|  |  |
| --- | --- |
| **Politechnika Białostocka**  **Wydział Informatyki** | Data: 08.10.2016 |
| **Przedmiot:** Techniki zapewniania poufności w Internecie.  **Sprawozdanie nr:** 2  **Temat:** Podpis cyfrowy. Krzyew eliptyczne.  **Autor:** Maciej Ziniewicz  **Studia:** stacjonarne II stopnia, semestr 2 | **Prowadzący:**  prof. dr hab. Vyacheslav Yarmolik  Ocena: |

Spis treści

[1. Treść zadania 2](#_Toc464143197)

[2. Część teoretyczna 2](#_Toc464143198)

[3. Rozwiązanie 3](#_Toc464143199)

# Treść zadania

1. Dla danego M =23 I a=10 zaprojektuj grupę Eliptyczną..

2. Wyznacz losowe b<M dla którego dwie dodatnie liczby całkowite, a i losowo wybrane be spełniają nierówność

3. Dla a i b zaimplementuj aplikację generującą elementy(punkty grupy eliptycznej.(Sekcja 1)

4. Zaimplementuj arytmetykę(Dodawanie punktów rozróżnienia(distinct) *P =* (*xP,yP*) i *Q =* (*xQ,yQ*) oraz punkt dublujący *P=*(*xP,yP*)) w grupie krzywej eliptycznej(Sekcja 2)

5. Określ generacje punktu G z porządkiem c reprezentowanym przez dużą liczbę pierwszą.

**PS\_#4 (2 hours)**

**Problems to be solved:**

Based on the results have been obtained within the framework of the PS\_#3.

1. Implement the ***Key Exchange procedure in Elliptic Curve Group*** (see section 5).

2. Implement ***Digital Signature Algorithm ECDSA*** (see section 7).

# Część teoretyczna

RSA jest algorytmem kryptograficznym z kluczem publicznym, który umożliwia zarówno szyfrowanie jak i podpisywanie cyfrowe (weryfikacja). Podstawową zaletą kryptografii z kluczem publicznym jest to, że klucze nie muszą być przekazywane lub ujawniane nikomu, w odróżnieniu od kluczy prywatnych (tajnych), które muszą być przekazywane, gdyż ten sam klucz służy do szyfrowania i deszyfrowania danych.

Nazwa algorytmu wzięła się od pierwszych liter nazwisk twórców – Rona Rivesta, Adi Shamira i Leonarda Adlemana. Opublikowany został w 1977 roku i jest powszechnie używany do dzisiaj. RSA składa się z trzech części:

* W pierwszym etapie tworzona jest para kluczy – prywatny i publiczny,
* Druga część zajmuje się zwykłą wiadomością – używa klucza publicznego odbiorcy i na jego podsawie szyfruje wiadomość,
* Trzecia, deszyfrowanie wiadomości – odbywa się za pomocą klucza prywatnego który posiadać powinien tylko odbiorca, klucz ten powinein być strzeżony.

Obecnie już ugruntowana jest świadomość, że użycie 512-bitowych kluczy nie zapewnia wystarczającego bezpieczeństwa. RSA Laboratories zaleca obecnie stosowanie kluczy o wielkości 1024 bitów dla firm i 2048 bitów dla instytucji certyfikujących. Kilka ostatnich standardów określa 1024-bitowe minimum dla firm. Dotychczas największym kluczem RSA, jaki rozłożono na czynniki pierwsze, jest klucz 768-bitowy

Aby wygenerować klucz należy zacząć od odnalezienia dwóch odpowiednio dużych liczb pierwszych (p i q). Odpowiednio dużych oznacza że np aby stworzyć klucz 1024 bitwowy należy znaleźć dwie 512 bitowe liczby pierwsze. Uzyskane liczby pierwsze mnożymi między sobą i uzyskujemy ich iloczyn:

Dalej do wyliczenia jest funkcja Eulera φ(phi) – przypisuje ona każdej liczbie naturalnej ilość liczb względnie z nią pierwszych, nie większych od niej samej. Względnie pierwszych oznacza że największym wspólnym dzielnikiem uzyskanych liczb pierwszych jest 1 (NWD(p,q)=1). W naszym przypadku sprowadza się to do wzoru:

Po tym etapie możemy usunąć nasze wygenerowane liczby p i q, ponieważ ich uzyskanie przez osoby trzecie może umożliwić odtwrozenie klucza prywatnego.

Kolejną częścią algorytmu jest odnalezienie odpowiedniej liczby e która ma spełniać następujące warunki – być nieparzysta oraz względnie pierwsza do wyliczonej wartości φ, oraz być z przedziału 1 < e < φ. Ostatnim etapem jest obliczenie liczby d odwrotnej do e modulo t:

Klucz publiczny stanowi parę liczb e i n, natomiast klucz prywatny to para liczb e i d.

Aby zaszyfrować wiadomość należy zamienić wiadomość (m) na liczby naturalne z zakresu

0 < m < n

Następnie na tak zamienionych liczbach należy wykonać oeprację

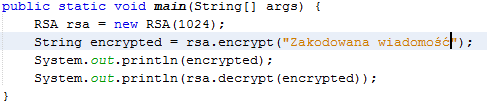
Otrzymany wynik to zaszyfrowana wiadomość.

Aby zdeszyfrować wiadomość należy na otrzymanym szyfrze (c) wykonać operację:

Otrzymany wynik jest zdeszyfrowaną wiadomością którą należy zamienić na tekst w takim sam sposób w jaki tekst był zamineniany na liczby przez nadawcę.

# Rozwiązanie

Rozwiązanie zostało zaimplementowane za pomocą język aktórego używam na codzień – Java.



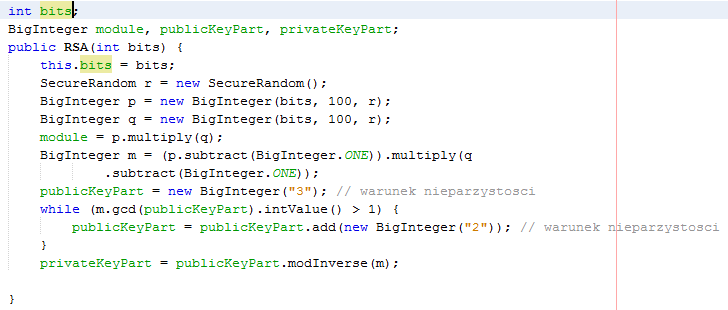
Aby uruchomić szyfrowanie trzeba stworzyć obiekt klasy RSA w konstruktorze wpisując liczbę która oznacz ilu bitowy ma być klucz. Klasa RSA posiada metody encrypt i decrypt, kolejno:

* Encrypt – szyfruje podany tekst
* Decrypt – deszyfruje podany szyfr.

Budowanie zestawów kluczy odbywa się w konstruktorze obiektu RSA.

Klasa ma zadeklarowane zmienne:

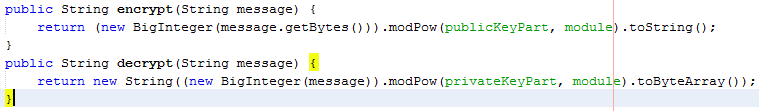
* „module” – moduł
* „publicKeyPart” – cześć klucza publicznego która z modułem będzie stanowiła klucz publiczny.
* „privateKeyPart” – część klucza prywanego która z modułem będzie stanowiłą klucz prywanty.



Algorytm generowanie klucza zgodnie z opisem początkowo generuje dwie losowe liczby pierwsze p i q. Liczby te są generowane za pomocą klasy BigInteger której paramtetry konstruktora kolejno określaja:

* Ile bitów ma miec generowana liczba w programie podana jest tam wartość bits czyli w tym przypadku 1024 bity.
* Jaka szansa że generowana liczba będzie liczbą pierwszą - ustawiona na 100%
* Obiekt generatora silnych liczb losowych.

Następnie zgodnie z algorytmem wyliczmy moduł oznaczany w opisach również jako n. Obliczany jest poprzez pomnożenie wygenerowanych liczb losowych p i q. Kolejna wartość to obliczenie - wyniku funkcji eulera ze wzoru z teorii, w programie zmienna m. Kolejnym elementem jest odnalezeinie liczby która razem z modułem będzie tworzyła klucz publiczy, liczba w teorii oznaczona jako e, tutaj w programie „publicKeyPart”. Przypisana jest do nie wartość 3 ponieważ to pierwsza większa niż 1 liczba nieparzysta. Następnie obliczany jest największy wspólny dzielnik dla tej liczby oraz dla obliczonego m, za pomocą metody gcd dostarczonej z przez obiekt BigInteger. Jeżeli największy NWD jest większy niż 1 oznacza to że liczby nie są względnie pierwsze i należy szukać innej liczby, więc do zmiennej „publicKeyPart” dodawana jest wartość dwa co daje kolejną liczbę nieparzystą, dla której obliczany jest NWD. Gdy NWD wynosi 1 oznacza to że obie liczby są względnie pierwsze. Następnie obliczana jest liczba „privateKeyPart” za pomocą metody modInverse, która zwraca odwrotność wyniku operacji modulo. W tym momencie pary („publicKeyPart”,n) oraz („privateKeyPart”,”n”) stanowią kolejno klucz publiczny i prywatny.



Metoda encrypt zamienia podaną wiadomość na tablicę bitów, na której wykonuje metodę modPow z klasy Biginteger. Metoda modPow zwraca wartość:

Zgodnie z algorytmem. Z kolei metoda encrypt wykonuje najpierw modPow na dostarczonym szyfrze, a po odkodowaniu wiadomości uzykaną tablicę bitów zamienia na tekst.