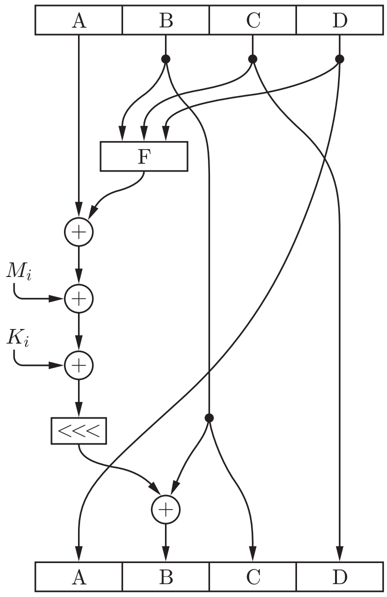

Rys.2.

Uruchamiając polecenie `md5sum` lub `sha1sum`, dostępne w większości instalacji systemów operacyjnych, można poeksperymentować samemu. Wynikiem tych poleceń jest skrót (MD5 lub SHA1), zapisany szesnastkowo (32 lub 40 cyfr). Obliczając skróty plików różniących się choćby jednym bajtem (albo nawet bitem), otrzymamy diametralnie różne wyniki.

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 kilkanaście lat.

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