ medzivýsledkov, takže expandovateľnú správu netreba pripájať na koniec štruktúry. Celková dĺžka suffixu bude $k + 1$ blokov a časová zložitosť celého útoku bude rovná približne:

$$2^{n/2+k/2+2} + 2^{n-k}.$$

[18]Vykonaná práca – počet volaní kompresnej funkcie – bude totiž najmenšia.

| veľkosť výstupu | príklad | šírka štruktúry ($k$) | dĺžka výstupu (blokov) | práca |
|---|---|---|---|---|
| 128 | MD5 | 41 | 48 | $2^{87}$ |
| 160 | SHA1 | 52 | 59 | $2^{108}$ |
| 192 | Tiger | 63 | 70 | $2^{129}$ |
| 256 | SHA-256 | 84 | 92 | $2^{172}$ |
| 512 | Whirpool | 169 | 178 | $2^{343}$ |
| $n$ | | $(n-5)/3$ | $k + lg(k) + 1$ | $2^{n-k}$ |

Tabuľka 2: Nostradamov útok.

Ak sa snažíme nadpojiť blok $m_{link}$ na ktorýkoľvek medzivýsledok štruktúry, tak sa zvýši pravdepodobnosť úspechu v treťom kroku na dvojnásobok (presnejšie na $(2^{k+1}-1)/2^n$), ale bude treba zostrojiť a pripojiť $(lg(k), lg(k)+k)$-expandovateľnú správu [19]. Dĺžka suffixu sa predĺži na $lg(k)+k+1$ a celková práca v tomto prípade bude:

$$2^{n/2+k/2+2} + 2^{n-k-1} + lg(k) \cdot 2^{n/2+1},$$

kde $lg(k) \cdot 2^{n/2+1}$ je zložitosť vytvorenia $(lg(k), lg(k)+k-1)$-expandovateľnej správy.

Našim cieľom je zvoliť parameter $k$ tak, aby bola celková práca bola čo najmenšia. Keďže zvyšovanie parametra $k$ zmenšuje prácu potrebnú na nájdenie spájajúceho bloku a zvyšuje prácu na vytvorenie diamantovej štruktúry, tak najmenšia celková práca bude dosiahnutá vtedy, ak sa tieto dve budú rovnať (vytvorenie expandovateľnej správy môžeme zanedbať). Niekoľko hodnôt pre konkrétne hašovacie funkcie je zobrazených v tabuľke 2.

V prípade predĺženej diamantovej štruktúry sa zvýši práca na vytvorenie štruktúry, uľahčí sa hľadanie bloku $m_{link}$ a bude sa musieť nájsť $(r, 2^r + r)$-expandovateľná správa. Dĺžka suffixu bude $2^r + k + 1 + r$ a práca:

$$2^{k+r} + 2^{n/2+k/2+2} + 2^{n-k-r} + r \cdot 2^{n/2+1}.$$

Veľkosť parametra $r$ ovplyvňuje dĺžku správy, ktorú týmto útokom vyprodukujeme. Príliš veľké hodnoty síce zmenšia prácu potrebnú na úspešný útok, ale vyprodukujú neprakticky dlhé správy, tak treba voliť vhodný kompromis. Časová zložitosť Nostradamovho útoku je pre niektoré hodnoty $r$ uvedená v tabuľke 3.

[19] Predpokladajme, že hašovacia funkcia neumožňuje rýchle hľadanie pevných bodov, tak zostrojujeme všeobecnú expandovateľnú správu.

| veľkosť výstupu | príklad | šírka štruktúry ($k$) | dĺžka výstupu (blokov) | práca |
|---|---|---|---|---|
| 128 | MD5 | 5 | $2^{55}$ | $2^{69}$ |
| 160 | SHA1 | 16 | $2^{55}$ | $2^{90}$ |
| 192 | Tiger | 27 | $2^{55}$ | $2^{111}$ |
| 256 | SHA-256 | 48 | $2^{55}$ | $2^{154}$ |
| 512 | Whirpool | 133 | $2^{55}$ | $2^{325}$ |
| 512 | Whirpool | 6 | $2^{246}$ | $2^{261}$ |
| $n$ | | $(n-2r-3)/3$ | $2^r$ | $2^{n-k-r+1}$ |

Tabuľka 3: Nostradamov útok pomocou predĺženej diamantovej štruktúry.

### 4.5.5 Dôsledky

Nostradamov útok je viac špecifický – má vymedzený scenár a užšie použitie – ako predchádzajúce dva, ktoré sme popísali v tejto časti. Ďalšie aplikácie tohto útoku sú podrobnejšie popísané v [KK05].

Ako si môžeme všimnúť, na zvýšenie efektívnosti elegantne využíva expandovateľné správy a takisto sa opera o pojem multikolízie – obavy z ďalšieho využitia expandovateľných správ a multikolízií sa teda naplnili.

Hlavné ponaučenie z Nostradamovho útoku je, že hašovacie funkcie, v ktorých je možné hľadať kolízie, nemožno už viac používať na dokazovanie vedomosti nejakej informácie alebo "commitment" protokoly.

## 4.6 Záver

V tejto časti práce sme predstavili hlavné nedostatky Merkle-Damgårdovej konštrukcie hašovacích funkcií. Zatiaľčo útok predlžovaním správy a predlžovanie kolízií boli známymi slabinami tejto konštrukcie takmer od jej vzniku, Jouxov útok sa objavil po dosť dlhom čase jej používania a obrazne povedané zatriasol touto konštrukciou. Nesporne tiež oživil diskusiu o bezpečnosti tejto konštrukcie a následne Kelsey a Schneier predstavili pojem expandovateľných správ a spôsob ako pomocou nich hľadať druhé vzory správ v čase $2^n/L$, kde $L$ je počet blokov pôvodnej správy.

Oba tieto útoky dokazujú, že hašovacia funkcia Merkle-Damgårdovej triedy sa nechová ako náhodné orákulum a aj keď prenáša odolnosť voči kolíziám z kompresnej funkcie $F$ na celú hašovaciu funkciu $h$, tak v prípade, že sa kompresná funkcia ukáže neodolná voči kolíziám, Merkle-Damgårdova konštrukcia túto chybu ešte rozšíri na ďalšie vlastnosti funkcie $h$, napríklad zníži odolnosť voči nájdeniu druhého vzoru alebo jednosmernosť.

Treba zároveň pripomenúť, že tieto útoky sú čisto teoretické a sami o sebe nie sú hrozbou pre nejakú aplikáciu. Odhaľujú však bezpečnostné diery, ktoré *možno* zúžitkujú iné útoky, ako tomu bolo napríklad v prípade Nostradamovho útoku.

Dôsledky týchto štrukturálnych slabín Merkle-Damgårdovej triedy funkcií ešte umocnili konkrétne útoky na kompresné funkcie niektorých hašovacích funkcií, menovite SHA-0 [Bih05] a [WY05] ďalej SHA-1 [Wa05] a MD5 [WH05], [Kli06]. Tieto útoky posunuli časovú zložitosť nájdenia kolízie do prakticky dosiahnuteľných medzí (neplatí to pre SHA-1, ale očakáva sa jej zlomenie v najbližšom čase) a znížili tým zložitosť útokov z tejto časti na vymenované hašovacie funkcie.

Z uvedených nedostatkov vyplynula nutnosť zmeniť, alebo aspoň vylepšiť aktuálnu Merkle-Damgårdovu konštrukciu. NIST zareagoval usporiadaním dvoch konferencií zameraných na hodnotenie a riešenie tohto problému. Výsledkom druhej z týchto konferencií bolo rozhodnutie o usporiadaní verejnej súťaže na novú hašovaciu funkciu[20].

[20]Inšpirované úspechom pri výbere AES.

# 5 Vylepšenia Merkle-Damgårdovej konštrukcie

Jedna rodina vylepšení je založená na pojmoch "nerozlíšiteľnosti" (*indistinguishability*) a "indiferentnosti" (*indifferentiability*), ktoré boli definované v práci [Ma03]. Na tomto princípe J.S.Coron a kol. [Co05] vytvorili štyri návrhy konštrukcie hašovacej funkcie, ktorá bola nerozlíšiteľná od náhodného orákula za predpokladu, že kompresná funkcia bola nerozlíšiteľná od náhodného orákula pevnej dĺžky. Tieto konštrukcie boli vylepšené M.Bellarom a T.Ristenpartom v [BR06]. Týmto pojmom a konštrukciám sa však v práci nebudeme venovať.

## 5.1 Nová štruktúra hašovacej funkcie

V roku 2004, Stefan Lucks v [Lu04] navrhol dva spôsoby úpravy Merkle-Damgårdovej konštrukcie, ktoré riešia doterajšie "generické" problémy popísané v časti 4. Sú to tzv. rozšírený výpočet – *Wide-pipe hash* – a dvojitý výpočet kompresnej funkcie – *Double-pipe hash*[21]. Obe tieto konštrukcie sú len štrukturálnym vylepšením Merkle-Damgårdovej konštrukcie, taktiež sa nezaoberajú konkrétnou kompresnou funkciou a predpokladajú, že kompresná funkcia je rovnako bezpečná ako náhodné orákulum pevnej dĺžky.

Keďže multikolízie aj expandovateľné správy vyžadovali hľadanie kolízií kompresnej funkcie, tak sťaženie vytvorenia obyčajnej kolízie kompresnej funkcie predĺži aj úspešné hľadanie multikolízie alebo expandovateľnej správy. A práve znemožnenie hľadania kolízií bolo cieľom týchto vylepšení.

Oba spôsoby takisto zamedzujú aj Nostradamovmu útoku [KK05], pretože je tiež založený na nájdení dostatočného množstva kolízií, aj keď v čase zverejnenia týchto konštrukcií Nostradamov útok ešte nebol známy.

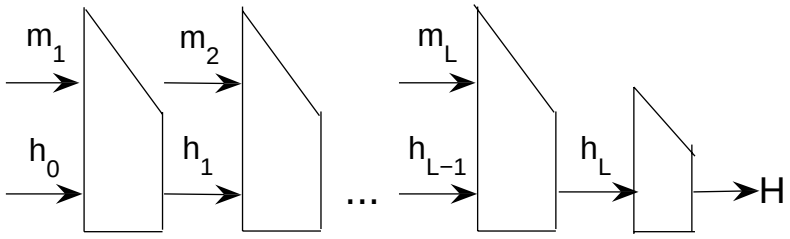
### 5.1.1 Wide-pipe hash

Jednoduché *rozšírenie*[22] výstupu kompresnej funkcie na $w$ bitov, kde $w > 2n$, je jedna možnosť, ako stanovený cieľ splniť. Preto boli v tejto konštrukcii definované dve kompresné funkcie $C'$ a $C''$

- $C' : \{0,1\}^w \times \{0,1\}^b \rightarrow \{0,1\}^w$ a
- $C'' : \{0,1\}^w \rightarrow \{0,1\}^n$

Výpočet takto rozšírenej $n$-bitovej hašovacej funkcie $h$ pozostáva z troch častí:

[21] budeme ich ďalej označovať pôvodnými anglickými názvami.
[22] odtiaľ pochádza názov konštrukcie.




Obr. 8: "Wide-pipe hash" podľa S.Lucksa.

– na predspracovanie správy sa použije Merkle-Damgårdove zosilnenie, správa $M$ sa rozdelí na $L$ blokov (každý dĺžky $b$), pričom posledný obsahuje dĺžku pôvodnej správy
– pre $i = 1, \ldots, L : H_i = C'(H_{i-1}, m_i)$
– výsledný odtlačok $h(M) = C''(H_L)$

Priebeh výpočtu je uvedený na obrázku 8.

**Hľadanie $k$-kolízie**

Ako horný odhad zložitosti útoku na $k$-kolízie funkcie $h$ založenej na konštrukcii wide-pipe slúži ohraničenie $min\{lg(k)2^{w/2}, 2^{n(k-1)/k}\}$. Prvý člen reprezentuje zložitosť Jouxovho útoku a druhý člen zátvorky je zložitosť klasického narodeninového útoku na $n$-bitovú funkciu $h$ ako na čiernu skrinku. Ako ukážeme[23] zachvíľu, tento odhad sa zhoduje s dolným až na násobok $lg(k)$.

**Lema 5.1.1.** *Nech $T'$ označuje počet operácií potrebných na nájdenie vnútornej kolízie – teda kolízie funkcie $C'$ a $T''(k)$ nech označuje čas na nájdenie $k$-kolízie vo funkcii $C''$. Potom nájdenie $k$-kolízie v hašovacej funkcii so štruktúrou wide-pipe vyžaduje $\Omega(min\{T', T''(k)\})$ operácií.*

*Dôkaz.* Uvažujme kolíziu $M \neq N$, $h(M) = h(N)$. $M$ a $N$ boli predspracované na postupnosti blokov $(m_1, \ldots, m_L) \neq (n_1, \ldots, n_{L'})$. Označme $H_i^M$ a

[23]Uvedené analýzy bezpečnosti konštrukcie je čerpaná z [Lu04], niektoré dôkazy sme doplnili o chýbajúce medzikroky s cieľom zvýšiť zrozumiteľnosť postupov.

$H_j^N$ medzivýsledky počítané počas hašovania $M$ a $N$. Rozlíšime tri možné prípady:

**rozdielna dĺžka:** $L \neq L'$ implikuje $m_L \neq n_{L'}$. Potom prípad, keď $H_L^M = H_{L'}^N$ implikuje internú kolíziu a prípad $H_L^M \neq H_{L'}^N$ implikuje koncovú kolíziu.

**koncová kolízia:** $H_L^M \neq H_{L'}^N$ a $C''(H_L^M) = C''(H_{L'}^N)$.

**vnútorná kolízia:** $(H_L^M, m_L) = (H_{L'}^N, n_{L'})$ a teda $L = L'$. Keďže $(m_1, \ldots, m_L) \neq (n_1, \ldots, n_{L'})$, tak existuje kolízia v $C' : (H_i^M, m_i) \neq (H_i^N, n_i)$ pričom $C'(H_i^M, m_i) = C'(H_i^N, n_i)$.

Každá $k$-kolízia funkcie $h$ sa teda redukuje buď na $k$-kolíziu kompresnej funkcie $C''$, alebo aspoň jednu (vnútornú) kolíziu $C'$. □

**Tvrdenie 5.1.2.** *Predpokladajme, že kompresné funkcie $C'$ a $C''$ použité vo wide-pipe konštrukcii sú náhodné orákulá. Potom nájdenie $k$-kolízie vyžaduje čas $\Omega(min\{2^{w/2}, 2^{n(k-1)/k}\})$ volaní kompresnej funkcie.*

*Dôkaz.* Tvrdenie vyplýva z toho, že vnútorná kolízia trvá aspoň $2^{w/2}$ operácií, $k$-kolízia funkcie $C''$ aspoň $2^{n(k-1)/k}$ a z Lemy 5.1.1. □

**Hľadanie $k$-vzoru a druhého $k$-vzoru**

Jouxov postup na hľadanie $k$-vzoru aj druhého $k$-vzoru funguje rovnako aj pri tejto rozšírenej konštrukcii. Jeho časová náročnosť je $O(lg(k)2^{w/2} + 2^n)$ a ukážeme, že dolný odhad je taký istý, až na člen $lg(k)$.

**Lema 5.1.3.** *Počet operácií potrebných na nájdenie vnútornej kolízie označíme $T'$ a $P''(k)$ nech označuje čas na nájdenie $k$-vzoru funkcie $C''$. Potom pre hašovaciu funkciu $h$ so štruktúrou wide-pipe platí:*

1. *Nájdenie vzoru funkcie $h$ trvá $\Omega(P''(1))$.*
2. *Nájdenie $k$-vzoru funkcie $h$ zaberie $\Omega(min\{T', P''(k)\})$.*

*Dôkaz.* Prvá časť Lemy vyplýva z toho, že nájdenie vzoru funkcie $h$ (nejakého $M$, aby $h(M) = Y$) implikuje nájdenie takého $H_L$, aby $C''(H_L) = Y$.

Druhá časť: hľadanie $k$ rôznych vzorov $m^1, \ldots, m^k$ hašovacej funkcie $h$ buď vyžaduje aspoň jednu kolíziu $C'$, alebo nájdenie $k$ rôznych medzivýsledkov $H_{L^1}^1, \ldots, H_{L^k}^k$ aby platilo $C''(H_{L^1}^1) = \ldots = C''(H_{L^k}^k) = Y$, teda $k$-vzor kompresnej funkcie $C''$. □

**Tvrdenie 5.1.4.** *Uvažujme rozšírenú hašovaciu funkciu $h$. Ak pre kompresné funkcie $C'$ a $C''$ predpokladáme, že sú náhodné nezávislé orákulá, tak*

– *nájdenie jednoduchého vzoru trvá* $\Omega(2^n)$,
– *nájdenie* $k$*-násobného vzoru a druhého vzoru trvá* $\Omega(min\{2^{w/2}, k2^n\})$ *volaní kompresnej funkcie.*

*Dôkaz.* Prvá časť tvrdenia vyplýva z toho, že $C''$ je náhodné orákulum a teda nájdenie vzoru bude trvať $\Omega(2^n)$ operácií. Druhá časť plynie z Lemy 5.1.3 a toho, že $P''(k) = k2^n$ pre náhodné orákulum $C''$. □

**Zhrnutie bezpečnosti wide-pipe konštrukcie**

Zvyšovanie parametra $w$ zvyšuje odolnosť hašovacej funkcie $h$ voči objaveniu vnútornej kolízie a tým aj voči všetkým útokom, ktoré sa o ne opierali. V našom modeli, kde kompresná funkcia je reprezentovaná náhodným orákulom postačuje $w \geq 2n$ na dosiahnutie ideálnej bezpečnosti $h$ voči multikolíziám. Pri hľadaní $k$-násobných vzorov sa (kvôli Jouxovmu útoku) nedá dosiahnuť ideálna bezpečnosť pre ľubovoľne veľké $k$, ale pre dané $k$ vieme zvoliť $w$, aby $h$ bola rovnako bezpečná ako náhodné orákulum.

### 5.1.2 Double-pipe hash

Predchádzajúca konštrukcia, opierajúca sa o rozšírenie výstupu vnútornej kompresnej funkcie, má jednu výraznú nevýhodu: na dosiahnutie želanej bezpečnosti $n$-bitovej hašovacej funkcie potrebujeme primitívum – stavebný prvok – s extrémne vysokou mierou bezpečnosti. Konkrétne sme potrebovali, aby nájdenie kolízie funkcie $C'$ zabralo aspoň $2^n$ operácií ($w \geq 2n$ je nutná, ale nie postačujúca podmienka).

Otázkou je či vieme navrhnúť iterovanú konštrukciu a dokázať jej stupeň bezpečnosti bez toho, aby sme predpokladali, že niektorý jej stavebný prvok je bezpečnejší, ako ona sama.

Pozitívnou odpoveďou by mala byť nasledovná konštrukcia double-pipe hash. Použitím jednej kompresnej funkcie $C$:

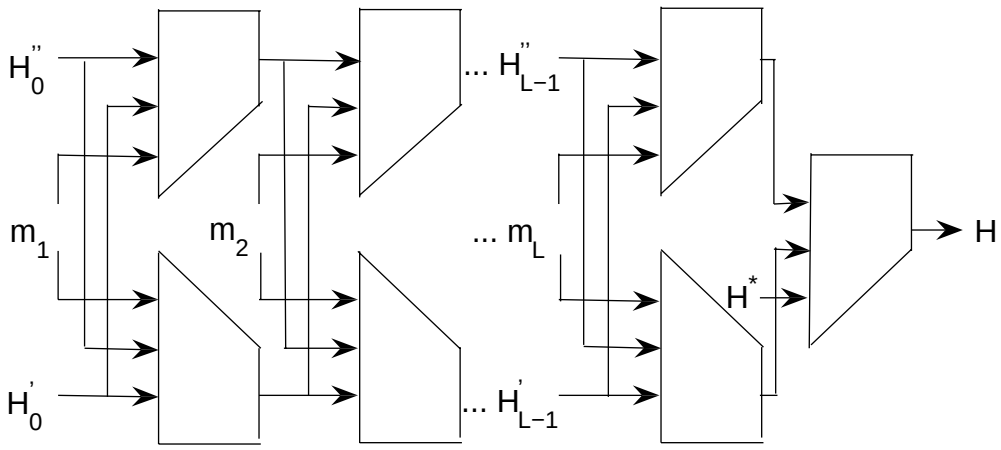
– $C : \{0,1\}^n \times \{0,1\}^{n+b} \rightarrow \{0,1\}^n$, $(b \geq n)$,

a troch rôznych (náhodných a nezávislých) inicializačných hodnôt $H_0{}', H_0{}'', H^* \in \{0,1\}^n$, definujeme výpočet hašovacej funkcie $h$ týmito troma krokmi:

– vstupnú správu $M$ predspracujeme podľa Merkle-Damgårdovho zosilnenia a rozdelíme na bloky $m_1, \ldots, m_L$,

– pre $i = 1, \ldots, L$ počítame

  • $H'_i = C(H'_{i-1}, H''_{i-1}||m_i)$,




Obr. 9: Štruktúra výpočtu "Double-pipe hash".

- $H''_i = C(H''_{i-1}, H'_{i-1}||m_i)$,

– výsledný odtlačok $h(M) = C(H^*, H_L{}'||H_L{}''||0^{b-n})$.

**Hľadanie $k$-kolízie**

Podobne ako v predchádzajúcom prípade rozlíšime vnútorné kolízie (tie, kde do kompresnej funkcie vstupuje blok $m_i$) a koncové kolízie (posledná iterácia $C$ s tretím inicializačným vektorom $H^*$).

**Vnútorná kolízia:** dve trojice $(H', H'', m_1) \neq (G', G'', m_2)$, pre ktoré:

$$C(H', H''||m_1) = C(G', G''||m_2),$$

$$C(H'', H'||m_1) = C(G'', G'||m_2).$$

**Koncová kolízia:** dvojice $(H', H'') \neq (G', G'')$, kde

$$C(H^*, H'||H''||0^{b-n}) = C(H^*, G'||G''||0^{b-n}).$$

Zvýšená bezpečnosť rozšírenej konštrukcie Wide-pipe vyplývala z faktu, že odolnost voči vnútorným kolíziám bola väčšia ako celková odolnosť voči kolíziám. Podobný argument sa tu nedá využiť, lebo nájdenie interných kolízií,

kde $H' = H''$ a $G' = G''$ je tak jednoduché, ako pri klasickej kompresnej funkcii s $n$-bitovým výstupom. Rozlíšime preto vnútorné kolízie ešte na striktné a krížové:

**striktná vnútorná kolízia:** je vnútorná kolízia, kde

$$H' \neq H'' \wedge G' \neq G'',$$

**krížová vnútorná kolízia:** je taká trojica $(H'_{i-1}, H''_{i-1}, m_i)$, aby platilo

$$C(H'_{i-1}, H''_{i-1}||m_i) = H'_i = H''_i = C(H''_{i-1}, H'_{i-1}||m_i).$$

Označme $T_S$ čas, potrebný na nájdenie striktnej vnútornej kolízie, $T_X$ čas nájdenia vnútornej krížovej kolízie a $T(K)$ nech označuje čas nutný na nájdenie koncovej $k$-kolízie.

**Lema 5.1.5.** *Uvažujme hašovaciu funkciu $h$ s kompresnou funkciou $C$ typu double-pipe. Potom:*

- *každá vnútorná kolízia vyžaduje buď striktnú, alebo krížovú kolíziu.*
- *nájdenie $k$-kolízie vyžaduje čas $\Omega(min\{T_S, T_X, T(k)\})$.*

*Dôkaz.* Prvú časť dokážeme sporom. Na začiatok si všimnime, že oba inicializačné vektory $H'_0$, $H''_0$ sú rôzne. Predpokladajme, že sa vyskytla vnútorná kolízia a nebola ani striktná, ani krížová. Keďže nebola striktná, tak to znamená, že sa v nejakom kroku vyskytla trojica vstupov $(H'_{i-1}, H''_{i-1}, m_i)$, pre ktorú $H'_{i-1} = H''_{i-1}$. Vezmime najmenšie také $i$, kde $H'_{i-1} = H''_{i-1}$. Potom museli existovať $H'_{i-2}, H''_{i-2}, m_{i-1}$ také, že

$$C(H'_{i-2}, H''_{i-2}||m_{i-1}) = H'_{i-1} = H''_{i-1} = C(H''_{i-2}, H'_{i-2}||m_{i-1}).$$

To ale znamená výskyt krížovej kolízie.

Druhú časť dokážeme podobne ako v Leme 5.1.1: $k$-kolízia $h$ totiž vyžaduje buď koncovú $k$-kolíziu (čas $\Omega(T(k))$), alebo nejakú vnútornú kolíziu. Z prvej časti tejto lemy vyplýva, že je to buď striktná (čas $\Omega(T_S)$), alebo krížová vnútorná kolízia (čas $\Omega(T_X)$). □

**Tvrdenie 5.1.6.** *Uvažujme double-pipe hašovaciu funkciu $h$. Nech kompresná funkcia je modelovaná náhodným orákulom pevnej dĺžky, potom:*

- $T_S = \Omega(2^n)$ *a* $T_X = \Omega(2^n)$,

– *nájdenie k-kolízie pre H vyžaduje čas* $\Omega(2^{n(k-1)/k})$.

*Dôkaz.* Prvá časť: nájdenie striktnej kolízie. Vezmime trojicu $t_1 = (H', H'', M)$, kde $H' \neq H''$. Definujme $H_{t1} = C(H', H''||M)$ a $H'_{t1} = C(H'', H'||M)$. Za predpokladu, že $C$ je náhodné orákulum pevnej dĺžky, tak $(H_{t1}, H'_{t1}) \in \{0,1\}^{2n}$ je náhodná premenná s rovnomerným rozdelením. Trojica $t_1$ tvorí s trojicou $t_2$ striktnú kolíziu práve vtedy, ak $H_{t1} = H_{t2}$ a $H'_{t1} = H'_{t2}$. Pravdepodobnosť tejto udalosti je $2^{-2n}$, pretože $H_{t1}$ je nezávislá od $H'_{t1}$. Čas jej nájdenia postupným skúšaním je teda $T_S = \Omega(2^n)$.

Pozrime sa teraz na množinu $q$ trojíc a označme $P_S$ pravdepodobnosť udalosti, že sa medzi nimi nachádza striktná kolízia. Potom $P_S \leq \sum_{0 \leq j \leq q} j/2^{2n} = \Omega(q^2/2^{2n})$. Z toho vyplýva, že počet pokusov $q$ musí byť aspoň $2^n$ a teda $T_S = \Omega(2^n)$.

Nájdenie krížovej kolízie. Vyberme $H', H''$ a $M$ tak, aby $H' \neq H''$. Pravdepodobnosť, že $C(H', H''||M) = C(H'', H'||M)$ je rovná $2^{-n}$, pretože $C$ je náhodné orákulum. Z toho ale vyplýva, že priemerný počet výberov trojíc na dosiahnutie rovnosti je $2^n$ a teda $T_X = \Omega(2^n)$.

Druhá časť vyplýva z predchádzajúcej lemy, prvej časti a faktu, že $T(k) = \Omega(2^{n(k-1)/k})$ (pretože $C$ je náhodné orákulum a nájdenie $k$-kolízie náhodného orákula trvá $\Omega(2^{n(k-1)/k})$). □

**Hľadanie $k$-vzoru**

Situácia je veľmi podobná, ako pri hľadaní $k$-vzoru v rozšírenej kompresnej funkcii. Čas potrebný na nájdenie striktnej kolízie označujeme $T_S$, symbolom $T_X$ čas na nájdenie krížovej. $P(k)$ bude značiť čas nájdenia $k$-vzoru pre funkciu $C$.

**Lema 5.1.7.** *Uvažujme hašovaciu funkciu h s dvojitým výpočtom kompresnej funkcie. Potom:*

– *hľadanie jednoduchého vzoru vo funkcii h vyžaduje* $\Omega(P(1))$ *operácií*

– *nájdenie k-vzoru funkcie h vyžaduje čas* $\Omega(min\{T_S, T_X, P(k)\})$.

*Dôkaz.* Dôkaz prvej časti je analogický, ako v Leme 5.1.3. Druhá časť vyplýva priamo z druhého tvrdenia Lemy 5.1.5. □

**Tvrdenie 5.1.8.** *Uvažujme hašovaciu funkciu h s dvojitým výpočtom kompresnej funkcie. Nech kompresná funkcia je modelovaná náhodným orákulom pevnej dĺžky, potom nájdenie k-vzoru ako aj druhého k-vzoru funkcie h vyžaduje* $\Omega(2^n)$ *volaní kompresnej funkcie.*

*Dôkaz.* Na nájdenie $k$-vzoru funkcie $h$ je potrebný buď $k$-vzor funkcie $C$ ($\Omega(k2^n)$), alebo aspoň jedna interná kolízia ($\Omega(2^n)$). □

### 5.1.3 Double-pipe hash na báze blokovej šifry

Nakoľko väčšina hašovacích funkcií má kompresnú funkciu založenú na nejakej blokovej šifre, najčastejšie pomocou Davies-Meyerovej konštrukcie, S.Luck v [Lu04] analyzoval bezpečnosť konštrukcie double-pipe pre kompresnú funkciu skonštruovanú pomocou ideálnej blokovej šifry[24].

Výsledky analýzy sú taktiež uspokojivé, nájdenie vnútornej kolízie vyžaduje čas $\Omega(2^n)$, $k$-kolízia vyžaduje $\Omega(2^{n(k-1)/k})$ a $k$-vzor, resp.druhý $k$-vzor potrebuje $\Omega(2^n)$ operácií, dôkazy pre oveľa komplikovanejšie značenie a technickú náročnosť vynechávame.

## 5.2 3C a 3C+ konštrukcia

Tieto konštrukcie boli navrhnuté v roku 2005 kolektívom P.Gauravaram, W.Millan, E.Dawson, J.G.Nieto a K.Viswanathan [GM05] a [GM06]. Ich cieľom je pomocou čo najmenších úprav zlepšiť Merkle-Damgårdov princíp vytvárania hašovacích funkcií, hlavne z pohľadu posledných útokov na SHA-0 [Bih05], [WY05], SHA-1 [Wa05] a MD5 [WH05], [Kli06] (skrátene MBCA) [25]. Taktiež chcú týmto dosiahnuť lepšiu odolnosť voči "generickým" útokom, ktoré sme prezentovali v časti **4**. Autori predstavujú konštrukciu $3C+$, ako ďalšie vylepšenie $3C$ z hľadiska MBCA.

### 5.2.1 Štruktúra 3C a 3C+

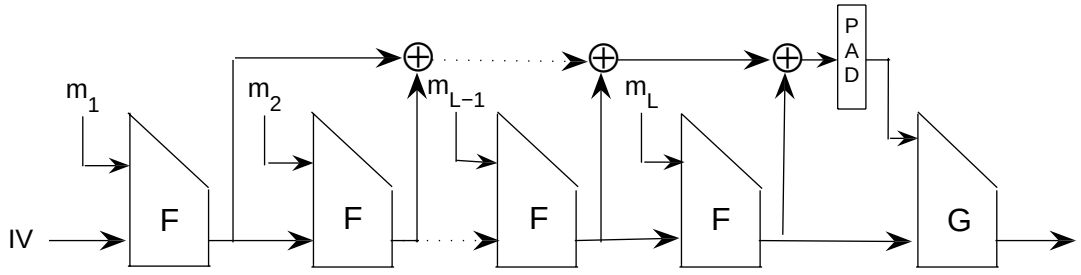
Obe konštrukcie si zachovávajú Merkle-Damgårdove predspracovanie správy a iterovanú štruktúru založenú na kompresnej funkcii $F : \{0,1\}^{b+n} \rightarrow \{0,1\}^n$, kde $b$ je dĺžka jednotlivých blokov a $b \geq n$. Očakávané zvýšenie bezpečnosti má priniesť tzv. akumulačná reťaz (obr. 10), ktorá pomocou funkcie XOR "akumuluje" jednotlivé medzivýsledky kompresnej funkcie, a výstupná transformácia $G : \{0,1\}^{b+n} \rightarrow \{0,1\}^n$.

Formálne, ak symbolom $B_i$ označíme medzivýsledok na akumulačnej reťazi po spracovaní $i$-teho bloku, tak priebeh výpočtu $n$-bitovej hašovacej funkcie $h$ definujeme takto:

- vstupná správa $M$ sa predspracuje na $L$ blokov (každý dĺžky $b$), pričom posledný obsahuje dĺžku pôvodnej správy $M$
- $H_0 = H_{IV}$
  $B_0 = 0$
  pre $i = 1, \ldots, L$:

[24] nazývanej aj Shannonove orákulum.

[25] Tieto útoky majú spoločnú črtu, že hľadajú kolízie v daných funkciách pomocou viacblokových správ, preto dostali názov multi-block collision attacks – MBCA.




Obr. 10: Konštrukcia 3C.

- $H_i = F(H_{i-1}, m_i)$
- $B_i = B_{i-1} \oplus H_i$

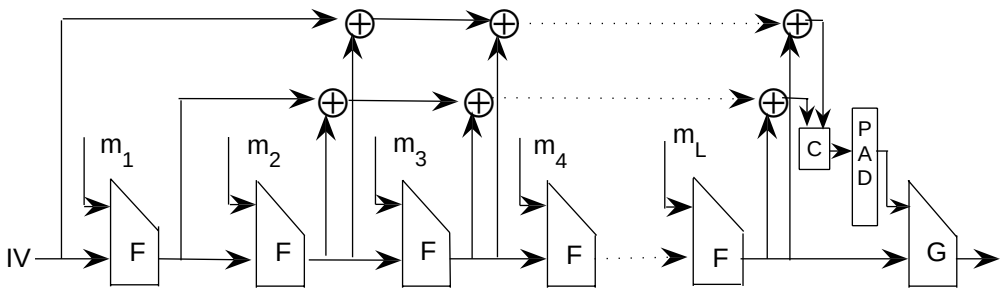
– výsledný odtlačok $h(M) = G(B_L||0^{b-n}, H_L)$

Veľkou výhodou konštrukcie $3C$ podľa autorov je, že ľubovoľný systém využívajúci nejakú hašovaciu funkciu Merkle-Damgårdovej triedy môže byť jednoducho a ľahko pozmenený na $3C$ štruktúru, bez nutnosti zmeny kompresnej funkcie [26]. Jednoduchosť implementácie zabezpečuje minimálna zmena v doterajšej Merkle-Damgårdovej konštrukcii, akumulovanie medzivýsledkov prebieha vďaka zvolenej funkcii XOR rýchlo a jediný krok navyše je už len posledný výpočet funkcie $G$.

Autori v pôvodnej verzii $3C$ namiesto funkcie XOR navrhovali použitie všeobecnej funkcie $f' : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ v akumulačnej reťazi, kvôli efektívnosti a zjednodušenej implementácii potom inštancovali $f'$ funkciou XOR.

Pridaním ďalšej akumulačnej reťaze (taktiež realizovanú funkciou XOR) do konštrukcie $3C$ vzniká jej vylepšenie $3C+$. Túto druhú reťaz nazývajú autori aj finálna, my ju budeme označovať ako vrchnú, resp. hornú reťaz. Ako je vidieť z obrázka 11, vrchná reťaz začína už inicializačným vektorom a vynecháva prvý medzivýsledok na rozdiel od pôvodnej prvej reťaze. Ak teda označíme symbolom $A_i$ medzivýsledok na hornej reťazi po spracovaní bloku $m_i$, tak platí $A_i = B_i \oplus H_{IV} \oplus H_1$. Funkcia $C$ zreťazuje posledné medzivýsledky $A_L$ a $B_L$, $C(a,b) = a||b$.

[26] Ako ukážeme neskôr, toto nie je celkom pravda.




Obr. 11: Vylepšená konštrukcia 3C+.

### 5.2.2 Bezpečnostná analýza 3C

Autori vo svojom článku [GM06] dokázali, že konštrukcia $3C$ je aspoň tak odolná voči MBCA, ako pôvodná Merkle-Damgårdova konštrukcia. Navyše ukázali, že $3C$ zamedzuje útoku predĺžením správy, ale že Jouxov spôsob hľadania multikolízií funguje rovnako efektívne ako pri Merkle-Damgårdovej konštrukcii. Z toho vyplýva aj nezmenená bezpečnosť $3C$ voči hľadaniu $k$-vzorov a druhých $k$-vzorov s využitím multikolízií oproti Merkle-Damgårdovej konštrukcii.

Táto konštrukcia je odolná voči útoku na druhý vzor [KS04]. Nájdenie bloku[27], ktorý je pevným bodom kompresnej funkcie $F$ a zároveň akumulačnej reťaze $B$, vyžaduje totiž $B_i = B_{i-1}$ a teda $H_i = 0$. To znamená, že blok správy $m_i$ musí spĺňať $F(0, m_i) = 0$ a to nastane s pravdpodobnosťou $2^{-n}$ pre ľubovoľný blok $m_i$. Podobná situácia nastáva, keď hľadáme expandovateľnú správu bez pevných bodov, kolízia oboch medzivýsledkov vyžaduje $2^n$ operácií (kvôli vnútornému stavu veľkosti $2n$). To znemožňuje úplne priamočiare využitie Kelsey-Schneierovho útoku. Autori však nevylúčili úspech modifikácie tohto útoku, ktorá by napadla viacero medzivýsledkov naraz[28].

### 5.2.3 Útok na druhý vzor dlhých správ konštrukcie 3C

V tejto časti prezentujeme výsledky vlastnej analýzy bezpečnosti 3C voči hľadaniu druhých vzorov. Ukážeme ako sa dajú hľadať pevné body výpočtu 3C pomocou pevných bodov použitej kompresnej funkcie $F$, popíšeme spôsob

[27] Chýbala však argumentácia, prečo by pevný bod musel pozostávať z jediného bloku správy.

[28] Presnejšie: "But if different parts of the internal state of 3C are attacked separately, 3C might not resist the second preimage attack." [GM06]

akým sa dajú hľadať expandovateľné správy tejto konštrukcie a analogickým spôsobom ako v [KS04] využiť na nájdenie druhého vzoru dlhých správ.

**Pevné body výpočtu 3C**

Predpokladajme, že je možné ľahko a rýchlo generovať pevné body kompresnej funkcie $F$. Označme $(H_1^p, m_1^p)$ nejaký pevný bod. Potom $F(H_1^p, m_1^p||m_1^p) = H_1^p$ a ak medzivýsledok na akumulačnej reťazi pred spracovaním $m_1^p||m_1^p$ označíme $B_i$, tak $B_i$ sa zmení na $B_i \oplus H_1^p \oplus H_1^p = B_i$. To znamená, že sme našli pevný bod výpočtu 3C konštrukcie, ktorý sa skladá z tej istej dvojice $(H_1^p, m_1^p)$, ako pôvodný pevný bod $F$.

**Expandovateľné správy konštrukcie 3C**

Najprv popíšeme spôsob, ktorým nadpojíme pevný bod za ľubovoľný medzivýsledok $H$ kompresnej funkcie $F$. Zostrojíme $2^{n/2}$ pevných bodov $(H_i^p, m_i^p)$ a $2^{n/2}$ správ (pre jednoduchosť jednoblokových) $m_i$. Vypočítame $2^{n/2}$ medzivýsledkov $H_i = F(H, m_i)$ a hľadáme zhodu medzi medzivýsledkami $H_i$ a $H_i^p$. Keďže všetkých dvojíc je $(2^{n/2})^2 = 2^n$, tak očákavame zhodu s vysokou pravdepodobnosťou. Časová náročnosť tohto postupu je $2^{n/2+1}$ volaní kompresnej funkcie.

Opakovaním tohto jednoduchého postupu môžeme vytvoriť ľubovoľne dlhú reťaz (označme ju $E^d$) pevných bodov, ktorá z ľubovoľne zvoleného medzivýsledku $H_{zac}$ skončí v nejakom náhodnom medzivýsledku $H_{kon}$ a bude tvaru

$$m_1||(m_1^p)^*||m_2||(m_2^p)^*||\ldots||m_d||(m_d^p)^*,$$

kde $d$ označuje počet pevných bodov reťaze $E^d$. Práca potrebná na jej vytvorenie popísaným spôsobom je $d \times 2^{n/2+1}$. Tento spôsob sa dá vylepšiť tým, že množinu pevných bodov $(H_i^p, m_i^p)$ vygenerujeme iba raz a to s mohutnosťou $2^{n/2} + d$. Takto nám stačí pri každom nadpájaní pevného bodu vytvárať iba $2^{n/2}$ správ a výsledná práca je o polovicu menšia.

Minimálna dĺžka takejto reťaze $E^d$ je $d$ blokov (spájajúce bloky $m_i$ tam byť musia, bloky pevných bodov $m_i^p$ nemusia). Označme $B_{zac}$ medzivýsledok akumulačnej reťaze konštrukcie 3C. Akumulačný medzivýsledok po spracovaní takejto minimálnej reťaze $E^d$ (dĺžky $d$) bude

$$B_{kon} = B_{zac} \oplus H_1^p \oplus H_2^p \oplus \ldots \oplus H_d^p.$$

Pridaním ľubovoľného bloku $m_i^p$ do reťaze $E^d$ sa pripočíta medzivýsledok $H_i^p$ do akumulačnej reťaze. Ak pridáme každý z blokov $m_i^p$ práve raz, dĺžka $E^d$ bude $2d$ blokov a akumulačný výsledok $B_{zac}$ sa nezmení:

$$\begin{aligned} B_{kon} &= B_{zac} \oplus H_1^p \oplus H_1^p \oplus \ldots \oplus H_d^p \oplus H_d^p \\ &= B_{zac} \end{aligned}$$

Vhodnou voľbou použitých blokov reťaze $E^d$ vieme na akumulačnom výsledku $B_{kon}$ dosiahnuť ľubovoľnú lineárnu kombináciu medzivýsledkov $B_{zac} + c_1H_1^p + c_2H_2^p + \ldots + c_dH_d^p$, kde $c_i \in \{0, 1\}$.

Medzivýsledky $H_i^p$ sú dĺžky $n$, môžeme sa na ne pozerať, ako na vektory $n$-rozmerného vektorového priestoru nad poľom $\mathbb{Z}_2$. Ak by sme dokázali vybrať do reťaze $E^d$ množinu $n$ nezávislých vektorov $H_i^p$, tak potom by bolo možné dosiahnuť ľubovoľný medzivýsledok $B_{kon}$. Toto sa dá dosiahnuť tým, že pri každom nadpojení nového pevného bodu $(H_i^p, m_i^p)$ skontrolujeme, či je množina vektorov $\{H_1^p, \ldots, H_i^p\}$ lineárne nezávislá a v prípade, že je závislá, vyberieme iný pevný bod $(H_i^p, m_i^p)$.

Pri prvom kroku vieme, že $H_1^p$ bude nezávislá množina. Pri $i$-tom kroku vieme, že $\{H_1^p, \ldots, H_{i-1}^p\}$ tvorili nezávislú množinu, takže stačí overiť, či $H_i^p$ nepatrí do podpriestoru generovaného touto množinou. Pravdepodobnosť že patrí, je rovná $2^{i-1}/2^n$ pre $i > 0$, pretože $i - 1$ nezávislých vektorov generuje podpriestor $\mathbb{Z}_2^n$ dimenzie $(i - 1)$. Pravdepodobnosť, že $n$-tý pevný bod $H_n^p$ bude patriť do podpriestoru generovaného množinou $(n - 1)$ predchádzajúcich vektorov bude najvačšia, a to $2^{n-1}/2^n = 1/2$. Pri každom nadpojení pevného bodu je teda pravdepodobnosť aspoň $1/2$, že bude vektor $H_i^p$ vyhovovať. Očakávaná zložitosť tohto postupu bude zhora ohraničená dvojnásobkom pôvodnej, stále je to však $O(2^{n/2})$.

Pre ľubovoľný $n$-bitový výsledok $B$ vieme z reťaze $E^n$ zostrojiť konkrétnu správu $E$, ktorej akumulačný medzivýsledok $B_{kon}$ bude rovný $B$ a to nezávisle na tom, aký bol začiatočný akumulačný medzivýsledok $B_{zac}$. Zvolíme jednoducho $c_1H_1^p \oplus \ldots \oplus c_nH_n^p = B \oplus B_{zac}$ a potom bude platiť $B_{kon} = B_{zac} \oplus B_{zac} \oplus B = B$.

Vytvorme reťaz $E^{n+1}$, ktorej vektory $H_1^p, H_2^p \ldots H_{n+1}^p$ sú rôzne a vygenerujú celý priestor $Z_2^n$. Tieto vektory sú lineárne závislé a teda každý z nich sa dá zapísať ako lineárna kombinácia ostatných. Nájdeme vektor $H_J^p$, ktorý sa dá napísať ako súčet párneho počtu iných vektorov. Taký sa vyskytuje s pravdepodobnosťou $1 - 2^n$ a ak tam nie je, musíme vytvoriť inú reťaz $E^{n+1}$. Vďaka $H_J^p$ vieme zvoliť paritu počtu vektorov, pomocou ktorých nasčítame ľubovoľný $n$-bitový vektor $B$. Napíšeme si $B \oplus B_{zac} = c_1H_1^p \oplus \ldots \oplus c_{n+1}H_{n+1}^p$. Ak počet nenulových $c_i$ nemá dobrú paritu, tak pripočítame k pravej strane výraz $H_J^p \oplus H_{i1}^p \oplus \ldots \oplus H_{ik}^p = 0$. Tým sa parita nenulových koeficientov $c_i$ – ako aj parita použitých blokov $m_i^p$ vo výslednej správe $E$ – zmenila a to sme chceli.

Takúto reťaz $E^{n+1}$ nazveme expandovateľná správa konštrukcie 3C. Časová náročnosť tohto postupu je $O(2^{n/2})$ volaní kompresnej funkcie $F$.

**Nájdenie druhého vzoru dlhej správy**

Útok na druhý vzor hašovacej funkcie $h$ založenej na konštrukcii 3C prebieha podobne ako pôvodný útok na Merkle-Damgårdovu konštrukciu. Označíme pôvodnú $L$-blokovú správu $M$ a jej predspracovanie $(m_1, \ldots, m_{L+1})$. Medzivýsledky po spracovaní bloku $m_i$ označme $(B_i, H_i)$.

Nájdeme expandovateľnú správu $E^{n+1}$, ktorá začína z inicializačného vektora $H_{IV}$. To znamená, že začiatočný akumulačný medzivýsledok reťaze $B_{zac} = 0$. Správa $E^{n+1}$ končí v nejakom medzivýsledku $H_{kon}$. Postupným preberaním nájdeme blok $m_{link}$, ktorý spĺňa $F(H_{kon}, m_{link}) = H_i$ pre nejaké $i = 2n + 2, \ldots, L$.

Na základe predpokladov o expandovateľnej správe $E^{n+1}$ potom vieme zabezpečiť, aby $B_{kon} = B_{i-1}$ a aby dĺžka správy $E$ vytvorenej z $E^{n+1}$ bola presne $i-1$ blokov. To znamená, že správy $M$ a $M' = E||m_{link}||m_{i+1}|| \ldots ||m_L$ budú rovnako dlhé. Z $B_{kon} = B_{i-1}$ vyplýva zhoda na skumulačnej reťazi po spracovaní časti $E||m_{link}$ a to znamená, že $h(M) = h(M')$ a našli sme druhý vzor správy $M$.

Časovú náročnosť celého postupu dominuje nájdenie bloku $m_{link}$, ktoré trvá podobne ako pri Kelseyho útoku $2^n/L$ volaní kompresnej funkcie $F$. Čas na nájdenie vhodnej expandovateľnej správy $O(2^{n/2})$ je oproti nemu zanedbateľný. Pripomíname, že tento útok funguje za predpokladu ľahkého hľadania pevných bodov použitej kompresnej funkcie $F$ uvedeného v dodatku A.

### 5.2.4 Kolízie v 3C a 3C+ za použitia kompresnej funkcie MD5, SHA-0 a SHA-1

Výsledky českých autorov D.Joščáka a J.Tůmu z [JT06] ukázali, že za použitia niektorých kompresných funkcií v $3C$ sa odolnosť voči MBCA prakticky nezvyšuje. Konkrétne, hľadanie kolízie v hašovacej funkcii založenej na $3C$, ktorá využíva kompresnú funkciu z MD5 (resp. SHA-0 alebo SHA-1), je iba dvakrát náročnejšie, ako pri pôvodnej funkcii MD5 (resp.SHA-0, či SHA-1).

Predtým, než prezentujeme samotný útok na $3C$ a $3C+$, zhrnieme kľúčové vlastnosti MBCA útokov na menované hašovacie funkcie.

**Tvar kolízií MBCA útoku na MD5,SHA-0,SHA-1**

Jednou z podstatných vlastností MBCA útoku na MD5 prezentovaného v [WH05] bolo, že nájdená kolízia pozostávala z dvoch blokov $m_1, m_2$ a začínala z ľubovoľného inicializačného vektora $H_{IV}$. Pre ľubovoľnú kolíziu $M = (m_1, m_2)$ a $M' = (m'_1, m'_2)$ navyše platilo, že rozdiel v medzivýsledkoch po spracovaní prvého bloku bol konštantný a nezávislý od $H_{IV}$

ani od $m_1$, či $m'_1$ [29]. Teda ak $F(H_{IV}, m_1) = h_1$ a $F(H_{IV}, m'_1) = H'_1$, tak $H_1 \oplus H'_1 = \delta = kont.$, pre ľubovoľnú kolíziu.

Útok na SHA-1 podľa [Wa05] produkuje z ľubovoľného inicializačného vektora multi-blokové kolízie takého istého tvaru, ako sme popísali pri MD5. Teda $H_1 \oplus H'_1 = \delta$, kde $\delta$ je konštantné pre každú kolíziu a teda nezávisí od použitého inicializačného vektora, či blokov $m_1$ a $m'_1$.

Pri Wangovom útoku na SHA-0 [WY05] je situácia ešte jednoduchšia, pretože nájdené kolízie pozostávajú taktiež z dvoch blokov, akurát prvý blok je rovnaký. Kolidujúce správy $M$ a $M'$ sú teda tvaru $M = (m_1, m_2)$ a $M' = (m_1, m'_2)$, čo znamená, že rozdiel medzivýsledkov po spracovaní prvého bloku kolidujúcich $M$ a $M'$ je dokonca nulový. Algoritmus hľadania kolízií funguje taktiež pre ľubovoľný inicializačný vektor $H_{IV}$.

**MBCA útok na 3C**

Algoritmus hľadania kolízií v $3C$, za použitia niektorej z kompresných funkcií MD5, SHA-0 alebo SHA-1, je jednoduchý. Najprv nájdeme pomocou Wangovho algoritmu kolidujúce správy $(m_1||m_2)$ a $(m'_1||m'_2)$ z ľubovoľne vybraného medzivýsledku $H_0$. Potom zadáme $H_2$ – spoločný medzivýsledok po spracovaní kolízie – ako vstup pre druhé volanie Wangovho algoritmu, ktorý vráti kolidujúcu dvojicu $(m_3||m_4)$ a $(m'_3||m'_4)$. Vďaka vlastnosti spomenutej v predchádzajúcom odseku tvoria štvorice blokov $(m_1||m_2||m_3||m_4)$ a $(m'_1||m'_2||m'_3||m'_4)$ kolíziu hašovacej funkcie založenej na $3C$ konštrukcii.

Formálnu správnosť uvedeného postupu dokazuje nasledujúca veta.

**Tvrdenie 5.2.1.** *Nech $h$ je hašovacia funkcia Merkle-Damgårdovej triedy, používajúca kompresnú funkciu $F$. Nech pre $r \geq 2$ existuje algoritmus $U$, ktorý je schopný hľadať kolidujúce správy dĺžky $r$ blokov pre ľubovoľný inicializačný vektor $H_{IV}$. Nech pre nájdené kolízie platí pre každé $1 \leq i \leq r$ spomínaná vlastnosť: $H_i \oplus H'_i$ je konštantné a nezávislé od $H_{IV}$ ani kolidujúcich správ.*

*Potom existuje algoritmus $U'$ na hľadanie $(2r)$-blokových kolízií funkcie $h^{3C}$ založenej na $3C$ konštrukcii s tou istou kompresnou funkciou $F$. Časová náročnosť algoritmu $U'$ je dvojnásobná oproti algoritmu $U$.*

*Dôkaz.* Nech $(m_1|| \ldots ||m_r)$ a $(m'_1|| \ldots ||m'_r)$ sú dve kolidujúce správy funkcie $h$ získané prvým volaním $U$ s ľubovoľným $H_{IV}$. Z toho vyplýva, že $H_r = H'_r$. Zavoláme $U$ druhýkrát s inicializačným vektorom $H_r$ a získame tým druhé dve $r$-blokové správy $(m_{r+1}|| \ldots ||m_{2r})$ a $(m'_{r+1}|| \ldots ||m'_{2r})$. Medzivýsledky $F$ počas spracovania druhej dvojice správ označíme $H_{r+i}$ a $H'_{r+i}$ pre $i = 1, \ldots, r$.

[29] Rozdiel medzivýsledkov po spracovaní druhého bloku bol samozrejme tiež konštantný a rovný $0^n$ – pretože $M$ a $M'$ tvoria kolíziu.

Ukážeme, že $M = (m_1||\ldots||m_{2r})$ a $M' = (m'_1||\ldots||m'_{2r})$ tvoria kolíziu $h^{3C}$. Keďže obe dvojice správ tvoria kolíziu kompresnej funkcie $F$, tak $H_{2r} = H'_{2r}$. Stačí ešte ukázať, že $2r$-tý medzivýsledok na akumulačnej reťazi vyjde v oboch prípadoch rovnaký. To nastáva práve vtedy, ak $B_{2r} \oplus B'_{2r} = 0$. Využijeme vlastnosť kolízií algoritmu $U$:

$$H_i \oplus H'_i = H_{r+i} \oplus H'_{r+i}$$

pre každé $i = 1, \ldots, r$. Takisto platí $B_{2r} = H_1 \oplus \ldots \oplus H_{2r}$ a $B'_{2r} = H'_1 \oplus \ldots \oplus H'_{2r}$. Z toho dostávame

$$\begin{aligned} B_{2r} \oplus B'_{2r} &= H_1 \oplus \ldots \oplus H_{2r} \oplus H'_1 \oplus \ldots \oplus H'_{2r} \\ &= (H_1 \oplus H'_1) \oplus \ldots \oplus (H_{2r} \oplus H'_{2r}) \\ &= 0. \end{aligned}$$

Posledná rovnosť platí preto, lebo každý z výrazov $H_i \oplus H'_i \oplus H_{r+i} \oplus H'_{r+i}$, $i = 1, \ldots, r$ sa v predposlednom riadku vyskytoval práve raz. Rovnosť $B_{2r} \oplus B'_{2r} = 0$ implikuje $B_{2r} = B'_{2r}$, to znamená, že $2r$-té medzivýsledky akumulačnej reťaze sú tiež rovnaké, teda $M$ a $M'$ tvoria kolíziu $h^{3C}$.

Algoritmus $U'$ potreboval dve volania algoritmu $U$, takže časová zložitosť $U'$ je dvojnásobná. $U'$ funguje pre ľubovoľný inicializačný vektor (pri prvom volaní $U$ sme mali voľnosť výberu $H_0$). □

**MBCA útok na 3C+**

Joščák a Tůma zároveň ukázali, že konštrukcia $3C+$ je rovnako bezpečná voči MBCA ako pôvodná $3C$. Zo 4-blokovej kolízie konštrukcie $3C$ sa totiž dajú jednoducho zostrojiť 5-blokové kolízie $3C+$.

Postup je založený na fakte, že pre medzivýsledok $A_i$ hornej reťaze platí vzťah $A_i = B_i \oplus H_0 \oplus H_1$, kde $H_0$ je inicializačný vektor. Zvolíme si prvý blok kolízie ľubovoľne a označíme ho $m_1$. Algoritmom $U'$ nájdeme kolíziu $(M, M')$ pre konštrukciu $3C$ s inicializačným vektorom $H_1 = F(H_0, m_1)$. Pre jednotlivé medzivýsledky správ $(m_1||M, m_1||M')$ teraz platí: $H_5 = H'_5$, $B_5 = B'_5$, pretože to bola kolízia konštrukcie $3C$. Pre medzivýsledok hornej reťaze platí $A_5 = B_5 \oplus H_0 \oplus H_1 = B'_5 \oplus H_0 \oplus H_1 = A'_5$ a teda $(m_1||M, m_1||M')$ tvoria kolíziu konštrukcie $3C+$ dĺžky 5 blokov. Uvedené úvahy sú zovšeobecnené v nasledujúcom tvrdení.

**Tvrdenie 5.2.2.** *Nech existuje algoritmus $U'$, ktorý nájde $k$-blokovú kolíziu v hašovacej funkcii $h^{3C}$ založenej na konštrukcii $3C$. Potom existuje algoritmus $U''$ na hľadanie $(k+1)$-blokových kolízií v konštrukcii $3C+$ založenej na rovnakej kompresnej funkcii $F$.*

*Časová náročnosť algoritmu $U''$ je rovná časovej zložitosti algoritmu $U'$ zväčšenej o jedno volanie kompresnej funkcie $F$.*

*Dôkaz.* Dôkaz vety je zopakovaním predchádzajúcej úvahy ($4 = k$, $5 = k + 1$). □

### 5.2.5 Zhrnutie bezpečnosti konštrukcií 3C a 3C+

Hlavná výhoda konštrukcií bola možnosť "záchrany" oslabených kompresných funkcií algoritmov MD5[30] a SHA-1 a zvýšenie bezpečnosti voči MBCA útokom. Ako však ukázali predchádzajúce výsledky, ak MBCA útok na ľubovoľnú hašovaciu funkciu Merkle-Damgårdovej triedy spĺňa určité dodatočné predpoklady, tak sa dá pomerne ľahko prispôsobiť aj proti $3C$ a $3C+$ a tým sa výhoda týchto koštrukcií stráca.

Z pohľadu štrukturálnych útokov tieto konštrukcie neprinášajú želané zvýšenie bezpečnosti, Jouxov útok je aplikovateľný bezo zmeny, Kelseyho iba na kompresné funkcie, ktoré umožňujú rýchle hľadanie pevných bodov.

Hlavnou slabinou $3C$ a $3C+$ sa ukázala funkcia XOR použitá na "akumulovanie" medzivýsledkov. Autori síce pôvodne navrhovali použitie inej (nelineárnej) funkcie $f'$, ale tým by konštrukcie stratili na rýchlosti výpočtu a začali by sa podobať na double-pipe hash z [Lu04] s vnútorným medzivýsledkom dĺžky $2n$ bitov a dvoma výpočtami kompresnej funkcie na spracovanie jedného bloku.

## 5.3 HAIFA

Konštrukciu HAIFA ( podľa *HAsh Iterative FrAmework*) navrhli E.Biham a O.Dunkelman v roku 2006 [BD06]. Vylepšuje doterajšiu Merkle-Damgårdovu, robí ju bezpečnejšou a variabilnejšou. Na rozdiel od predchádzajúcich konštrukcií sa mení aj spôsob výplne a inicializačný vektor, čo sú dve na sebe nezávislé zmeny a dali by sa ľahko využiť aj pri iných konštrukciách.

Doterajšia kompresná funkcia Merkle-Damgårdovej triedy zobrazovala $(n+b)$-bitové reťazce na $n$-bitové, jej vstupnými parametrami bol blok správy a predchádzajúci medzivýsledok. V HAIF-e do kompresnej funkcie navyše vstupuje *počet* doteraz *spracovaných bitov* a reťazec označovaný ako "soľ"[31]. Formálne:

$$F_{HA} : \{0,1\}^n \times \{0,1\}^b \times \{0,1\}^d \times \{0,1\}^s \rightarrow \{0,1\}^n,$$

kde $n$ je dĺžka medzivýsledku, $b$ dĺžka bloku, $d$ maximálna dĺžka správy a $s$ dĺžka "soli". Výpočet kompresnej funkcie je lineárny, ako tomu bolo pri Merkle-Damgårdovej konštrukcii: $H_i = F(H_{i-1}, m_i, \#bitov, sol)$.

[30] V prípade MD5 možno hovoriť o rozbitej kompresnej funkcii.

[31] Autori sa pri výbere názvu nechali inšpirovať ukladaním odtlačkov hesiel v UNIXe.

### 5.3.1 Počet doteraz spracovaných bitov

Autori zdôrazňujú, že počet doteraz spracovaných bitov sa počítal aj v súčasných hašovacích funkciách kvôli zisteniu dĺžky správy a doplneniu posledného bloku Merkle-Damgårdovej výplne. Týmto parametrom sa teda pamäťová náročnosť nezvýši, len sa predĺži výpočet kompresnej funkcie.

Do spracovaných bitov sa zarátava aj aktuálne spracovávaný blok, teda $H_i = F(H_{i-1}, m_i, b \cdot i, sol)$ pre každé $i$, okrem posledného bloku. Pri poslednom bloku platí, že $\#bitov \leq n \cdot i$.

Táto vlastnosť zabraňuje útoku predĺžením správy, pretože útočník už nemôže použiť $h(M)$ ako inicializačný vektor pri výpočte $h(M||y)$. Ak bol počet bitov správy $M$ násobkom $b$, tak sa na záver spracovania $h(M)$ pridal blok výplne $10 \ldots 0$. Tento blok bol spracovaný kompresnou funkciou s rovnakým počtom spracovaných bitov, ako predchádzajúci blok. Blok $10 \ldots 0$ však nemá byť vo výpočte pri spracovaní $h(M||y)$. Ak by počet bitov $M$ nebol násobkom $b$, tak by posledný výpočet kompresnej funkcie odtlačku $h(M)$ obsahoval $\#bitov < b \cdot i$. Pri výpočte $h(M||y)$ by bol posledný blok správy $M$ spracovaný s počtom bitov rovným $b \cdot i$ a to by vrátilo iný medzivýsledok kompresnej funkcie.

### 5.3.2 "Soľ"

"Soľ" slúži ako parameter nejakej triedy hašovacích funkcií, zvolí sa pri počítaní odtlačku dokumentu. Pridanie $s$-bitového reťazca do každého bloku znáhodní výpočet a znemožní akékoľvek predvýpočty týkajúce sa daného dokumentu (útočník si napríkad nemôže vytvoriť expandovateľnú predtým, než sa vytvorí daný odtlačok.). Taktiež jednoduchou zmenou dĺžky "soli" môžeme meniť bezpečnosť konštrukcie, čo zvyšuje jej variabilnosť.

### 5.3.3 Rôzna dĺžka odtlačkov a výplň

Konštrukcia HAIFA podporuje ľubovoľné skrátenie dĺžky výstupného odtlačku (v prípade, že nejaká aplikácia vyžaduje inú dĺžku). Pre každú dĺžku výstupu $n' \leq n$ je definovaný inicializačný vektor $IV_{n'} = F(n', IV, 0, 0)$, kde $IV$ je globálny inicializačný vektor celej funkcie. Výpočet vždy prebehne s $n$-bitovou kompresnou funkciou, len $IV_{n'}$ sa mení. Na konci výpočtu sa výsledok skráti na želanú dĺžku. Vďaka inému $IV_{n'}$ nebude kratší odtlačok tej istej správy prefixom dlhšieho. Zároveň, ak by sa používala iba jedna dĺžka výstupu, zodpovedajúci inicializačný vektor sa môže predvypočítať a neovplyvní rýchlosť výpočtu.

Spôsob výplne sa definuje podobne, ako pri Merkle-Damgårdovej triede. Za správu sa doplní jednotkový bit a toľko núl, aby výsledná dĺžka bola

násobkom $b$. Potom sa pridá dĺžka výstupného odtlačku a dĺžka vstupnej správy (obe majú pevnú dĺžku zápisu, tak počet núl vieme vypočítať ešte pred ich pripojením).

### 5.3.4 Bezpečnosť konštrukcie

Za podmienky, že kompresná funkcia je silne odolná voči kolíziám, vieme analogicky ako pri Merkle-Damgårdovej konštrukcii dokázať, že aj výsledná funkcia bude silne odolná voči kolíziám. Takisto vieme, že Jouxov útok platí aj na túto konštrukciu, keďže nemení ani dĺžku správy, ani jednotlivé medzivýsledky. Nostradamov útok je za podmienky, že útočník pozná počet blokov prefixu, tiež aplikovateľný bezo zmeny, pretože "soľ" vyberá ten, kto vytvára odtlačok, v tomto prípade je to útočník.

Pridanie počtu spracovaných bitov do kompresnej funkcie znemožnilo manipulovanie s poradím blokov a takisto pridávanie alebo uberanie ďalších blokov. To zabránilo použitiu Kelsey-Schneierovho útoku na druhý vzor a využitiu expandovateľných správ.

## 5.4 Vyhodnotenie nových konštrukcií

V tejto časti sme uviedli viacero návrhov konštrukcií nových hašovacích funkcií s cieľom ich porovnania. Pri ich vzájomnom porovnaní sa zameriame na najdôležitejšiu vlastnosť – bezpečnosť.

| | útok predĺžením správy (4.1) | nájdenie $k$-kolízie |
|---|---|---|
| MD konštrukcia | áno | $O(\lg k 2^{n/2})$ |
| Wide-pipe hash | nie | $\Omega(min(2^{w/2}, 2^{n(k-1)/k}))$ |
| Double-pipe hash | nie | $\Omega(2^{n(k-1)/k})$ |
| 3C | nie | $O(\lg k 2^{n/2})$ |
| 3C+ | nie | $O(\lg k 2^{n/2})$ |
| HAIFA | nie | $O(\lg k 2^{n/2})$ |

Tabuľka 4: Bezpečnosť nových konštrukcií. V tabuľke je čas najlepšieho útoku na nájdenie kolízie. Ak je uvedený údaj $O(f(n))$, znamená to, že nie je dokázaný dolný odhad útoku.

Bezpečnosť týchto konštrukcií určíme časovou náročnosťou najlepšieho útoku na nájdenie $k$-kolízie, vzoru a druhého vzoru. Vo väčšine prípadov je na nájdenie $k$-kolízie použitý Jouxov útok. Jediné konštrukcie, ktoré sa

ukázali odolné voči multikolíznemu útoku sú Wide-pipe hash a Double-pipe hash. Samozrejme, cenou za túto odolnosť je znížená rýchlosť výpočtu a vyššie nároky na pamäť. V prípade Wide-pipe je veľkosť medzivýsledku – a teda potrebnej pamäte – daná parametrom $w$ a výpočet Dual-pipe hash je v tomto ohľade dvakrát pomalší a náročnejší ako pôvodná Merkle-Damgårdova konštrukcia – potrebujeme si pamätať dva $n$-bitové medzivýsledky a spracovanie jedného bloku správy vyžaduje dve volania kompresnej funkcie $C$.

| | nájdenie druhého $k$-vzoru | nájdenie $k$-vzoru |
|---|---|---|
| MD konštrukcia | $O(2^n)$ | $O(2^n)$ |
| Wide-pipe hash | $\Omega(min(2^{w/2}, 2^n))$ | $\Omega(min(2^{w/2}, 2^n))$ |
| Double-pipe hash | $\Omega(2^n)$ | $\Omega(2^n)$ |
| 3C | $O(2^n)$ | $O(2^n)$ |
| 3C+ | $O(2^n)$ | $O(2^n)$ |
| HAIFA | $O(2^n)$ | $O(2^n)$ |

Tabuľka 5: Bezpečnosť nových konštrukcií. Údaj $O(f(n))$ znamená, že nebol dokázaný dolný odhad útoku.

Konštrukcie 3C a 3C+ sa zásluhou útoku [JT06] ukázali rovnako odolné voči spomínaným útokom na kompresnú funkciu, ako bola pôvodná Merkle-Damgårdova konštrukcia. Za cenu dvojnásobne (trojnásobne v prípade 3C+) väčšieho medzivýsledku poskytujú odolnosť "iba" voči priamočiaremu Kelseyho útoku na druhý vzor a aj to len v niektorých prípadoch – ako ukázali naše výsledky (5.2.3). Jednoduchosť implementácie v tomto prípade stráca na význame.

| | veľkosť vnútorného medzivýsledku | počet volaní $F$ na 1 blok správy | pamäťová náročnosť |
|---|---|---|---|
| MD konštrukcia | $n$ | 1 | $n$ |
| Wide-pipe hash | $w$ | 1 | $w$ |
| Double-pipe hash | $2n$ | 2 | $2n$ |
| 3C | $2n$ | 1 + XOR | $2n$ |
| 3C+ | $3n$ | 1 + 2×XOR | $3n$ |
| HAIFA | $n$ | $1^{32}$ | $n + s$ |

Tabuľka 6: Efektívnosť, pamäťová náročnosť a rýchlosť spracovania správy.

Veľmi dobrý kompromis medzi odolnosťou a jednoduchosťou zvolili tvor-

covia konštrukcie HAIFA. Táto konštrukcia je ľahko implementovateľná namiesto pôvodnej Merkle-Damgårdovej, nakoľko počet spracovanych bitov sa v súčasných funkciách aj tak počíta a pridanie "soli" možno realizovať na úkor spracovávaného bloku. Znížime dĺžku bloku o $s$ bitov a takto skrátený blok zreťazíme so "soľou". Pamäťové nároky sa týmto nezvýšia vôbec, klesne však rýchlosť výpočtu, pretože stúpol počet blokov, na ktoré sa rozdelí správa. Chýbali však horné ohraničenia odolnosti konštrukcie.

# 6 Záver

Na začiatku práce sme uviedli základné definície a značenie. Potom sme popísali všeobecnú konštrukciu všetkých hašovacích funkcií spolu s Merkle-Damgårdovým zosilnením. Výhodou tejto konštrukcie je dokázateľné prenášanie silnej odolnosti voči kolíziám z kompresnej funkcie na výslednú hašovaciu funkciu.

V časti 4 sme zozbierali a vysvetlili všetky doteraz objavené chyby Merkle-Damgårdovho vylepšenia konštrukcie hašovacích funkcií. Tieto nedostatky sú síce len teoretické – sami o sebe nepredstavujú reálnu hrozbu nejakej aplikácii – ale za pomoci objavených slabín v kompresnej funkcii aktuálnych funkcií MD5 a SHA-1, ktoré umožňujú hľadanie kolízií, ohrozili bezpečné použitie hašovacích funkcií v niektorých oblastiach kryptografie. Tieto nedostatky svedčia o tom, že Merkle-Damgårdova konštrukcia v pôvodnom tvare nie je vhodná na ďalšie používanie a je potrebné jej vylepšenie.

V nasledujúcej časti sme uviedli konštrukcie, ktoré rozširujú pôvodnú Merkle-Damgårdovu. Niektoré dokázateľne zvyšujú jej odolnosť, alebo úplne zamedzujú spomínaným útokom (napríklad konštrukcia Double-pipe hash), iné iba zamedzili priamočiaremu aplikovaniu týchto útokov, pričom bezpečnosť voči nejakej modifikácii ostáva nezmenená alebo otázna. V prípade 3C ukázali výsledky našej analýzy 5.2.3 ako aj českých kryptológov [JT06], že odolnosť sa nezlepšila.

Porovnaním týchto konštrukcií z pohľadu bezpečnosti získavame (očakávateľne) opačné poradie, ako porovnaním z pohľadu rýchlosti výpočtu. Najlepším kompromisom sa javí konštrukcia HAIFA, ktorá síce nie je odolná voči Jouxovmu útoku na multikolízie, ale pridaním počtu doposiaľ spracovaných bitov ako vstupného parametra do každého kroku kompresnej funkcie obmedzila možnosti ich využitia. Aj tejto konštrukcii však chýbajú dôkazy a nie je vylúčený podobný osud ako u 3C.

Ukázalo sa, že niektoré vylepšenia iterovanej konštrukcie sú na sebe nezávislé a môžu sa navzájom kombinovať, napríklad spôsob výplne, štruktúra iterácie ainicializačný vektor. Tým vznikajú ďalšie možnosti zlepšovania Merkle-Damgårdovej konštrukcie, ktoré sa oplatí hlbšie skúmať.

# A Hľadanie pevných bodov kompresných funkcií založených na Davies-Meyerovej konštrukcii

Mnoho hašovacích funkcií má kompresnú funkciu založenú na Davies-Meyerovej konštrukcii, alebo inej konštrukcii, ktorá sa jej veľmi podobá a umožňuje využitie tohto spôsobu hľadania pevných bodov. Takéto hašovacie funkcie sú pomerne rozšírené a patrí medzi ne aj MD5 a SHA-1.

Ak si označíme šifrovaciu funkciu $E_k(m)$, kde $k$ je kľúč použitý k zašifrovaniu a $m$ je správa, tak kompresná funkcia založená na Davies-Meyerovej konštrukcii je definovaná nasledovne:

$$F(h_i, m) = E_m(h_i) \oplus h_i,$$

kde $\oplus$ predstavuje operáciu XOR[33]. Potom platí:

$$F(h_i, m) = h_i \iff E_m(h_i) \oplus h_i = h_i \iff E_m(h_i) = 0 \iff h_i = E_m^{-1}(0),$$

pričom $E_k^{-1}(y)$ označuje dešifrovaciu (teda inverznú) funkciu k $E_k$.

Z uvedeného je vidieť, že na správu $m$ nie sú kladené žiadne podmienky, ako aj to, že pre každú správu možno nájsť takýmto spôsobom práve jedno $h_i$. Nad tvarom nájdeného $h_i$ nemáme žiadnu kontrolu.

Na záver poznamenávame, že sú aj konštrukcie kompresnej funkcie založené na blokovej šifre, ktoré neumožňujú zvoliť všeobecný postup na hľadanie pevných bodov. Príkladom je Miyaguchi-Preneelova konštrukcia, používaná vo funkciách Whirpool a N-Hash.

[33] Môže byť použitá aj iná invertovateľná operácia.

# Literatúra


[Pre03] B.Preneel: *Analysis and Design of Cryptographic Hash Functions.* Doctoral Dissertation, Katholieke Universiteit Leuven, 2003.

[Rom04] B.V.Rompay: *Analysis and Design of Cryptographic Hash Functions, Mac Algorithms and Block Ciphers.* Doctoral Dissertation, Katholieke Universiteit Leuven, 2004.

[MOV97] A.J.Menezes, P.C.Van Oorschot and S.A.Vanstone: *Handbook of Applied Cryptography*, chapter Hash Functions and Data Integrity, p. 321–383. CRC Press, 1997.

[MS04] M.Stanek: *Základy kryptológie.* 2004. Prístupné online na `http://www.dcs.fmph.uniba.sk/%7Estanek/crypto/main2.pdf`.

[Sti02] D.R.Stinson: *Cryptography: Theory and Practice.* Second Edition, chapter Hash Functions, p. 233–258. CRC Press, 2002.

[BS96] B.Schneier*Applied Cryptography.* Second Edition. Wiley, 1996.

[Mer89] R.Merkle: *One-way hash functions and DES.* Advances in Cryptology-CRYPTO 1989, volume 435 of Lecture Notes in Computer Science, p. 428–446, Springer, 1989.

[Dam89] I.Damgård: *A design principle for hash functions.* Advances in Cryptology-CRYPTO 1989, volume 435 of Lecture Notes in Computer Science, p. 416–427, Springer, 1989.

[Jo04] A.Joux: *Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions.* Advances in Cryptology-CRYPTO 2004, volume 3152 of Lecture Notes in Computer Science, p. 306–316, Springer, 2004.

[DM89] P.Diaconis and F.Mosteller: *Methods for studying coincidences.* Journal of the American Statistical Association 84, p. 853–861, 1989.

[KS04] J.Kelsey and B.Schneier: *Second Preimages on n-bit hash function for much less than $2^n$ work.* IACR Cryptology ePrint Archive, Report 2004/304, 2004. Prístupné online na `http://eprint.iacr.org/2004/304`.

[KK05] J.Kelsey and T.Kohno: *Herding Hash Functions and the Nostradamus Attack.* IACR Cryptology ePrint Archive, Report 2005/281, 2005. Prístupné online na `http://eprint.iacr.org/2005/281`.

[BK04] M.Bellare and T.Kohno: *Hash Function Balance and its Impact on Birthday Attacks.* IACR Cryptology ePrint Archive, Report 2004/065, 2004. Prístupné online na `http://eprint.iacr.org/2004/065`.

[Wie05] M.J.Wiener:*Bounds on Birthday Attack Times.* IACR Cryptology ePrint Archive, Report 2005/318, 2005. Prístupné online na `http://eprint.iacr.org/2005/318`.

[Bih05] E.Biham, R.Chen, A.Joux, P.Carribault, C.Lemuet and W.Jalby: *Collisions of SHA-0 and Reduced SHA-1.* In Ronald Cramer, editor, Advances in Cryptology-EUROCRYPT 2005, volume 3494 of Lecture Notes in Computer Science, p. 36–57, Springer, 2005.

[WY05] X.Wang, Y.L.Yin and H.Yu: *Efficient collision search attacks on SHA-0.* In Victor Shoup, editor, Advances in Cryptology-CRYPTO 2005, volume 3621 of Lecture Notes in Computer Science, p. 1–16, Springer 2005.

[Wa05] X.Wang, Y.L.Yin and H.Yu: *Finding collisions in the full SHA-1.* In Victor Shoup, editor, Advances in Cryptology-CRYPTO 2005, volume 3621 of Lecture Notes in Computer Science, p. 17–36, Springer 2005.

[WH05] X.Wang and H.Yu: *How to Break MD5 and Other Hash Functions.* In Ronald Cramer, editor, Advances in Cryptology-EUROCRYPT 2005, volume 3494 of Lecture Notes in Computer Science, p. 19–35, Springer 2005.

[Kli06] V.Klima: *Tunnels in Hash Functions: MD5 collisions Within a Minute.* IACR Cryptology ePrint Archive, Report 2006/105. Prístupné online na `http://eprint.iacr.org/2006/105`.

[JT06] D.Joščák and J.Tůma: *Multi-block Collisions in Hash functions based on 3C and 3C+ Enhancements of the*

*Merkle-Damgård Construcion.* 2007. Prístupné online na http://www.karlin.mff.cuni.cz/ka-preprints/2007-pap/alg07_02.pdf.

[NS06] M.Nandi and D.R.Stinson: *Multicollision Attacks on Some Generalized Sequential Hash Functions.* IACR Cryptology ePrint Archive, Report 2006/055. Prístupné online na http://eprint.iacr.org/2006/055.

[NS04] M.Nandi and D.R.Stinson: *Multicollision Attacks on Some Generalized Sequential Hash Functions.* IACR Cryptology ePrint Archive, Report 2004/330. Prístupné online na http://eprint.iacr.org/2004/330.

[Lu04] S.Lucks: *Design Principles for Iterated Hash Functions.* IACR Cryptology ePrint Archive, Report 2004/253. Prístupné online na http://eprint.iacr.org/2004/253.

[Lu05] S.Lucks: *A Failure-Friendly Design Principle for Hash Functions.* Advances in Cryptology-Asiacrypt 2005, volume 3788 of Lecture Notes in Computer Science, p. 474–494, Springer 2005.

[GM05] P. Gauravaram, W.Millan, J.G.Nieto and E.Dawson: *3C - A Provably Secure Pseudorandom Function and Message Authentication Code. A New mode of operation for Cryptographic Hash Function.* IACR Cryptology ePrint Archive, Report 2005/390. Prístupné online na http://eprint.iacr.org/2005/390.

[GN05] P. Gauravaram, W.Millan and J.G. Nieto: *Some thoughts on collision attacks in the hash functions MD5, SHA-0 and SHA-1.* IACR Cryptology ePrint Archive, Report 2005/391. Prístupné online na http://eprint.iacr.org/2005/391.

[GM06] P. Gauravaram, W.Millan, E.Dawson and K. Viswanathan: *Constructing Secure Hash Functions by Enhancing Merkle-Damgård Construction.* IACR Cryptology ePrint Archive, Report 2006/061. Prístupné online na http://eprint.iacr.org/2006/061.

[BD06] E.Biham and O.Dunkelman: *A Framework for Iterative Hash Functions.* Prístupné online na http://csrc.nist.gov/pki/HashWorkshop/2006/Papers/DUNKELMAN_NIST3.pdf.

[Ma03] U.M.Maurer, R.Renner and C.Holenstein: *Indifferentiability, Impossibility Results on Reductions and Applications to the Random Oracle Methodology.* IACR Crypto-

logy ePrint Archive, Report 2003/161. Prístupné online na http://eprint.iacr.org/2003/161.

[Co05] J.S.Coron, Y.Dodis, C.Malinaud and P.Puniya: *Merkle-Damgård revisited: How To Construct a Hash Function.* Advances in Cryptology-CRYPTO 2005, volume 3621 of Lecture Notes in Computer Science, Springer 2005.

[BR06] M.Bellare and T.Ristenpart:*Multi-Property Preserving Hash Domain Extension and the EMD Transform.* Prístupné online na http://www.cs.ucsd.edu/%7Etristenp/papers/hashdomextACff.pdf.