

Laboratorium nr 3-4

Badanie RSA

Termin wykonania: **do terminu laboratorium nr 5**

Liczba punktów: **6 + 4 + 2 dodatkowe**

PRK: T-L-3

1. Opis laboratorium

Celem laboratorium jest zbadanie kwestii związanych z bezpieczną implementacją algorytmu podpisu RSA oraz zapoznanie się z modelami bezpieczeństwa schematów podpisów cyfrowych. Zadanie obejmuje implementację, przeprowadzanie eksperymentów związanych z najbardziej znanymi atakami oraz przygotowanie analizy dotyczącej bezpiecznej implementacji RSA.

2. Materiały

- J. Katz, Y. Lindell, Introduction to Modern Cryptography, Second Edition, CRC Press 2015
- A. Menezes, P. van Oorschot, S. Vanstone, Kryptografia Stosowana, WNT 2005
- B. Schneier, Applied Cryptography, Second Edition, Wiley & Sons 2015
- <https://crypto.stanford.edu/~dabo/papers/RSA-survey.pdf>
- <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf>
- <https://tools.ietf.org/html/rfc3447>
- https://www.cryptosys.net/pki/manpki/pki_rsaschemes.html

3. Zadania do wykonania

- 1) **Zadanie 3.1** – (1 pkt + 2 pkt do lab3) Zaimplementuj podstawowy algorytm podpisu RSA (**Plain RSA**), tj. bez użycia funkcji skrótu.
 - a. *Wskazówka:* w BouncyCastle należy używać metody *compareTo* do porównywania wartości *BigInt*.
 - b. *Wskazówka:* Potęgowanie dużych liczb w ciele skończonym wymaga algorytmu potęgowania modularnego (*pow mod*).
 - c. *Wskazówka:* Dzielenie liczb w ciele skończonym to obliczanie ich odwrotności. Służy do tego algorytm *mod inverse*. Algorytm ten przyjmuje jako wejście liczbę dodatnią. W przypadku, gdy mamy liczbę ujemną należy dodać do niej moduł.

```

Konsola debugowania programu Microsoft Visual Studio
Bezpieczeństwo: 2048 bitów
Klucz publiczny:
65537
Test e*d == 1 OK
Klucz prywatny:
202645846738165524155131505979028843486327359614789143242020758653868892153293
709832716264699468454030513239724203751495351268693276874502765273768947064278
759070777512803301287453444871501149862815775918106438572798503499823881971509
019407385085128733859487989190581691785102654801700532764273094165693385759082
993922684521451646339948867860652344081410114928111115689996359692162190044398
636236172670002695664331308062404866036093479848049166082130871240989401208838
843309512976875893835411519486989512415481426326638485132762325886071487673052
505102901767170016212052957871285988951837183984325910961300523560177

Wiadomość:
Za oknem pada deszcz..
Wiadomość w bajtach:
5A61206F6B6E656D2070616461206465737A637A2E2E
Podpis
723577708614157157363038039151089314833839529591940648263184692447650307397016
553760883667853372309411954998696549921525966175211882147737937313364165285317
705357310613577764878338548144352541015110338050743045257010647339820741767773
529209214358093972332630098646359584964799475915217318090258062098057069054651
910315301518840190592760618357227815411319756077436096589404521441738594816468
188405245740242765774557879548987855201310065160307771060834337046303575032069
130579280458353768423825661910774039930508085216292242422868875266973340267754
1770132534241662296768165283445528793487151864337950776130012181998194
Weryfikacja: podpis się zgadza
C:\Program Files\dotnet\dotnet.exe (proces 31344) zakończono z kodem 0.
Aby automatycznie zamykać konsolę po zatrzymaniu debugowania, wybierz pozycję
Narzędzia -> Opcje -> Debugowanie i włącz opcję Automatycznie zamknij konsolę
po zatrzymaniu debugowania.
Naciśnij dowolny klawisz, aby zamknąć to okno...

```

2) **Zadanie 3.2** – (1 pkt) Odpowiedz na następujące pytania:

- Wiadomość o jakiej maksymalnie długości można podpisać za pomocą tego algorytmu?
- Sprawdź czas wykonania operacji podpisywania i weryfikacji podpisu dla różnych wartości modułu, tj. dla 2048, 3072, 4096, 7680 bitów.
- Umieść w pliku z odpowiedziami przykładową liczbę składającą się z 4096 bitów, jak duża jest to liczba?
- Jak ustala się wartość klucza publicznego?

3) **Zadanie 4.1** – Przeprowadź następujące eksperymenty (2 + 2 pkt do lab4)

- Eksperyment 1** – *no-message attack* – wygeneruj losowy podpis s i oblicz $m = s^e \bmod N$. Na wyjściu otrzymasz parę podpis, wiadomość (s, m) wygenerowaną pomimo braku użycia klucza prywatnego.
- Eksperyment 2** – Adwersarz wybiera dwie wiadomości m_1, m_2 takie, że $m = m_1 m_2 \bmod N$. Następnie uzyskuje podpisy dla wiadomości m_1, m_2 odpowiednio s_1 i s_2 . Adwersarz oblicza podpis dla m jako $s = s_1 s_2 \bmod N$.
- Eksperyment 3** – *Atak na szyfrowanie z wykorzystaniem RSA*. Dany jest szyfrogram **2829246759667430901779973875** (zapis dziesiętny). Został on zaszyfrowany algorytmem *PlainRSA* z kluczem publicznym $e=3$ oraz $N =$
7486374846663627918089811394557316880016731434900733973466
4557033677222985045895878321130196223760783214379338040678
2339080107477732640032376205901411740283301540121395970682
3612154294544242607436701783834990586691512046997836198600

**2240362282392181726265023378796284600697013635003150020012
763665368297013349**

Odszyfruj wiadomość (uzyskane odszyfrowane liczby dziesiętne zamień na zapis szesnastkowy i z tablicy ASCII odczytaj litery).

- d. (opcjonalny, **Bonus +2pkt!**) Eksperyment 4 – zademonstruj działanie ataku o nazwie *Håstad's broadcast attack* na schemat *PlainRSA* w którym ta sama wiadomość jest szyfrowana różnymi kluczami publicznymi trzykrotnie, przy czym każdy z tych kluczy jest taki sam i wynosi $e = 3$, a moduły N_1 , N_2 , N_3 są różne. Dokładny opis ataku znajdziesz w materiałach poniżej:
- https://cims.nyu.edu/~regev/teaching/lattices_fall_2004/ln/rsa.pdf
 - <https://crypto.stackexchange.com/questions/6713/low-public-exponent-attack-for-rsa>
 - https://en.wikipedia.org/wiki/Coppersmith%27s_attack
- 4) **Zadanie 4.2** - (1 pkt) Ulepsz schemat **PlainRSA** do wersji z załącznikiem (**RSA-FDH**), tj. algorytm ten używa funkcji skrótu (podpisywany (szyfrowany) kluczem prywatnym jest skrót z wiadomości a nie sama wiadomość). Odpowiedz dodatkowo na pytania:
- Jak dodanie funkcji skrótu wpływa na czas wykonania podpisywania i weryfikacji podpisu?
 - Jak dodanie funkcji wpływa na bezpieczeństwo? Czy któryś z powyższych ataków może się nadal powieść?
- 5) **Zadanie 4.3** - (1 pkt) Co należy zrobić, aby implementacja RSA była bezpieczna? Odpowiedz na poniższe pytania:
- Co to znaczy, że schemat podpisu jest bezpieczny? Jaka jest przyjęta definicja?
 - Wyjaśnij dlaczego ataki z pkt. 2 są możliwe do przeprowadzenia.
 - Jaka wartość klucza publicznego należy wybrać, czy e może być stałe?
 - Jaka musi być wielkość modułu, aby uzyskać bezpieczeństwo na poziomie 256 bitów (*256 bit security*) i co znaczy tak określony poziom bezpieczeństwa?
 - Co to jest za schemat RSA-PSS? Dlaczego zaleca się jego używanie zamiast schematu RSA PKCS#1.5?
 - Po co w formacie klucza prywatnego ANSI zachowuje się wartości p i q ?
 - Jakie są inne ataki na schemat RSA oprócz tych opisanych w pkt. 2?
 - Dlaczego moduł N nie może być używany więcej niż raz?