

# Nalézání kolizí MD5 na notebooku pomocí mnohonásobných modifikací zprávy

Vlastimil Klíma<sup>1,2</sup>  
Prague, Czech Republic  
<http://cryptography.hyperlink.cz>  
[v.klima@volny.cz](mailto:v.klima@volny.cz)

31. března 2005  
verze 1

## Abstrakt

V tomto stručném příspěvku shrnujeme výsledky našeho tříměsíčního výzkumu kolizí hašovací funkce MD5. Inspirováni prací Wangové a kol. [1] jsme vyvinuli metody nalézání kolizí, které pracují pro libovolný inicializační vektor a jsou rychlejší než metody v [1, 8]. To umožňuje nalézt kolizi MD5 na standardním notebooku asi za 8 hodin [7]. Nezávisle na [1, 8] jsme objevili a navrhli několik metod mnohonásobné modifikace zpráv, které jsou efektivnější než v [1, 8], a ukazujeme jejich podstatu.

**Klíčová slova:** MD5, kolize, mnohonásobná modifikace zprávy

## 1. Úvod

Jednou z nejvýznamnějších kryptologických událostí posledních let bylo objevení kolizí pro sérii hašovacích funkcí MD4, MD5, HAVAL-128 a RIPEMD čínským týmem v srpnu 2004 [1]. Jejich autoři (Wangová a kol.) však utajili metodu nalézání kolizí a zveřejnili pouze strohá data a informace. V říjnu 2004 se australský tým (Hawkes a kol.) pokusil tuto metodu zrekonstruovat ve skvělé práci [3]. Nejdůležitější "čínský trik" se nepodařilo objevit, ale na základě dat z [1] bylo dobře popsáno diferenční schéma, kterému uveřejněné čínské kolize vyhovují. Naplnění podmínek tohoto schématu bylo však ještě příliš náročné a výpočetně složitější, než ukazovaly výsledky z [1].

## 2. Výsledky našeho výzkumu [7]

V našem výzkumu [7] jsme také analyzovali dostupná data diferenční kryptoanalýzou. Nalezli jsme cestu, jak generovat kolize prvního bloku 1000 - 2000 krát rychleji než čínský tým, což odpovídá nalezení jedné kolize prvního bloku na běžném notebooku za 2 minuty. Čínskému týmu tato fáze trvá jednu hodinu na počítači IBM p690. Naproti tomu byl čínský tým 2 - 80 krát rychlejší při vyhledávání kolizí druhého bloku. Obě metody se proto mohou lišit nejen časově, ale i obsahově. Celkově je naše metoda 3 - 6 krát rychlejší. Konkrétně nalezení první (úplné) kolize nám na notebooku (Intel Pentium 1.6 GHz) trvalo pouze 8 hodin.

---

<sup>1</sup> Tento výzkum byl dělán o vánoční dovolené a v lednu - březnu 2005. Autor v této době pracoval pro firmu LEC, s.r.o., Praha, Česká republika, která tento projekt materiálně i finančně podpořila.

<sup>2</sup> Tento příspěvek bude prezentován na: 3rd International Scientific Conference *Security and Protection of Information*, Brno, Czech Republic, May 3 - 5, 2005, <http://www.unob.cz/spi/defaulten.asp>

Poznamenejme, že naše metoda pracuje pro jakoukoli zvolenou inicializační hodnotu. To je velmi zneužitelné pro falšování podpisů SW balíků nebo padělání certifikátů, jak ukazují některé současné práce ([4], [5], [6]). Ukázali jsme, že vyhledávání kolizí hašovací funkce MD5 je možné provádět na domácím počítači. To by mělo být varováním před dalším používáním této hašovací funkce. V příloze uvádíme nové příklady kolizí MD5 pro standardní a zvolenou inicializační hodnotu.

### 3. Výsledky Wangové a kol. [1]

Hašovací funkce jsou velmi užitečným kryptografickým nástrojem. Pro zajištění jejich vlastností jednocestnosti a bezkoliznosti musí být hašovací funkce velmi robustní a složité. Proto je vždy velmi vzrušující, když je nalezena nějaká kolize. Jednou z nejvýznamnějších kryptoanalytických prací v minulém roce byla práce čínského týmu [1]. Nejsložitější útok si vyžádala hašovací funkce MD5, proto se budeme dále věnovat jen této funkci.

Připomeňme, že Wangová a kol. [1] nezveřejnily postup hledání kolizí, jen strohé údaje, které zde zopakujeme. Kolidující zprávy  $(M, N)$  a  $(M', N')$  se skládají ze dvou bloků, přičemž první bloky zpráv se liší o předem definovaný konstantní vektor  $C1$  ( $M' = M + C1$ ) a druhé bloky se liší o předem definovaný konstantní vektor  $C2 = -C1 \bmod 2^{32}$  ( $N' = N + C2$ ), přičemž  $MD5(M, N) = MD5(M', N')$ .

Wangová a kol. uvedli, že na počítači IBM p690 jim trvá zhruba hodinu než naleznou blok  $M$ . Nalezení bloku  $N$  pak trvá od 15 sekund do 5 minut. V první verzi [1] uvedli dva páry kolidujících zpráv. Jimi zvolená hodnota inicializačního vektoru ( $IV$ ) však neodpovídala popisu MD5, neboť měla obrácené pořadí bajtů (Little vs. Big Endian). V opravené verzi příspěvku den poté uvedli opět dvě dvojice kolidujících zpráv pro MD5, tentokrát se správnou  $IV$ . Navíc poznamenali, že jejich útok pracuje pro jakoukoli hodnotu  $IV$ .

#### 3.1. Zneužitelnost kolizí

Po uveřejnění jejich výsledků jsme měli k dispozici pouze čtyři páry kolidujících zpráv. Přesto bylo ukázáno, že dokonce i pouze tato data lze využít ke konstrukci úspěšných útoků [4], [5]. V [4] je ukázáno, že postačí jediná kolize k vytvoření páru různých samorozbalovacích archivů s identickou haší. To může být zneužitelné například při vkládání zadních vrátek do velkých balíků SW při jejich distribuci. Později bylo za účasti jednoho z autorů [1] ukázáno jak s využitím schopnosti vytvářet kolize pro libovolný inicializační vektor padělat digitální certifikát [6].

### 4. Pokus odhalit čínský trik

V říjnu 2004 byla publikována skvělá práce [3] Hawkese a kol., kde se její autoři snaží odhalit "čínskou metodu" hledání kolizí na základě strohých dat a informací, uvedených v [1]. V práci vyšetřují vnitřní diference a podmínky pro zprávy, které by měly být splněny, aby došlo ke kolizi čínským postupem [1]. Byla to první analýza a pokus vysvětlit čínskou metodu. Na základě jednoho páru kolidujících zpráv se správnou inicializační hodnotou  $IV$  autoři popsali diferenční schéma, které publikovaná kolize splňuje, a které pravděpodobně bylo v pozadí kolize. Nepodařilo se však vysvětlit, jak toto schéma vzniklo. Dále popsali podmínky, které musí splňovat jedna zpráva z kolidujícího páru tak, aby diferenční schéma bylo splněno. Obdrželi dlouhý seznam podmínek, které musí zpráva splňovat. První sada (tzv. ft-podmínky a Tt-podmínky) vzniká u prvního bloku zprávy při průchodu 64 kroky MD5. Pokud se splní ft-podmínky a Tt-podmínky v prvních 16 krocích (přes 200 podmínek) vhodnou volbou bloku  $M$ , zbývá ještě naplnit 39 ft-podmínek a "3.2" Tt-podmínek ve zbývajících krocích, které jsou splněny pouze pravděpodobnostně. Celkem je tak potřeba generovat cca  $2^{42.2}$  zpráv  $M$ , aby jedna z nich splňovala všechny ft-podmínky a všechny Tt-podmínky z rund 17 - 64. Podobně pro splnění ft- a Tt-podmínek druhého bloku zprávy  $N$  je

nutné podle [3] generovat  $2^{42.2}$  zpráv. Složitost celého útoku je pak  $2^{43}$ . Hawkes a kol. se domnívají, že toto je příliš velká složitost, aby se kolize dala vygenerovat za jednu hodinu. Proto dovozují, že Wangová a kol. museli použít ještě nějaký další trik. Tento trik je pochopitelně klíčový.

## 5. Naše metoda

V našem výzkumu [7] jsme vyšli z výsledků [3] a diferenční schéma jsme také zkoumali z hlediska diferencí aditivních (aritmetický rozdíl modulo  $2^{32}$ ) i diferencí binárních (XOR, mod 2), stejně jako [3]. Navíc jsme zkoumání podrobili i další kolidující pár, který byl v [1] vytvořen pro špatnou inicializační hodnotu. Potvrdili jsme, že diferenční schéma platí pro oba kolidující páry, neboť více dat nebylo k dispozici. V našem výzkumu se ukázalo, že některé ft- a Tt- podmínky mohou být splněny více cestami, než zvolil Hawkes a kol. To by mohlo teoreticky vést ke snížení výpočetní složitosti. Narostla by však složitost paměťová a složitost příslušného programu na generování kolizí, a tak jsme touto cestou nešli. Nicméně analýza ft- a Tt-podmínek naznačila, že skutečná složitost nalezení kolize by mohla být ve skutečnosti menší, než v teoretickém modelu. V dalším výzkumu jsme pak (nezávisle na [8], publikované později) našli vlastní metodu mnohonásobné modifikace zpráv, umožňující generovat první bloky kolidujících zpráv velmi rychle. Na standardním notebooku jsme obdrželi první blok zprávy během dvou minut, oproti jedné hodině na počítači IBM p690 [1]. Vzhledem ke krátkosti výzkumu jsme nepokročili v urychlení hledání kolizí v druhém bloku tak jako u prvního bloku, i když jsme dosáhli složitosti významně nižší než  $2^{42}$  podle [3] a nižší než  $2^{32}$  podle [8]. O tom svědčí i nalezení první kolize na notebooku za 8 hodin. Podle [1] by však hledání kolizí druhého bloku mělo být 12 - 240 krát rychlejší než u prvního bloku. Pak by kolize byla na notebooku místo za 8 hodin nalezena během dvou minut. V hledání kolizí druhého bloku tak ještě zůstávají časové rezervy.

### 5.1. Výsledky našich experimentů

K nalezení kolizí jsme nepoužili žádný superpočítač, pouze běžné domácí počítače. Autor prováděl své experimenty výhradně na notebooku, kde našel jak desetitisíce kolizí prvního bloku, tak i úplné kolize MD5 pro platnou inicializační hodnotu i volené inicializační hodnoty. Pro ověření funkčnosti programu jsme také požádali několik přátel o vyzkoušení na jejich domácích počítačích. Za týden experimentování na počátku března tak byly nalezeny desetitisíce kolizí prvních bloků a desítky úplných kolizí.

Výsledek na běžném notebooku (Acer TravelMate 450LMi, Intel Pentium 1.6 GHz) je tento: během 8 hodin bylo nalezeno 331 kolizí prvního bloku a 1 úplná kolize MD5. Vzhledem k tomu, že nalezení 1 kolize prvního bloku trvalo čínskému týmu 1 hodinu na počítači IBM p690, nalezení 331 těchto kolizí by trvalo cca 331 hodin, což je 40 krát více. Výkony notebooku a velkého počítače lze těžko srovnávat z důvodu různých architektur, ale když uvažujeme, že uvedený počítač je 25 - 50 krát rychlejší než notebook (odhad poskytl Ondrej Mikle na základě poměru bogomips), dostáváme velmi hrubý odhad, že naše metoda hledání kolize prvního bloku je 1000 - 2000 krát rychlejší než v [1]. Naproti tomu hledání kolize druhého bloku je 2 - 80 krát pomalejší. Pokud srovnáme celkový čas hledání úplné kolize u čínského týmu (1.0 až 1.08 hodiny) s naším (8 hodin) na 25 - 50 krát pomalejším stroji, je naše metoda celkově 3 - 6 krát rychlejší. Všechna tato srovnání jsou orientační a autor si nečiní žádný nárok na jejich přesnost (přesné jsou pouze časové údaje).

### 5.2. Podstata čínského triku

Výsledky našeho výzkumu [7] byly publikovány před umístěním příspěvků [8] a [9] na web. V [8] byla popsána metoda hledání kolizí MD5, ale bez detailů, umožňujících ji

naprogramovat a zrekonstruovat. Klíčové myšlenky uvedené v [8] si zde zopakujeme a vysvětlíme rozdíly naší a čínské metody.

### 5.3. Výběr difference mezi bloky

Základem [8] bylo rozhodnutí hledat kolize u zpráv se dvěma bloky a výběr difference mezi určitými slovy těchto bloků. Podle [8] byla difference vybírána tak, aby se co nejvíce zvýšila pravděpodobnost naplnění diferenciálu v rundách 3 a 4 schématu MD5. Pravděpodobně existuje více těchto základních diferencí, vedoucích k efektivním diferenčním schématům.

### 5.4. Proměnné $Q$ ( $Q^*$ ), stacionární podmínky a diferenční schéma

Označme první bloky kolidujících zpráv  $M$  a  $M^*$  a jim odpovídající proměnné  $Q$  podle [3] jako  $Q$  a  $Q^*$ .  $Q[1]$  (resp.  $Q^*[1]$ ) tak označují první hodnotu, která je počítána v prvním kroku zpracování bloku  $M$  (resp.  $M^*$ ).  $Q[64]$  (resp.  $Q^*[64]$ ) je poslední hodnota, počítaná při zpracování bloku  $M$  (resp.  $M^*$ ). Dále označme  $x[i]$ ,  $i = 0, \dots, 15$  jednotlivá slova bloku  $M$ . Podle [3] máme následující vztahy mezi  $Q$  a  $x$ . Na konci rovnic pro  $Q[17 - 20]$  uvádíme počet podmínek pro dané  $Q$  (v prvním bloku), které mají být splněny podle diferenčního schématu z [3]. RL, resp. RR, označuje cyklickou rotaci doleva (resp. doprava).

$$\begin{aligned} Q[1] &= Q[0] + \text{RL}(F(Q[0], Q[-1], Q[-2])) + Q[-3] + x[0] + 0xd76aa478, 7); \\ Q[2] &= Q[1] + \text{RL}(F(Q[1], Q[0], Q[-1])) + Q[-2] + x[1] + 0xe8c7b756, 12); \\ Q[3] &= Q[2] + \text{RL}(F(Q[2], Q[1], Q[0])) + Q[-1] + x[2] + 0x242070db, 17); \\ Q[4] &= Q[3] + \text{RL}(F(Q[3], Q[2], Q[1])) + Q[0] + x[3] + 0xc1bdceee, 22); \\ Q[5] &= Q[4] + \text{RL}(F(Q[4], Q[3], Q[2])) + Q[1] + x[4] + 0xf57c0faf, 7); \\ Q[6] &= Q[5] + \text{RL}(F(Q[5], Q[4], Q[3])) + Q[2] + x[5] + 0x4787c62a, 12); \\ Q[7] &= Q[6] + \text{RL}(F(Q[6], Q[5], Q[4])) + Q[3] + x[6] + 0xa8304613, 17); \\ Q[8] &= Q[7] + \text{RL}(F(Q[7], Q[6], Q[5])) + Q[4] + x[7] + 0xfd469501, 22); \\ Q[9] &= Q[8] + \text{RL}(F(Q[8], Q[7], Q[6])) + Q[5] + x[8] + 0x698098d8, 7); \\ Q[10] &= Q[9] + \text{RL}(F(Q[9], Q[8], Q[7])) + Q[6] + x[9] + 0x8b44f7af, 12); \\ Q[11] &= Q[10] + \text{RL}(F(Q[10], Q[9], Q[8])) + Q[7] + x[10] + 0xfffff5bb1, 17); \\ Q[12] &= Q[11] + \text{RL}(F(Q[11], Q[10], Q[9])) + Q[8] + x[11] + 0x895cd7be, 22); \\ Q[13] &= Q[12] + \text{RL}(F(Q[12], Q[11], Q[10])) + Q[9] + x[12] + 0x6b901122, 7); \\ Q[14] &= Q[13] + \text{RL}(F(Q[13], Q[12], Q[11])) + Q[10] + x[13] + 0xfd987193, 12); \\ Q[15] &= Q[14] + \text{RL}(F(Q[14], Q[13], Q[12])) + Q[11] + x[14] + 0xa679438e, 17); \\ Q[16] &= Q[15] + \text{RL}(F(Q[15], Q[14], Q[13])) + Q[12] + x[15] + 0x49b40821, 22); \end{aligned}$$

$$\begin{aligned} Q[17] &= Q[16] + \text{RL}(G(Q[16], Q[15], Q[14])) + Q[13] + x[1] + 0xf61e2562, 5); & 4 \text{ podm.} \\ Q[18] &= Q[17] + \text{RL}(G(Q[17], Q[16], Q[15])) + Q[14] + x[6] + 0xc040b340, 9); & 3 \text{ podm.} \\ Q[19] &= Q[18] + \text{RL}(G(Q[18], Q[17], Q[16])) + Q[15] + x[11] + 0x265e5a51, 14); & 2 \text{ podm.} \\ Q[20] &= Q[19] + \text{RL}(G(Q[19], Q[18], Q[17])) + Q[16] + x[0] + 0xe9b6c7aa, 20); & 1 \text{ podm.} \end{aligned}$$

Uvažujme nejprve diferenční schéma v prvním bloku. Obě dvě práce [3] a [8] uvádí podobné diferenční schéma, z něhož plyne mnoho podmínek na bity proměnných  $Q[1]$  až  $Q[64]$  a čtyři mezihodnoty součtů (mezi prvním a druhým blokem). Některé bity slov  $Q$  jsou nastaveny přímo na nulu nebo jedničku, mezi některými musí platit vztahy rovnosti nebo negace. Tyto podmínky označujeme jako stacionární. Jsou v [3] i [8] voleny tak, aby byly postačující pro splnění diferenčního schématu v krocích 1 - 64. Diferenční schéma pak budou splňovat všechny zprávy  $M$  a  $M^*$ , kde  $M$  splňuje stacionární podmínky a  $M^*$  se od  $M$  liší definovaným způsobem v  $x[1]$  (o  $2^{31}$ ),  $x[4]$  (o  $2^{15}$ ) a  $x[11]$  (o  $2^{31}$ ).

Poznamenejme, že je více cest, jak naplnit diferenční schéma, takže (mírně se odlišující) schémata z [3] a [8] jsou pouze dva příklady možných řešení. Schémat je velké množství.

### 5.5. Naplnění stacionárních podmínek pro co nejvíce

## kroků

Náš postup je obecný a nezávislý na konkrétním vybraném schématu. V naší metodě volíme proměnné  $Q[1 - 16]$  libovolně tak, aby splňovaly stacionární podmínky. Odtud vypočítáme  $x[0 - 15]$  a zkontrolujeme, zda jsou splněny i ostatní stacionární podmínky pro  $Q[17 - 64]$  (je jich 36 a 7 podmínek na čtyři součty hodnot mezi prvním a druhým blokem). Tyto podmínky jsou splněny pouze pravděpodobnostně a podle [8] je složitost takového postupu  $2^{43}$ .

Cílem je proto nalézt takové hodnoty  $x[0 - 15]$ , aby bylo splněno  $Q[1 - 16]$  a co nejvíce podmínek pro  $Q[17], Q[18], \dots$  deterministicky.

V [8] se uvádí, že metodou mnohonásobné modifikace zprávy lze v prvním bloku docílit naplnění 6 podmínek v krocích  $Q[17], Q[18], \dots$ . Složitost nalezení kolize v prvním bloku se tak snižuje z  $2^{43}$  na  $2^{37}$ .

Naše metoda [7] vyvinutá nezávisle na ([8]), snižuje složitost na  $2^{33}$ , neboť deterministicky splňuje 10 podmínek v krocích  $Q[17]$  až  $Q[20]$ .

## 5.6. Naše metoda mnohonásobné modifikace zprávy v prvním bloku

V naší metodě mnohonásobné modifikace zprávy [7] splňujeme podmínky  $Q[1 - 20]$  deterministicky. Zbývajících 31 podmínek je splněno pravděpodobnostně.

Postup:

1. Zvolíme  $Q[3 - 16]$  splňující stacionární podmínky

2. Vypočítáme  $x[6 - 15]$ :

$$x[6] = \text{RR}(Q[7] - Q[6], 17) - \text{F}(Q[6], Q[5], Q[4]) - Q[3] - 0\text{xa}8304613;$$

$$x[7] = \text{RR}(Q[8] - Q[7], 22) - \text{F}(Q[7], Q[6], Q[5]) - Q[4] - 0\text{xfd}469501;$$

$$x[8] = \text{RR}(Q[9] - Q[8], 7) - \text{F}(Q[8], Q[7], Q[6]) - Q[5] - 0\text{x}698098d8;$$

$$x[9] = \text{RR}(Q[10] - Q[9], 12) - \text{F}(Q[9], Q[8], Q[7]) - Q[6] - 0\text{x}8b44f7af;$$

$$x[10] = \text{RR}(Q[11] - Q[10], 17) - \text{F}(Q[10], Q[9], Q[8]) - Q[7] - 0\text{xffff}5bb1;$$

$$x[11] = \text{RR}(Q[12] - Q[11], 22) - \text{F}(Q[11], Q[10], Q[9]) - Q[8] - 0\text{x}895cd7be;$$

$$x[12] = \text{RR}(Q[13] - Q[12], 7) - \text{F}(Q[12], Q[11], Q[10]) - Q[9] - 0\text{x}6b901122;$$

$$x[13] = \text{RR}(Q[14] - Q[13], 12) - \text{F}(Q[13], Q[12], Q[11]) - Q[10] - 0\text{xfd}987193;$$

$$x[14] = \text{RR}(Q[15] - Q[14], 17) - \text{F}(Q[14], Q[13], Q[12]) - Q[11] - 0\text{xa}679438e;$$

$$x[15] = \text{RR}(Q[16] - Q[15], 22) - \text{F}(Q[15], Q[14], Q[13]) - Q[12] - 0\text{x}49b40821;$$

3. Měníme  $Q[17]$  do té doby, dokud nejsou splněny podmínky  $Q[17 - 19]$ . V případě nutnosti lze měnit i hodnoty  $Q[3 - 16]$ .  $Q[18]$  a  $[19]$  vypočteme podle vztahů

$$Q[18] = Q[17] + \text{RL}(G(Q[17], Q[16], Q[15]) + Q[14] + x[6] + 0\text{xc}040b340, 9)$$

$$Q[19] = Q[18] + \text{RL}(G(Q[18], Q[17], Q[16]) + Q[15] + x[11] + 0\text{x}265e5a51, 14).$$

Potom vypočteme

$$x[1] = \text{RR}(Q[17] - Q[16], 5) - G(Q[16], Q[15], Q[14]) - Q[13] - 0\text{xf}61e2562;$$

a

$$x[2] = \text{RR}(Q[3] - Q[2], 17) - \text{F}(Q[2], Q[1], Q[0]) - Q[-1] - 0\text{x}242070db;$$

$$x[3] = \text{RR}(Q[4] - Q[3], 22) - \text{F}(Q[3], Q[2], Q[1]) - Q[0] - 0\text{xc}1bdceee;$$

$$x[4] = \text{RR}(Q[5] - Q[4], 7) - \text{F}(Q[4], Q[3], Q[2]) - Q[1] - 0\text{xf}57c0faf;$$

$$x[5] = \text{RR}(Q[6] - Q[5], 12) - \text{F}(Q[5], Q[4], Q[3]) - Q[2] - 0\text{x}4787c62a;$$

4. Nyní jsou splněny všechny stacionární podmínky pro  $Q[3 - 19]$ . Navíc je volná hodnota  $x[0]$ .

5. Volíme libovolně  $Q[20]$ , splňující jednu stacionární podmínku. Dopočteme  $x[0]$ :

$$x[0] = \text{RR}(Q[20] - Q[19], 20) - G(Q[19], Q[18], Q[17]) - Q[16] - 0\text{xe}9b6c7aa;$$

6. Zbývajících 33 podmínek je splněno pravděpodobnostně. Máme  $2^{31}$  možností volby  $Q[20]$ . Pokud nenalezneme kolizi, vrátíme se k bodu nové volby  $Q[17]$ . Se složitostí  $2^{33}$  nalezneme kolizi prvního bloku.

**Poznámka.** Pokud bychom popisovali metodu jako v [8], bylo by to méně přehledné. Šlo by v první řadě o mnohonásobnou změnu bloků  $x[1], x[6]$  a  $x[11]$  tak, aby byly splněny

stacionární podmínky  $Q[17]$ ,  $Q[18]$  a  $Q[19]$ . Dalším krokem by byla změna bloku  $x[0]$  tak, aby byla splněna podmínka pro  $Q[20]$ .

## 5.7. Naše metoda mnohonásobné modifikace zprávy v druhém bloku

Nyní uvedeme metodu, která není optimální, ale která byla použita v [7]. Poté uvedeme metody mnohonásobné modifikace zprávy, které jsou rychlejší, ale které nebyly experimentálně ověřeny.

Naše metoda mnohonásobné modifikace zprávy [7] spočívá v deterministickém splnění podmínek  $Q[1 - 18]$ . Zbývajících 29 podmínek je splněno pravděpodobnostně.

Postup:

1. Zvolíme libovolně  $Q[3 - 16]$ , splňující stacionární podmínky
2. Změníme libovolně bit č. 22 (váha  $2^{22}$ ) proměnných  $Q[4 - 16]$
3. Zvolíme libovolně hodnotu  $Q[1]$  splňující stacionární podmínky (je zde minimálně  $2^{24}$  možných hodnot  $Q[1]$ )
4. Vypočteme  $x[0]$ ,  $x[7 - 15]$

$x[0]=RR(Q[1]-Q[0],7)-F(Q[0],Q[-1],Q[-2])-Q[-3]-0xd76aa478;$

$x[7]=RR(Q[8]-Q[7],22)-F(Q[7],Q[6],Q[5])-Q[4]-0xfd469501;$

$x[8]=RR(Q[9]-Q[8],7)-F(Q[8],Q[7],Q[6])-Q[5]-0x698098d8;$

$x[9]=RR(Q[10]-Q[9],12)-F(Q[9],Q[8],Q[7])-Q[6]-0x8b44f7af;$

$x[10]=RR(Q[11]-Q[10],17)-F(Q[10],Q[9],Q[8])-Q[7]-0xffff5bb1;$

$x[11]=RR(Q[12]-Q[11],22)-F(Q[11],Q[10],Q[9])-Q[8]-0x895cd7be;$

$x[12]=RR(Q[13]-Q[12],7)-F(Q[12],Q[11],Q[10])-Q[9]-0x6b901122;$

$x[13]=RR(Q[14]-Q[13],12)-F(Q[13],Q[12],Q[11])-Q[10]-0xfd987193;$

$x[14]=RR(Q[15]-Q[14],17)-F(Q[14],Q[13],Q[12])-Q[11]-0xa679438e;$

$x[15]=RR(Q[16]-Q[15],22)-F(Q[15],Q[14],Q[13])-Q[12]-0x49b40821;$

5. Zvolíme libovolně hodnotu  $Q[2]$ , splňující stacionární podmínky (je zde minimálně  $2^{12}$  možných hodnot)

6. Vypočteme  $x[1]$  z  $Q[1 - 2]$  a zkontrolujeme, zda jsou splněny stacionární podmínky pro  $Q[17]$ :

$x[1]=RR(Q[2]-Q[1],12)-F(Q[1],Q[0],Q[-1])-Q[-2]-0xe8c7b756,$

$Q[17]=Q[16]+RL(G(Q[16],Q[15],Q[14])+Q[13]+x[1]+0xf61e2562,5).$

Jestliže nikoli, jdeme na bod 5 a zvolíme nové  $Q[2]$  (je-li to nezbytné, jdeme na bod 3 a volíme i nové  $Q[1]$ ).

7. Nastavíme dolní bity  $Q[3]$  jako u  $Q[2]$  podle stacionárních podmínek

8. Z  $Q[3 - 7]$  vypočteme  $x[2 - 6]$ :

$x[2]=RR(Q[3]-Q[2],17)-F(Q[2],Q[1],Q[0])-Q[-1]-0x242070db;$

$x[3]=RR(Q[4]-Q[3],22)-F(Q[3],Q[2],Q[1])-Q[0]-0xc1bdceee;$

$x[4]=RR(Q[5]-Q[4],7)-F(Q[4],Q[3],Q[2])-Q[1]-0xf57c0faf;$

$x[5]=RR(Q[6]-Q[5],12)-F(Q[5],Q[4],Q[3])-Q[2]-0x4787c62a;$

$x[6]=RR(Q[7]-Q[6],17)-F(Q[6],Q[5],Q[4])-Q[3]-0xa8304613;$

Zkontrolujeme, zda jsou splněny stacionární podmínky pro  $Q[18]$ , jinak se vracíme o tři kroky zpět na novou volbu  $Q[2]$  v kroku 5 (resp.  $Q[1]$  v kroku 3). Podle terminologie [8] zde použijeme mnohonásobnou modifikaci  $x[2 - 6]$ .

9. V tomto okamžiku jsou deterministicky splněny stacionární podmínky pro  $Q[1 - 18]$ .

10. Zbývajících 29 podmínek podle [8] (resp. 27 podmínek podle [3]) je splněno pravděpodobnostně. Pokud nenalezneme kolizi, máme více než  $2^{29}$  možností volby  $Q[1 - 2]$ , eventuelně další možnosti volby 22. bitu  $Q[3 - 16]$ . Se složitostí  $2^{29}$  ( $2^{27}$ ) nalezneme kolizi druhého bloku. Složitost nalezení kolize 2. bloku v [8] je uváděna jako  $2^{30}$ .

## 5.8. Další mnohonásobné modifikace zpráv v druhém bloku

### 5.8.1. Deterministické splnění podmínek Q[1 -19]

Navrhujeme další postup, spočívající v deterministickém splnění podmínek Q[1] až Q[19].

1. Zvolíme libovolně Q[1 - 16], splňující stacionární podmínky
2. Vypočteme x[0 - 15]
3. Dokud není splněno Q[17] měníme Q[2] (vypočteme x[1 - 5] a zjistíme, zda nové x[1] splňuje Q[17])
4. Dokud není splněno Q[18], měníme Q[7] (vypočteme x[6 - 10] a zjistíme, zda nové x[6] splňuje Q[18])
5. Dokud není splněno Q[19], měníme Q[12] (vypočteme x[11 - 15] a zjistíme, zda nové x[11] splňuje Q[19])

Nyní jsou podmínky Q[1 - 19] splněny deterministicky. Zbývajících 27 podmínek podle [8] (resp. 25 podmínek podle [3]) je splněno pravděpodobnostně. Pokud nenalezneme kolizi, máme více než  $2^{27}$  možností volby Q[1 -16]. Se složitostí  $2^{27}$  nalezneme kolizi druhého bloku. To je další snížení složitosti. Tento postup nebyl experimentálně ověřen.

### 5.8.2. Deterministické splnění podmínek Q[1 - 21]

Další postup spočívá v deterministickém splnění podmínek Q[1 - 21].

1. Zvolíme libovolně Q[2 - 16], splňující stacionární podmínky.
2. Vypočteme x[5 -15].
3. Měníme Q[1] dokud není splněno Q[17 - 21] (z Q[1] vypočteme nové x[0 - 4]).
4. Nyní měníme Q[8 - 12] tak, aby splňovaly stacionární podmínky a neměnily hodnotu x[11]. Protože se nemění hodnoty x[1, 6, 11, 0, 5], podmínky Q[17 - 21] zůstávají splněny. Vypočteme x[7 -15]. Zbývajících 24 stacionárních podmínek je splněno pravděpodobnostně.

Máme více než  $2^{24}$  možností volby Q[8 -12]. Se složitostí  $2^{24}$  nalezneme kolizi druhého bloku. To je další snížení složitosti. Tento postup nebyl experimentálně ověřen.

## 6. Závěr

Nezávisle na popisu metody Wangové a kol. [1, 8] jsme navrhli nové metody hledání kolizí MD5. Naše metody jsou rychlejší. Dosáhli jsme složitosti  $2^{33}$  vs.  $2^{37}$  v prvním bloku a  $2^{29}$  (nebo dokonce  $2^{24}$ ) vs.  $2^{30}$  v druhém bloku. To umožňuje nalézat kolize i na notebooku během několika hodin. Metody pracují pro jakoukoli zvolenou inicializační hodnotu.

### Poděkování

Rád bych poděkoval svým přátelům za jejich pomoc. Milanu Nosálovi (LEC, s.r.o.) za jeho pomoc při ladění programu, Tomáši Rosovi a Ondřeji Pokornému a Milanu Nosálovi za provádění experimentů na jejich domácích počítačích, Tomáši Jabůrkovi za technickou pomoc s experimenty a Ondřeji Mikle a Tomáši Rosovi za pomoc při překladu příspěvku, a všem za cenné připomínky.

### Poznámka

V posledním experimentu, který dělal Ondřej Pokorný na svém domácím PC (Intel Pentium, 1 GHz), obdržel za 58 hodin a 32 minut 14 kolizí. To dává ještě optimističtější čas pro nalezení kolize (1 kolize za 4 hodiny a 11 minut) než na autorově notebooku.

### Domácí stránka projektu

[http://cryptography.hyperlink.cz/MD5\\_collisions.html](http://cryptography.hyperlink.cz/MD5_collisions.html)

## 7. Příloha: Příklady

### 7.1. Příklad: Kolize MD5 se standardní inicializační hodnotou IV

IV podle [2]:

```
context->state[0] = 0x67452301;  
context->state[1] = 0xefcdab89;  
context->state[2] = 0x98badcfe;  
context->state[3] = 0x10325476;
```

#### První zpráva:

```
0xA6, 0x64, 0xEA, 0xB8, 0x89, 0x04, 0xC2, 0xAC,  
0x48, 0x43, 0x41, 0x0E, 0x0A, 0x63, 0x42, 0x54,  
0x16, 0x60, 0x6C, 0x81, 0x44, 0x2D, 0xD6, 0x8D,  
0x40, 0x04, 0x58, 0x3E, 0xB8, 0xFB, 0x7F, 0x89,  
0x55, 0xAD, 0x34, 0x06, 0x09, 0xF4, 0xB3, 0x02,  
0x83, 0xE4, 0x88, 0x83, 0x25, 0x71, 0x41, 0x5A,  
0x08, 0x51, 0x25, 0xE8, 0xF7, 0xCD, 0xC9, 0x9F,  
0xD9, 0x1D, 0xBD, 0xF2, 0x80, 0x37, 0x3C, 0x5B,  
0x97, 0x9E, 0xBD, 0xB4, 0x0E, 0x2A, 0x6E, 0x17,  
0xA6, 0x23, 0x57, 0x24, 0xD1, 0xDF, 0x41, 0xB4,  
0x46, 0x73, 0xF9, 0x96, 0xF1, 0x62, 0x4A, 0xDD,  
0x10, 0x29, 0x31, 0x67, 0xD0, 0x09, 0xB1, 0x8F,  
0x75, 0xA7, 0x7F, 0x79, 0x30, 0xD9, 0x5C, 0xEB,  
0x02, 0xE8, 0xAD, 0xBA, 0x7A, 0xC8, 0x55, 0x5C,  
0xED, 0x74, 0xCA, 0xDD, 0x5F, 0xC9, 0x93, 0x6D,  
0xB1, 0x9B, 0x4A, 0xD8, 0x35, 0xCC, 0x67, 0xE3.
```

#### Druhá zpráva:

```
0xA6, 0x64, 0xEA, 0xB8, 0x89, 0x04, 0xC2, 0xAC,  
0x48, 0x43, 0x41, 0x0E, 0x0A, 0x63, 0x42, 0x54,  
0x16, 0x60, 0x6C, 0x01, 0x44, 0x2D, 0xD6, 0x8D,  
0x40, 0x04, 0x58, 0x3E, 0xB8, 0xFB, 0x7F, 0x89,  
0x55, 0xAD, 0x34, 0x06, 0x09, 0xF4, 0xB3, 0x02,  
0x83, 0xE4, 0x88, 0x83, 0x25, 0xF1, 0x41, 0x5A,  
0x08, 0x51, 0x25, 0xE8, 0xF7, 0xCD, 0xC9, 0x9F,  
0xD9, 0x1D, 0xBD, 0x72, 0x80, 0x37, 0x3C, 0x5B,  
0x97, 0x9E, 0xBD, 0xB4, 0x0E, 0x2A, 0x6E, 0x17,  
0xA6, 0x23, 0x57, 0x24, 0xD1, 0xDF, 0x41, 0xB4,  
0x46, 0x73, 0xF9, 0x16, 0xF1, 0x62, 0x4A, 0xDD,  
0x10, 0x29, 0x31, 0x67, 0xD0, 0x09, 0xB1, 0x8F,  
0x75, 0xA7, 0x7F, 0x79, 0x30, 0xD9, 0x5C, 0xEB,  
0x02, 0xE8, 0xAD, 0xBA, 0x7A, 0x48, 0x55, 0x5C,  
0xED, 0x74, 0xCA, 0xDD, 0x5F, 0xC9, 0x93, 0x6D,  
0xB1, 0x9B, 0x4A, 0x58, 0x35, 0xCC, 0x67, 0xE3.
```

#### Společná haš:

0x2B,0xA3,0xBE,0x5A,0xA5,0x41,0x00,0x6B,  
0x62,0x37,0x01,0x11,0x28,0x2D,0x19,0xF5.

## 7.2. Příklad: Kolize MD5 se zvolenou inicializační hodnotou IV

```
context->state[0] = 0xabaaaaaa;  
context->state[1] = 0xaaacaaaa;  
context->state[2] = 0xaaaadaaa;  
context->state[3] = 0xaaaaaaaa;
```

### První zpráva:

0x9E,0x83,0x2A,0x4C,0x95,0x64,0x5E,0x2B,  
0x2E,0x1B,0xB0,0x70,0x47,0x1E,0xBA,0x13,  
0x7F,0x1A,0x53,0x43,0x22,0x34,0x25,0xC1,  
0x40,0x04,0x58,0x3E,0xB8,0xFB,0x7F,0x89,  
0x55,0xAD,0x34,0x06,0x09,0xF4,0xB3,0x02,  
0x83,0xE4,0x88,0x83,0x25,0x71,0x41,0x5A,  
0x08,0x51,0x25,0xE8,0xF7,0xCD,0xC9,0x9F,  
0xD9,0x1D,0xBD,0xF2,0x80,0x37,0x3C,0x5B,  
0x89,0x62,0x33,0xEC,0x5B,0x0C,0x8D,0x77,  
0x19,0xDE,0x93,0xFA,0xA1,0x44,0xA8,0xCC,  
0x56,0x91,0x9E,0x47,0x00,0x0C,0x00,0x4D,  
0x40,0x29,0xF1,0x66,0xD1,0x09,0xB1,0x8F,  
0x75,0x27,0x7F,0x79,0x30,0xD5,0x5C,0xEB,  
0x42,0xE8,0xAD,0xBA,0x78,0xCC,0x55,0x5C,  
0xED,0xF4,0xCA,0xDD,0x5F,0xC5,0x93,0x6D,  
0xD1,0x9B,0x0A,0xD8,0x35,0xCC,0xE7,0xE3.

### Druhá zpráva:

0x9E,0x83,0x2A,0x4C,0x95,0x64,0x5E,0x2B,  
0x2E,0x1B,0xB0,0x70,0x47,0x1E,0xBA,0x13,  
0x7F,0x1A,0x53,0xC3,0x22,0x34,0x25,0xC1,  
0x40,0x04,0x58,0x3E,0xB8,0xFB,0x7F,0x89,  
0x55,0xAD,0x34,0x06,0x09,0xF4,0xB3,0x02,  
0x83,0xE4,0x88,0x83,0x25,0xF1,0x41,0x5A,  
0x08,0x51,0x25,0xE8,0xF7,0xCD,0xC9,0x9F,  
0xD9,0x1D,0xBD,0x72,0x80,0x37,0x3C,0x5B,  
0x89,0x62,0x33,0xEC,0x5B,0x0C,0x8D,0x77,  
0x19,0xDE,0x93,0xFA,0xA1,0x44,0xA8,0xCC,  
0x56,0x91,0x9E,0xC7,0x00,0x0C,0x00,0x4D,  
0x40,0x29,0xF1,0x66,0xD1,0x09,0xB1,0x8F,  
0x75,0x27,0x7F,0x79,0x30,0xD5,0x5C,0xEB,  
0x42,0xE8,0xAD,0xBA,0x78,0x4C,0x55,0x5C,  
0xED,0xF4,0xCA,0xDD,0x5F,0xC5,0x93,0x6D,  
0xD1,0x9B,0x0A,0x58,0x35,0xCC,0xE7,0xE3.

### Společná haš:

0xef,0x2e,0xae,0x54,0xe0,0x34,0x70,0x7c,  
0xa2,0x6e,0xb0,0x9b,0x45,0xc7,0xe4,0x87.

## Literatura

- [1] Xiaoyun Wang, Dengguo Feng , Xuejia Lai, Hongbo Yu: Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD, rump session, CRYPTO 2004, *Cryptology ePrint Archive*, Report 2004/199, first version (August 16, 2004), second version (August 17, 2004), <http://eprint.iacr.org/2004/199.pdf>
- [2] Ronald Rivest: The MD5 Message Digest Algorithm, RFC1321, April 1992, <ftp://ftp.rfc-editor.org/in-notes/rfc1321.txt>
- [3] Philip Hawkes, Michael Paddon, Gregory G. Rose: Musings on the Wang et al. MD5 Collision, *Cryptology ePrint Archive*, Report 2004/264, 13 October 2004, <http://eprint.iacr.org/2004/264.pdf>
- [4] Ondrej Mikle: Practical Attacks on Digital Signatures Using MD5 Message Digest, *Cryptology ePrint Archive*, Report 2004/356, <http://eprint.iacr.org/2004/356.pdf>, 2nd December 2004
- [5] Dan Kaminsky: MD5 To Be Considered Harmful Someday, *Cryptology ePrint Archive*, Report 2004/357, <http://eprint.iacr.org/2004/357.pdf>, 6 December 2004
- [6] Arjen Lenstra, Xiaoyun Wang and Benne de Weger: Colliding X.509 Certificates, *Cryptology ePrint Archive*, Report 2005/067, <http://eprint.iacr.org/2005/067.pdf>
- [7] Vlastimil Klima: Finding MD5 Collisions – a Toy For a Notebook, *Cryptology ePrint Archive*, Report 2005/075, <http://eprint.iacr.org/2005/075.pdf>, March 5, 2005
- [8] Xiaoyun Wang and Hongbo Yu: How to Break MD5 and Other Hash Functions, <http://www.infosec.sdu.edu.cn/paper/md5-attack.pdf>, published on the web on March 6, 2005
- [9] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, Xiuyuan Yu: Cryptanalysis of the Hash Functions MD4 and RIPEMD, <http://www.infosec.sdu.edu.cn/paper/md4-ripemd-attck.pdf>, published on the web on March 6, 2005
- [10] Xiaoyun Wang, Yiqun Lisa Yin, Hongbo Yu: Collision Search Attacks on SHA1, <http://theory.csail.mit.edu/~yiqun/shanote.pdf>, February 2005