5. Funkcje skrótu

5. 1. Podstawowe definicje

Funkcje skrótu (ang. hash function, One Way Hash Function czyli OWHF) nazywa się również jednokierunkowymi funkcjami skrótu lub jednokierunkowymi fukcjami haszującymi lub po prostu funkcjami haszującymi. Funkcje skrótu są bardzo ważnym narzędziem współczesnej kryptografii. Kryptograficzne funkcje haszujące należy odróżniać od funkcji haszujących znanych z przedmiotu „algorytmy komputerowe i struktury danych".

Funkcja jednokierunkowa (ang. one way function) to z definicji taka funkcja $f : X \to Y$, że obliczenie dla danego $x \in X$ wartości $f(x) = y$ jest łatwe ale jeśli $f(x) = y$ i znamy tylko wartość y, to praktycznie powinno być niemożliwe (ze względu na złożoność obliczeniową) obliczenie argumentu x takiego, że $f(x) = y$. Nie żądamy przy tym różnowartościowości funkcji $f : X \to Y$.

Funkcja skrótu to odwzorowanie $h : V_1^* \to V_1^n$, gdzie V_1 jest ustalonym alfabetem, określone na przestrzeni wiadomości V_1^* będące funkcją jednokierunkową. V_1 jest alfabetem w którym zapisujemy wiadomości ale z reguły w praktyce $V_1 = \{0,1\}$ tak, że funkcja skrótu jest odwzorowaniem $h : \{0,1\}^* \to \{0,1\}^n$. Wartość funkcji skrótu nazywa się skrótem wiadomości, skrótem dokumentu czy po prostu skrótem (angielskie nazwy wartości funkcji skrótu to „hash functiom value" „message digest", „digital fingerprint", „imprint").

Staramy się zdefiniować funkcję skrótu $h : V_1^* \to V_1^n$ dla wszystkich wiadomości $m \in V_1^*$ ale definiuje się ją często tylko dla wiadomości $m \in V_1^m$, gdzie $m > n$ ustalając sposób rozszerzania funkcji $h : V_1^m \to V_1^n$ na cały zbiór V_1^*. Funkcja $h : V_1^m \to V_1^n$ jest tzw. blokową funkcja skrótu ale nazywamy ją też po prostu funkcją skrótu.

Funkcja skrótu $h : V_1^* \to V_1^n$ ma więc 3 zasadnicze cechy:

1. Realizuje kompresję danych zamieniając słowo dowolnej długości na słowo o stałej długości n.

2. Łatwo można obliczyć wartość $h(m)$ dla każdego $m \in V_1^*$.

3. Nie możemy praktycznie obliczyć m znając tylko wartość $h(m)$ funkcji skrótu. (ang. preimage resistance)

Sformułowanie „nie możemy praktycznie obliczyć m" oznacza, że złożoność obliczeniowa znanych algorytmów służących do tego celu jest zbyt duża by były realizowalne.

Od funkcji skrótu żąda się by była **silnie bezkonfliktowa** tzn. by praktycznie nie można było znaleźć takich różnych wiadomości m, m', żeby $h(m) = h(m')$.

Silna bezkonfliktowość Silna bezkonfliktowość funkcji hashującej polega na tym, że nie potrafimy dobrać (ze względu na zbyt dużą złożoność obliczeniową algorytmów służących do tego celu) takich argumentów x, x', $x' \neq x$, że $h(x) = h(x')$.

Przykład 1. Funkcja skrótu SHA-1 (Secure Hash Algorithm 1), funkcja skrótu SHA-2.

Przykład 2. Funkcje skrótu MD2, MD4, MD5. Skrót MD pochodzi od ang. Message Digest. Funkcja skrótu MD 5 ma z nich najlepsze własności i jest najczęściej wykorzystywana w praktyce.

Przykład 3. Funkcja skrótu HMAC

Przykład 4. Omawiając funkcje skrótu warto wspomnieć o jednokierunkowej funkcji skrótu o bardzo dobrych własnościach o nazwie HAVAL. Jest to modyfikacja funkcji skrótu MD4. Pracuje na blokach 1024 bitowych. Wytwarzany skrót może być równy 128, 160, 192, 224 i 256 bitów (od 128 bitów co 32 bity do 256).

Plik, tekst, wiadomość m o dowolnej długości „obłożone" funkcją skrótu h dają ciąg bitów $h(m)$ o ustalonej długości n (np.: 16 bajtów czyli 128 bitów lub 20 bajtów, czyli 160 bitów).

Jeśli znamy wartość funkcji skrótu i samą funkcję skrótu i dysponujemy parą $(m', f(m))$, to możemy sprawdzić integralność danych, czyli sprawdzić czy $m = m'$ obliczając dla m' wartość $f(m')$. Jeśli $f(m) = f(m')$, to przyjmujemy, że $m = m'$ (choć tak być nie musi), czyli stwierdzamy, że „integralność danych jest zachowana". Prawdopodobieństwo pomyłki (przy odpowiednio długim n) jest bardzo małe.

Uwaga 1. Ze względów praktycznych (wiadomość m może być przecież bardzo długa) podpis cyfrowy wiadomości m jest tworzony z reguły dla funkcji skrótu tej wiadomości $h(m)$, rzadko pełnej wiadomości m.

Uwaga 2. Intuicja funkcji skrótu jest prosta. Dla danej wiadomości jawnej m, wartość funkcji skrótu $f(m)$ to zwarta reprezentacja wiadomości, „streszczenie" (message digest) tej wiadomości. lub bardziej obrazowo „cyfrowy odcisk palca" (digital fingerprint) wiadomości m. Wartość $h(m)$ stanowi przy tym słowo o stałej liczbie bitów. Typowa długość $h(m)$ to 20 bajtów, tzn. 160 bitów. Praktycznie $h(m)$ służy do jednoznacznej identyfikacji wiadomości m.

Po co stosujemy funkcje skrótu.

1. zapewnienie integralności danych (data integrity)

2. uwierzytelnienie wiadomości (message authentication)

3. uwierzytelnienie strony czyli identyfikacja (entity authentication)

Mamy 2 zasadnicze kategorie funkcji skrótu:

1. funkcje skrótu bez klucza lub po prostu funkcje skrótu oznaczane skrótem MDC (ang. Message Digest Code)

2. kluczowane funkcje skrótu oznaczane skrótem MAC (ang. Message Authentication Code)

Funkcje skrótu MAC tak konstruujemy by równość dla danego *m* można było uzyskać tylko dla tej samej wartości klucza.

5.2. Funkcja hashująca Chaum –van Heijst –Pfitzmanna

Funkcja hashująca Chaum–van Heijst –Pfitzmanna. jest przykładem funkcji skrótu dla której możemy wykazać, że jest silnie bezkonfliktowa o ile pewne założenia są spełnione. Takie funkcje hashujące nazywamy funkcjami o dowodliwych właściwościach.

Niech p będzie dużą liczbą pierwszą taką, że $p = 2q+1$, gdzie q jest również liczbą pierwszą (są takie p i q na przykład 2 i 5, 3 i 7, 5 i 11, 11 i 23).

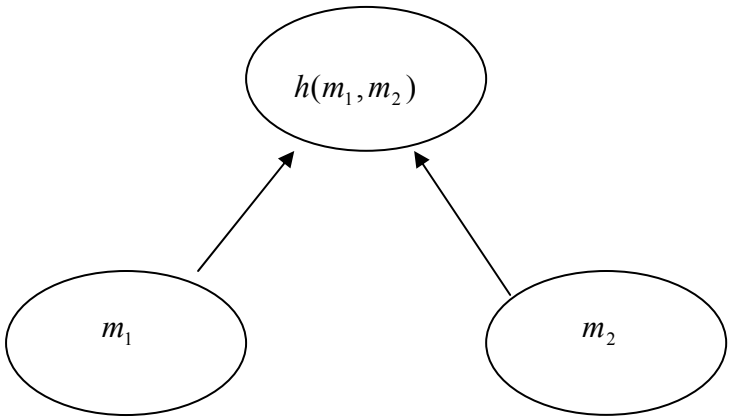
α, β - generatory grupy multiplikatywnej Z_p^*, ale są one takie, że nie znamy $\log_\alpha \beta$.

Definiujemy funkcję $h : Z_q \times Z_q \to Z_p$ wzorem

$h(m_1, m_2) = \alpha^{m_1} \otimes_p \beta^{m_2} \in Z_p^*$ (podnoszenie do potęgi w pierścieniu Z_p)

Argumenty "m_1, m_2" wchodzą więc do wykładnika.

Uwaga. Wartość funkcji $h(m_1, m_2)$ ma w przybliżeniu połowę długości argumentu funkcji h. Mamy więc efekt skracania czyli kompresji.



Rys. 5.1 Efekt skracania dla funkcji hashującej $h : Z_q \times Z_q \to Z_p$. Argumenty m_1, m_2 zapisujemy na na $2\lceil \log_2(q-1) \rceil$ bitach a wynik na $\lceil \log_2(2q+1) \rceil \le \lceil \log_2(q-1) \rceil + 2$ bitach .

Okazuje się, (wykażmy to w dalszym ciągu), że tak zdefiniowana funkcja $h : Z_q \times Z_q \to Z_p$ jest silnie bezkonfliktowa, o ile obliczenie $\log_\alpha \beta$ w Z_p jest **praktycznie niewykonalne**.

Zauważmy, że $\beta^q = -1$ (potęgowanie w Z_p) . Istotnie z własności generatora grupy multiplikatywnej w Z_p^* mamy $\beta^{p-1} = 1$ a więc $\beta^{2q} = 1$ ale równanie $x^2 = 1$ w Z_p ma tylko 2 rozwiązania $x_1 = 1$ i $x_2 = -1$ zatem $\beta^q = 1$ lub $\beta^q = -1$. Pierwszy przypadek zajść nie może bo β jest generatorem stąd musi być $\beta^q = -1$.

Twierdzenie Jeśli potrafimy znaleźć argumenty (m_1, m_2) i (m_1', m_2') (różne) i takie, że

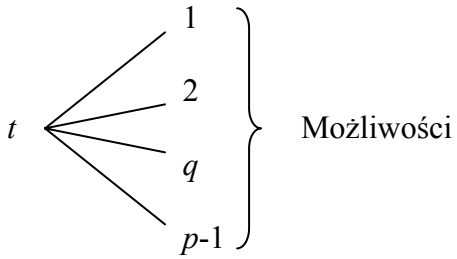
$$h(m_1, m_2) = h(m_1', m_2'),$$

(co oznacza że funkcja h nie jest silnie bezkonfliktowa) to potrafimy już łatwo obliczyć logarytm dyskretny $\log_\alpha \beta$ w Z_p, czyli rozwiązać problem logarytmów dyskretnych w Z_p.

Dowód. Załóżmy, że dla różnych (m_1, m_2) i (m_1', m_2') mamy $h(m_1, m_2) = h(m_1', m_2')$ czyli $\alpha^{m_1} \otimes_p \beta^{m_2} = \alpha^{m_1'} \otimes_p \beta^{m_2'}$. Zatem:

$$\alpha^{m_1 - m_1'} = \beta^{m_2 - m_2'} \text{ w } Z_p$$

Niech $t \overset{ozn.}{=} NWD(m_2 - m_2', p-1)$. Z założenia jednak $p - 1 = 2q$ zatem możemy mieć tylko następujące możliwości



Rozważmy przypadek 1: $t = 1$**.** Za pomocą algorytmu Euklidesa bardzo łatwo wyznaczamy takie k, że:

$$(m_2' - m_2) \cdot k \equiv 1 \pmod{p-1},$$

bo $\left[m_2'-m_2\right]_{p-1}$ ma w tej sytuacji element odwrotny w Z_{p-1}.

Wówczas $\alpha^{(m_1-m_1')\cdot k} = \beta^{(m_2'-m_2)\cdot k} = \beta^{1}$ w Z_p

Redukcja wykładnika modulo $\varphi(p) = p-1$

Zatem: $\log_\alpha \beta = (m_1 - m_1')\cdot k$

ewentualnie $\bmod(p-1)$

Znaleźliśmy więc logarytm!

Przypadek 3 $t = q$ NIE ZACHODZI. Istotnie, niech będzie $NWD(m_2'-m_2, p-1) = q$. Przypadek $m_2' - m_2 = 0$ nie zachodzi, gdyż wówczas byłoby $NWD(m_2' - m_2, p-1) = p-1$. Jednocześnie z faktu, że $m_2, m_2' \in Z_q$ wynika, że $-(q-1) \le m_2'-m_2 \le q-1$ a więc $m_2'-m_2$ nie jest podzielne przez *q* a zatem sprzeczność.

Przypadek 4 $t = p-1$ NIE ZACHODZI.. Istotnie, niech będzie $t = p-1$, ale to jest możliwe wtedy i tylko wtedy, jeśli $m_2 = m_2'$. Jednocześnie na wstępie założyliśmy, że:

$$\alpha^{m_1} \otimes_p \beta^{m_2} = \alpha^{m_1'} \otimes_p \beta^{m_2'}$$

zatem

$$\alpha^{m_1} = \alpha^{m_1'} \quad \text{w } Z_p$$

i ponieważ α jest generatorem w Z_p^*, to $m_1 = m_1' \pmod{p-1}$ i stąd $m_1 = m_1'$ ale to daje sprzeczność, gdyż z założenia $(m_1, m_2) \neq (m_1', m_2')$.

Przypadek 2: $t = 2$. Mamy wówczas oczywiście $NWD(m_2'-m_2, q) = 1$. Za pomocą rozszerzonego algorytmu Euklidesa łatwo możemy wyznaczyć takie *k*, że

$$(m_2'-m_2)\cdot k \equiv 1 (\bmod q)$$

wówczas istnieje takie $i \in Z$, że $(m_2' - m_2)\cdot k = 1 + i\,q$ mamy więc

$$\alpha^{(m_1-m_1')\cdot k} = \beta^{(m_2'-m_2)\cdot k} = \beta^{1+i\cdot q} \text{ w } Z_p .$$

Łatwo wykazać, że $\beta^q = -1$ w Z_p. Istotnie $\beta^{p-1} = 1$, bo β to generator w Z_p^*, czyli $\beta^{2q} - 1 = 0$ w Z_p a zatem:

$$(\beta^q - 1)(\beta^q + 1) = 0$$

Zauważmy, że $(\beta^q - 1)$ nie może być równe 0, bo β to generator, zatem $\beta^q + 1$ jest równe 0 czyli $\beta^q = -1$ w Z_p. Zatem:

$$\alpha^{(m_1-m_1')\cdot k} = \beta(-1)^i \text{ w } Z_p .$$

Ostatecznie: Jeśli *i* jest parzyste, to:

$$\alpha^{(m_1-m_1')\cdot k} = \beta \text{ w } Z_p, \text{ czyli } \quad \log_\alpha \beta = (m_1 - m_1')\cdot k \qquad (*)$$

Jeśli *i* jest nieparzyste, to: $\alpha^{(m_1-m_1')\cdot k} = -\beta$ w Z_p,
a ponieważ $\alpha^q = -1$ w Z_p (to samo rozumowanie jak w przypadku β), to otrzymujemy:

$$\alpha^{(m_1-m_1')\cdot k+q} = \beta \text{ w } Z_p, \text{ czyli } \log_\alpha \beta = (m_1 - m_1')\cdot k + q \quad (**)$$

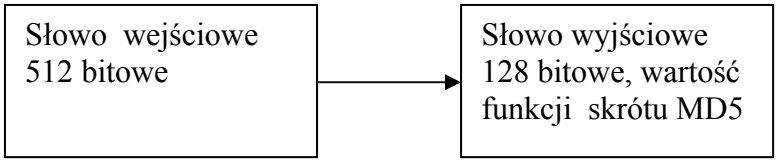
Łatwo sprawdzić, która z równości (*) czy (**) zachodzi. Znaleźliśmy więc logarytm dyskretny $\log_\alpha \beta$.

■

5.3. Funkcja haszująca MD 5

Twórcą funkcji haszującej MD 5 jest R. Rivest (współautor szyfru RSA). MD 5 jest typowym przedstawicielem całej klasy algorytmów funkcji skrótu. MD 5 stała się punktem wyjścia do standardu amerykańskiego SHA (Secure Hash Algorithm).

- MD 5 jest uważany za algorytm bardzo pewny. Jest wykorzystywany między innymi w znanym pakiecie PGP (ang. Pretty Good Privacy).
- MD 5 przekształca ciągi binarne 512 bitowe na 128 bitowe jest więc odwzorowaniem



Rys.5.2 Sposób przekształcania słowa wejściowego przez MD5

Algorytm MD 5 jest łatwy do implementacji zarówno softwarowej jak i hardware'owej – operuje blokami 32 bitowymi.

MD 5 składa się z 4 tzw. rund; czyli etapów obliczeń

1.Wstępnie definiujemy 4 elementarne operacje po współrzędnych na 32 bitowych słowach.

OR (czyli lub)

XOR (czyli suma modulo 2) $\oplus$;

$\neg$ (negacja)

- (iloczyn)

Każdy typowy mikroprocesor 32 bitowy (np. Pentium 4) ma rozkazy OR, XOR, AND i NOT realizujące powyższe działania.

2. Za pomocą powyższych działań definiujemy 4 podstawowe dla dalszego opisu algorytmu funkcje F , G , H , I (działania 3 argumentowe w $\{0,1\}^{32}$), gdzie argumenty X,Y,Z są słowami 32 bitowymi a wynik jest również 32 bitowy.

$F(X,Y,Z)=X\cdot Y \ OR \ (\neg X)\cdot Z$; $F:\{0,1\}^{32}\times\{0,1\}^{32}\times\{0,1\}^{32}\to\{0,1\}^{32}$

$G(X,Y,Z)=X\cdot Z \ OR \ Y\cdot(\neg Z)$; $G:\{0,1\}^{32}\times\{0,1\}^{32}\times\{0,1\}^{32}\to\{0,1\}^{32}$

$H(X,Y,Z)=X \ XOR \ Y \ XOR \ Z$; $H:\{0,1\}^{32}\times\{0,1\}^{32}\times\{0,1\}^{32}\to\{0,1\}^{32}$

$I(X,Y,Z)=Y \ XOR \ (X \ OR \ \neg Z)$; $I:\{0,1\}^{32}\times\{0,1\}^{32}\times\{0,1\}^{32}\to\{0,1\}^{32}$

3. Za pomocą powyższych czterech funkcji *F*, *G*, *H* i *I* definiujemy cztery procedury: *FF, GG, HH, II*

$$FF(\overbrace{a,b,c,d}^{4},M_j,s,t_i)$$

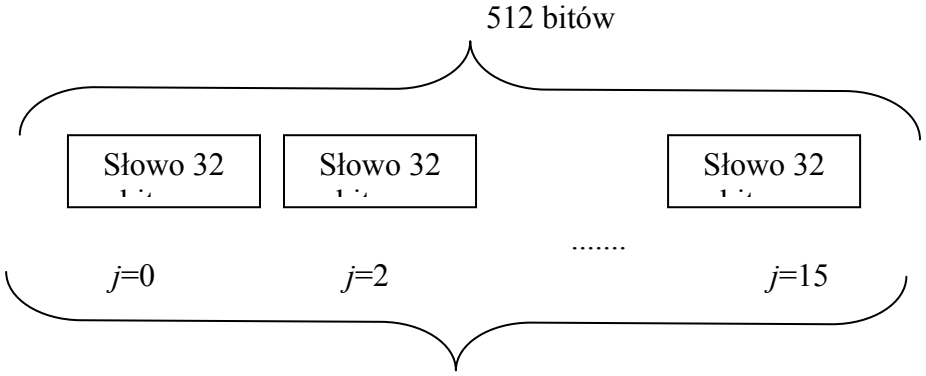
co robi:

$$a := b + shift_s(a + F(b,c,d) + M_j + t_i)$$

gdzie $shift_s:\{0,1\}^{32}\to\{0,1\}^{32}$ jest przesunięciem cyklicznym w lewo (rotacją w lewo) o *s* pozycji słowa 32 bitowego. Przesunięcie to zadane jest wzorem

$$shift_s(a_1,a_2,...a_{32}) = (a_{s+1},a_{s+2},...,a_{32},a_1,a_2,...,a_s)$$

M_j dla j =0,2,...,15 oznacza *j*-ty 32 bitowy blok ciągu wejściowego a + oznacza tu dodawanie modulo 2^{32} liczb 32 bitowych.



Rys. 5.3 Podział słowa wejściowego na 32 bitowe bloki M_j dla j =0,2,...,15

t_i jest rozwinięciem dwójkowym liczby $t_i = \lfloor 2^{32} \cdot |\sin i| \rfloor$ (podłoga z iloczynu 2^{32} przez moduł sinusa z indeksu *i*).

UWAGA 1. Definiujemy procedury GG, HH oraz II tak, jak FF zastępując F przez G, potem przez H a na końcu przez przez I.

UWAGA 2. Algorytm używa 32 bitowych zmiennych *a, b, c, d* lub inaczej 32 bitowych rejestrów *a, b, c, d*. Ich początkowe wartości w zapisie heksadecymalnym są takie:

a=01 23 45 67, *b*=89 AB CD EF
c=FE DC BA 98, *d*= 76 54 32 10

zmienne te nazywamy zmiennymi łańcuchowymi, ang. „chaining variabes". Algorytm funkcji skrótu SHA jest identyczny jak algorytm MD5 z tym, że zmienne *a, b, c, d* inicjowane są innymi wartościami podanymi poniżej

a= 67 45 23 01 , *b*= EF CD AB 89
c=98 BA DC FE , *d*=10 32 54 76

RUNDA 1. Po kolei wywołujemy 16 razy procedurę *FF*:

FF(a,b,c,d,M$_0$,7,t_1)
FF(d,a,b,c,M$_1$,12,t_2)
FF(c,d,a,b,M$_2$,17,t_3)
FF(b,c,d,a,M$_2$,22,t_4)
.
.
.
FF(b,c,d,a,M$_{15}$,22, t_{16})

16 wywołań

UWAGA 1. Jak widać pierwsze cztery zmienne *a*,*b*,*c*,*d*, przesuwamy w prawo cyklicznie o 1 pozycję w kolejnych wywołaniach procedury *FF*.

UWAGA 2. Przedostatnie argumenty tworzą ciąg okresowy o okresie 4.

$$\overbrace{7,12,17,22}^{4\,wyrazy},\overbrace{7,12,17,22,}^{4\,wyrazy}...$$

RUNDA 2 Jest podobna do rundy 1, ale:
1) Zamiast procedury FF, stosujemy procedurę GG.
2) Zamiast ciągu: 7,12,17,22, stosujemy ciąg 5,9,14,20.
3) *i*-te wywołanie procedury GG używa t_{16+i} zamiast t_i, oraz $M_{(1+5i)(\mathrm{mod}16)}$ zamiast M_i.

Runda 2 polega na wywołaniu 16 razy procedury GG z parametrami zmodyfikowanymi jak wyżej.

RUNDA 3 Jest podobna do rundy 1, ale:
1) Zamiast procedury FF, stosujemy HH.
2) Zamiast ciągu: 7,12,17,22 stosujemy ciąg 4,11,16,23.
3) *i*-te wywołanie procedury HH używa t_{32+i} zamiast t_i oraz $M_{(5+3i)(\mathrm{mod}16)}$ zamiast M_i.

Runda 3 polega na wywołaniu 16 razy procedury HH z parametrami zmodyfikowanymi jak wyżej.

RUNDA 4 Jest podobna do rundy 1, ale:
1) Zamiast procedury FF, stosujemy procedurę II.
2) Zamiast ciągu 7,12,17,22 stosujemy ciąg: 6,10,15,21
3) *i*-te wywołanie procedury II używa t_{48+i} zamiast t_i oraz $M_{(10+7i)(\mathrm{mod}16)}$ zamiast M_i.

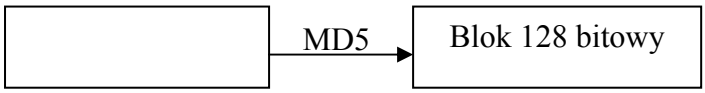
Runda 4 polega na wywołaniu 16 razy procedury HH z parametrami zmodyfikowanymi jak wyżej.

Ostatecznie wartość funkcji hashującej MD 5 jest ciągiem 128 bitów otrzymanych z połączenia zawartości czterech 32 bitowych rejestrów *a,b,c,d*.

5.4. Schematy ogólne tworzenia funkcji skrótu

Schematy ogólne tworzenia funkcji skrótu podają metodę jak obliczać funkcję skrótu dla tekstów o dowolnej długości.

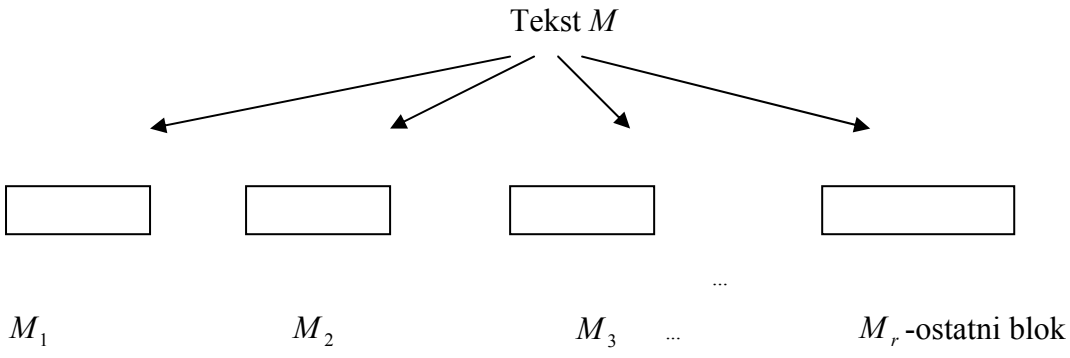
1.Zasadniczo funkcję skrótu MD5 stosujemy do ciągów binarnych o długości 512 bitów uzyskując ciągi 128 bitowe. MD5 jest więc tzw. blokową funkcją skrótu czyli funkcją postaci $h:\{0,1\}^t \to \{0,1\}^n$, gdzie $t > n$.



Rys. 5.4 Funkcja skrótu MD5

Jednak w sposób naturalny można MD5 rozszerzyć na kilka sposobów na teksty o dowolnej długości. Mówiąc o MD5 myślimy z reguły o takich rozszerzeniach.

2.Omówimy teraz sposoby obliczenia funkcji skrótu dla tekstów o dowolnej długości. Widać analogię z trybami pracy szyfrów blokowych, tam też staramy się rozszerzyć blokową funkcję szyfrującą na funkcję szyfrującą pracującą na tekstach dowolnej długości a blokową funkcję deszyfrującą na funkcję deszyfrującą teksty o dowolnej długości. Rys. 2 wyjaśnia jak dzielimy tekst wejściowy na bloki:



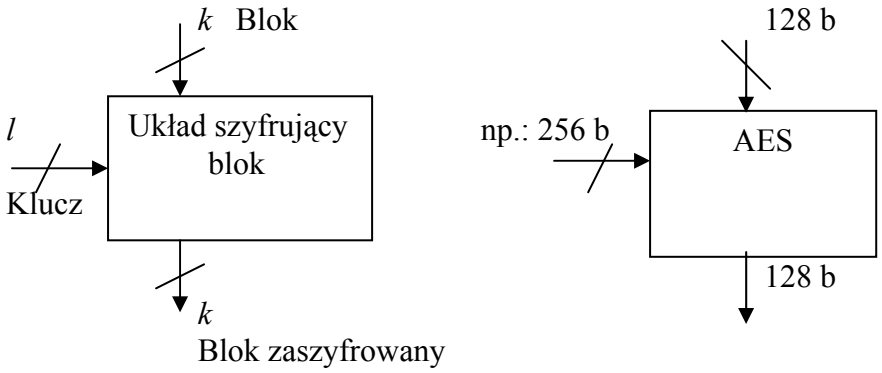
Rys. 5.5 Podział tekstu wejściowego na bloki

Blok M_k dla $k = 1,2,...,r$ ma zawsze ustaloną długość d np.: 512 bitów. r dobieramy tak, by $r \cdot d \geq |\text{tekst}|$. Blok ostatni tzn. blok o numerze r uzupełniamy jeśli trzeba w ustalony sposób do pełnej długości (np.: zerami lub spacjami). Od tego momentu pracujemy na pełnych blokach wykorzystując tzw. schematy ogólne tworzenia funkcji skrótu. Do schematów takich zaliczamy schematy Rabina, Daviesa oraz Matyasa-Meyera-Oseasa. Omówimy je po kolei.

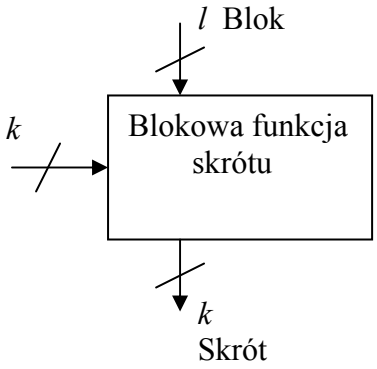
Schemat Rabina

Niech $E : V_1^k \times V_1^l \to V_1^k$ będzie funkcją opisującą blokową funkcję skrótu lub blokową funkcję szyfrującą . Alfabet V_1 w którym zapisujemy tekst wejściowy jest z reguły alfabetem binarnym tzn. $V_1 = \{0,1\}$. E może być to np.: zdefiniowana poprzednio funkcja MD5 lub blokowa funkcja szyfrującą np. z DES'a czy AES'a.

a) blokowa funkcja szyfrująca



b) blokowa funkcja skrótu

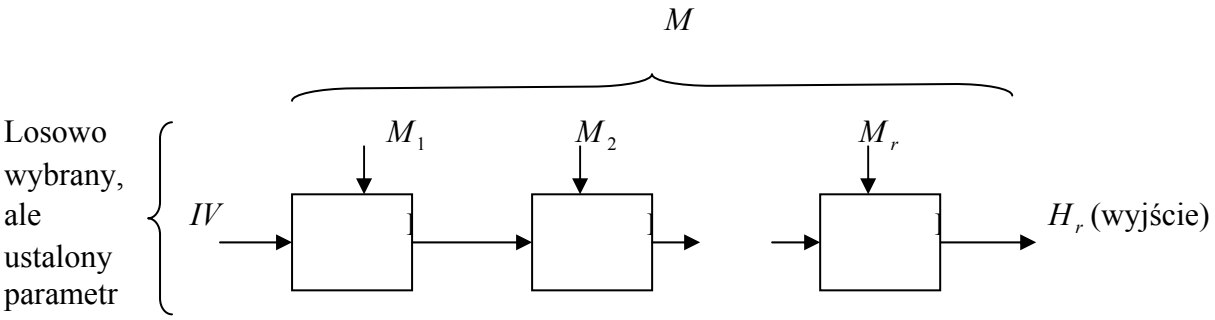


Rys. 5.6 Bloki elementarne w schemacie Rabina

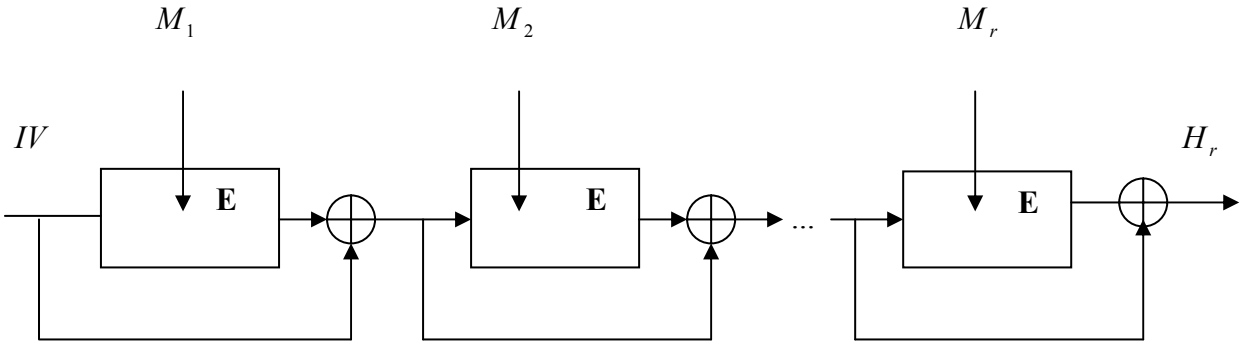
Schemat Rabina jest taki. Niech $M = M_1 M_2 ... M_r$ będzie słowem wejściowym (M jest konkatenacją słów $M_1, M_2, ..., M_r$) a $H_0 = IV$ (ang. Initial Value) wartością początkową.

$H_k = E(H_{k-1}, M_k)$ dla $k = 1,2,...,r$ przy czym H_r jest wartością funkcji skrótu dla M

IV – Initial Value jest losowym ciągiem, ale publicznie znanym



Rys. 5.7. Schemat Rabina obliczania funkcji skrótu dla dowolnie długiego tekstu.



Rys. 5.8 Schemat Daviesa obliczania funkcji skrótu. H_r jest wartością funkcji skrótu dla słowa wejściowego $M = M_1 M_2 ... M_r$.

$H_0 = IV$ (wartość początkowa ustalona i ogólnie znana)

$H_k = E(M_k, H_{k-1}) \oplus H_{k-1}$ dla $k = 1,2,...,r$

H_r = wartość funkcji skrótu dla słowa wejściowego $M = M_1 M_2 ... M_r$

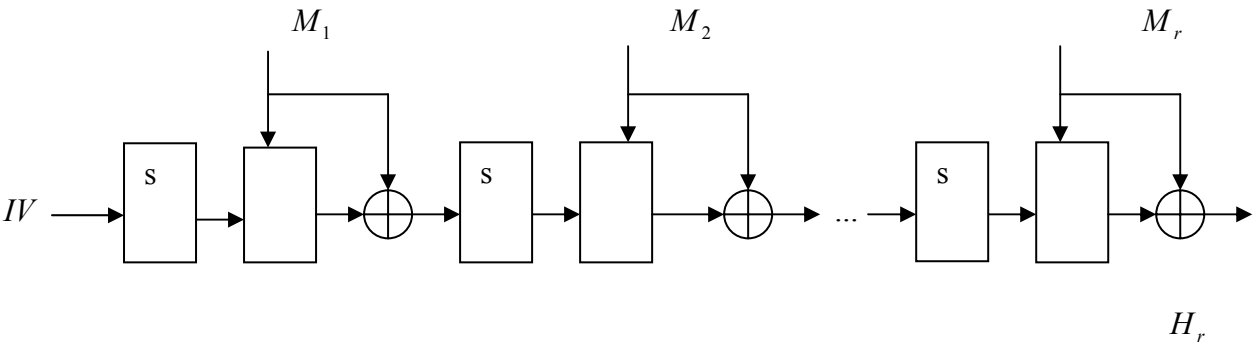
Schemat Matyasa – Meyera – Oseasa

- Uważany jest powszechnie za bezpieczny standard
- s jest funkcją, która kryptogram przekształca na klucz (bloki klucza i kryptogramu mogą być różnej długości). Jeśli blok klucza i blok kryptogramu mają tę samą długość to funkcja s może być tożsamością.

$H_0 = IV$ (IV to Initial Value czyli wartość początkowa)

$H_k = E\left(s(H_{k-1}), M_k\right) \oplus M_k$ dla każdego $k = 1,2,...,r$

H_r = wartość funkcji skrótu dla słowa wejściowego $M = M_1 M_2 ... M_r$



Rys. 5.9 Schemat Matyasa – Meyera – Oseasa. H_r jest wartością funkcji skrótu dla słowa wejściowego $M = M_1 M_2 ... M_r$.

5.5. Paradoks dnia urodzin

1. Zanim zajmiemy się atakami na funkcje skrótu omówimy tzw. paradoks dnia urodzin. Rozważmy następujący problem. Windą jedzie 7 osób z parteru, a pięter jest 10. Chcemy obliczyć prawdopodobieństwo, że te osoby wysiądą na różnych piętrach tzn. każda osoba na innym. zakładamy przy tym, że każda z osób podejmuje decyzję o wyjściu z windy niezależnie o pozostałych.

Obliczymy szukane prawdopodobieństwo jako stosunek liczby zdarzeń elementarnych sprzyjających zajściu zdarzenia do wszystkich możliwych zdarzeń elementarnych. Wszystkich możliwych, równie prawdopodobnych zdarzeń elementarnych jest tu tyle ile 7 elementowych wariancji z powtórzeniami ze zbioru 10 elementowego a więc 10^7. Warto może w tym miejscu przypomnieć znane z kombinatoryki ogólne wzory: liczba k elementowych wariacji z powtórzeniami ze zbioru n elementowego jest równa n^k a liczba k elementowych wariacji bez powtórzeń ze zbioru n elementowego oznaczana symbolem V_n^k jest równa $V_n^k = \frac{n!}{(n-k)!}$.

Zdarzeń elementarnych sprzyjających jest V_{10}^7 (V_{10}^7 to liczba siedmioelementowych wariancji bez powtórzeń ze zbioru 10 elementowego). Zatem poszukiwane prawdopodobieństwo jest równe

$$p = \frac{V_{10}^7}{10^7} = \frac{\frac{10!}{3!}}{10^7} = \frac{10 \cdot 9 \cdot 8 \cdot \ldots \cdot 4}{10^7} \approx 0,018$$

Ogólnie dla n-pięter i $1 \le k \le n$ osób mamy

$$p = \frac{V_n^k}{n^k} = \frac{\frac{n!}{(n-k)!}}{n^k}$$

Rozważane doświadczenie możemy również opisać ciągiem skończonym siedmiu zmiennych losowych $(X_1, X_2, \ldots, X_7)$, określonych na pewnej przestrzeni probabilistycznej $(\Omega, \mathfrak{M}, P)$, gdzie $X_i(\omega) \in \{1,2,\ldots,10\}$ i $\omega \in \Omega$. Naturalne jest założenie, że te zmienne losowe są niezależne i mają rozkład równomierny na $\{1,2,\ldots,10\}$. Prościej jest jednak potraktować zadanie jako zdanie z elementarnego rachunku prawdopodobieństwa i rozumować tak jak powyżej.

2. Podobnie możemy zapytać o prawdopodobieństwo tego, że w grupie studenckiej *k*-osobowej wszystkie osoby obchodzą urodziny różnego dnia. Analogia z windami jest oczywista. Zatem przyjmijmy, że liczba osób $1 \le k \le 365$ (oczywisty warunek konieczny niezerowego prawdopodobieństwa), wówczas korzystając z wyników p.1 mamy

$$p_k = \frac{V_{365}^k}{365^k} = \frac{365 \cdot \ldots \cdot (365-k+1)}{365^k} = \left(1-\frac{1}{365}\right) \cdot \ldots \cdot \left(1-\frac{k-1}{365}\right)$$

Rozważaną sytuację z dniem urodzin możemy również opisać ciągiem skończonym *k* zmiennych losowych $(X_1, X_2, ..., X_k)$, określonych na pewnej przestrzeni probabilistycznej $(\Omega, \mathfrak{M}, P)$, gdzie $X_i(\omega) \in \{1,2,...,365\}$ i $\omega \in \Omega$. Naturalne jest założenie, że te zmienne losowe $X_1, X_2, ..., X_k$ są niezależne i mają rozkład równomierny na zbiorze $\{1, 2, ..., 365\}$.

3. Podobnie możemy zapytać o prawdopodobieństwo tego, że w grupie *k*-osobowej co najmniej 2 osoby obchodzą urodziny tego samego dnia. Oczywiście to prawdopodobieństwo jest równe:

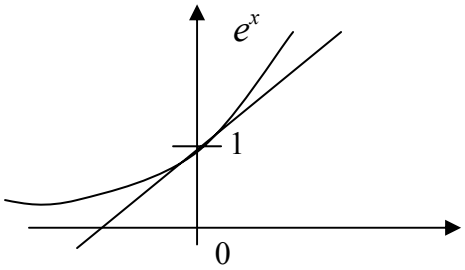
$$p_k^{'} = 1 - p_k$$

4. Można teraz zadać jeszcze jedno pytanie: jaka musi być liczba *k*, $1 \le k \le n$, by z prawdopodobieństwem $> \frac{1}{2}$ co najmniej 2 osoby obchodziły urodziny tego samego dnia. Oczywiście szukamy takiego k_0, by dla $k \ge k_0$ było:

$$p_k^{'} = 1 - p_k = 1 - \underbrace{\left(1-\frac{1}{365}\right)\left(1-\frac{2}{365}\right) \cdot \ldots \cdot \left(1-\frac{k-1}{365}\right)}_{k-1} > \frac{1}{2}$$

Obliczenia są nieco żmudne, ale stosując oszacowanie $1 + x \le e^x$ por. rys. 1 dostajemy następujące oszacowanie na p_k

$$p_k \le e^{\frac{-1}{365}} \cdot e^{\frac{-2}{365}} \cdot \ldots \cdot e^{-\frac{k-1}{365}} = e^{-\frac{1+2+\ldots+(k-1)}{365}} = e^{-\frac{k \cdot (k-1)}{2 \cdot 365}} = e^{-\frac{k \cdot (k-1)}{730}}$$



Rys. 5.10 Geometryczna interpretacja oszacowania $1 + x \le e^x$

Na to, by $p_k \le \frac{1}{2}$ (czyli $p_k^{'} = 1 - p_k > \frac{1}{2}$) wystarczy więc wziąć takie k, by $k \cdot (k-1) \ge 730 \ln 2 \approx 505{,}99$. Łatwo sprawdzić, że pierwszym k_0 spełniającym tę nierówność będzie $k_0 = 23$, czyli dla $k \ge k_0$ będzie dobre.

To jest zaskakujące bo uzyskany wynik oznacza, że już w stosunkowo małej grupie osób prawdopodobieństwo zdarzenia polegającego na tym, że 2 osoby obchodzą urodziny tego samego dnia jest stosunkowo duże. Uzyskany wynik nosi nazwę „paradoksu dnia urodzin".

5. Podobne zadanie jak w p.4 możemy sformułować dla funkcji skrótu. Ile trzeba wziąć – tzn. wylosować wartości k funkcji skrótu $h : \{0,1\}^m \to \{0,1\}^n$ by z prawdopodobieństwem $> \frac{1}{2}$ udało się znaleźć 2 identyczne wartości funkcji skrótu.

5.6. Ataki na funkcje skrótu

Rozważmy blokową funkcję skrótu $h:\{0,1\}^t \to \{0,1\}^n$, gdzie $t > n$. Jeśli na przestrzeni $\{0,1\}^t$ zdefiniowaliśmy równomierny rozkład prawdopodobieństwa oraz dla każdego $i \neq j$ $i, j \in \{0,1\}^n$ mamy $card(f^{-1}(\{i\})) = card(f^{-1}(\{j\}))$ to prawdopodobieństwo wybrania takich różnych argumentów $x, y \in \{0,1\}^t$, że $h(x) = h(y)$ jest równe $\frac{1}{2^n}$.

Długość wartości funkcji skrótu *n* z uwagi na możliwość tzw. ataku urodzinowego w praktyce jest ≥ 128 bitów.

Atak urodzinowy polega na obliczeniu wartości funkcji skrótu dla wielu argumentów a następnie poszukiwaniu 2 jednakowych wartości funkcji $h:\{0,1\}^t \to \{0,1\}^n$.

Plik, tekst, wiadomość *m* o dowolnej długości „obłożona" funkcją skrótu f daje ciąg bitów o ustalonej długości *n* (np.: 20 bajtów, czyli 20×8=160 bitów).

Weryfikacja integralnośći danych.

Jeśli znamy wartość funkcji skrótu $f(m)$ i samą funkcję skrótu i dysponujemy parą $(m', f(m))$, to możemy sprawdzić integralność danych, czyli sprawdzić czy $m = m'$ obliczając dla m' wartość $f(m')$. Jeśli $f(m) = f(m')$, to przyjmujemy, że $m = m'$ (choć tak być oczywiście nie musi bo funkcja f nie jest różnowartościowa), czyli stwierdzamy, że „integralność danych jest zachowana".

Uwaga 2. Intuicja funkcji skrótu jest prosta. Dla danej wiadomości jawnej *m*, wartość $f(m)$ to „streszczenie" tej wiadomości zawarte w słowie o stałej liczbie *n* bitów. Typowe „streszczenie" ma długość 20 bajtów, tzn. 160 bitów. Z reguły *n* wybiera się tak duże by prawdopodobieństwo zdarzenia $\{\omega \in \Omega; f(X(\omega)) = f(Y(\omega))\}$, (gdzie X i Y są niezależnymi zmiennymi losowymi o rozkładzie jednostajnym na *M* określonymi na pewnej przestrzeni probabilistycznej $(\Omega, \mathfrak{M}, P)$) było bardzo małe. Wybór właściwego *n* umożliwia rozwiązanie tzw. problemu (lub paradoksu) dnia urodzin.

Dwa pojęcia są używane do opisywania własności funkcji skrótu: tzw. słaba bezkonfliktowość i mocna bezkonfliktowość.

Słaba bezkonfliktowość funkcji skrótu oznacza , że jeśli $f(m) = y$ to praktycznie nie jesteśmy w stanie znaleźć takiego $m' \neq m$, że $f(m') = y$. Mocna bezkonfliktowość oznacza z kolei, że praktycznie nie można znaleźć takich różnych wiadomości m, m', że $f(m) = f(m')$. Z mocnej bezkonfliktowości wynika oczywiście słaba bezkonfliktowość.

Literatura

[1] A. Menezes, P. Oorschot, S. Vanstone; Handbook of Applled Cryptography; CRC Press Inc., 1997. (treść jest na stronie: http://cacr.math.uwaterloo.ca/hac)

[2] J.Stokłosa, T.Bilski,T.Pankowski; Bezpieczeństwo danych w systemach informatycznych; PWN, Warszawa 2001.

[3] N.Koblitz; Algebraiczne aspekty kryptografii; WNT, Warszawa 2000.

[4] N.Koblitz; A Course in Number Theory and Cryptography; Springer Verlag, New York 1994. (jest przekład polski p.t. Wykład z teorii liczb i kryptografii; WNT, Warszawa 1995.)

[5] M.Kutyłowski, W.Strothmann; Kryptografia, teoria i praktyka zabezpieczania systemów komputerowych; Oficyna Wydawnicza Read Me, Warszawa 2001.

[6] B.Schneier; Kryptografia dla praktyków; WNT, 2002.

Zadania

Zadanie 8.1
Pokazać, że funkcja skrótu Chaum – van Heijst – Pfitzmanna jest silnie bezkonfliktowa o ile obliczenie logarytmu dyskretnego $\log_\alpha \beta$ w Z_p jest praktycznie niewykonalne, gdzie α i β są generatorami grupy multiplikatywnej Z_p^* ciała Z_p.

Rozwiązanie
1.Niech p będzie dużą liczbą pierwszą postaci $2q+1$, gdzie q jest również liczbą pierwszą. Łatwo sprawdzić, że takie liczby pierwsze p i q istnieją np.: 2 i 5,5 i 11. Osobnym problemem jest jednak generacja dużych liczb pierwszych p i q spełniających warunek $p = 2q+1$.
Funkcją skrótu Chaum – van – Heijst – Pfitzmanna jest zdefiniowana wzorem

$$h : Z_q^2 \ni (m_1, m_2) \to h(m_1, m_2) = \alpha^{m_1} \beta^{m_2} \pmod p \in Z_p$$

Jeżeli liczby $m_1, m_2 \in Z_q$ są zapisywane za pomocą r bitów w kodzie NKB to $h(m_1, m_2)$ da się zapisać za pomocą co najwyżej $r + 1$ bitów. Zatem funkcja h może być traktowana jako przekształcająca ciągi 2 r bitowe w ciąg $r + 1$ bitowy. Słowo będące wartością funkcji h jest $\frac{r+1}{2r}$ razy krótsze od słowa będącego argumentem funkcji h. Przy dużych r jest to $\approx 1/2$.

Uwaga. Grupa multiplikatywna Z_p^* jest cykliczna, istnieją więc odpowiednie generatory α i β.

2.Silna bezkonfliktowość h (z definicji) oznacza, że znalezienie dwóch różnych argumentów (m_1, m_2) i (m'_1, m'_2) takich, że $h(m_1, m_2) = h(m'_1, m'_2)$ jest praktycznie niemożliwe tzn. nie jest znany skuteczny algorytm wyszukiwania takich argumentów.

3.Wykażemy teraz, że znajomość takich różnych argumentów (m_1, m_2) i (m'_1, m'_2), że

$$h(m_1, m_2) = h(m'_1, m'_2) \qquad (*)$$

pozwala w czasie liniowym (realizacja rozszerzonego algorytmu Euklidesa) obliczyć $\log_\alpha \beta$ w grupie multiplikatywnej ciała Z_p. Gdyby więc funkcja h nie była silnie bezkonfliktowa, to znaczy znalezienie różnych argumentów (m_1, m_2), (m'_1, m'_2) takich, że (*) zachodzi, byłoby łatwe obliczeniowo, to i obliczanie $\log_\alpha \beta$ byłoby łatwe obliczeniowo.

4. Równość (*) oznacza, że

$$\alpha^{m_1} \otimes_p \beta^{m_2} = \alpha^{m'_1} \otimes_p \beta^{m'_2}$$

gdzie $\otimes_p$ oznacza mnożenie modulo w Z_p. Zatem

$$\alpha^{m_1-m_1^{'}} = \alpha^{m_2^{'}-m_2} \text{ (równość w pierścieniu } Z_p) \qquad (**)$$

Niech $t \overset{ozn}{=} NWD(m`_2-m_2, p-1)$. Ponieważ $p-1=2q$, więc t może być równa 1, 2, q, $p-1$.

5. Rozważmy po kolei powyższe możliwości. Niech więc t =1. Możemy wówczas znaleźć element odwrotny k do $(m`_2-m_2)$ w pierścieniu $Z_{p-1}$ tzn. takie $k \in Z_{p-1}$, że $k \otimes_{p-1} (m`_2-m_2) = 1$ lub równoważnie $k \cdot (m`_2-m_2) \equiv 1 (\mathrm{mod}\ p-1)$ (zauważmy, że $m`_2-m_2 \in Z_{p-1}$).

Podnieśmy obie strony równości (**) do potęgi k dostaniemy wówczas równość w Z_p dla pewnego całkowitego s.

$$\alpha^{(m_1-m_2^{'})k} = \beta^{(m_2^{'})k} = \beta^{1+s(p-1)}$$

i dalej z małego twierdzenia Fermata mamy, że

$$\beta^{1+s(p-1)} = \beta^1$$

zatem

$$\alpha^{(m_1-m_1^{'})k} = \beta$$

Oznacza to, że $(m_1-m`_1)k = \log_\alpha \beta$ czyli $(m_1-m`_1)k$ jest logarytmem dyskretnym z β przy podstawie α.

Do znalezienia elementu odwrotnego k w pierścieniu Z_{p-1} można wykorzystać rozszerzony algorytm Euklidesa, który w czasie liniowym oblicza k.

6. Rozważmy teraz przypadek t= 2, $NWD(m`_2 - m_2, p-1) = 2$ pociąga za sobą to, że $NWD(m`_2 - m_2, q) = 1$. Istnieje więc element odwrotny k do $m`_2 - m_2$, w pierścieniu $Z_q$ (zauważmy, że $m`_2 - m_2 \in Z_q$) tzn.

$$k \otimes_q (m`_2 - m_2) = 1$$

lub równoważnie $k \cdot (m`_2 - m_2) \equiv 1 \pmod{q}$. Element k tak jak poprzednio można wyznaczyć w czasie liniowym za pomocą algorytmu Euklidesa.

Podnosząc do k –tej potęgi obie strony równośći (**) dostajemy dla pewnego $j \in Z$

$$\alpha^{(m_1 - m`_1)k} = \beta^{(m`_2 - m_2)k} = \beta^{1 + j \cdot k} \qquad (***)$$

Zauważmy teraz, że β jako generator grupy multiplikatywnej Z_p^* ($p-1$ elementowej) przy podniesieniu do potęgi $p-1$ w Z_p musi dawać jedynkę, ale nie otrzymamy 1 dla potęg naturalnych mniejszych od $p-1$.

Jeśli $\beta^{p-1} = 1$ to $\beta^{2q} = 1$ zatem $\beta^q = -1$. Zatem z (***) wynika, że $\alpha^{(m_1 - m`_1)k} = \beta \cdot (-1)^j$.

Jeśli j jest parzyste, to $\alpha^{(m_1 - m`_1)k} = \beta$ czyli $\log_\alpha \beta = (m_1 - m`_1)k$.

Jeśli j jest nieparzyste, to $\alpha^{(m_1 - m`_1)k} = -\beta$, ale ponieważ $\alpha^q = -1$ (podobnie jak $\beta^q = -1$) otrzymujemy $\alpha^{(m_1 - m`_1)k + q} = \beta$ czyli $(m_1 - m`_1)k + q = \log_\alpha \beta$.

Nie wiemy ogólnie rzecz biorąc, czy j jest parzyste czy nie, ale w obu przypadkach potrafimy obliczyć $\log_\alpha \beta$. Mamy dwie propozycje wartości logarytmu dyskretnego, łatwo sprawdzić, która jest właściwa podnosząc generator α do odpowiedniej potęgi i sprawdzając czy wynik jest równy β.

Uwaga. Czas obliczania $b^n \pmod{m}$ dla $n, m \in N, m \geq 2$ algorytmem iterowanego podnoszenia do kwadratu jest równy $O((\log_2 n)(\log_2^2 m))$ por. zadanie xx.

7. Rozważmy teraz przypadek $t = q$, tzn. $NWD(m`_2 - m_2, p-1) = q$. Przypadek ten nie może zajść, ponieważ $m_2, m`_2 \in Z_q$ zatem $-q < m_2 - m`_2 < q$, a NWD nie może być większa niż moduł mniejszego z argumentów.

8. Rozważmy przypadek ostatni $t = p$-1, czyli $NWD(m`_2 - m_2, p-1) = p-1$ wynika stąd, że $m`_2 - m_2 = 0$ czyli $m_2^{'} = m_2$.

Mnożąc uzyskaną już równość

$$\alpha^{m_1} \otimes \beta^{m_2} = \alpha^{m_1^{'}} \otimes \beta^{m_2^{'}}$$

przez β^{-m_2} i uwzględniając fakt $m_2^{'} = m_2$ dostajemy

$$\alpha^{m_1} = \alpha^{m_1^{'}}.$$

Ponieważ α jest generatorem grupy multiplikatywnej Z_p^* (jest to grupa cykliczna rzędu p-1) a wykładniki $0 \le m_1, m_2 \le q-1 < p-1$ więc równość potęg pociąga za sobą równość wykładników, a więc $m_1 = m`_1$ co przeczy założeniu, że $(m_1, m_2) \neq (m_1^{'}, m_2^{'})$. Zatem przypadek ostatni $NWD(m`_2 - m_2, p-1) = p-1$ nie może zachodzić.

9. Pokazaliśmy, że jeśli potrafimy łatwo zaleźć takie $(m_1, m_2) \neq (m`_1, m`_2)$, że $h(m_1, m_2) = h(m`_1, m`_2)$ to potrafimy łatwo obliczać logarytmy dyskretne w pierścieniu Z_p. ■