Kodowanie i bezpieczeństwo informacji

Laboratorium - lista nr 4, 29 XI

Zadanie 1 (15 pkt.) Zaimplementuj[1] "książkowy" schemat szyfrowania RSA $\langle \mathsf{Gen}(), \mathsf{Enc}(), \mathsf{Dec}() \rangle$ dla wersji RSA, w której moduł jest postaci: $n = p_1 \cdots p_k$ (d jest długością w bitach liczb p_i).

Program uruchomiony z parametrami *prog gen k d* generuje klucz prywatny i publiczny (i przechowuje je wpliku).

Zaimplementuj dwie wersje szyfrowania i deszyfrowania:

1. "tradycyjną" ,
2. wykorzystującą CRT (Chińskie Twierdzenie o Resztach).

Wersja CRT ma działać równolegle na k wątkach. Porównaj efektywność wersji "tradycyjnej" z wersją CRT dla różnych wartości k i d (sprawdź te same wartości k, d co w zadaniu 2).

(3 pkt.) Wykorzystaj swoją implementację RSA (dla różnych parametrów) z implementacją AES z poprzedniej listy (wybierz takie długości kluczy, aby to porównanie miało sens – vide *NIST SP 800-57*, str. 60: `http://csrc.nist.gov/publications/drafts/800-57/sp800-57p1r4_draft.pdf`). W tym celu potraktuj RSA jako szyfr blokowy wykorzystując np. CBC. Spróbuj zaszyfrować/deszyfrować 1, 10, 100, 1000, 10000 wiadomości o długości 128 bitów, 1MB, 10MB.

Zadanie 2 (5 pkt.) Stwórz program generujący równolegle k liczb pierwszych zadanej długości (w bitach) d. Każda z liczb ma być generowana przez osobny wątek. W tym celu (m. in.) zaimplementuj wybrany przez siebie algorytm testowania pierwszości.

Musisz wykorzystać dobre źródła losowości np. systemowe jak: */dev/random*, tudzież wbudowane w wybrany przez Ciebie język np. *java.security.SecureRandom*.

Wykonaj testy wydajnościowe dla $k = 1, \ldots, 8$ oraz $d = 256, 512, 1024, 2048, 3072, 7680$.

[1] Nie trzeba zrobić zadania 2, aby zrobić zadanie 1 – można "pożyczyć" liczby pierwsze konieczne do generowania klucza.