Kryptologie pro praxi – opět kolize MD5

Nejznámější použití hašovacích funkcí je pro digitální otisky dat. Soubor dat můžeme považovat za digitálně identifikovaný jeho haší, pokud nejsme technicky schopni nalézt jiný soubor se stejným digitálním otiskem (haší), neboli kolizi. Pokud u hašovací funkce neumíme hledat kolize, je možné místo zpráv a souborů digitálně podepisovat jejich hašovací kódy, neboť víme, že žádný jiný soubor k dané haši nalézt nedokážeme. Pokud ale u hašovací funkce kolize nalezeny byly nebo je lze dokonce hledat systematicky, měli bychom ji z používání pro digitální otisky vyloučit nebo alespoň zvážit rizika z toho vyplývající.

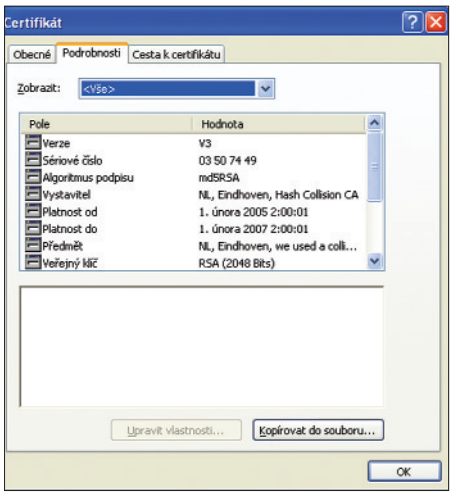
Před rokem byly objeveny kolize

Právě před rokem byly ohlášeny kolize u několika hašovacích funkcí, z nichž nejznámější byla MD5, jak jsme informovali v *ST 12/2004*. Tým profesorky Wangové z Číny je schopen kolizi MD5 nalézt řádově za hodinu s použitím multiprocesorového počítače IBM P690. Dodejme, že v prvotním ohlášení [1] v srpnu 2004 čínský tým nepopsal metodu, ale pouze poskytl kolidující 1024bitové řetězce. Analýzou dat byl ale odhalen jejich základní trik, tzv. diferenční cesta, a poté, v březnu 2005 byla nezávisle objevena metoda mnohonásobné modifikace zpráv [7]. Ta vedla ke kolizím 3 až 6 krát rychleji (kolize jsou generovány na notebooku) než v té době utajovaná čínská metoda [1]. Poznamenejme, že zatím jen dvě místa na světě oznámila, že umí generovat kolize MD5, i když kompletní popis české metody byl již v [7] uveřejněn.

Protože MD5 nelze jen tak ze stovek systémů, protokolů, certifikačních autorit a standardů odstranit, je poněkud zlehčována možnost využití kolizí. Ukážeme si argumenty protivníků i zastánců MD5. K tomu budeme potřebovat stručně popsat, jak se kolize hledá a nachází, abychom pochopili rizika a možnosti jejich tvorby.

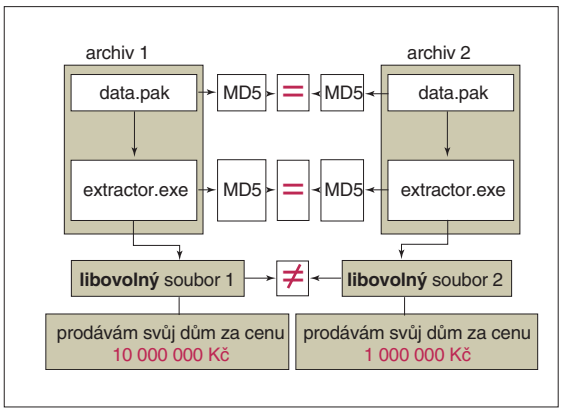
Kde je slabé místo

MD5 vzniká iterativním procesem, který využívá tzv. kompresní funkci f a inicializační konstantu IV. Hašovaná zpráva je doplněna na celistvé bloky o 512 bitech $m_1,\ldots,m_n$, a ty jsou iterativně zpracovány takto: $h_0 = IV$, $h_1 = f(h_0, m_1)$, $h_2=f(h_1, m_2),\ldots, h_n=f(h_{n-1}, m_n)$. Hodnota h_n je pak výslednou haší. Metody nalézání kolizí [1], [7] pak pro libovolný inicializační vektor IV umí nalézt dva různé bloky m_1 a m_1' a dva různé bloky m_2 a m_2' tak, že h_2 a h_2' jsou stejné, tedy po zhašování 1024bitových zpráv (m_1, m_2) a (m_1', m_2') nastává kolize. Samozřejmě, že obě zprávy mohou pokračovat libovolným (stejným) obsahem $m_3 = m_3$', $m_4 = m_4$', atd. Protože kolize umíme nalézt pro libovolnou inicializační hodnotu, můžeme



Obr. 1 Jeden certifikát pro dva různé klíče

také začít od libovolného stejného počátku zprávy $m_1,\ldots, m_k$ a poté pro $IV = h_k$ generovat dva různé kolizní fragmenty (m_{k+1}, m_{k+2}) a (m_{k+1}',m_{k+2}') tak, aby h_{k+2} bylo stejné. Obecněji řečeno zprávy, které mají kolidovat, mohou začínat libovolnými stejně dlouhými prefixy vedoucími k libovolné stejné hodnotě $IV = h_k$. Útočník pouze za tento „stejný" začátek vloží jím vykonstruované (různé) 1024bitové bloky a obě zprávy pak mohou zase pokračovat libovolným stejným obsahem. Zdůrazněme důsledek této konstrukce, a sice že výsledné kolidující zprávy se mohou lišit hned v několika kolizních fragmentech, nikoliv pouze v jednom, jak bývá občas mylně tvrzeno.



Obr. 2 Na bázi jediné kolize MD5 lze vytvořit "kolizi" souborů libovolného obsahu

Kolidující certifikáty

Uvedeného postupu lze využít k přípravě dvou různých klíčů RSA, jejichž certifikáty, coby datové soubory v příslušném formátu, budou mít stejnou (případně zaměnitelnou) hodnotu podpisu certifikační autority (CA). Důsledkem je, že útočník může od nic netušící certifikační autority získat certifikát, ve kterém může kdykoliv zaměnit jednu hodnotu veřejného klíče za druhou, přičemž certifikát zůstane platným. Přímočaré využití nabízí možnost získání dvou certifikátů za cenu jednoho, avšak našli bychom i komplikovanější postupy založené na možnosti porušit jednoznačnou vazbu mezi identitou uživatele a hodnotou veřejného klíče, kterou řada architektur IM (Identity Management) implicitně předpokládá. Zde útočník použije jednoduchý trik: Protože CA druhý klíč dosud „neviděla", dovolí ho bez problémů přiřadit k nějakému dalšímu certifikátu. Tím se tento klíč může naráz pojit k dvěma různým identitám uživatele, a to sice k identitě z kolizního a z řádného certifikátu. Realizovatelnost v praxi není ještě tak jednoduchá, protože certifikační autorita do certifikátu často přidává položky, které útočník není schopen dopředu odhadnout a připravit podle nich kolidující hodnotu veřejného klíče. Může se jednat například o náhodné sériové číslo certifikátu nebo o kontrolní hašový otisk klíče (s jinou hodnotou IV). Příklad, který byl publikován, vychází z předpokladu, že žádný z těchto prvků není aplikován, což na druhou stranu není tak nereálný požadavek. Zejména pokud se bude mluvit o certifikačních autoritách kdesi v pralesech podnikových intranetů a IM. Ukážeme, jak probíhá příprava dvou různých kolidujících klíčů. Útočník si připraví datový obsah certifikátu tak, že vyplní všechna pole, která se udávají v žádosti o certifikát, tj. datum platnosti, svoje jméno, organizaci a další údaje. Jediné, co neví, je číslo certifikátu, avšak předpokládá, že certifikační autorita používá snadno předpověditelnou monotónní posloupnost. Takže toto číslo odhadne a přidá k ostatním datům. Připravený balíček dat eventuelně doplní na celistvý násobek 512bitových bloků například mezerami za jménem apod. Útočník pak tento datový obsah zhašuje a obdrží průběžnou hodnotu h_k. Za datovým obsahem následuje už číslo – modul RSA, který volí 2048bitový. Prvních 1024 bitů volí jako dva různé kolidující fragmenty

zprávy (B_1, B_1'), které mu dá algoritmus generování kolizí pro inicializační hodnotu h_k. Zbylých 1024 bitů (B_2) dopočítá tak, aby bloky (B_1, B_2) a (B_1', B_2) dávaly dva různé moduly RSA pro smysluplné klíče (výpočet trvá několik minut až dnů, podle zvolených parametrů klíče). Pokud za náš datový balíček vloží modul RSA $N = (B_1, B_2)$ nebo $N' = (B_1', B_2)$, má kolidujícím algoritmem zajištěno, že po zhašování obou různých modulů bude mít stejnou hodnotu haše. Nyní může doplnit zbývající data certifikátu, a to stejně u obou balíčků. Nyní dá útočník jeden z klíčů podepsat certifikační autoritě. Pokud certifikační autorita přidělí tomuto certifikátu stejné sériové číslo, jako útočník předvídal, vytvoří si před podpisem u sebe naprosto shodný balíček dat k podpisu jako útočník a při jeho hašování obdrží stejnou hašovací hodnotu jako on. Data certifikátu obsahující jako nejdůležitější položku připravený RSA modul (B_1, B_2) pak podepíše a podpis přiloží, čímž vznikne certifikát. Nyní je zřejmé, že v tomto certifikátu může útočník vyměnit blok (B_1, B_2) za (B_1', B_2) a má platný certifikát i pro tento falešný klíč. Vše lze ověřit na skutečných datech a certifikátech, které jsou k dispozici na webové stránce a v článku [5].

Kolidující dokumenty libovolného typu

Školním příkladem, který by přesvědčil o nepoužitelnosti MD5 i zaryté zastánce MD5, by bylo k danému dokumentu umět vytvořit jiný se stejnou haší. Toto je ale úloha jiná a složitější než nalezení kolize. U kolize hledáme jakékoliv (i nesmyslné) dva soubory nebo bloky dat, které kolidují, zatímco zde bychom chtěli k danému souboru, který už má haš danou, najít jiný (tzv. druhý vzor), který má stejnou haš. A určitě by bylo hezké, kdyby ten druhý soubor byl také smysluplný text. Tak to zatím neumíme, ale možná, že se k tomu za několik let propracujeme. Umíme však takovou situaci navodit jinak. Princip je v tom, že současné formáty typu .doc, .ps, .pdf aj. jsou poměrně složité, proto k nim vždy máme příslušné překladače, programy (word, acrobat apod.), které tato složitá data zpracují a prezentují na displeji. Stačí tedy znát příslušná pravidla toho kterého překladače, abychom ho pouze na základě rozdílných 1024bitových bloků v datovém souboru přinutili zobrazit zcela jiné dokumenty. To bylo ukázáno na příkladu samorozbalovacího archivu (tak se dělá například distribuce SW balíků), který byl popsán v [2]. V tomto případě dva uživatelé obdrží soubory *extractor.exe* a *data.pak*. Mohou si zkontrolovat, že jejich haše jsou stejné (přesto se soubory *data.pak* liší). *Data.pak* totiž obsahuje jen kolidující řetězec a za ním data souboru číslo 1 a data souboru číslo 2. Tyto soubory mohou být libovolné soubory jiného obsahu, typu i jména. Navíc se zde využívá pouze jedné jediné kolize MD5, umístěné na počátku kolidujících dat *data.pak*. Překladačem je zde program *extractor.exe* (u obou stejný), který jen na základě jediného rozdílného bitu na počátku souboru *data.pak* z něho vyextrahuje buď soubor číslo 1 nebo soubor číslo 2. Není tedy nutné ani mít schopnost kolize generovat. Tento postup lze zjednodušit tak, aby uživatelé dostali jen jeden soubor *extractor.exe*, ale to není podstatné. Existují i jiné příklady využití, například [3] a [6].

Nutné podmínky pro útok pomocí MD5

Ve všech zatím využitelných příkladech je nutné, aby útočník měl možnost data, která budou později kolidovat, sám vytvářet (v předchozím příkladu měl možnost vytvářet kolidující *data.pak*). Není ovšem nic neobvyklého, když někdo jiný dokumenty, programy apod. tvoří a někdo jiný je kontroluje a podepisuje, zejména u složitějších programů a dokumentů. Útočník tedy může v části díla, které tvoří, vyčlenit místo, kde bude později umístěn 1024bitový kolidující řetězec. Když pak oprávněný kontrolor prověří platnost kódu nebo dokumentu, vytvoří jeho haš a navíc ji třeba digitálně podepíše, útočník může vyměnit onen 1024bitový řetězec za kolidující, přičemž program nebo dokument rázem začne dělat jiné věci, například vytvoří zadní vrátka, zobrazí jinou smlouvu apod.

Vývoj věcí příštích

V současné době se pracuje na projektu kolidujících certifikátů používajících SHA-1. Složitost algoritmu nalezení kolize je zatím pro tuto funkci velmi vysoká – asi 2^{69}, ale jedná se o typicky distribuovaný výpočet, který již byl zahájen, a příslušný výsledek bude nakonec dosažen. Certifikační autority by se na tuto situaci měly připravit, alespoň řádným proškolením tiskových mluvčích. Nemá cenu strkat hlavu do písku a říkat si, že to nějak dopadne. Může ale nastat i situace, že dříve než tento útok hrubou silou skončí, dojde k pokroku teoretickému a kolize budou nalezeny rychleji jinou metodou, stejně jako tomu bylo u MD5. V poslední době se totiž na hašovací funkce soustřeďuje více pracovišť, takže takový vývoj nelze vyloučit.

Vlastimil Klíma, Tomáš Rosa,
v.klima@volny.cz, trosa@ebanka.cz

LITERATURA

[1] Wang X., xFeng X., Lai X., Yu H., "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD", rump session, CRYPTO 2004, srpen 2004, http://eprint.iacr.org/2004/199

[2] Mikle O.: Practical Attacks on Digital Signatures Using MD5 Message Digest, Cryptology ePrint Archive, Report 2004/356, 2nd December 2004, http://cryptography.hyperlink.cz/2004/collisions.htm

[3] Kaminsky D.: MD5 To Be Considered Harmful Someday, Cryptology ePrint Archive, Report 2004/357, http://eprint.iacr.org/2004/357, 6 December 2004

[4] Lenstra A., Wang X., Weger B.: Colliding X.509 Certificates, Cryptology ePrint Archive, Report 2005/067, http://eprint.iacr.org/2005/067

[5] Lenstra A., Wang X., Weger B.: Colliding X.509 Certificates based on SHA1-collisions, http://www.win.tue.nl/~bdeweger/CollidingCertificates/index.html

[6] Down M., Lucks S.: Attacking Hash Functions by Poisoned Messages, "The Story of Alice and her Boss", http://www.cits.rub.de/MD5Collisions/

[7] Klima V.: Finding MD5 Collisions on a Notebook PC Using Multi-message Modifications, March 31, 2005, IACR ePrint archive, http://eprint.iacr.org/2005/102

[8] E-archivy http://cryptography.hyperlink.cz, http://crypto.hyperlink.cz