Kryptografia na procesorach wielordzeniowych

Andrzej Chmielowiec

andrzej.chmielowiec@cmmsigma.eu

Centrum Modelowania Matematycznego Sigma

Plan prezentacji

- Wprowadzenie
- Algorytmy asymetryczne
- Algorytmy symetryczne i tryby szyfrowania
- Narzędzia wspierające implementację algorytmów równoległych
- Podsumowanie

Czy równoległość jest potrzebna?

Światowe zapotrzebowanie szacuję na około pięć komputerów – *Thomas Watson (1943)*

640 kilobajtów powinno każdemu wystarczyć. – *Bill Gates (1981)*

Gdzie można stosować równoległość?

- Procesory wielordzeniowe.
- Procesory i mikrokontrolery obsługujące instrukcje typu SIMD (Single Instruction Multiple Data).
- Układy programowalne FPGA.
- Klastry.

Co nam to daje?

Zasadniczym celem implementacji algorytmów równoległych jest zwiększenie szybkości przetwarzania danych.

- Lepsze wykorzystanie sprzętu:
  - większa szybkość przetwarzania,
  - mniejsze koszty infrastruktury.
- Utrzymanie dotychczasowej wydajności przy mniejszym poborze mocy:
  - mniejsze rachunki za prąd.

Generowanie kluczy RSA

Aby wygenerować klucz RSA wykonujemy następujące kroki:

1. znajdujemy dwie liczby pierwsze p i q,
2. wyznaczamy $n = pq$ oraz liczby e i d takie, że $ed \equiv 1 \bmod (p-1)(q-1)$.

Po wygenerowaniu liczb możemy przyjąć, że pary (e, n) i (d, n) stanowią odpowiednio klucz publiczny i prywatny.

Szyfrowanie RSA

Zaszyfrowanie wiadomości x polega wtedy na wykorzystaniu klucza publicznego (e, n) i wyznaczeniu liczby

$$c \equiv x^e \bmod n.$$

Do zdeszyfrowania wiadomości potrzebujemy klucza prywatnego (d, n), który pozwala na odtworzenie przesłanej wiadomości

$$c^d \equiv (x^e)^d \equiv x^{ed} \equiv x \bmod n.$$

Zrównoleglone szyfrowanie RSA (1)

Zrównoleglenie potęgowania przy użyciu wykładnika d jest możliwe jeżeli tylko zastosujemy inną postać klucza prywatnego. Jeśli zapiszemy go w postaci (p, q, d_{p-1}, d_{q-1}), gdzie

$$d_{p-1} \equiv d \bmod p - 1,$$

$$d_{q-1} \equiv d \bmod q - 1,$$

to zasadniczą część obliczeń będziemy mogli wykonać w dwóch oddzielnych wątkach.

Zrównoleglone szyfrowanie RSA (2)

Algorytm:

P1 wyznaczamy $x_p \equiv c^{d_{p-1}} \bmod p$,

P2 wyznaczamy $x_q \equiv c^{d_{q-1}} \bmod q$,

P1 wykorzystując twierdzenie chińskie o resztach znajdujemy takie x, że

$$\begin{cases} x \equiv x_p \equiv c^{d_{p-1}} \bmod p, \\ x \equiv x_q \equiv c^{d_{q-1}} \bmod q. \end{cases}$$

Podwajanie punktu krzywej eliptycznej (1)

Załóżmy, że mamy dany punkt

$$P_1 = (X_1, Y_1, Z_1, X_1^2, Z_1^2, Z_1^4)$$

i chcemy znaleźć jego podwojenie w postaci

- $P_3 = (X_3, Y_3, Z_3, X_3^2, Z_3^2, Z_3^4)$ jeżeli kolejnym działaniem będzie również podwojenie,
- $P_3 = (X_3, Y_3, Z_3, X_3^2, Z_3^2, Z_3^3)$ jeżeli kolejnym działaniem będzie dodawanie.

Podwajanie punktu krzywej eliptycznej (2)

Kolejne współrzędne punktu P_3 są postaci

$$\begin{aligned} X_3 &= \alpha^2 - 2\beta, \\ Y_3 &= \alpha(\beta - X_3) - 8Y_1^4, \\ Z_3 &= (Y_1 + Z_1)^2 - Y_1^2 - Z_1^2, \end{aligned}$$

gdzie

$$\begin{aligned} \alpha &= 3(X_1^2 - Z_1^4), \\ \beta &= 2\left((X_1 + Y_1^2)^2 - X_1^2 - Y_1^4\right). \end{aligned}$$

Podwajanie punktu krzywej eliptycznej (3)

Kolejne etapy obliczeń rozdzielone pomiędzy 3 procesory

Procesor P1	Procesor P2	Procesor P3
α^2	$(Y_1 + Z_1)^2$	Y_1^2
Y_1^4	Z_3^2	$(X_1 + Y_1^2)^2$
X_3^2	$\alpha \cdot (\beta - X_3)$	Z_3^3 lub Z_3^4

Dodawanie punktu w postaci afinicznej (1)

Załóżmy, że mamy dane punkty

$$P_1 = (X_1, Y_1, Z_1, X_1^2, Z_1^2, Z_1^3),$$

$$P_2 = (X_2, Y_2)$$

i chcemy znaleźć ich sumę postaci

$$P_3 = (X_3, Y_3, Z_3, X_3^2, Z_3^2, Z_3^4).$$

Dodawanie punktu w postaci afinicznej (2)

Kolejne współrzędne punktu P_3 są postaci

$$\begin{aligned} X_3 &= \alpha^2 - 4\beta^3 - 8X_1\beta^2, \\ Y_3 &= \alpha(4X_1\beta^2 - X_3) - 8Y_1\beta^3, \\ Z_3 &= (Z_1 + \beta)^2 - Z_1^2 - \beta^2, \end{aligned}$$

gdzie

$$\begin{aligned} \alpha &= 2(Z_1^3 Y_2 - Y_1), \\ \beta &= Z_1^2 X_2 - X_1. \end{aligned}$$

Dodawanie punktu w postaci afinicznej (3)

Kolejne etapy obliczeń rozdzielone pomiędzy 3 procesory

Procesor P1	Procesor P2	Procesor P3
$Z_1^3 \cdot Y_2$	$Z_1^2 \cdot X_2$	—
α^2	$(Z_1 + \beta)^2$	β^2
$4X_1 \cdot \beta^2$	$4\beta \cdot \beta$	$2\beta \cdot Y_1$
X_3^2	—	Z_3^2
$4Y_1\beta \cdot 2\beta^2$	$\alpha \cdot (4X_1\beta^2 - X_3)$	Z_3^4

Krotność punktu i potęgowanie (1)

Załóżmy, że interesuje nas podniesienie pewnej losowej liczby do potęgi k, która ma n bitów

$$k = \sum_{i=0}^{n-1} k_i 2^i.$$

Aby rozdzielić skorzystamy z następującego prostego faktu. Jeżeli $k = r + s$, to

$$x^k = x^r x^s.$$

Krotność punktu i potęgowanie (2)

Algorytm:

P1 wyznaczamy $x_r = x^r$,

P2 wyznaczamy $x_s = x^s$,

P1 wyznaczamy $y = x_r x_s$.

Krotność punktu i potęgowanie (3)

Okazuje się, że jeśli odpowiednio dobierzemy indeks m, to liczby

$$r = \sum_{i=0}^{m-1} k_i 2^i,$$

$$s = \sum_{i=m}^{n-1} k_i 2^i,$$

pozwolą nam na szybsze wykonanie potęgowania.

Krotność punktu i potęgowanie (4)

Załóżmy, że mnożenie modularne jest β razy wolniejsze od kwadratowania, a niezerowy bit w reprezentacji liczby k pojawia się z prawdopodobieństwem α.

$$T_r = \alpha\beta m + m = \alpha\beta(n-m) + n = T_s$$

$$m(1 + 2\alpha\beta) = n(1 + \alpha\beta)$$

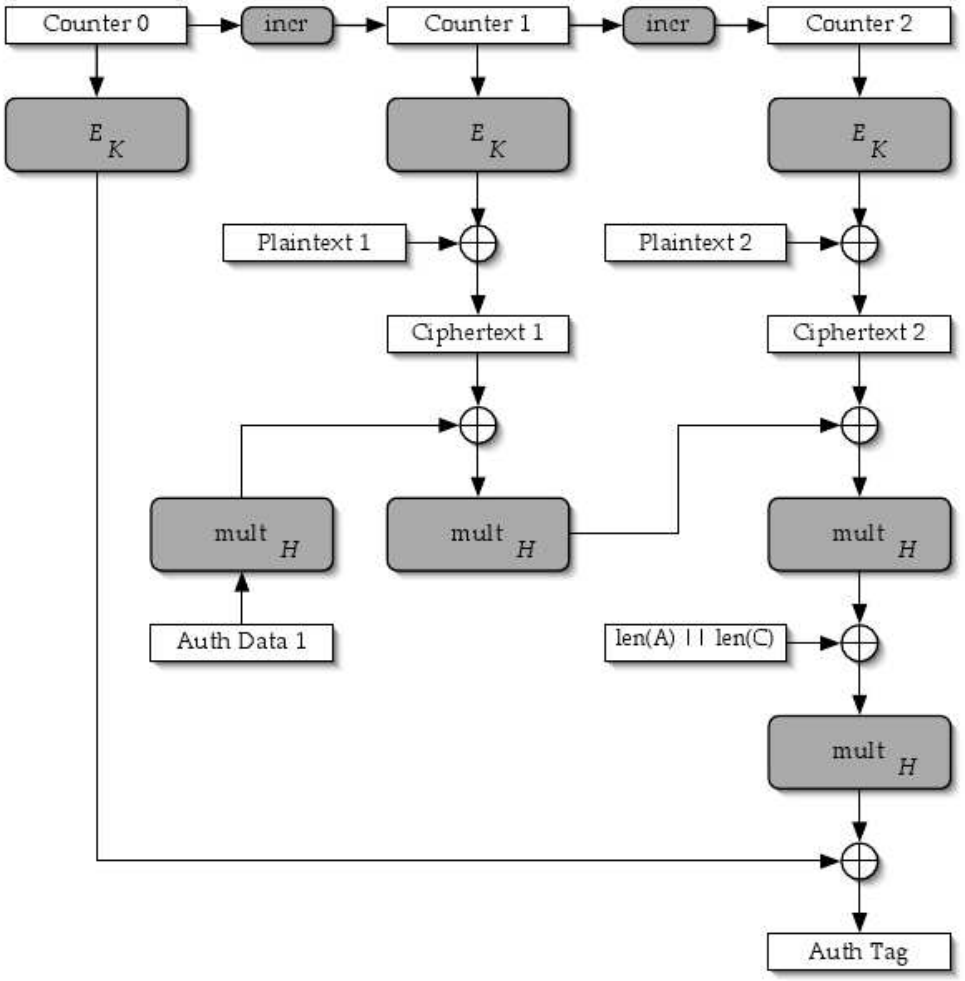
$$m = n\frac{1 + \alpha\beta}{1 + 2\alpha\beta}.$$

Algorytmy symetryczne i tryby szyfrowania

Możliwość zrównoleglenia algorytmu symetrycznego jest ściśle uzależniona od trybu pracy tego szyfru.

- ECB (Electronic Code Book),
- CBC (Cipher Block Chaining),
- OFB (Output Feedback),
- CFB (Cipher Feedback),
- CTR (Counter),
- GCM (Galois Counter Mode).

Schemat trybu GCM



Narzędzia wspierające implementację

- Biblioteka POSIX threads – standardowy interfejs obsługujący przetwarzanie wielowątkowe.
- Przemysłowy standard OpenMP – zestaw dyrektyw preprocesora, których głównym zadaniem jest zrównoleglanie pętli.
- Biblioteki Intel Ct – zestaw bibliotek, których zadaniem jest udostępnianie mechanizmów wspierających programowanie równoległe.

Podsumowanie

Idea programowania równoległego wraca znowu do łask po kilkunastu latach nieobecności. Zmiany, które nadchodzą, z całą pewnością nie ominą również implemetnacji mechanizmów kryptograficznych.