Crypto-World

Informační sešit GCUCMP

Ročník 6, číslo 11/2004 **15. listopadu 2004**

11/2004 - speciál

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

Nedůvěřujte kryptologům

Vlastimil Klíma

(http://cryptography.hyperlink.cz , v.klima@volny.cz)

Příspěvek je určen těm, kdo nemají podrobné znalosti o hašovacích funkcích, ale přitom se jich nějakým způsobem týká jejich bezpečnost, poněkud otřesená v srpnu t. r. nalezením kolizí u několika hašovacích funkcí.

Obsah

1. Úvod

Tento příspěvek je určen těm, kdo nemají podrobné znalosti o hašovacích funkcích, ale přitom se jich nějakým způsobem týká jejich bezpečnost, poněkud otřesená v srpnu t. r. nalezením kolizí u několika hašovacích funkcí. Je určen zejména pro technické pracovníky v oblasti IS/IT. Cílem budou proto zcela praktická doporučení, ale uvedeme si i kryptologický kontext, aby bylo vidět, z čeho tato doporučení pramení. Tento kontext bude také trochu sloužit pro pochopení míry rizik vyplývajících z vývoje v kryptologii.

Zatímco v jiných oblastech bezpečnosti se počítá s nedůvěrou, tj. s tím, že dílčí funkce mohou být špatně naprogramovány, opravovány nebo vyměňovány, kryptografické nástroje jsou ještě chápány jako něco neobvyklého. Příspěvek vybízí k tomu, aby se tento postoj ke kryptografickým technikám změnil a pracovalo se s nimi jako s jakýmikoliv jinými bezpečnostními nástroji, to znamená a priori jim nedůvěřovat, sledovat vývoj v dané oblasti a běžně provádět update nebo upgrade.

Tím se naplňuje název příspěvku i následující motto:

> To nejlepší, co pro vás kryptologové mohou udělat, je, když vás přesvědčí, abyste jim slepě nedůvěřovali. Je to nutná podmínka pro to, abyste ani vy ani oni neusnuli na vavřínech.

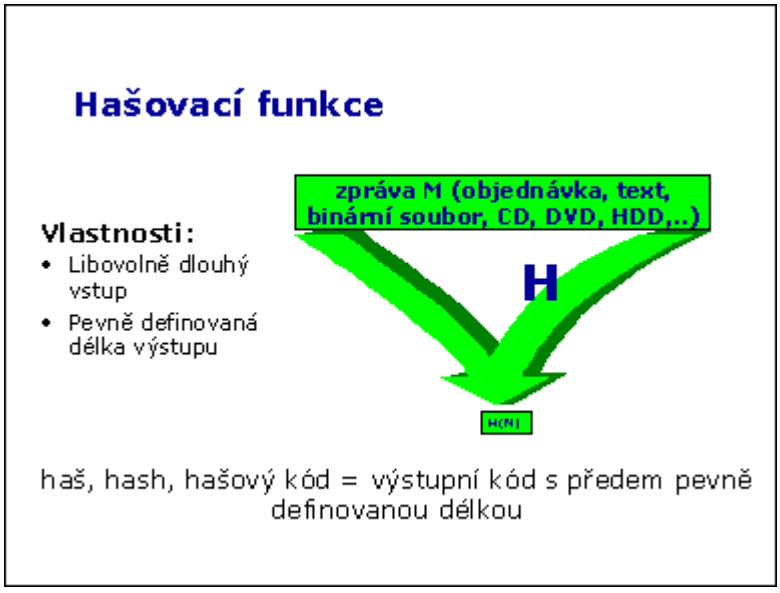
Aby se ale příslušný update a upgrade mohl u oslabené kryptografické techniky provést, je nutné, aby je architekti bezpečnostních produktů už tak navrhovali. **Změna filozofie produktu na vyměnitelnost kryptografických technik je koncepční a trvá roky**, proto se jedná dnes spíše o vizi než reálný program. Kdo ale tento fakt pochopí jako první, může získat náskok před ostatními. Řada standardů na toto pamatuje tím, že zavádí identifikátory různých algoritmů nebo dokonce identifikátory pro seskupení algoritmů do tzv. sad (cipher suites). Pak prostým přidáním nové knihovní funkce a změnou konfigurace lze vyměnit jeden algoritmus za druhý. Tak by tomu mělo být všude.

Problémy nastávají, pokud algoritmy nejsou "datově kompatibilní", čímž máme na mysli například když je výstup nového (hašovacího, šifrovacího) algoritmu širší než předchozí nebo když nová technika vyžaduje zavedení změn v protokolu. Například přechod ze šifrování v modu ECB na šifrování v modu CBC vyžaduje přenos inicializačního bloku navíc a dále přítomnost generátoru náhodných znaků pro vytvoření inicializační hodnoty. To už je zásah do datových struktur i do hardwaru.
Příkladem budiž také přechod od blokové šifry DES k nové AES. Zde je také více zásadních zásahů, které modularitu ztěžují. Za prvé je to změna šířky zpracovávaného bloku dat ze 64 na 128 bitů, za druhé delší klíč (z 56 na 128 bitů a výše) a za třetí variabilita klíče (tři možné délky klíče - 128, 192, 256 bitů). Proto výměna DES za AES probíhá postupně, a to tak, že stará zařízení se nechávají "dožít" nebo se změní DES na TripleDES (kde šířka bloku zůstává stejná a mění se jen délka klíče) a AES se zavádí zejména do nových verzí zařízení. U hašovacích funkcí probíhal z důvodu bezpečnosti přechod od MD4, u níž byly nalezeny kolize v roce 1995 (publikováno v r. 1996), k MD5 poměrně velmi rychle, protože datově byla tato změna jednoduchá - stejné vstupy i výstupy, a měnil se jen kryptografický vnitřek. Po nalezení slabiny u MD5 (tzv. kolize kompresní funkce, viz dále) se začalo přecházet na funkci SHA-1, ale spíše také jen u nových zařízení, a to ve stylu přechodu od DES k AES. Protože výstup SHA-1 je 160bitový a MD5 128bitový, zasahuje tato změna do "datové

kompatibility" a vývojáři a bezpečnostní architekti nebyli ochotni kvůli "nějaké teoretické slabině" tuto změnu prosazovat. Myslí se pochopitelně plný přechod, nikoli jen použití SHA-1 oříznuté na 128 bitů - v takovém případě by šlo výměnu za MD5 provést bez problémů. Díky neochotě cokoli měnit však **MD5 zůstala v desítkách standardů a zařízení a bude těžké ji vyměnit**. Z toho důvodu je nutné blíže pochopit čínský útok, který MD5 dále oslabuje, a příslušná rizika zhodnotit.

2. Hašovací funkce a kryptografická hašovací funkce

V úvodu si uvedeme nezbytné pojmy a zároveň si je doplníme o nové souvislosti, podrobněji se lze s nimi seznámit např. v [ST0204] a [ST1204], odkud jsme do tohoto textu převzali několik odstavců.



Obr.: Hašovací funkce v původní definici

2.1. Hašovací funkce

Hašovací funkce byla původně označením pro funkci, která libovolně velkému vstupu přiřazovala krátký hašovací kód o pevně definované délce.

2.2. Kryptografická hašovací funkce

Nyní se pojem hašovací funkce používá v kryptografii pro *kryptografickou hašovací funkci*, která má oproti původní definici ještě navíc vlastnost jednosměrnosti a bezkoliznosti.

Pojmy jednosměrnost a jednocestnost používáme jako synonyma.

2.3. Mnoho k jedné

Hašovací funkce h zpracovává prakticky neomezeně dlouhá vstupní data M na krátký výstupní hašový kód $h(M)$. Například u hašovacích funkcí MD5/SHA-1/SHA-256/SHA-512 je to 128/160/256/512 bitů.

2.4. Jednosměrnost

Hašovací funkce musí být *jednosměrná* (one-way) a *bezkolizní* (collission-free). Jednosměrnost znamená, že z M lze jednoduše vypočítat $h(M)$, ale obráceně to není pro náhodně zadaný hašový kód H výpočetně zvládnutelné.

2.5. Bezkoliznost (odolnost proti kolizi), bezkoliznost prvního řádu

Bezkoliznost (stručnější označení pro "odolnost proti kolizi") požaduje, aby bylo výpočetně nezvládnutelné nalezení libovolných dvou různých (byť naprosto nesmyslných) zpráv M a M' tak, že $h(M) = h(M')$. Pokud se toto stane, říkáme, že jsme nalezli **kolizi**.
Bezkoliznosti, kterou jsme právě popsali, říkáme bezkoliznost 1. řádu nebo jednoduše bezkoliznost.

2.6. Bezkoliznost druhého řádu

Existuje několik dalších variant definic bezkoliznosti, stupňující sílu požadavků na hašovací funkci. Další nejznámější definice je tzv. odolnost proti nalezení *druhého* vzoru neboli bezkoliznost druhého řádu. Řekneme, že hašovací funkce h je odolná proti nalezení druhého vzoru, jestliže pro *daný* náhodný vzor x je výpočetně neuskutečnitelné nalézt druhý vzor $y \neq x$ tak, že $h(x) = h(y)$.

2.7. Vztah bezkoliznosti prvního a druhého řádu

Setkáváme se často s tím, že prolomení hašovací funkce se spojuje až se schopností útočníka konstruovat kolize druhého řádu. Zejména se požaduje demonstrovat, že "právě k tomuto dokumentu" je útočník schopen vytvořit jiný (lišící se nejlépe ve výši peněžní částky v něm uvedené), se stejnou haší.

Tento požadavek je poněkud ze staré doby a připomíná amatérské konstruktéry proprietárních šifer, kteří svůj systém považují za prolomený jen pokud je kryptoanalytik schopen rozšifrovat poskytnutý šifrový text. Přitom tvůrce šifry není ochoten předat ani popis svého algoritmu. Funkce, u níž lze nacházet kolize 2. řádu je už tak slabá, že její nepoužitelnost uznají i laikové.

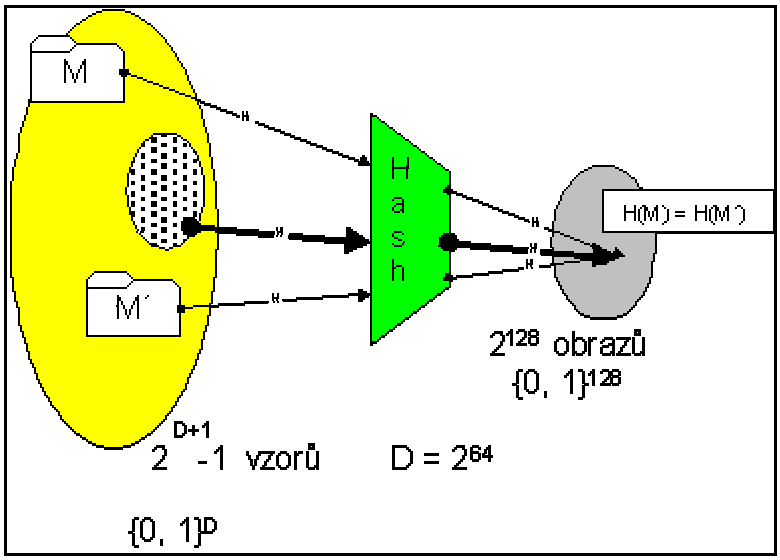
Moderní kryptografie však považuje za bezpečné pouze ty funkce, které jsou odolné proti nalezení jakýchkoliv kolizí, tedy kolizí 1. řádu, protože takové funkce jsou pochopitelně silnější.

Proto pokud je u hašovacích funkcí nalezena jakákoliv kolize, je podkopána základní vlastnost z definice hašovací funkce. Je podkopána vlastnost, na níž se spoléhá - a to, že můžeme podepisovat haše namísto dokumentů. Najednou dva dokumenty (nebo binární soubory) mají stejnou haš, a tudíž lze popřít, že dotyčný podepsal dokument číslo jedna s tím, že podepsal dokument číslo dva. Tedy je ohrožena vlastnost a bezpečnostní služba nepopiratelnosti. Požadavek odolnosti proti kolizi 1. řádu je proto zcela přirozený.

2.8. Aplikace

Jednosměrnost umožňuje, aby se v operačních systémech kontrolovala **přihlašovací hesla** bez nutnosti jejich přímého uložení v systému. Pak by byla přístupná například administrátorům systému. Místo toho se ukládají hesla ve formě jejich haší (v praxi ještě s

použitím soli). Z haše díky jednosměrnosti není možné určit heslo, pokud ovšem heslo není zvoleno příliš jednoduché (v tom případě je možné aplikovat slovníkový útok na haše hesel ze slovníku).



Obr. : Kolize

Bezkoliznost se, jak jsme uvedli, zásadním způsobem využívá **k digitálním podpisům**. Nepodepisuje se přímo zpráva, často velmi dlouhá (u MD5/SHA-1/SHA-256 prakticky až do délky $D = 2^{64}$-1 bitů), ale pouze její haš. Můžeme si to dovolit, protože bezkoliznost zaručuje, že není možné nalézt dva dokumenty se stejnou haší. Proto můžeme podepisovat haš. Často ale dochází k nepochopení základního principu hašovacích funkcí. Každý cítí, že je tady něco divného, protože možných zpráv je mnoho ($1 + 2^1 + ... + 2^D = 2^{D+1}$ -1) a hašovacích kódů málo (u MD5 například pouze 2^{128}). Musí proto existovat ohromné množství zpráv, vedoucích na tentýž hašový kód - v průměru je to řádově 2^{D-127}. **Kolizí tedy existuje ohromné množství**. Pointa je v tom, že hašovací funkce jsou konstruovány tak, abychom kolize nedokázali nalézt, a to díky ohromné výpočetní složitosti takového postupu. Nalézání kolizí je tedy možné, ale nad naše výpočetní možnosti. I když uvidíme, že vytváření hašovacích kódů je opravdu neskutečně složité, nalezení kolizí je přesto pouze otázkou intelektuální výzvy, neboť není prokázáno, že je *nelze* nalézt.

2.9. Bezpečnost hašovacích funkcí

A nyní se dostáváme k jádru věci, které bylo zapomenuto. **Hašovací funkce nejsou prokazatelně bezpečné nástroje a jejich bezpečnost (jednosměrnost, bezkoliznost) závisí pouze na stavu vědy v oblasti kryptografie a kryptoanalýzy.** Čas od času se štěstí zvrtne a některá šifra, hašovací funkce nebo podpisové schéma padne díky odhalené slabině. Na to nejsme, ale měli bychom být připraveni - na úrovni techniků, vývojářů, administrátorů, bezpečnostních architektů i manažerů, odpovědných za bezpečnost. **Prolomení některých kryptografických technik musí být přijímáno nikoli jako nedůvěra v kryptologii, ale jako průvodní jev rozvoje poznání v této oblasti.**

Jakmile někdo nalezne kolizi hašovací funkce, je nejjednodušší a nejbezpečnější tuto funkci vyměnit za jinou. To je ideální řešení, které v praxi někdy prostě není možné. Potom je nutné zkoumat, v jakých aplikacích je hašovací funkce použita a které vlastnosti celého systému jsou ohroženy.

Taková hašovací funkce generálně ztrácí smysl, neboť hypotéza o její bezkoliznosti byla vyvrácena. **Zejména by neměla být používána k digitálním podpisům, neboť tam kolize znamená, že je možné předložit dvě různé zprávy s tímtéž digitálním podpisem**, platným pro obě zprávy. Existují ale techniky, kde nejsou využity všechny vlastnosti hašovací funkce a kde porušení bezkoliznosti (nebo částečné porušení bezkoliznosti) nevadí.

3. Čínský útok na hašovací funkce MD4, MD5, RIPEMD, HAVAL-128 a SHA-0

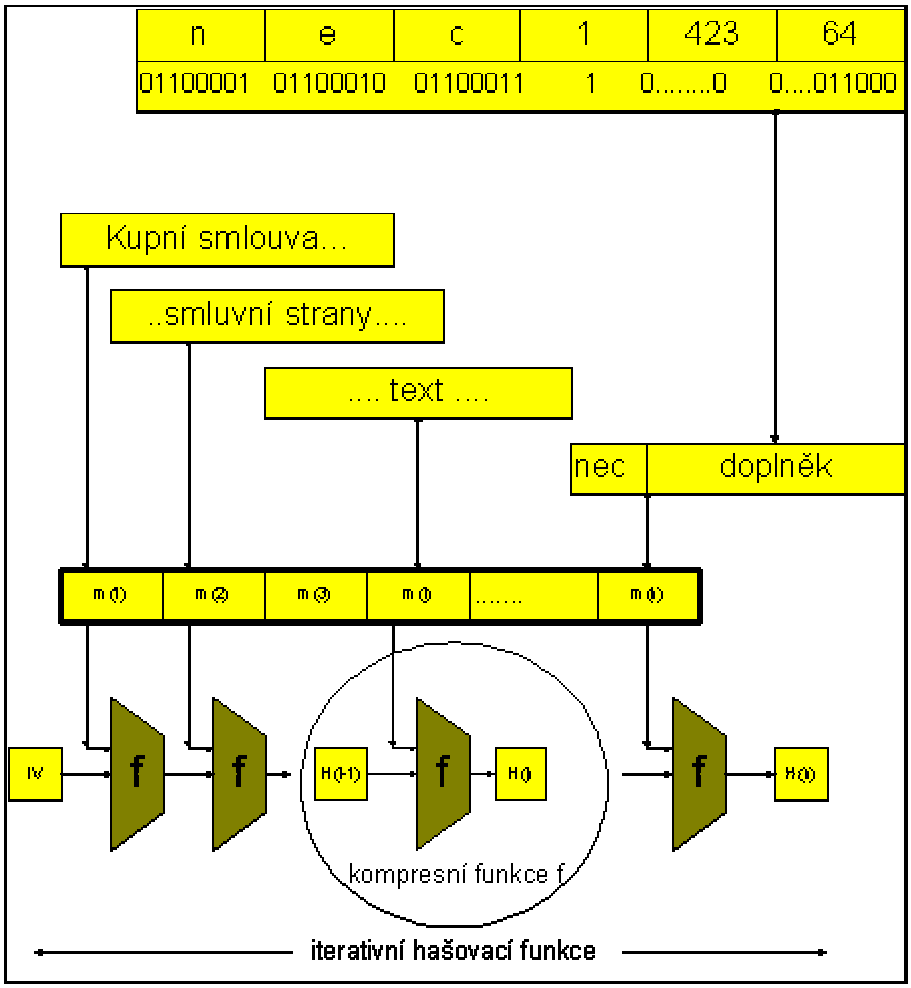
Zatím stručně předesíláme, že Číňané předvedli, že umí najít velkou třídu kolizí u hašovacích funkcí MD4, MD5, RIPEMD a HAVAL-128 s časovou náročností od sekund po 1-2 hodiny [WFLY04]. U SHA-0 by nalezli kolize se složitostí 2^{40} (výpočtů SHA-0). Pro HAVAL-160 může být kolize nalezena s pravděpodobností 2^{-32}. Jak jsme už zmínili, neznáme jejich postup, ale ukážeme si jejich výsledky na MD5.

4. Princip většiny moderních hašovacích funkcí, příklad MD5

Princip většiny moderních hašovacích funkcí je podobný a vysvětlíme si ho na prolomené funkci MD5. Hašovací funkce, o kterých se dále zmíníme, jsou všechny iterativního typu. Jsou definovány pomocí kompresní funkce f a inicializační hodnoty H_0. Při hašování zprávy M je tato předepsaným způsobem doplněna a poté rozdělena na bloky m_i (i = 1,..., n) pevné délky. U námi popisovaných funkcí (MD4, MD5, SHA-0, SHA-1, SHA-256) je to 512 bitů.

Doplnění vstupní zprávy M se provádí tak, že nejprve je povinně doplněna bitem 1, poté co nejmenším počtem nulových bitů (může jich být 0 - 447) tak, aby do celistvého násobku 512 bitů zbývalo ještě 64 bitů, a nakonec je těchto 64 bitů vyplněno 64bitovým vyjádřením počtu bitů původní zprávy M. Délka zprávy tedy také vstupuje do hašovacího procesu, viz obrázek.

Hašování potom probíhá postupně po jednotlivých blocích m_i v cyklu podle i od 1 do n. Kompresní funkce f v i-tém kroku (i = 1,..., n) zpracuje vždy daný kontext H_{i-1} a blok zprávy m_i na nový kontext H_i. Vidíme, že název kompresní funkce je vhodný, neboť funkce f zpracovává širší vstup (H_{i-1}, m_i) na mnohem kratší H_i, tedy blok zprávy m_i se sice funkčně promítne do H_i, ale současně dochází ke ztrátě informace (šířka kontextu H_0, H_1,...H_i,... zůstává stále stejná). Kontextem bývá obvykle několik 16bitových nebo 32bitových slov, u MD5 jsou to čtyři slova A, B, C, D (dohromady 128 bitů).



Obr.: Doplňování, kompresní funkce a iterativní hašovací funkce

Po zhašování posledního bloku m_n dostáváme kontext H_n, z něhož bereme buď celou délku nebo část jako výslednou haš. U funkce MD5 je šířka kontextu 128 bitů a výslednou haš tvoří všech 128 bitů kontextu H_n.

Pro další výklad si povšimněme, že když začínáme hašovat druhý blok m_2, je to jako bychom začínali hašování od 1. bloku, ale s inicializační hodnotou H_1 nikoli H_0. Čili výsledek hašování zprávy m_1, m_2, m_3 ... s inicializační konstantou H_0 je stejný jako výsledek hašování zprávy m_2, m_3, ... s inicializační konstantou H_1, atd.

Kolize hašovací funkce h spočívá v nalezení různých zpráv M1 a M2 tak, že h(M1) = h(M2).

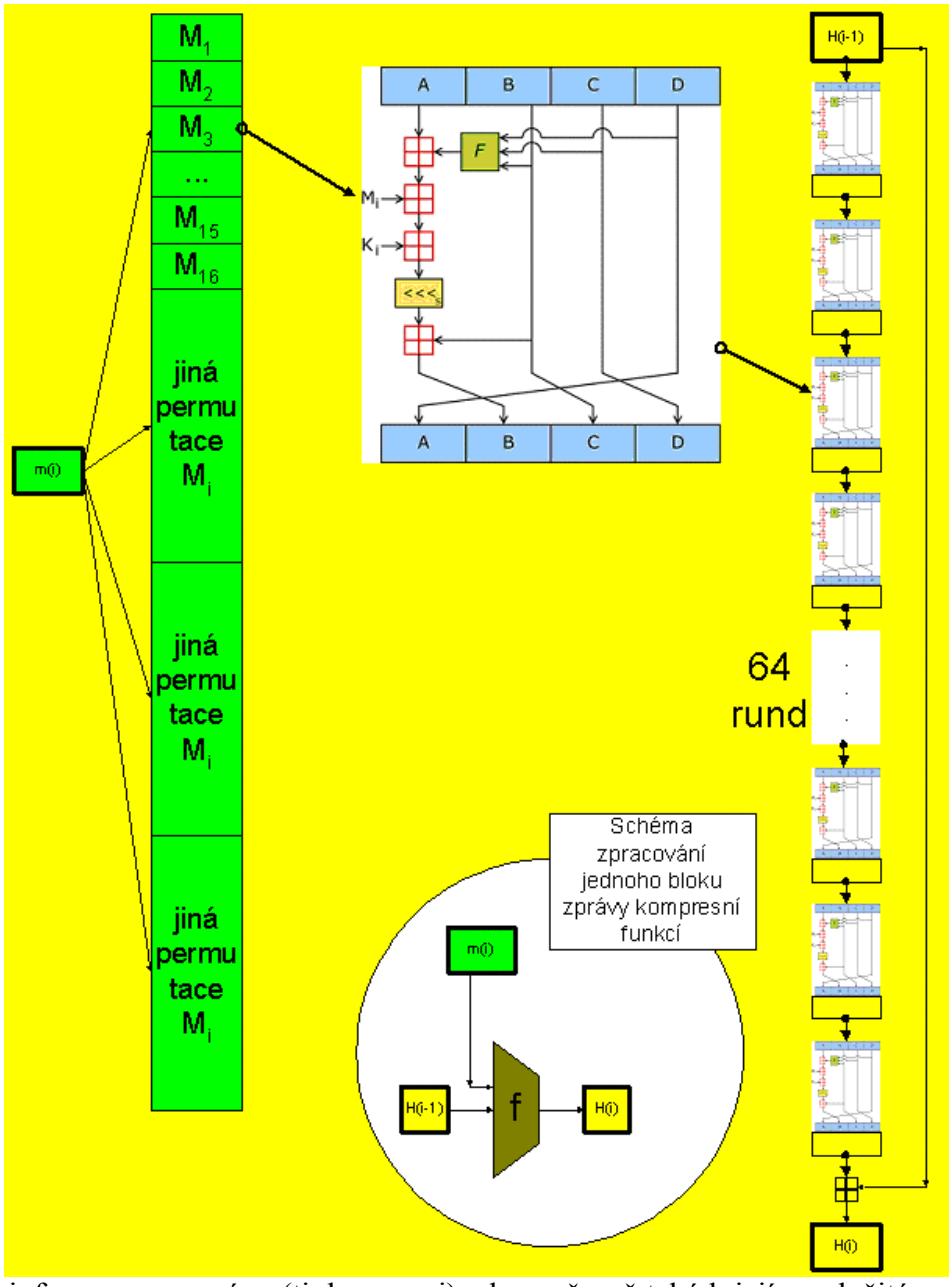
Kolize kompresní funkce f spočívá v nalezení inicializační hodnoty H a dvou různých bloků B1 a B2 tak, že $f(H, \text{B1}) = f(H, \text{B2})$.

Rozdíl je tedy v tom, zda si můžeme libovolně volit inicializační hodnotu H_0. Jak je vidět, nalezení kolize kompresní funkce je obecně „snazší“ úloha, nicméně nebývá tak „daleko“ od nalezení kolize haše.

Kompresní funkce je velmi složitá, aby zajistila míchání bitů zprávy s kontextem a jednocestnost. Kontext tak postupně vstřebává jednotlivé bity a bloky zprávy a ukládá je v sobě složitým způsobem. Právě tato složitost má zajistit jak nemožnost invertovat tento

proces, tak odolnost proti nalézání kolizí. Ukážeme si kompresní funkci pro MD5, jednu z těch jednodušších.

4.1. Příklad hašovací funkce: MD5

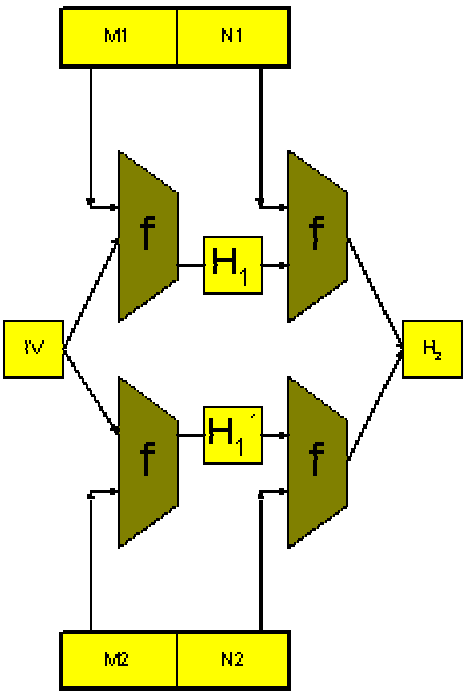


U MD5 tvoří kontext 4 32bitová slova A, B, C a D. Na obrázku vidíme jednu tzv. rundu hašování u MD5. Poznamenejme, že m_i je jeden 512 bitový blok zprávy. Ten je rozdělen na 16 slov M_0, M_1, ..., M_{15}, a tato posloupnost je opakována 4x za sebou (v různých permutacích). Každé slovo M_i je tak zpracováno v kompresní funkci celkem 4x. Na obrázku vidíme, že v kompresní funkci se ke kontextu složitým způsobem vždy přimíchá jedno 32bitové slovo typu M_i a po zpracování všech 16 slov (16 rund) se toto opakuje celkem čtyřikrát. V každé rundě dojde k určité ztrátě informace ze zprávy (tj. kompresi), ale současně také k jejímu složitému včlenění do kontextu. Poznamenejme jen, že na místě F se po 16 rundách střídají 4 různé nelineární funkce (F, G, H, I) a v každé rundě se využívá jiná konstanta K_i. Po uvedených 64 rundách dojde ještě k přičtení původního kontextu (H_{i-1}) k výsledku. Vznikne nový kontext H_i. Pokud by zpráva M měla jen jeden blok, byl by tento kontext (A, B, C, D) celkovým výsledkem. Pokud ne, je tento kontext považován jakoby za inicializační konstantu a pokračuje se stejným způsobem v hašování druhého bloku zprávy m_2. Po zpracování bloku m_n máme v registrech výslednou 128bitovou haš.

5. Interpretace čínského útoku

Současný stav hašovacích funkcí nejvíce ovlivnily útoky a práce prezentované v období konání nejprestižnější kryptografické konference CRYPTO 2004 v srpnu t. r. v Kalifornii. Jednak to byly příspěvky řádné, jednak mimořádné, velmi krátké neformální příspěvky, které se prezentují na tzv. rump session ve velmi rychlém sledu (pár minut na příspěvek). Čínský tým zde dostal, vzhledem k mimořádnému výsledku, 15 minut. Jak by ne, když to, co předvedli, mělo mimořádnou hodnotu pro kryptografickou komunitu [WFLY04]. Ukážeme si to opět na MD5.

Čínští výzkumníci přišli s metodou, jak nalézt kolize dvou různých 1024bitových zpráv. Jejich metoda spočívá v tom, že nejprve naleznou dvě **různé** 512bitové půlzprávy (bloky) M1, M2, což jim trvá cca hodinu, a potom k nim naleznou dvě **různé** 512bitové půlzprávy N1, N2 (což trvá už jen sekundy) tak, že složené zprávy (M1, N1) a (M2, N2) o dvou blocích mají stejnou haš.



Obr.: Princip čínského útoku na MD5

Dokonce uvádí předpis, jak se mají lišit první poloviny M1 a M2 těchto komponovaných zpráv a jak se mají lišit druhé poloviny N1 a N2. Jedním z velkých překvapení bylo, že tyto zprávy (M1, N1) a (M2, N2) se v obou polovinách liší pouze o tři bity, a že změny v první polovině a druhé polovině tvoří inverzní (!!!) konstanty, konkrétně M2 = M1 + C, N2 = N1 - C, kde C je vhodná 512bitová konstanta, mající v binárním vyjádření pouze 3 bity nenulové. Hlavní myšlenkou útoku tedy bylo najít takovou zvláštní konstantu C a pak během první hodiny útoku nalézt zprávu M1 tak, aby M1 a M2 = M1 + C při hašování vedly na určitý odlišný kontext H_1 (viz iterativní proces výše) takový, že tato odlišnost je srovnána při hašování následných bloků N1 a N2 = N1 - C, tj. kontexty H_2 jsou už totožné. Protože zprávy jsou 1024bitové, jsou doplněny ještě dalším blokem (viz definice doplňování) 10.....0 0...10000000000, který je ale pro obě zprávy stejný, neboť mají stejné délky 1024 bitů, takže i kontext H_3 (výsledná haš) je stejný.

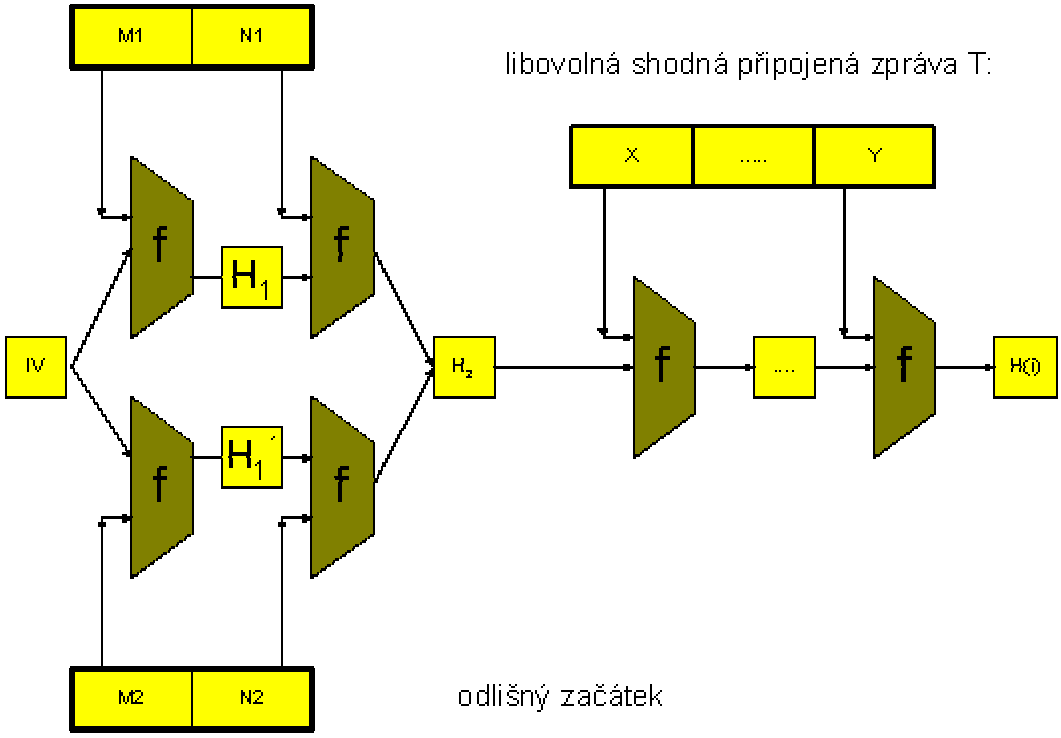
Uvedená vlastnost však **neplatí** pro všechny bloky M1, N1 a všechny konstanty C, takže je nutné je zvláštním způsobem konstruovat. To tvoří neznámé know-how čínského týmu. Nalezení M1 a C probíhá současně a trvá cca hodinu. Nalezení N1 pak už jen řádově vteřiny.

Avšak aby byla vidět síla útoku, autoři ukázali, že po první (hodinu trvající) fázi, kdy naleznou (M1, M2), jsou k této dvojici schopni nalézt **více dvojic** (N1 a N2), (N1´, N2´), ... vedoucích ke kolizi, tj. h(M1, N1) = h(M2, N2), h(M1, N1´) = h(M2, N2´). Tedy prokazují, že původní různé kontexty umí dovést ke stejné haši několika cestami.

5.1. Rozvíjení kolizí - 1. varianta

Hlavní myšlenka tedy je, že první odlišné bloky zpráv vytvoří sice různé kontexty H_1 a H_1´, ale druhé bloky to srovnají na celkový stejný výsledek H_2. Pokud nyní hašování buď

ukončíme nebo budeme v obou případech pokračovat už jen stejnými bloky zprávy, obdržíme v obou případech stejný kontext H_3, H_4,..., tedy kolizi. Proto za bloky (M1, N1) a (M2, N2) můžeme připojit libovolnou společnou zprávu T (zarovnanou na bloky), přičemž zprávy (M1, N1, T) a (M2, N2, T) povedou na stejnou haš.

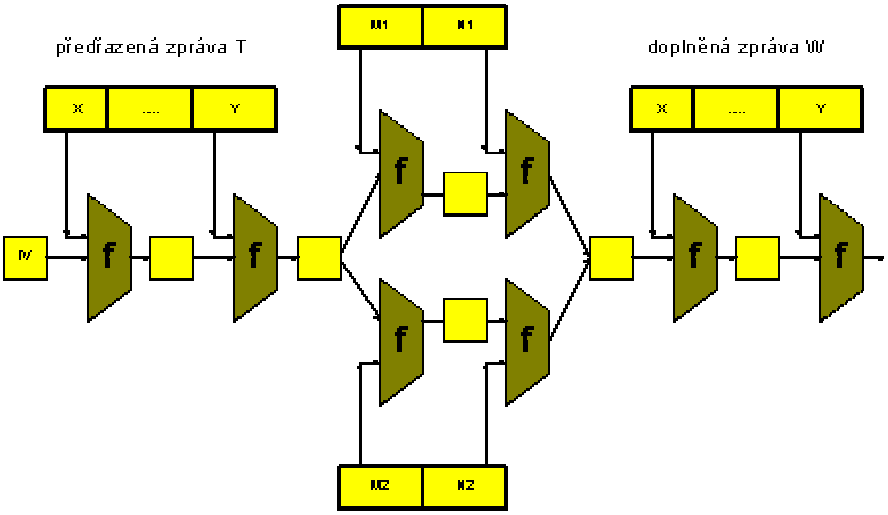


Obr.: Rozšiřování kolizí připojením libovolné zprávy

5.2. Rozvíjení kolizí - 2. varianta

Číňané zřejmě omylem prozradili více, a totiž, že jsou schopni svůj útok provést s libovolnou inicializační hodnotou IV, nejen tou, která je pevně definována pro MD5. Když to už bylo zřejmé, nakonec to i oficiálně prohlásili. Pro libovolnou inicializační hodnotu IV (H_0) jsou schopni nalézt zprávu M_{IV}, N_{IV} a 512bitovou konstantu C_{IV} (obsahující pouze 3 nenulové bity) tak, že zprávy (M_{IV}, N_{IV}) a ($M_{IV} + C_{IV}$, $N_{IV} - C_{IV}$) kolidují.

Mohou si proto také **zvolit libovolnou smysluplnou zprávu** T a poté k ní konstruovat jak jsou zvyklí (M1, N1) a (M2, N2) tak, že h(T, M1, N1) = h(T, M2, N2), čili zahajovat kolidující zprávy libovolnou zvolenou zprávou. Proč? Když hašují zprávu (T, ...) dojdou po zhašování T (uvažujeme T zarovnanou na bloky) k určitému kontextu H_n. Ten prohlásí za novou inicializační konstantu H_0' (je jim jedno, jakou má hodnotu) a začnou svůj útok od ní. Zkonstruují (M1, N1) a (M2, N2) tak, že *pro tuto inicializační konstantu* vedou ke kolizi, jinými slovy **dosáhnou toho, že *h*(T, M1, N1) = *h*(T, M2, N2) pro libovolnou zprávu T** zarovnanou na bloky. Připojíme-li předchozí výsledek, vedou na stejnou haš i zprávy (T, M1, N1, W) a (T, M2, N2, W) pro libovolné T a W, jak ukazuje obrázek.

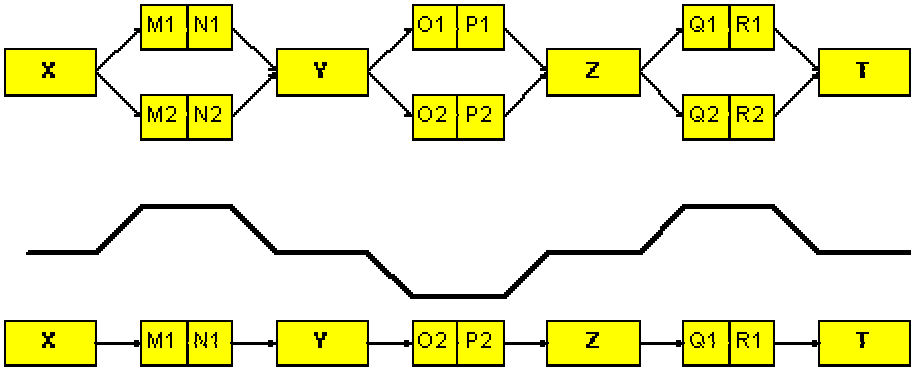


Obr.: Možnost rozšiřování kolizí před a za základní kolizí

> "Objevení kolizí nijak neotřáslo naší důvěrou v MD, otřáslo rakví, v níž MD5 už devět let leží."
>
> Hugo Krawczyk

6. Možnosti zneužití kolizí pro digitální podpisy

Na základě předchozího odstavce si lze představit i konstrukce kolidujících zpráv na následujícím obrázku.



Obr.: Možnosti sestavování kolidujících zpráv

Způsoby, jak využít tento útok konkrétně, ještě nebyly prezentovány, ale jistě se časem objeví. V tomto odstavci se budeme věnovat zneužití uvedené techniky nalézání kolizí přímo v digitálních podpisech.

6.1. Obecný postup

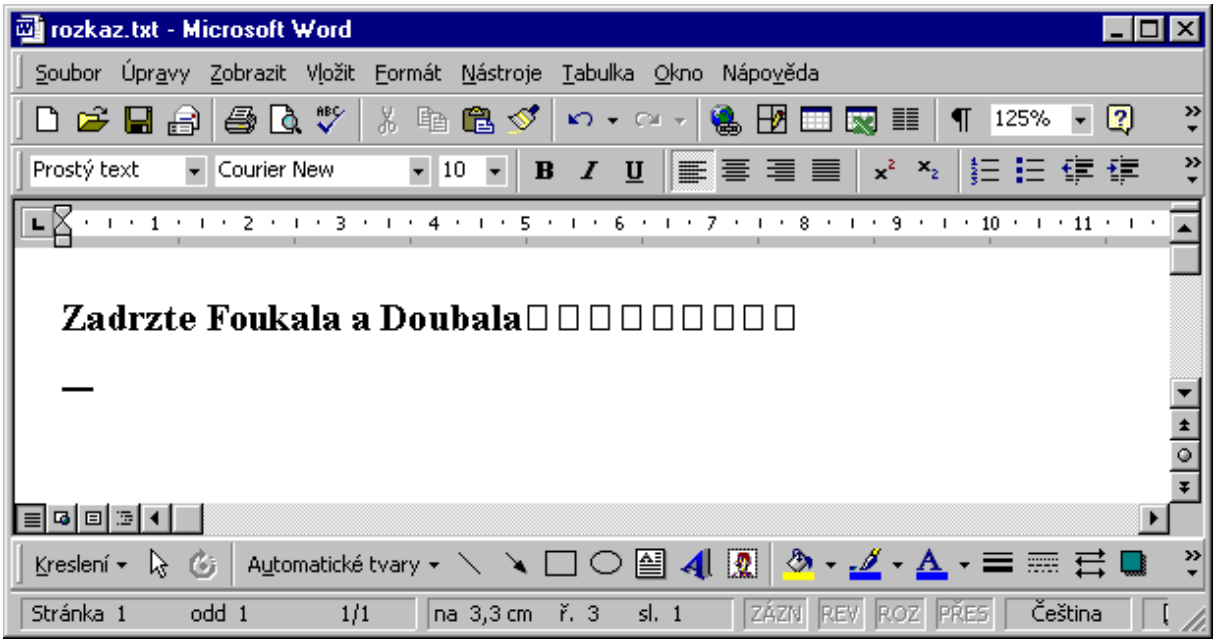
Lze si představit, že útočník může umístit rozdílné části zpráv (viz předchozí obrázek ....) na taková místa, která příslušný interpret daného dokumentu (WinZip, Word, Excel aj.) bude interpretovat ve prospěch útočníka. Útočník pak například vezme výchozí soubor **file.ext**

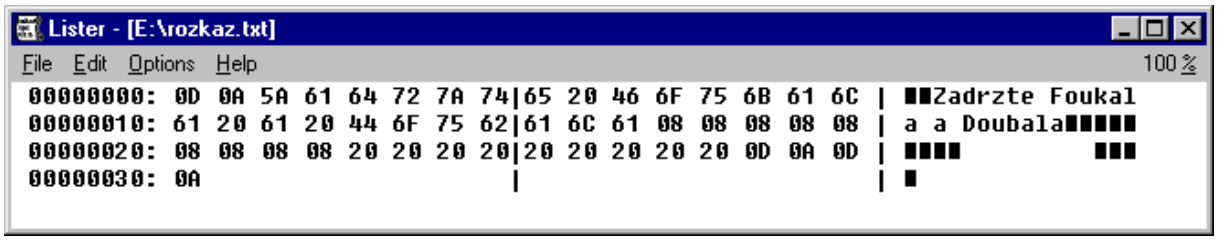
(například tabulku platů, seznam příkazů bance apod.) a pozmění ho na soubor, skládající se z dat
(**X**, M1, N1, **Y**, O1, P1, **Z**, Q1, R1, **T**), (Soubor 1)

který nechá podepsat. Přitom X, Y, Z a T jsou nějaké části původního souboru file.ext a zbylé bloky pochází z konstrukce kolizí. Musí zajistit, aby interpret souboru s koncovkou .ext uživateli zobrazil zamýšlený text (obsah), zatímco text souboru

(**X**, M2, N2, **Y**, O2, P2, **Z**, Q2, R2, **T**) (Soubor 2)

bude interpretován jinak, ve prospěch útočníka, a to v důsledku jiné interpretace odlišných bloků obou souborů. Záleží velmi na tom, jaký program daný soubor interpretuje (zobrazuje) uživateli.
Další útok může spočívat ve změně interpreta vůbec. To je například viditelné na obrázku u textových souborů [R04]. Soubor interpretujeme v prvním případě pomocí MS Word a v druhém případě pomocí příkazu type MS DOS. Dále lze kombinovat jak změnu obsahu, tak změnu interpreta tím, že soubory, které mají stejnou haš, můžeme označit jinou koncovkou. Lze konstruovat i složitější případy než u koncovky typu txt, například .zip, .doc, .xls apod. ¨



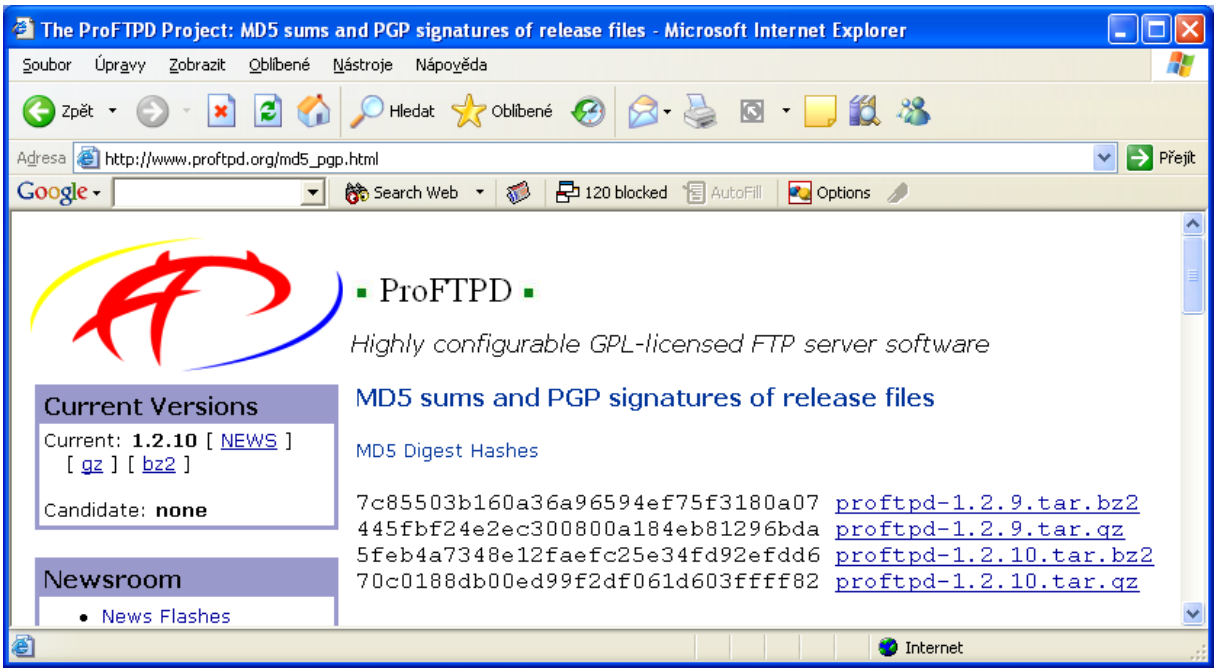


Obr.: Zobrazení stejného obsahu souboru různými interprety

7. Možnosti zneužití kolizí k podvodné výměně originálních programů (útok na integritu)

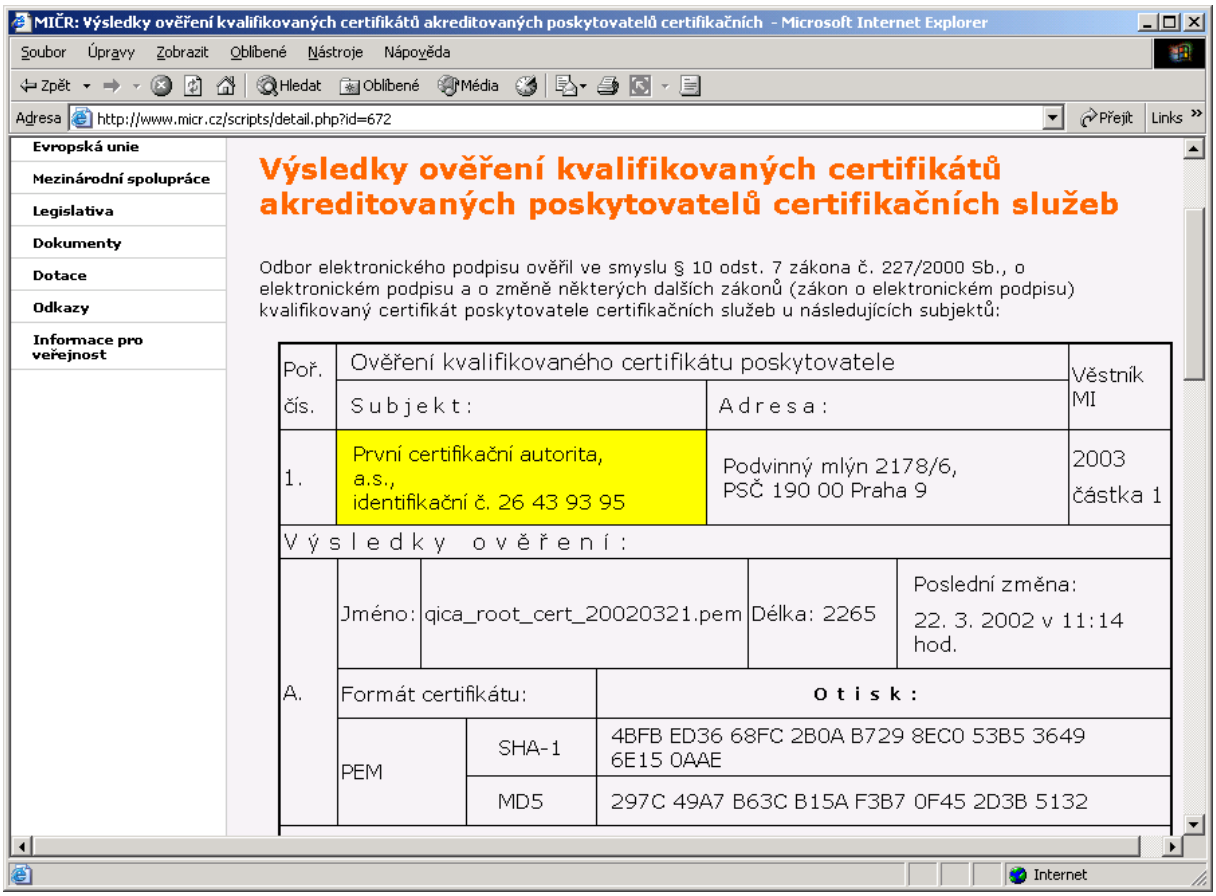
Příkladem situace, kdy dokumenty vytváří útočník, může být třeba nějaký program vystavený na internetu, jehož zdrojový tvar je ověřen, že neobsahuje zadní vrátka (například jádro Linux, programy PGP, obecné "open source" programy apod.).

Hašovací funkce je v tomto případě použita ke kontrole integrity, jak ukazuje ilustrativní obrázek.



Obr.: Kontrolní haše veřejně dostupných souborů na http://www.proftpd.org/md5_pgp.html

Útočník v tomto případě vytvoří dva takové soubory mající stejnou haš, přičemž docílí toho, aby část odlišných bloků byla součástí nějakého komentáře zdrojového programového kódu a část bloku vytvářela zadní vrátka (například pomocí instrukcí typu jmp). První ze souborů si nechá ověřit, že neobsahuje zadní vrátka, dále pod kontrolou vytvoří jeho haš a umístí soubor společně s haší na internet. Poté soubory vymění, haš zůstává v platnosti. Uživatelé si ale budou stahovat programy se zadními vrátky. Horší by byla taková výměna u certifikátů, tam je však manévrovací schopnost menší.



Obr.: Kontrolní haše u kvalifikovaných certifikátů certifikačních autorit, viz http://www.micr.cz/scripts/detail.php?id=672

8. Co čínský útok neumí při útoku na digitální podpisy aj.

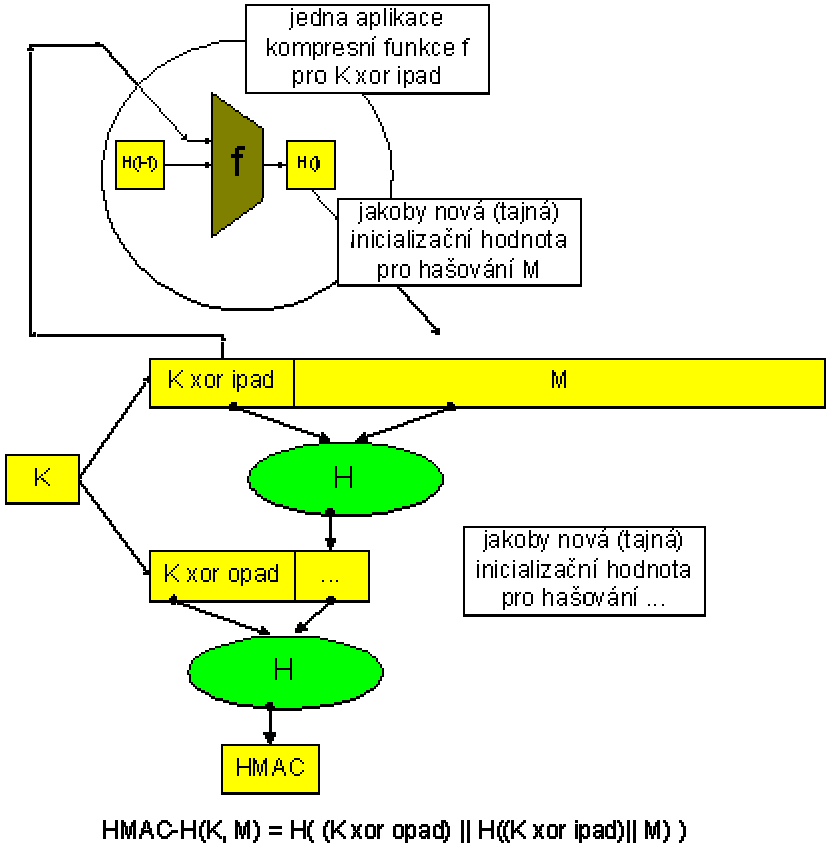
Čínský útok neumí **k danému dokumentu** nalézt jiný, se stejnou haší! Umí "pouze" najít dva různé dokumenty se stejnou haší. **Útočník musí obě kolidující zprávy vytvářet sám**. Při konstrukci útoků například na digitální podpisy na podkladě kolizí je tedy nutné, aby útočník oba kolidující dokumenty vytvářel sám. V karikatuře si můžeme představit například jak zaměstnanec dává nadřízenému podepsat žádost o dovolenou, se kterou mu „náhodou“ koliduje příkaz na zvýšení platu.

Je to přesně rozdíl mezi bezkolizností 1. a 2. řádu. Číňané nalezli kolize 1. řádu, ale ne 2. řádu.

9. HMAC - klíčovaný hašový autentizační kód zprávy

Avšak je tu trochu háček. **Čínský útok umí nalézt kolizi pro libovolnou inicializační hodnotu.** To Číňané prozradili nedopatřením. Když 16. 8. 2004 Číňané zveřejnili svůj útok, dopustili se chyby v interpretaci pořadí bajtů v originální hodnotě konstanty IV, definované v popisu MD5. Když na tuto skutečnost byli upozorněni, 17. 8. publikovali nové kolize pro novou, správnou, konstantu IV a navíc (když už to tím pádem bylo jasné) připsali větu, že *jejich útok funguje pro libovolnou hodnotu IV*.

Jejich útok tedy funguje pro libovolnou hodnotu, ale k útoku (alespoň to tak vypadá), ji potřebují znát, aby vygenerovali příslušné kolidující zprávy. Klíčovaný hašový autentizační kód zprávy HMAC ale ve skutečnosti tuto konstantu jakoby skrývá, viz obrázek.



Obr.: Klíčovaný hašový autentizační kód zprávy HMAC-H(K, M)

Klíčovaný hašový autentizační kód zprávy HMAC-H(K, M) je funkčně podobný autentizačnímu kódu zprávy MAC, ale místo blokové šifry využívá hašovací funkci (H). Označuje se konkrétně podle toho, jakou hašovací funkci používá, např. HMAC-SHA-1(K, M). M označuje zprávu a K klíč. Je definován ve standardu FIPS 198 (kde je popsán o něco obecněji než v RFC 2104 a ANSI X9.71) a jeho definice závisí na délce bloku kompresní funkce v bajtech (např. u MD5/SHA-1/SHA-256 je to B = 64 bajtů, u SHA-384/SHA-512 je to B = 128 bajtů) a na délce hašového kódu hašovací funkce H. HMAC používá dvě konstanty, a to *ipad* jako řetězec B bajtů s hodnotou 0x36 a *opad* jako řetězec B bajtů s hodnotou 0x5C. Klíč K se doplní nulovými bajty do plného bloku délky B a poté definujeme HMAC-H(K, M) = H((K xor *opad*) || H((K xor *ipad*)|| M)), kde || označuje zřetězení. Na obrázku je schéma HMAC.
Tajný klíč se modifikuje konstantou *ipad* a výsledek (K xor *ipad*) tvoří začátek vstupu do hašování. Je to definováno tak, že K xor *ipad* je přesně jeden blok kompresní funkce, takže po jeho zpracování dostáváme kontext H_1. Následuje zpracování zprávy M, čili její hašování jakoby začínalo z (útočníkovi neznámé) inicializační hodnoty IV = H_1, bez uvažování předsazeného řetězce (K xor *ipad*). Tento princip se použije ještě jednou, ale nikoli na zprávu jako takovou, nýbrž na obdrženou haš. Uvědomme si, že stačí nalézt jen kolizi pro H((K xor *ipad*)|| M), protože ta se automaticky projeví v celém HMAC.

Protože se předpokládá, že Číňané neznají metodu, jak konstruovat kolize pro tajné nastavení inicializační konstanty, **konstrukce HMAC je považována jejich útokem za nedotčenou**. Avšak odhaduje se, že i při tajné inicializační hodnotě by nalezení kolize mohlo být výpočetně méně náročné než by mělo teoreticky mělo být.

Proti použití prolomených hašovacích funkcí ve funkci HMAC v současné době není námitek, neboť není známo žádné oslabení funkce autentizačního kódu. Je však nutné sledovat, zda se útok neprohloubí i na tajné inicializační hodnoty, a dále vyhodnotit i jeho složitost, jakmile budou publikovány podrobnosti čínského útoku.

10. Další příklady použití hašovacích funkcí a odhad jejich oslabení čínským útokem

I když z teoretického hlediska nejsou hašovací funkce náhodné funkce, prakticky se tak jeví. Každá změna byť i jednoho bitu na vstupu má za následek nepredikovatelnou náhodnou změnu všech bitů na výstupu s pravděpodobností 1/2. A naopak jakákoliv změna na výstupu by měla vést k nepredikovatelné a náhodné změně na vstupu. Náhodnosti a nepredikovatelnosti se začalo využívat v různých technikách, a to i **ve spojení s tajným klíčem**. Jedná se o pseudonáhodnou funkci **PRF**, která se používá v pseudonáhodných generátorech (**PRNG**). Uvedeme si některé příklady a odhadneme, jaký vliv na jejich bezpečnost má nalezení kolizí.

10.1. Hašovací funkce při tvorbě klíčů z passwordů, PKCS#5

Standard PKCS#5 umožňuje využít hašovací funkci k tvorbě "náhodného" šifrovacího klíče z passwordu. Předpis je vidět z obrázku a vynecháme-li hodnotu soli, spočívá v hašování passwordu a následném mnohonásobném hašování výsledku. Počet hašování je dán konstantou c, jejíž hodnota se doporučuje minimálně 1000, ale používá se i 2000.
Výsledkem je krátký "náhodně vyhlížející" klíč DK, který je možné využít lépe než původní password. Jednak má pevnou délku a jednak z něho lze využít tolik bitů, kolik potřebujeme. Hodnota DK má pochopitelně lepší statistické vlastnosti než původní password. Tento postup se využívá ke tvorbě krátkých klíčů.

$$T_1 = Hash\ (P \parallel S)$$
$$T_2 = Hash\ (T_1)$$
$$\dots$$
$$T_c = Hash\ (T_{c\text{-}1})$$

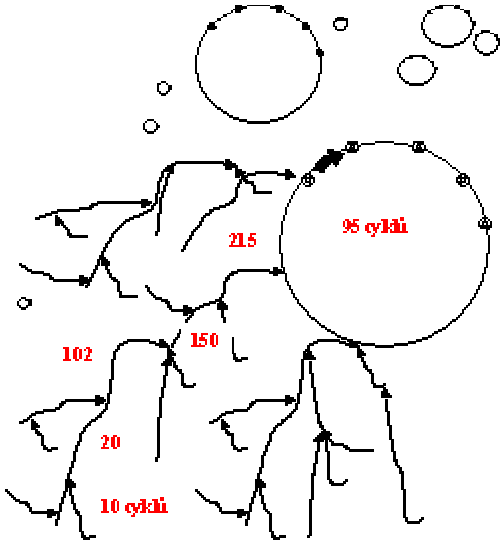
šifrovací klíč $DK = T_c\langle 0..dkLen\text{-}1\rangle$

Obr.: Tvorba klíče DK z passwordu P podle PKCS#5 (S je sůl)

Zde nalézání kolizí čínskou metodou nevadí, i když je to "vada na kráse", protože bychom rádi, aby použitá hašovací funkce byla co nejkvalitnější ze všech hledisek. Proto připojujeme dovětek ve formě: **...ale uvedené prolomené funkce by se zde měly používat obezřetně.**

Zde by nejvíce vadilo, kdyby byly porušeny vlastnosti hašovací funkce jako náhodné funkce. Například kdyby graf jejího chování při mnohonásobném hašování měl velmi málo krátkých

cyklů, jak ukazuje obrázek. Takovým utajeným chováním hašovací funkce lze do systémů vpašovat zadní vrátka. Stačí nastavit konstantu c "hodně bezpečnou", například oněch několik tisíc, a věděli bychom, že výsledkem může být jen malá množina (třeba tisíc) výsledných haší - bodů v cyklech, kde hašování nakonec vždy skončí, nezávisle na hodnotě passwordu (P) a soli (S).

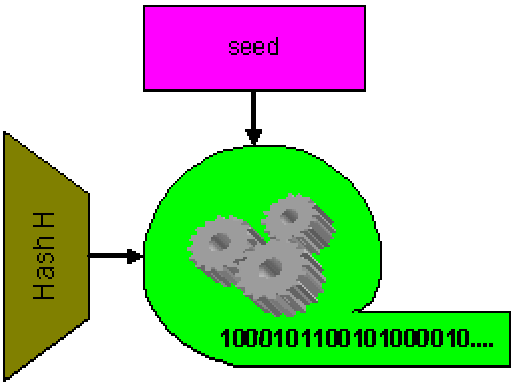


Obr.: Možnost krátkých cyklů v iterovaném hašování nekvalitní hašovací funkce

Pokud krátké cykly nehrozí, je pravděpodobně i konstrukce z PKCS#5 bezpečná.

10.2. Pseudonáhodné funkce (PRF) a pseudonáhodné generátory (PRNG)

Typické použití hašovacích funkcí jako pseudonáhodných funkcí je v případech, kdy máme k dispozici krátký řetězec dat (seed) s dostatečnou entropií. Může se jednat například o "krátký" 256bitový náhodný šifrovací klíč, záznam náhodného pohybu myši na displeji, časový profil náhodných stisků kláves apod. Přitom potřebujeme z tohoto vzorku získat pseudonáhodnou posloupnost o velké délce, například 1 GByte apod. A k promítnutí entropie původního vzorku (seedu) do delší posloupnosti se právě používají hašovací funkce.



Hlavní rozdíl oproti minulému použití je v tom, že vstupem je náhodný zdroj, a další rozdíl je, že výstupem je relativně dlouhá posloupnost.

Obr.: Hašovací funkce v konstrukci PRNG

Tím, že je vstupem náhodný (útočníkovi neznámý) řetězec, dostáváme se s využitím kolizí do podobné situace jako u HMAC, protože útočník nezná inicializační hodnotu. Navíc jsou zde další složitosti. Uvedeme si příklady.

10.3. PRNG ve spojení s hašovací funkcí H, PKCS#1 v.2.1

Například standard PKCS#1 v.2.1 definuje pseudonáhodný generátor MGF1 (Mask Generation Function) pomocí hašovací funkce H s počátečním - **většinou náhodným - nastavením** seed takto:
H(seed || 0x00000000), H(seed || 0x00000001), H(seed || 0x00000002), H(seed || 0x00000003), ...
Protože seed je náhodný, konstrukce bude pravděpodobně bezpečná a čínským útokem nedotčená.

10.4. PRNG ve spojení s hašovací funkcí podle PKCS#5

Tam, kde seed není zcela náhodný, se používá komplikovanější postup, viz například funkce PBKDF2(P, S, c, dklen) z PKCS#5. Ta na základě passwordu P a soli S generuje pseudonáhodnou posloupnost (c je konstanta - hodnota čítače, např. 1000 nebo 2000):
T_1, T_2, T_3, ..., kde T_i, je vždy součet (xor) sloupce v následující tabulce. Počet řádků v tabulce odpovídá počtu (c) iterací.

$U_1 = H(P, S, 0x00000001)$	$U_1 = H(P, S, 0x00000002)$		$U_1 = H(P, S, i\ (4Byte))$
$U_2 = H(P, U_1)$	$U_2 = H(P, U_1)$		$U_2 = H(P, U_1)$
$U_3 = H(P, U_2)$	$U_3 = H(P, U_2)$		$U_3 = H(P, U_2)$
$U_4 = H(P, U_3)$	$U_4 = H(P, U_3)$		$U_4 = H(P, U_3)$
....	....	....	....
$U_c = H(P, U_{c-1})$	$U_c = H(P, U_{c-1})$		$U_c = H(P, U_{c-1})$
$\mathbf{T_1}$ = součet (xor) sloupce	$\mathbf{T_2}$ = součet (xor) sloupce		$\mathbf{T_i}$ = součet (xor) sloupce

Tab.: Pseudonáhodný generátor PBKDF2(P, S, c, dklen) podle PKCS#5

Protože tato konstrukce je velmi robustní, není tu vůbec zřejmé, jak by se mohla schopnost nalézání kolizí čínským útokem projevit na bezpečnostních vlastnostech těchto konstrukcí. Pochopitelně, pokud by hašovací funkce byla velmi primitivní (například kód CRC), i tyto konstrukce by byly ohroženy. Současné hašovací funkce ale velmi primitivní rozhodně nejsou, ani u nich nejsou známy takové uvedené slabiny

Obecně můžeme opět říci, že není znám žádný útok na **PRNG ve spojení s prolomenými hašovacími funkcemi**. Je samozřejmě lepší je nahradit za kvalitní, pokud to lze.

Poznamenejme, že dodatek o obezřetnosti platí pro všechny hašovací funkce (viz úvod - není prokázána jejich "bezpečnost", jak bychom si představovali), takže je jenom větším zdůrazněním obecné vlastnosti v tomto konkrétním případě, kde by nějaká nežádoucí slabina mohla eventuelně vzniknout.

10.5. PRNG ve spojení s HMAC

Poznamenejme, že namísto H v předchozích nebo jiných PRNG je možné použít HMAC. Taková konstrukce je ještě robustnější než PRNG ve spojení s H, proto **ani u PRNG ve spojení s HMAC zde nejsou známy žádné negativní důsledky čínského útoku**.

11. Kde kolize nevadí (některá autentizační schémata)

Existují i různé příklady využívající dokonce digitální podpisy, kde slabší odolnost proti kolizi tolik nevadí. Jedná se většinou o autentizační schémata. Můžeme si představit primitivní model autentizace, kdy ten, kdo autentizuje (řekněme nějaká webová aplikace), vyšle nějakou náhodnou výzvu RND uživateli. Ten ji zhašuje funkcí *h* a aplikaci vrátí digitálně podepsanou hašovací hodnotu h(RND), čímž se autentizuje, neboť aplikace si jeho podpis může ověřit. V tomto případě nalézání kolizí útočníkovi příliš nepomůže, protože výzvu RND nevytváří on, ale aplikace. I kdyby útočník dokonce uměl nalézt kolizi 2. řádu, tj. k dané (už existující) výzvě RND nalézt její kolidující dvojče RND´ tak, že h(RND) = h(RND´), nemůže se za uživatele přihlásit.

Poznámka.
Kdyby ovšem útočník uměl **narušit jednosměrnost**, mohl by s nějakým úsilím z h(RND) získat RND. Potom záleží na aplikaci, jak RND využívá. Jestliže RND využívá například pro vytváření šifrovacích klíčů, může to útočníkovi napomoci k dešifrování následné komunikace. Čili v tomto případě je jednosměrnost důležitější než bezkoliznost.

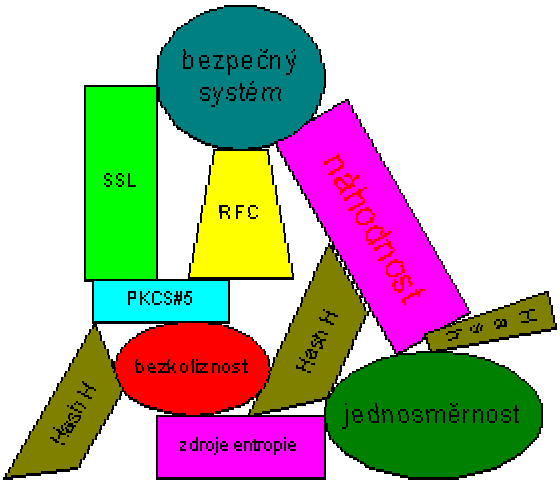
12. Jaká vlastnost hašovacích funkcí je nejdůležitější ?

Hašovací funkce jsou používány k mnoha účelům a moderní protokoly a standardy je používají v několika různých technikách. Každá technika má své vlastní předpoklady o síle a vlastnostech hašovací funkce, jednou je to požadavek na bezkoliznost, podruhé na (pseudo)náhodnost, jindy na jednosměrnost. Nemůžeme říci, která z vlastností je nepodstatnější. Nejčastěji je pravděpodobně spoléháno na odolnost proti kolizi pro zajištění služby nepopiratelnosti (většinou pomocí digitálních podpisů). Také jsme viděli, že prolomení jedné z vlastností (jednosměrnost, bezkoliznost, pseudonáhodnost) se může i nemusí dotýkat jiné vlastnosti. Popis vzájemných vazeb je složitý, takže dopady jednotlivých útoků se uvádějí pro každou vlastnost zvlášť.

12.1. Jak moc byla funkce prolomena?

Praxe je ovšem mnohem komplikovanější, protože v jednom standardu, protokolu nebo zařízení se hašovací funkce obvykle použije v mnoha variantách a využívají se všechny nebo několik jejích vlastností.
Při nalezení *kolize* nějaké hašovací funkce se také v praxi můžeme setkat s otázkou, jak „moc“ byla funkce prolomena. Jádro odpovědi je v tomto případě v matematické logice, která stojí za argumenty o bezpečnosti toho kterého systému. Tato logika je dvouhodnotová: výrok buď platí, nebo neplatí. Mravenčí matematickou prací se na jednodušších tvrzeních stavějí složitější, až nakonec na vrcholku pyramidy stojí výrok: Tento systém je pravděpodobně bezpečný. Někde v hloubi této konstrukce přitom stojí výrok: Hašovací funkce je bezkolizní.

Pokud někdo ukáže, že kolizi našel, pyramida se hroutí společně s výrokem na vrcholu a po formální stránce je z celého schématu pouhá ruina. **Je přitom jedno, jestli kolidující zprávy nám jako lidem připadají smysluplné nebo ne.** Na druhou stranu zhroucení pyramidy s důkazem bezpečí v jednom konkrétním systému ještě neznamená, že pro jiný systém nemůže vyrůst jiná pyramida, kde nalezení kolizí vyústí v mírné zvýšení reálného rizika



nebezpečnosti. Zde většinou bývá jistý časový odstup. Přísně logicky vzato tak lze nějakou chvíli používat i prolomené funkce. Praktické zkušenosti ovšem ukazují, že je to jen poslední večírek na Titaniku.

Obr.: Různé logické předpoklady vytváří základ bezpečnostní pyramidy

12.2. Zvláštní posouzení v každé aplikaci?

Jistěže by v každé aplikaci mohlo dojít ke zvláštnímu posouzení, zda nalezení té či oné slabiny hašovací funkce má vliv nebo ne, ale tolik kryptologů na světě není. Proto kryptologové připravují pro programátory, vývojáře a bezpečnostní pracovníky funkce, které jsou pak univerzálněji použitelné. Padne-li nějaká bezpečnostní vlastnost takové funkce, není jiné obecné rady, než ji okamžitě nahradit. Je-li si někdo jistý, že v jeho aplikaci nalezená slabina nevadí, nechť ji na své riziko používá dále.

13. Nalezené slabiny ohrožují zatím "pouze" budoucí digitální podpisy

Největší dopad má čínský útok na zprávy, které teprve "budou podepsány" pomocí prolomených hašovacích funkcí. Například sekretářka předloží řediteli k podpisu nevinně vyhlížející objednávku kancelářských potřeb, která má stejný hašový kód (třeba MD5) jako nějaká nevýhodná smlouva, datovaná do budoucnosti.

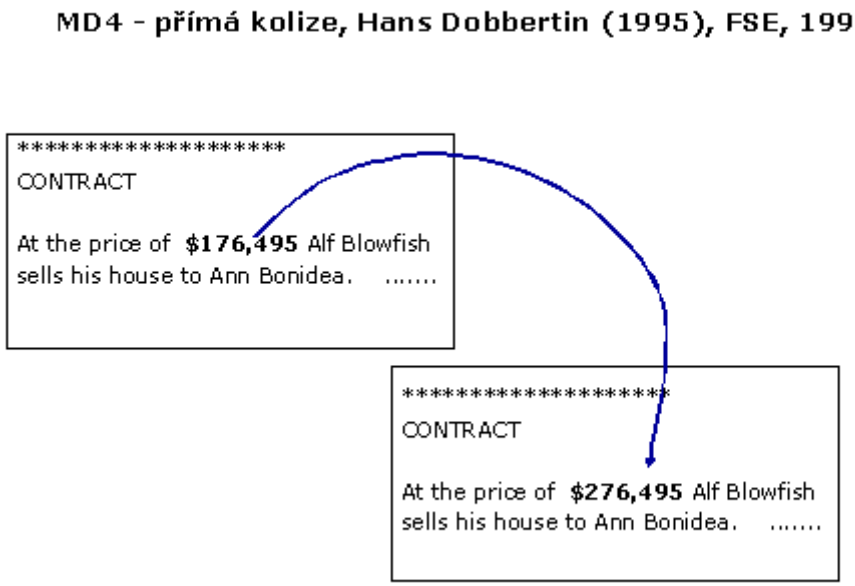
14. Jsou ohroženy i podpisy vytvořené v minulosti?

Jestliže se Číňanům podaří útok zesílit tak, že budou umět nalézt kolize druhého řádu, tedy k (jakékoli) *dané* zprávě (M1) nalézt druhou (M2) vedoucí na stejnou haš (h(M1) = h(M2)), ohrožovalo by to i podpisy, pořízené v minulosti. Využití by mohlo být takové, že k dané smlouvě, která se později ukázala nevýhodnou, lze vytvořit kolidující smlouvu se stejnou haší, ale jiným obsahem. Útočník by pak tvrdil, že podepsal smlouvu číslo 2, nikoli 1, s tím, že smlouvu číslo 1 mohl dodatečně vytvořit jeho protějšek. Zatím se však nepředpokládá, že by se toto podařilo, neboť se jedná o úlohu mnohem složitější. Nicméně i tento aspekt se u napadených funkcí musí sledovat.

15. Konference CRYPTO 2004 a její výsledky

Číňané na konferenci předvedli, že umí najít velkou třídu kolizí u hašovacích funkcí MD4, MD5, RIPEMD a HAVAL-128 s časovou náročností od sekund do 1-2 hodin [WFLY04]. Joux [Joux04] zde prezentoval dvě kolidující zprávy pro SHA-0, které získal se složitostí 2^{51}. Číňané poznamenali, že je umí nalézt se složitostí pouze 2^{40} výpočtů. Joux v další diskusi poznamenal, že při neznámé inicializační konstantě předpokládá nalezení kolize u SHA-0 se složitostí 2^{60} místo očekávaných 2^{80}. Tato úvaha vede na hledání kolizí v HMAC-SHA-0 s touto složitostí. Nic takového zatím není známo o SHA-1 a HMAC-SHA-1. Pokud by se však

jejich metoda dala použít i tímto směrem, byla by trochu oslabena (snížena) i bezpečnost HMAC-SHA-1. Biham a Chen zlepšili na této konferenci své dřívější teoretické úvahy o útocích na SHA-0 [BC04a] a redukovanou verzi SHA-1 [BC04b]. Joux poukázal na to, že současná konstrukce hašovacích funkcí na bázi iterativního procesu s využitím kompresní funkce nemá vlastnosti náhodné funkce [Joux04]. Tuto konstrukci používá drtivá většina hašovacích funkcí. Poukázal na to například tím, že pokud lze konstruovat kolize, je konstrukce multikolizí (mnoha zpráv vedoucích na tutéž kolizi) méně náročnější, než by u náhodné funkce mělo být. Jedná se o teoretickou vlastnost, jejíž praktické využití silně závisí na předpokladu, že je možné nacházet kolize.



Poznamenejme ještě, že (jedna) kolize MD4 byla nalezena (a to ještě poměrně pracně) Dobbertinem v roce 1995 [D96a]. Tehdy to vzbudilo opravdu velký rozruch a od MD4 se rychle upustilo. Dnes to Číňanům u MD4 i MD5 trvá pár sekund až hodin a naleznou kolizí celou řadu. K výměně MD5 ovšem už není taková vůle, protože je implementovaná v desítkách standardů, protokolů a zařízení.

Dále poznamenejme, že Číňané nejen nalezli kolize, ale vlastně techniky konstrukce kolizí u uvedených hašovacích funkcí. Protože jsou všechny prolomené hašovací funkce iterativní, je výsledkem trochu mrazení v zádech, zda i ostatní moderní iterativní hašovací funkce ustojí tyto neznámé nové techniky a na nich založené útoky. Zejména taková otázka vyvstává u SHA-1, která je nejrozšířenější, ale i u třídy funkcí SHA-2, které se začínají nasazovat a jako perspektivní se prosazují.

16. Které techniky jsou a nejsou dotčeny a zůstávají bezpečné

Ukázali jsme si, že prolomené hašovací funkce by se neměly používat tam, kde se jedná o nepopiratelnost, tedy u digitálních podpisů.
Klíčované hašové autentizační kódy zpráv HMAC ani pseudonáhodné funkce PRF a pseudonáhodné generátory PRNG zatím nejsou čínským výsledkem dotčeny. Je tu ale možné

riziko pramenící z toho, že neznáme téměř nic o technikách prolomení a že po jejich zveřejnění může být všechno jinak. To je hrozba, kterou si každý musí ohodnotit.

Zatím se domníváme, že funkce HMAC používající MD5, tj. **HMAC-MD5, nemusí být vyměňovány** a že HMAC v kombinaci s funkcemi SHA-1 nebo třídou SHA-2 zůstávají zcela bezpečné.

Pokud se týká ostatních prolomených hašovacích funkcí, tak SHA-0 byla krátkou dobu oficiálním standardem, ale v roce 1994 byla rychle nahrazena SHA-1, proto by se v systémech neměla vyskytovat. RIPEMD a HAVAL (včetně varianty HAVAL-128) se příliš neujaly (RIPEMD byl nahrazen bezpečnějším RIPEMD-160), takže jejich nahrazení by nemělo vůbec nastat. Pokud jsou přesto někde používány, měly by být vyměněny.

Mnoho technických otázek bylo vyjasněno v diskusi k článku na serveru root "Hašovací funkce MD5 a další prolomeny!", http://www.root.cz/clanek/2368. Jeho text a další linky a informace je také možné nalézt na stránce věnované tématu hašovacích funkcí http://cryptography.hyperlink.cz/2004/kolize_hash.htm, která je průběžně aktualizována.

17. Nový vzor chování při používání kryptografických technik

Poučení tedy je, že je nutné se na problémy tohoto typu připravit jako na reálné jevy tak, jako se reálně u složitých programů vydávají záplaty. Nové paradigma by mělo být nedůvěřovat slepě jen jedné funkci, ale systémy budovat tak, aby se kryptografické nástroje v nich mohly pružně měnit. Bezpečnost není konstantní, je to komplikovaná veličina, která se postupem času vyvíjí. Proto ji ošetřujeme procesem řízení rizika, který bezpečnost monitoruje a včas provádí příslušné korekce. Tuhle poučku sice každý zná a v některých oblastech se už i rutinně uplatňuje, v oblasti používání kryptografických nástrojů ale jako by všichni ztuhli.

18. NIST plánuje přechod na SHA-2 do r. 2010

Americký standardizační úřad NIST, který za standardy hašovacích funkcí odpovídá, vydal prohlášení k současným výsledkům na *http://csrc.ncsl.nist.gov/hash_standards_comments.pdf*, z něhož vyjímáme:

- SHA-1 zůstává bezpečná.
- Doporučuje se používat třídu funkcí SHA-2.
- Do roku 2010 se předpokládá opuštění i SHA-1 a přechod na SHA-2.

19. Distribuovaný útok MD5CRACK zastaven

Možná nevíte, že o nalezení kolizí se hrubou silou pokoušel i projekt MD5CRACK na http://www.md5crk.com/, kde Češi patřili k významným přispěvatelům strojového času. Cílem bylo najít kolizi MD5 hrubou silou a přesvědčit tak bezpečnostní architekty, aby od ní konečně ustoupili. Jakmile byl publikován čínský výsledek, projekt byl pochopitelně zastaven. Číňani ukázali, že geniální nápad skály proráží.

20. Závěrečný souhrn pro manažery

- Je nutné provést revizi všech aplikací, kde jsou použity hašovací funkce MD4, MD5, SHA-0, RIPEMD a HAVAL-128.
- Je-li některá z těchto funkcí použita pro účely digitálních podpisů (s klasickým účelem zajištění nepopiratelnosti), je nutno tuto funkci nahradit. Podle okolností provést náhradu za některou z funkcí, které jsou považovány za bezpečné: SHA-1, SHA-256, SHA-384 nebo SHA-512, nejlépe SHA-512 [SHA-1,2].
- Je-li některá z těchto funkcí použita pro účely HMAC nebo PRNG, nechat pro jistotu posoudit, zda je toto užití bezpečné nebo ne.
- Nové paradigma by mělo být nedůvěřovat slepě jen jedné funkci, ale systémy budovat tak, aby se kryptografické nástroje v nich mohly pružně měnit. Pokud je to možné, přejít na toto pravidlo postupně i u stávajících systémů.
- Zajistit průběžné sledování vývoje aplikované kryptologie a zavést mechanismus pravidelného hodnocení používaných kryptografických technik ve světle tohoto vývoje.

21. Literatura

[ARCHIV] Archiv autora obsahující články o kryptologii a bezpečnosti, http://cryptography.hyperlink.cz

[BC04a] Biham, Eli, Chen, Rafi: Near Collisions of SHA-0, CRYPTO 2004
http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/2004/CS/CS-2004-09.ps.gz

[BC04b] [4] Eli Biham, Rafi Chen: New results on SHA-0 and SHA-1, CRYPTO 2004 Rump Session

[D96a] H. Dobbertin, Cryptanalysis of MD4, Fast Software Encryption, LNCS, Vol. 1039, Springer-Verlag, 1996

[HAVAL] Y. Zheng, J. Pieprzyk, J. Seberry, HAVAL - A One-way Hashing Algorithm with Variable Length of Output,Auscrypt 92

[HMAC] FIPS PUB 198, The Keyed-Hash Message Authentication Code (HMAC), NIST, US Department of Commerce, Washington D. C., March 6, 2002, http://csrc.nist.gov/CryptoToolkit/tkhash.html, resp. RFC 2104, http://www.rfc-editor.org/

[Joux04] Antoine Joux: Collisions in SHA-0, CRYPTO 2004 Rump Session

[MD245] MD2, MD4, MD5 - RFC 1319, 1320, 1321, http://www.rfc-editor.org/

[PKCS#5] PKCS #5 v2.0: Password-Based Cryptography Standard, RSA Laboratories, March 25, 1999

[PKCS#1 v.2.1] PKCS #1 v2.1: RSA Cryptography Standard, RSA Laboratories, June 14, 2002

[R04] Rosa T.: Nepopiratelnost digitálních podpisů, Druhá vědecká a pedagogická konference ZMVS: Právní regulace informační společnosti, Třebíč, září 2004, http://crypto.hyperlink.cz/files/rosa_ZMVS04.pdf

[RIPEMD-160] H. Dobbertin, A. Bosselaers, B. Preneel, "RIPEMD-160: A Strengthened Version of RIPEMD," Fast Software Encryption, LNCS 1039, D.Gollmann, Ed., Springer-Verlag, 1996, pp. 71-82

[SHA-0] FIPS 180 (superseded by FIPS 180-1 and FIPS 180-2), Secure hash standard (SHS), NIST, US Department of Commerce, Washington D. C., May 1993

[SHA-1] FIPS 180-1 (superseded by FIPS 180-2), Secure hash standard (SHS), NIST, US Department of Commerce, Washington D. C., April 1995

[SHA-2] FIPS 180-2, Secure Hash Standard (SHS), NIST, US Department of Commerce, Washington D. C., August 2002 (change notice: February 2004), http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf, platný standard, obsahuje definice SHA-1, SHA-224, SHA-256, SHA-384 a SHA-512

[ST1204] Klíma V., Rosa T.: Kryptologie pro praxi - Hašovací funkce MD5 & spol. definitivně prolomeny, Sdělovací technika 12/2004

[ST0204] Klíma V., Rosa T.: Kryptologie pro praxi - funkce HMAC, Sdělovací technika 02/2004

[WFLY04] X. Wang, D. Feng, X. Lai, H. Yu, "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD", rump session, CRYPTO 2004, *Cryptology ePrint Archive*, Report 2004/199, http://eprint.iacr.org/2004/199