W poszukiwaniu funkcji skrótu

Michał ADAMASZEK
Stanisław RADZISZOWSKI[*]

Zacznijmy od wyjaśnienia tytułu. *Funkcja skrótu* (nazwijmy ją H) służy do przyporządkowywania dowolnie długim wiadomościom krótkich liczb („odcisków palca"). Wiadomością może być plik, obrazek, film w postaci cyfrowej lub inna, zupełnie dowolna porcja danych (x), którą będziemy traktować po prostu jako ciąg bitów. Kodowany ciąg x może być bardzo długi, za to wynik $H(x)$ powinien być krótki, na przykład kilkusetbitowy.

Do czego przydają się takie funkcje? Mogą służyć do ochrony przed modyfikacjami danych, a na ich popularność wpływa rosnąca dostępność Internetu, a wraz z nim możliwość wymieniania się przeróżnymi plikami. Wiąże się to z różnymi zagrożeniami: skąd mamy wiedzieć, że pobieramy kopię identyczną z oryginalną, a nie taką, która została celowo zmodyfikowana (np. poprzez wprowadzenie wirusa)? Dobra funkcja skrótu powinna służyć między innymi do wykrywania takich machlojek: jeżeli autor pliku x opublikuje $H(x)$, to ktoś, kto chce stwierdzić, czy ma wierną kopię x, może sam szybko obliczyć $H(x)$ i porównać ją z opublikowaną wartością.

Zamiast funkcja skrótu mówi się nieraz funkcja haszująca, choć to drugie pojęcie jest bardziej ogólne, bo funkcja haszująca musi po prostu dobrze rozpraszać dane i nie pełni funkcji kryptograficznej.

Można pomyśleć, że dobrą sygnaturą pliku jest, na przykład, jego rozmiar. Niestety, istnieją rozmaite sposoby podrabiania plików w taki sposób, aby ich rozmiar nie zmieniał się. Podobną wadę mają inne zbyt proste techniki. Dobra funkcja skrótu H powinna być

(1) szybko obliczalna on-line, w niewielkiej pamięci,

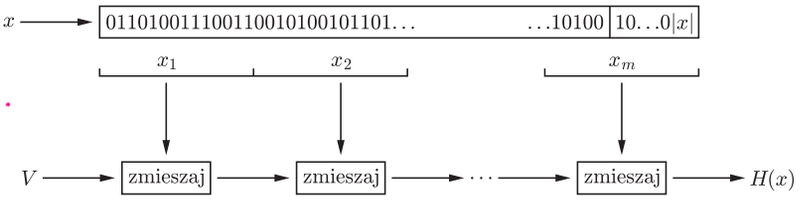
(2) bardzo trudno odwracalna: dla danego z znalezienie x takiego, że $H(x) = z$, powinno być bardzo kosztowne obliczeniowo,

(3) odporna na kolizje: znalezienie dwóch wiadomości x i y, dla których $H(x) = H(y)$, także powinno być bardzo trudne.

Zatem bardzo łatwo obliczyć wartość skrótu, ale niesłychanie trudno wykonać operację odwrotną. Warunek (3) oznacza, że funkcja H jest „praktycznie różnowartościowa" – w zasadzie nie mamy szans trafić na dwa ciągi bitów o tym samym skrócie. Jak widać, funkcja H powinna dobrze „mieszać" wszystkie bity wejściowego ciągu. Funkcje haszujące mają wiele zastosowań: do obliczania sum kontrolnych, uwierzytelniania, przechowywania haseł czy w technologii podpisów cyfrowych.

Większość używanych funkcji skrótu stosuje tzw. schemat Merkle–Damgarda. Polega on na tym, że ciąg wejściowy x jest dzielony na bloki ustalonej długości b: $x = x_1 x_2 \dots x_m$, $|x_i| = b$. Ostatni blok dopełniamy jedynką, zerami i binarnym zapisem długości tekstu x na ustalonej liczbie bitów, tak aby również osiągnąć długość b. Ponadto mamy określony pewien początkowy ciąg bitów V, po czym wykonujemy pętlę (rys. 1):

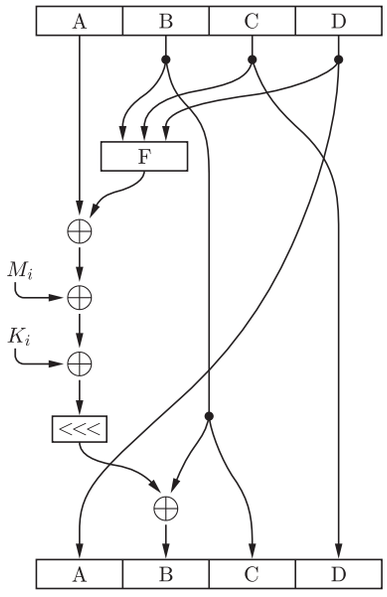
$H_0 := V;$
for $k := 1$ **to** m **do**
  $H_k = zmieszaj(H_{k-1}, x_k);$
return $H_m;$



Rys. 1

Cała sztuka polega na doborze odpowiedniej funkcji kompresującej *zmieszaj*, której zadaniem jest zmieszanie zawartości k-tego bloku ze skrótem poprzednich bloków. Podkreślmy, że nie ma na to żadnego uniwersalnego sposobu – wszystko opiera się na kreatywności twórców pomysłu i intuicji dotyczącej tego, jaki sposób mieszania będzie trudno odwracalny. W praktyce popularne są dwa

[*]Rochester Institute of Technology



Rys. 2.

standardy: MD5 (MD od „Message Digest”, autorstwa Rivesta, rok 1992) oraz SHA-1 (Secure Hash Algorithm, 1995). W obu algorytmach wejście jest przetwarzane blokami po $b = 512$ bitów. MD5 generuje klucze długości $|H| = 128$, a skrót typu SHA-1 jest dłuższy, $|H| = 160$. W obu przypadkach sama funkcja *zmieszaj* także ma strukturę iteracyjną, przy czym w MD5 składa się ona z 64, w SHA-1 zaś z 80 rund.

Z ciekawości przyjrzyjmy się jednej z 64 rund funkcji $zmieszaj_{MD5}$ (rys. 2). Przetwarzana wartość 128-bitowa jest dzielona na 4 bloki A, B, C, D po 32 bity, których zawartość jest poddawana operacjom zaznaczonym na rysunku. F oznacza jedną z czterech funkcji trójargumentowych, takich jak na przykład $F(a, b, c) = (a \wedge b) \vee (\neg a \wedge c)$ i podobnych. W każdej z 64 rund używana jest jedna z tych funkcji. W węzłach oznaczonych $\oplus$ obliczany jest XOR argumentów. K_i jest pewną stałą zależną od numeru rundy $i = 1, \ldots, 64$. Możemy nawet podać ją dokładnie:

$$K_i = \lfloor |\sin i| \cdot 2^{32} \rfloor.$$

Z kolei M_i jest pewnym 32-bitowym fragmentem mieszanego akurat 512-bitowego bloku x_k – jego wybór jest uzależniony od numeru rundy i. Operator <<< przesuwa liczbę cyklicznie o s_i pozycji, gdzie znów s_i ($i = 1, \ldots, 64$) jest pewnym określonym z góry ciągiem (jego początek to $7, 12, 17, 22, \ldots$). Na koniec otrzymujemy nową liczbę 128-bitową, która jest poddawana przekształceniom w kolejnej rundzie i tak 64 razy. We wszystkim tym wyraźnie daje się odczuć pewną aurę tajemniczości. Nie sposób, na przykład, nie zastanowić się, dlaczego niby dobrano takie, a nie inne „magiczne” wartości przeróżnych parametrów (i co by się stało, gdyby je nieco zmienić), czemu porcje danych są przetwarzane akurat w takiej kolejności itp. Z drugiej strony intuicyjnie wydaje się, że taka „maszynka” miele wejściowy ciąg bitów bardzo dokładnie.

Algorytm SHA-1 działa na podobnej zasadzie, z tym że jest nieco bardziej skomplikowany. Dlaczego? Dość szybko okazało się, że MD5 nie jest całkiem bezpieczny. Co to znaczy i jak można atakować taki algorytm? Próba znalezienia przeciwobrazu jakiegoś klucza z (czyli tekstu x, dla którego $H(x) = z$), tzw. *preimage attack*, jest bardzo trudna i jak na razie pod tym względem MD5 trzyma się świetnie. Dużo łatwiej znaleźć kolizję, czyli jakiekolwiek dwa ciągi x i y, dla których $H(x) = H(y)$. Jedna ze sztuczek opiera się na znanym paradoksie urodzin: wystarczą 23 osoby, aby prawdopodobieństwo tego, że któreś dwie z nich mają urodziny tego samego dnia, przekroczyło 50%. Ogólniej, jeżeli mamy próbki przyjmujące M wartości, to wystarczy przetestować około

$$q \approx \sqrt{2M \ln 2} \approx 1{,}17\sqrt{M}$$

z nich, aby z prawdopodobieństwem co najmniej 1/2 trafić na dwie próbki o tej samej wartości (dla $M = 365$ mamy $q = 23$). Wynika stąd, że dla 128-bitowej funkcji skrótu, po przejrzeniu około $\sqrt{2^{128}} = 2^{64}$ tekstów, z dużym prawdopodobieństwem trafimy na kolizję. Przeprowadzenie 2^{64} testów to spory, ale dziś już wyobrażalny nakład obliczeń. Dla SHA-1 trzeba by wykonać około $\sqrt{2^{160}} = 2^{80}$ prób, co, jak się obecnie przewiduje, nie będzie wykonalne jeszcze przez kilka lat.

Uruchamiając polecenie `md5sum` lub `sha-1sum`, dostępne w większości instalacji systemów operacyjnych, można poeksperymentować samemu. Wynikiem tych poleceń jest skrót (MD5 lub SHA-1), zapisany szesnastkowo (32 lub 40 cyfr). Obliczając skróty plików różniących się choćby jednym bajtem (albo nawet bitem), otrzymamy diametralnie różne wyniki.

Ta metoda jest dość uniwersalna, ale funkcja MD5 „padła” zupełnie inaczej, przy użyciu technik dobranych specjalnie pod jej kątem. Wang, Yu i Yin w latach 1995–2006 wypracowali heurystyki, dzięki którym znaleźli przykłady kolizji dla MD5. Była to bardzo żmudna praca, polegająca na dokładnym śledzeniu przepływu bitów w kolejnych rundach mieszania, a dziś pomysły te zostały rozwinięte i w sieci można znaleźć programy, które generują w kilka sekund ciągi o kolidujących kluczach. Co gorsza (i to jest właśnie naprawdę niebezpieczne!), mając dane dwa ciągi bitów, na przykład dokumenty PDF lub programy wykonywalne, potrafimy nieznacznie, w sposób niezauważalny dla człowieka, zmodyfikować każdy z nich tak, aby sumy MD5 nowych plików były równe. Daje to możliwość podrabiania dokumentów i podmieniania programów (na przykład na złośliwe) w sposób niewykrywalny przy użyciu sumy MD5. Jest to tym bardziej niebezpieczne, że tzw. podpisy cyfrowe są często stosowane nie

w odniesieniu do całych dokumentów, a jedynie do ich skrótów, bo tak jest szybciej (ta technika nosi nazwę „hash then sign"). Przez to podpis może być wykorzystany przez fałszerza dokumentu wbrew intencjom podpisującego. Więcej na ten temat można poczytać na stronie [1], gdzie znaleźć można także przykład takiego fałszerstwa: dwa pliki PS o bardzo różnej treści i tym samym skrócie MD5.

Algorytm SHA-1 pozostaje póki co bezpieczny, choć powstały techniki ataku wymagające znacznie mniej niż 2^{80} testów. Próby takie są podejmowane, ale kolizje znaleziono na razie jedynie dla uproszczonych wersji tego algorytmu. Istnieje także nowszy standard, SHA-2, którego różne wersje generują klucze 224, 256, 384 i 512-bitowe. Stopień skomplikowania tego algorytmu jest jeszcze większy niż w przypadku MD5 i SHA-1; występują w nim także różne dziwne stałe, na przykład części ułamkowe liczb postaci $\sqrt[3]{p}$, gdzie p jest liczbą pierwszą. W związku z zagrożeniami atakami National Institute of Standards and Technology (NIST, amerykańska jednostka rządowa zajmująca się między innymi standaryzacją) zaleca całkowitą rezygnację z MD5 na rzecz SHA-1, a po roku 2010 rezygnację z SHA-1 na rzecz wariantów SHA-2. Nie będzie to takie proste, bo SHA-1 jest dziś częścią powszechnie używanego standardu podpisów cyfrowych Digital Signature Standard.

Niedawno (w listopadzie 2007) NIST ogłosił konkurs na nową funkcję skrótu, która zastąpi MD5 i warianty SHA. Będzie ona nosiła nazwę SHA-3. Kandydatury można zgłaszać do 31 października 2008 roku, po czym, po czterech latach przeznaczonych na kolejne rundy publicznej dyskusji, nowy standard ma zostać wybrany i ogłoszony w 2012 roku. Projekty muszą zawierać dokładną specyfikację wraz z implementacją oraz uzasadnienie takiej a nie innej konstrukcji algorytmu, doboru parametrów i stałych oraz, w miarę możliwości, analizę odporności na różne rodzaje ataków. Nowa funkcja musi być zdolna do generowania skrótów o długościach 224, 256, 384 i 512 bitów. Podstawowymi kryteriami decydującymi o wyborze zwycięzcy będą: bezpieczeństwo, koszt czasowy i pamięciowy obliczania funkcji oraz dodatkowe cechy algorytmu, takie jak np. łatwość wykonania równoległego.

[1] http://www.cits.rub.de/MD5Collisions/
[2] http://www.nist.gov/hash-competition
[3] http://en.wikipedia.org/wiki/{MD5,SHA}

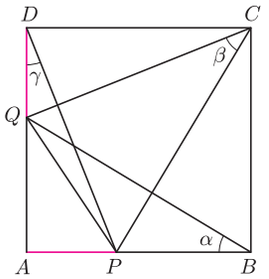
Zadania

Redaguje Ewa CZUCHRY

F 715. Dwie gwiazdy okrążają wspólny środek masy ze stałymi co do wartości prędkościami liniowymi v_1 i v_2 oraz okresem T. Znaleźć masy gwiazd oraz odległość między nimi.
Rozwiązanie na str. 8

F 716. Dwie gwiazdy A i B o masie M każda, znajdujące się w stałej odległości r, pod działaniem wzajemnego przyciągania grawitacyjnego poruszają się po orbitach kołowych. W pewnej nieznanej odległości od gwiazd, w tej samej płaszczyźnie, znajduje się lekka planeta C. Znaleźć tę odległość, wiedząc, że $AC = BC = x$, oraz że rozmiary trójkąta ABC nie zmieniają się w czasie ruchu układu.
Rozwiązanie na str. 9

Redaguje Waldemar POMPE



M 1204. Każdy punkt okręgu pomalowano na jeden z dwóch kolorów. Wykazać, że istnieje trójkąt równoramienny wpisany w ten okrąg, którego wszystkie wierzchołki mają ten sam kolor.
Rozwiązanie na str. 10

M 1205. Punkty P i Q leżą odpowiednio na bokach AB i AD kwadratu $ABCD$, przy czym $AP = DQ$ (rysunek). Obliczyć

$$\sphericalangle PBQ + \sphericalangle PCQ + \sphericalangle PDQ\,.$$

Rozwiązanie na str. 11

M 1206. Rozstrzygnąć, czy istnieje takich 100 liczb naturalnych $a_1, a_2, \ldots, a_{100}$, że dla dowolnych $i \neq j$ liczba a_i^2 jest podzielna przez a_j, a liczba a_i nie jest podzielna przez a_j.
Rozwiązanie na str. 15