Kryptografia

z elementami kryptografii kwantowej

Ryszard Tanaś

http://zon8.physd.amu.edu.pl/~tanas

Wykład 7

Spis treści

11 Algorytm ElGamala

U podstaw działania algorytmu ElGamala leży matematyczny problem logarytmu dyskretnego.

11.1 Wybór klucza

- Wybieramy odpowiednio dużą liczbę pierwszą p, taką, że obliczenie logarytmu dyskretnego jest praktycznie niewykonalne, wybieramy liczbę całkowitą $0 < a < p - 1$ oraz liczbę g, następnie obliczamy $b \equiv g^a \pmod{p}$.
- Liczby $\{b, g, p\}$ stanowią klucz publiczny,
- zaś liczby $\{a, g, p\}$ klucz prywatny.

11 Algorytm ElGamala

U podstaw działania algorytmu ElGamala leży matematyczny problem logarytmu dyskretnego.

11.1 Wybór klucza

- Wybieramy odpowiednio dużą liczbę pierwszą p, taką, że obliczenie logarytmu dyskretnego jest praktycznie niewykonalne, wybieramy liczbę całkowitą $0 < a < p - 1$ oraz liczbę g, następnie obliczamy $b \equiv g^a \pmod{p}$.
- Liczby $\{b, g, p\}$ stanowią klucz publiczny,
- zaś liczby $\{a, g, p\}$ klucz prywatny.

11 Algorytm ElGamala

U podstaw działania algorytmu ElGamala leży matematyczny problem logarytmu dyskretnego.

11.1 Wybór klucza

- Wybieramy odpowiednio dużą liczbę pierwszą p, taką, że obliczenie logarytmu dyskretnego jest praktycznie niewykonalne, wybieramy liczbę całkowitą $0 < a < p - 1$ oraz liczbę g, następnie obliczamy $b \equiv g^a \pmod{p}$.
- Liczby $\{b, g, p\}$ stanowią klucz publiczny,
- zaś liczby $\{a, g, p\}$ klucz prywatny.

11 Algorytm ElGamala

U podstaw działania algorytmu ElGamala leży matematyczny problem logarytmu dyskretnego.

11.1 Wybór klucza

- Wybieramy odpowiednio dużą liczbę pierwszą p, taką, że obliczenie logarytmu dyskretnego jest praktycznie niewykonalne, wybieramy liczbę całkowitą $0 < a < p - 1$ oraz liczbę g, następnie obliczamy $b \equiv g^a \pmod{p}$.
- Liczby $\{b, g, p\}$ stanowią klucz publiczny,
- zaś liczby $\{a, g, p\}$ klucz prywatny.

11.2 Szyfrowanie

- Aby zaszyfrować wiadomość M wybieramy losowo liczbę k względnie pierwszą z $p-1$,
- a następnie obliczamy

$$\begin{aligned} c_1 &\equiv g^k \pmod{p} \\ c_2 &\equiv M\, b^k \pmod{p} \end{aligned}$$

- Para liczb c_1 i c_2 tworzy kryptogram, który jest dwukrotnie dłuższy od tekstu jawnego.

11.2 Szyfrowanie

- Aby zaszyfrować wiadomość M wybieramy losowo liczbę k względnie pierwszą z $p-1$,
- a następnie obliczamy

$$\begin{aligned} c_1 &\equiv g^k \pmod{p} \\ c_2 &\equiv M\, b^k \pmod{p} \end{aligned}$$

- Para liczb c_1 i c_2 tworzy kryptogram, który jest dwukrotnie dłuższy od tekstu jawnego.

11.2 Szyfrowanie

- Aby zaszyfrować wiadomość M wybieramy losowo liczbę k względnie pierwszą z $p-1$,
- a następnie obliczamy

$$\begin{aligned} c_1 &\equiv g^k \pmod{p} \\ c_2 &\equiv M\, b^k \pmod{p} \end{aligned}$$

- Para liczb c_1 i c_2 tworzy kryptogram, który jest dwukrotnie dłuższy od tekstu jawnego.

11.2 Szyfrowanie

- Aby zaszyfrować wiadomość M wybieramy losowo liczbę k względnie pierwszą z $p-1$,
- a następnie obliczamy

$$\begin{aligned} c_1 &\equiv g^k \pmod{p} \\ c_2 &\equiv M\, b^k \pmod{p} \end{aligned}$$

- Para liczb c_1 i c_2 tworzy kryptogram, który jest dwukrotnie dłuższy od tekstu jawnego.

11.3 Deszyfrowanie

$$M \quad = \quad c_2(c_1^a)^{-1} \pmod{p}$$

11.3 Deszyfrowanie

$$M \quad = \quad c_2(c_1^a)^{-1} \pmod{p}$$

11.4 Uzasadnienie

$$c_2(c_1^a)^{-1} \equiv M\, b^k (g^{ka})^{-1} \equiv M g^{ka}(g^{ka})^{-1} \equiv M \pmod{p}$$

11.5 Prosty przykład

Znajdowanie klucza

Niech $p = 229$ i $g = 6$, wybieramy $a = 70$, wtedy $b \equiv 6^{70} \pmod{229} = 97$, zatem klucz publiczny stanowią liczby $\{97, 6, 229\}$, zaś klucz prywatny stanowią liczby $\{70, 6, 229\}$

11.5 Prosty przykład

Znajdowanie klucza

Niech $p = 229$ i $g = 6$, wybieramy $a = 70$, wtedy $b \equiv 6^{70} \pmod{229} = 97$, zatem klucz publiczny stanowią liczby $\{97, 6, 229\}$, zaś klucz prywatny stanowią liczby $\{70, 6, 229\}$

Szyfrowanie

Niech wiadomość $M = 130$, wybieramy $k = 127$ takie, że $NWD(127, 228) = 1$ (liczby tej nie ujawniamy)

$$c_1 \equiv 6^{127} \pmod{229} = 155$$

$$c_2 \equiv 130 \cdot 97^{127} \pmod{229} = 169$$

11.5 Prosty przykład

Znajdowanie klucza

Niech $p = 229$ i $g = 6$, wybieramy $a = 70$, wtedy $b \equiv 6^{70} \pmod{229} = 97$, zatem klucz publiczny stanowią liczby $\{97, 6, 229\}$, zaś klucz prywatny stanowią liczby $\{70, 6, 229\}$

Szyfrowanie

Niech wiadomość $M = 130$, wybieramy $k = 127$ takie, że $NWD(127, 228) = 1$ (liczby tej nie ujawniamy)

$$c_1 \equiv 6^{127} \pmod{229} = 155$$

$$c_2 \equiv 130 \cdot 97^{127} \pmod{229} = 169$$

Deszyfrowanie

$$M = 169 \cdot (155^{70})^{-1} \pmod{229} \equiv 169 \cdot 36 \pmod{229} = 130$$

12 Jednokierunkowe funkcje hashujące (skrótu)

- Dla każdego X łatwo jest obliczyć $H(X)$
- $H(X)$ ma taką samą długość dla wszystkich tekstów X
- Dla zadanego Y znalezienie takiego X, że $H(X) = Y$ jest praktycznie niemożliwe; funkcja jednokierunkowa
- Dla danego X trudno znaleźć X' takie, że $H(X) = H(X')$; funkcja słabo bezkonfliktowa
- Nie jest praktycznie możliwe znalezienie X i X' takich, że $X \neq X'$ oraz $H(X) = H(X')$; funkcja silnie bezkonfliktowa

12 Jednokierunkowe funkcje hashujące (skrótu)

- Dla każdego X łatwo jest obliczyć $H(X)$
- $H(X)$ ma taką samą długość dla wszystkich tekstów X
- Dla zadanego Y znalezienie takiego X, że $H(X) = Y$ jest praktycznie niemożliwe; funkcja jednokierunkowa
- Dla danego X trudno znaleźć X' takie, że $H(X) = H(X')$; funkcja słabo bezkonfliktowa
- Nie jest praktycznie możliwe znalezienie X i X' takich, że $X \neq X'$ oraz $H(X) = H(X')$; funkcja silnie bezkonfliktowa

12 Jednokierunkowe funkcje hashujące (skrótu)

- Dla każdego X łatwo jest obliczyć $H(X)$
- $H(X)$ ma taką samą długość dla wszystkich tekstów X
- Dla zadanego Y znalezienie takiego X, że $H(X) = Y$ jest praktycznie niemożliwe; funkcja jednokierunkowa
- Dla danego X trudno znaleźć X' takie, że $H(X) = H(X')$; funkcja słabo bezkonfliktowa
- Nie jest praktycznie możliwe znalezienie X i X' takich, że $X \neq X'$ oraz $H(X) = H(X')$; funkcja silnie bezkonfliktowa

12 Jednokierunkowe funkcje hashujące (skrótu)

- Dla każdego X łatwo jest obliczyć $H(X)$
- $H(X)$ ma taką samą długość dla wszystkich tekstów X
- Dla zadanego Y znalezienie takiego X, że $H(X) = Y$ jest praktycznie niemożliwe; funkcja jednokierunkowa
- Dla danego X trudno znaleźć X' takie, że $H(X) = H(X')$; funkcja słabo bezkonfliktowa
- Nie jest praktycznie możliwe znalezienie X i X' takich, że $X \neq X'$ oraz $H(X) = H(X')$; funkcja silnie bezkonfliktowa

12 Jednokierunkowe funkcje hashujące (skrótu)

- Dla każdego X łatwo jest obliczyć $H(X)$
- $H(X)$ ma taką samą długość dla wszystkich tekstów X
- Dla zadanego Y znalezienie takiego X, że $H(X) = Y$ jest praktycznie niemożliwe; funkcja jednokierunkowa
- Dla danego X trudno znaleźć X' takie, że $H(X) = H(X')$; funkcja słabo bezkonfliktowa
- Nie jest praktycznie możliwe znalezienie X i X' takich, że $X \neq X'$ oraz $H(X) = H(X')$; funkcja silnie bezkonfliktowa

12 Jednokierunkowe funkcje hashujące (skrótu)

- Dla każdego X łatwo jest obliczyć $H(X)$
- $H(X)$ ma taką samą długość dla wszystkich tekstów X
- Dla zadanego Y znalezienie takiego X, że $H(X) = Y$ jest praktycznie niemożliwe; funkcja jednokierunkowa
- Dla danego X trudno znaleźć X' takie, że $H(X) = H(X')$; funkcja słabo bezkonfliktowa
- Nie jest praktycznie możliwe znalezienie X i X' takich, że $X \neq X'$ oraz $H(X) = H(X')$; funkcja silnie bezkonfliktowa

12.1 Typowe zastosowania

- Przechowywanie haseł w komputerach
- Ochrona integralności danych

12.1 Typowe zastosowania

- Przechowywanie haseł w komputerach
- Ochrona integralności danych

12.1 Typowe zastosowania

- Przechowywanie haseł w komputerach
- Ochrona integralności danych

12.2 Algorytm MD5 (Message Digest)

Algorytm, zaprojektowany przez Rivesta, jest modyfikacją wcześniejszego algorytmu MD4. Wiadomość dowolnej długości jest przekształcona w jej 128 bitowy „odcisk palca" (sumę kontrolną, skrót wiadomości). Zasadnicza procedura algorytmu działa na blokach 512 bitowych przekształcając je w 128 bitowe skróty.

12.2.1 Etapy MD5

- Krok 1

  Wiadomość dowolnej długości jest uzupełniana w taki sposób, że na końcu dodawany jest bit „1” i odpowiednia ilość zer, tak aby ostatni blok miał długość 448 bitów.

- Krok 2

  Do ostatniego bloku dodawana jest 64 bitowa liczba reprezentująca długość wiadomości (w bitach) i w ten sposób przygotowana wiadomość ma długość będącą całkowitą wielokrotnością 512 bitów. Tym samym wiadomość jest wielokrotnością 16 słów 32-bitowych. Niech $M_0, M_1, \ldots M_{N-1}$ oznaczają kolejne słowa wiadomości, gdzie N jest jest wielokrotnością 16.

12.2.1 Etapy MD5

- Krok 1

  Wiadomość dowolnej długości jest uzupełniana w taki sposób, że na końcu dodawany jest bit „1” i odpowiednia ilość zer, tak aby ostatni blok miał długość 448 bitów.

- Krok 2

  Do ostatniego bloku dodawana jest 64 bitowa liczba reprezentująca długość wiadomości (w bitach) i w ten sposób przygotowana wiadomość ma długość będącą całkowitą wielokrotnością 512 bitów. Tym samym wiadomość jest wielokrotnością 16 słów 32-bitowych. Niech $M_0, M_1, \ldots M_{N-1}$ oznaczają kolejne słowa wiadomości, gdzie N jest jest wielokrotnością 16.

12.2.1 Etapy MD5

- Krok 1

  Wiadomość dowolnej długości jest uzupełniana w taki sposób, że na końcu dodawany jest bit „1” i odpowiednia ilość zer, tak aby ostatni blok miał długość 448 bitów.

- Krok 2

  Do ostatniego bloku dodawana jest 64 bitowa liczba reprezentująca długość wiadomości (w bitach) i w ten sposób przygotowana wiadomość ma długość będącą całkowitą wielokrotnością 512 bitów. Tym samym wiadomość jest wielokrotnością 16 słów 32-bitowych. Niech $M_0, M_1, \ldots M_{N-1}$ oznaczają kolejne słowa wiadomości, gdzie N jest jest wielokrotnością 16.

- Krok 3

  Algorytm operuje na 32-bitowych zmiennych a, b, c, d, których wartości początkowe A, B, C, D w zapisie szestnastkowym są następujące:

  $A =$ 0x67452301
  $B =$ 0xefcdab89
  $C =$ 0x98badcfe
  $D =$ 0x10325476

- Krok 4

  Główna pętla algorytmu składa się z 4 rund, w każdej rundzie 16 razy wykonywane są operacje na 32 bitowych słowach. Operacje te zdefiniowane są przez 4 funkcje

- Krok 3

  Algorytm operuje na 32-bitowych zmiennych a, b, c, d, których wartości początkowe A, B, C, D w zapisie szestnastkowym są następujące:

  $A =$ 0x67452301
  $B =$ 0xefcdab89
  $C =$ 0x98badcfe
  $D =$ 0x10325476

- Krok 4

  Główna pętla algorytmu składa się z 4 rund, w każdej rundzie 16 razy wykonywane są operacje na 32 bitowych słowach. Operacje te zdefiniowane są przez 4 funkcje

- Krok 3

  Algorytm operuje na 32-bitowych zmiennych a, b, c, d, których wartości początkowe A, B, C, D w zapisie szestnastkowym są następujące:

  $A =$ 0x67452301
  $B =$ 0xefcdab89
  $C =$ 0x98badcfe
  $D =$ 0x10325476

- Krok 4

  Główna pętla algorytmu składa się z 4 rund, w każdej rundzie 16 razy wykonywane są operacje na 32 bitowych słowach. Operacje te zdefiniowane są przez 4 funkcje

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X) \wedge Z$$
$$G(X,Y,Z) = (X \wedge Z) \vee Y \wedge (\neg X)$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$
$$I(X,Y,Z) = Y \oplus (X \vee (\neg Z))$$

gdzie $\oplus$ oznacza operację xor, $\wedge$ operację and, $\vee$ operację or, zaś $\neg$ operację not.

Dla przypomnienia:

$$\begin{array}{llll} 0 \oplus 0 = 0 & 0 \wedge 0 = 0 & 0 \vee 0 = 0 & \neg 0 = 1 \\ 0 \oplus 1 = 1 & 0 \wedge 1 = 0 & 0 \vee 1 = 1 & \neg 1 = 0 \\ 1 \oplus 0 = 1 & 1 \wedge 0 = 0 & 1 \vee 0 = 1 & \\ 1 \oplus 1 = 0 & 1 \wedge 1 = 1 & 1 \vee 1 = 1 & \end{array}$$

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X) \wedge Z$$
$$G(X,Y,Z) = (X \wedge Z) \vee Y \wedge (\neg X)$$
$$H(X,Y,Z) = X \oplus Y \oplus Z$$
$$I(X,Y,Z) = Y \oplus (X \vee (\neg Z))$$

gdzie $\oplus$ oznacza operację xor, $\wedge$ operację and, $\vee$ operację or, zaś $\neg$ operację not.

Dla przypomnienia:

$$\begin{array}{llll} 0 \oplus 0 = 0 & 0 \wedge 0 = 0 & 0 \vee 0 = 0 & \neg 0 = 1 \\ 0 \oplus 1 = 1 & 0 \wedge 1 = 0 & 0 \vee 1 = 1 & \neg 1 = 0 \\ 1 \oplus 0 = 1 & 1 \wedge 0 = 0 & 1 \vee 0 = 1 & \\ 1 \oplus 1 = 0 & 1 \wedge 1 = 1 & 1 \vee 1 = 1 & \end{array}$$

Niech $X_j = M_{i*16+j}$ oznacza j-te słowo w i-tym bloku wiadomości M $(j = 0, \ldots, 15, i = 0, \ldots, N/16 - 1)$, $\hookleftarrow s$ niech oznacza cykliczne przesunięcie w lewo o s bitów oraz zdefiniujmy liczby T_k $(k = 1, \ldots, 64)$ tak, że $T_k = \lfloor 2^{32} \cdot |\sin k| \rfloor$, gdzie k jest w radianach. Mamy wtedy:

Runda 1

Niech $[abcd\ j\ s\ k]$ oznacza operację

$a = b + ((a + F(b, c, d) + X_j + T_k) \hookleftarrow s)$,

wtedy 16 operacji tej rundy to:

$$
\begin{array}{llll}
[abcd\ \ 0\ 7\ \ 1] & [dabc\ \ 1\ 12\ \ 2] & [cdab\ \ 2\ 17\ \ 3] & [bcda\ \ 3\ 22\ \ 4] \\
[abcd\ \ 4\ 7\ \ 5] & [dabc\ \ 5\ 12\ \ 6] & [cdab\ \ 6\ 17\ \ 7] & [bcda\ \ 7\ 22\ \ 8] \\
[abcd\ \ 8\ 7\ \ 9] & [dabc\ \ 9\ 12\ 10] & [cdab\ 10\ 17\ 11] & [bcda\ 11\ 22\ 12] \\
[abcd\ 12\ 7\ 13] & [dabc\ 13\ 12\ 14] & [cdab\ 14\ 17\ 15] & [bcda\ 15\ 22\ 16]
\end{array}
$$

Runda 2

Niech $[abcd\ j\ s\ k]$ oznacza operację

$a = b + ((a + G(b, c, d) + X_j + T_k) \hookleftarrow s)$,

wtedy 16 operacji tej rundy to:

$$
\begin{array}{llll}
[abcd\ \ 1\ 5\ 17] & [dabc\ \ 6\ 9\ 18] & [cdab\ 11\ 14\ 19] & [bcda\ \ 0\ 20\ 20] \\
[abcd\ \ 5\ 5\ 21] & [dabc\ 10\ 9\ 22] & [cdab\ 15\ 14\ 23] & [bcda\ \ 4\ 20\ 24] \\
[abcd\ \ 9\ 5\ 25] & [dabc\ 14\ 9\ 26] & [cdab\ \ 3\ 14\ 27] & [bcda\ \ 8\ 20\ 28] \\
[abcd\ 13\ 5\ 29] & [dabc\ \ 2\ 9\ 30] & [cdab\ \ 7\ 14\ 31] & [bcda\ 12\ 20\ 32]
\end{array}
$$

Runda 3

Niech $[abcd\ j\ s\ k]$ oznacza operację

$a = b + ((a + H(b, c, d) + X_j + T_k) \hookleftarrow s)$,

wtedy 16 operacji tej rundy to:

$$
\begin{array}{llll}
[abcd\ \ 5\ 4\ 33] & [dabc\ \ 8\ 11\ 34] & [cdab\ 11\ 16\ 35] & [bcda\ 14\ 23\ 36] \\
[abcd\ \ 1\ 4\ 37] & [dabc\ \ 4\ 11\ 38] & [cdab\ \ 7\ 16\ 39] & [bcda\ 10\ 23\ 40] \\
[abcd\ 13\ 4\ 41] & [dabc\ \ 0\ 11\ 42] & [cdab\ \ 3\ 16\ 43] & [bcda\ \ 6\ 23\ 44] \\
[abcd\ \ 9\ 4\ 45] & [dabc\ 12\ 11\ 46] & [cdab\ 15\ 16\ 47] & [bcda\ \ 2\ 23\ 48]
\end{array}
$$

Runda 4

Niech $[abcd\ j\ s\ k]$ oznacza operację

$a = b + ((a + I(b, c, d) + X_j + T_k) \hookleftarrow s)$,

wtedy 16 operacji tej rundy to:

$$
\begin{array}{llll}
[abcd\ \ 0\ 6\ 49] & [dabc\ \ 7\ 10\ 50] & [cdab\ 14\ 15\ 51] & [bcda\ \ 5\ 21\ 52] \\
[abcd\ 12\ 6\ 53] & [dabc\ \ 3\ 10\ 54] & [cdab\ 10\ 15\ 55] & [bcda\ \ 1\ 21\ 56] \\
[abcd\ \ 8\ 6\ 57] & [dabc\ 15\ 10\ 58] & [cdab\ \ 6\ 15\ 59] & [bcda\ 13\ 21\ 60] \\
[abcd\ \ 4\ 6\ 61] & [dabc\ 11\ 10\ 62] & [cdab\ \ 2\ 15\ 63] & [bcda\ \ 9\ 21\ 64]
\end{array}
$$

Po tych 4 rundach wartości a, b, c, d są dodawane do wartości A, B, C, D i algorytm przechodzi do następnego bloku M.

- Krok 5

  Po przejściu wszystkich 512-bitowych bloków wiadomości M algorytm łączy rejestry a, b, c, d dając 128 bitową liczbę będącą wartością funkcji hashującej.

- Przykład:

  880a292abed7febe7ec4c6a2e42f0bbf

  krypt.pdf

Po tych 4 rundach wartości a, b, c, d są dodawane do wartości A, B, C, D i algorytm przechodzi do następnego bloku M.

- Krok 5

  Po przejściu wszystkich 512-bitowych bloków wiadomości M algorytm łączy rejestry a, b, c, d dając 128 bitową liczbę będącą wartością funkcji hashującej.

- Przykład:
  880a292abed7febe7ec4c6a2e42f0bbf
  krypt.pdf

Po tych 4 rundach wartości a, b, c, d są dodawane do wartości A, B, C, D i algorytm przechodzi do następnego bloku M.

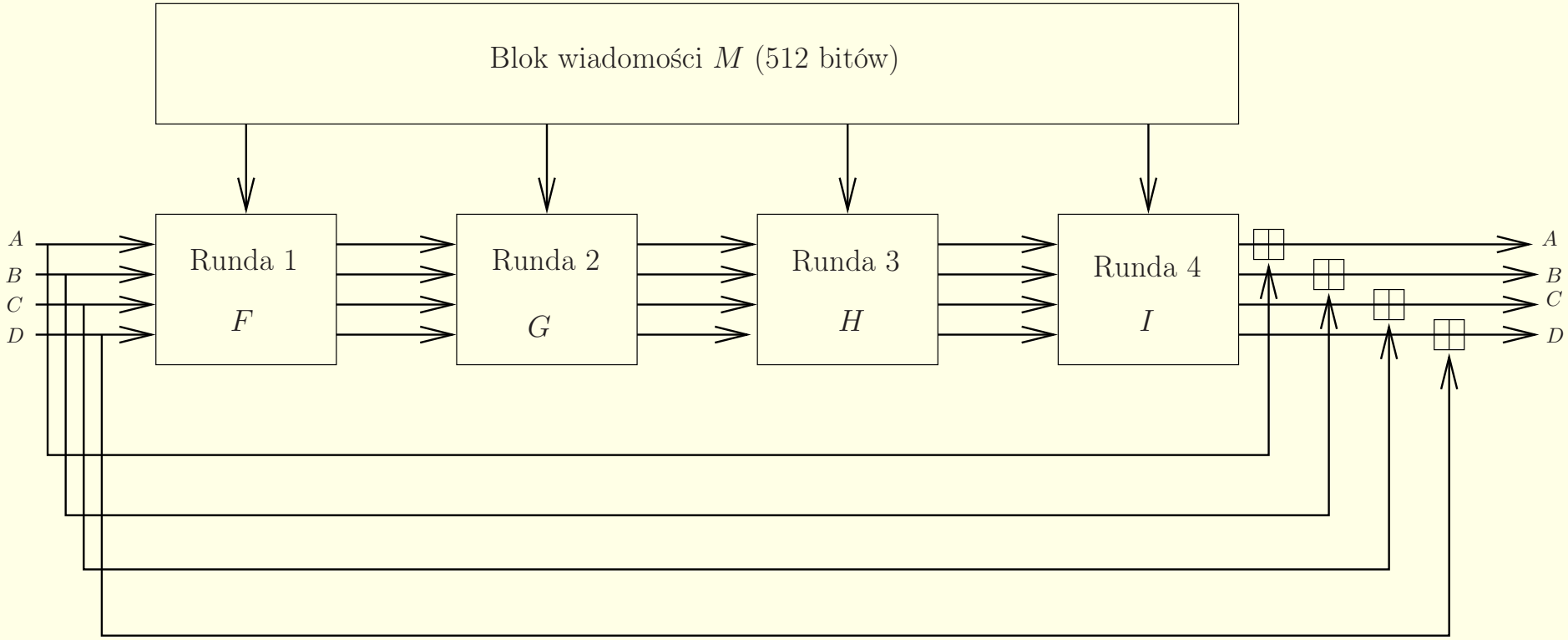
- Krok 5

  Po przejściu wszystkich 512-bitowych bloków wiadomości M algorytm łączy rejestry a, b, c, d dając 128 bitową liczbę będącą wartością funkcji hashującej.

- Przykład:

  880a292abed7febe7ec4c6a2e42f0bbf

  krypt.pdf



Główna pętla algorytmu MD5

12.3 SHA (Secure Hash Algorithm)

- Algorytm opracowany przez NIST przy udziale NSA opublikowany w 1993. Nowsza wersja SHA-1 opublikowana w 1995 r.
- Idea tego algorytmu oparta jest na MD4 i MD5.
- Wartość funkcji hashującej to liczba 160 bitowa i w związku z tym algorytm wymaga 5 rejestrów zamiast 4.
- Używa też w nieco inny sposób nieliniowych funkcji transformujących, dokonuje dodatkowych operacji na poszczególnych słowach wiadomości, w każdej rundzie wykonuje 20 operacji zamiast 16.

12.3 SHA (Secure Hash Algorithm)

- Algorytm opracowany przez NIST przy udziale NSA opublikowany w 1993. Nowsza wersja SHA-1 opublikowana w 1995 r.
- Idea tego algorytmu oparta jest na MD4 i MD5.
- Wartość funkcji hashującej to liczba 160 bitowa i w związku z tym algorytm wymaga 5 rejestrów zamiast 4.
- Używa też w nieco inny sposób nieliniowych funkcji transformujących, dokonuje dodatkowych operacji na poszczególnych słowach wiadomości, w każdej rundzie wykonuje 20 operacji zamiast 16.

12.3 SHA (Secure Hash Algorithm)

- Algorytm opracowany przez NIST przy udziale NSA opublikowany w 1993. Nowsza wersja SHA-1 opublikowana w 1995 r.
- Idea tego algorytmu oparta jest na MD4 i MD5.
- Wartość funkcji hashującej to liczba 160 bitowa i w związku z tym algorytm wymaga 5 rejestrów zamiast 4.
- Używa też w nieco inny sposób nieliniowych funkcji transformujących, dokonuje dodatkowych operacji na poszczególnych słowach wiadomości, w każdej rundzie wykonuje 20 operacji zamiast 16.

12.3 SHA (Secure Hash Algorithm)

- Algorytm opracowany przez NIST przy udziale NSA opublikowany w 1993. Nowsza wersja SHA-1 opublikowana w 1995 r.
- Idea tego algorytmu oparta jest na MD4 i MD5.
- Wartość funkcji hashującej to liczba 160 bitowa i w związku z tym algorytm wymaga 5 rejestrów zamiast 4.
- Używa też w nieco inny sposób nieliniowych funkcji transformujących, dokonuje dodatkowych operacji na poszczególnych słowach wiadomości, w każdej rundzie wykonuje 20 operacji zamiast 16.

12.3 SHA (Secure Hash Algorithm)

- Algorytm opracowany przez NIST przy udziale NSA opublikowany w 1993. Nowsza wersja SHA-1 opublikowana w 1995 r.
- Idea tego algorytmu oparta jest na MD4 i MD5.
- Wartość funkcji hashującej to liczba 160 bitowa i w związku z tym algorytm wymaga 5 rejestrów zamiast 4.
- Używa też w nieco inny sposób nieliniowych funkcji transformujących, dokonuje dodatkowych operacji na poszczególnych słowach wiadomości, w każdej rundzie wykonuje 20 operacji zamiast 16.

- Zasada działania jest jednak bardzo podobna do MD5.
- Ogólnie uważa się, że jest bezpieczniejszy niż MD5 ze względu na „dłuższą” wartość funkcji hashującej i pewne ulepszenia samego algorytmu.
- Przykład:
  12676c135a75eed1d794b5f49843742ff039eb47
  krypt.pdf

- Zasada działania jest jednak bardzo podobna do MD5.
- Ogólnie uważa się, że jest bezpieczniejszy niż MD5 ze względu na „dłuższą” wartość funkcji hashującej i pewne ulepszenia samego algorytmu.
- Przykład:
  12676c135a75eed1d794b5f49843742ff039eb47
  krypt.pdf

- Zasada działania jest jednak bardzo podobna do MD5.
- Ogólnie uważa się, że jest bezpieczniejszy niż MD5 ze względu na „dłuższą” wartość funkcji hashującej i pewne ulepszenia samego algorytmu.
- Przykład:
  12676c135a75eed1d794b5f49843742ff039eb47
  krypt.pdf