|  |  |
| --- | --- |
| **Politechnika Białostocka**  **Wydział Informatyki** | Data: 17.11.2016 |
| **Przedmiot:** Techniki zapewniania poufności w Internecie.  **Sprawozdanie nr:** 5  **Temat:** Krzywe eliptyczne. Podpis cyfrowy i wymiana kluczy.  **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. Zaimplementuj wymianę kluczy w grupie krzywej eliptycznej.
2. Zaimplementuj algorytm podpisu cyfrowego.

# Część teoretyczna

Do wymiany kluczy potrzebna jest poprzednia część zadania a konkretnie liczba pierwsza M oraz paramtery a i b, stworzona grupa eliptyczna (a,b) , oraz wybrany punkt generujący G oraz jego rząd c. Grupa eliptyczna i punkt generujący wraz z rzędem jest dostępny publicznie.

Przy wymianie kluczy mamy dwóch użytkownikó A i B.

* Użytkownik A wybiera liczbę taką że <c, liczba ta jest kluczem prywatnym użytkownika A, następnie użytkownik a generuje klucz publiczny(punkt) Pa = .
* Użytkownik B wykonuje te same czynności wybierając i generując Pb
* Następnie użytkownik A wysyła swój klucz publiczny Pa użytkownikowi B, użytkownik B wysyła klucz publiczny Pb użytkownikowi A.
* Użytkownik A generuje klucz prywatny(punkt) używając klucza publicznego użytkownika B oraz swojego wybranego klucza prywatnego K=
* Użytkownik B generuje klucz prywatny(punkt) używając klucza publicznego użytkownika A oraz swojego wybranego klucza prywatnego K=

Aby wykonać podpis wiadomości algorytmem ECDSA potrzebne są publiczne parametry: grupa eliptyczna (a,b), punkt generujący G i jego rząd c. Oraz funkcja hashująca wiadomość H(M).

Do podpisu wiadomości należy wykonać następujące czynności:

* Wybierany jest losowy numer k, taki że 1<k<c-1
* Użytkownik oblicza wartość kG=(x1,y1) czyli k-ty wynik operacji dodawania/dublowania punktu generacyjnego G, a następnie oblicza r=x1%c, gdzie r jest takie że 0≤ r ≤c-1. W przypadku gdy r=0 zaczynamy od początku.
* Następnie obliczana jest wartość działania
* Obliczany jest wynik funkcji hashującej na wiadomości M, m = H(M)
* Obliczana jest wartość s=. Jeżeli s= 0zaczynamy od początku.
* Podpis wiadomości to para (r,s)

Do weryfikacji podpisu potrzebujemy wcześniej wymienione publiczne parametry oraz popis wiadomości który chcemy zweryfikować.

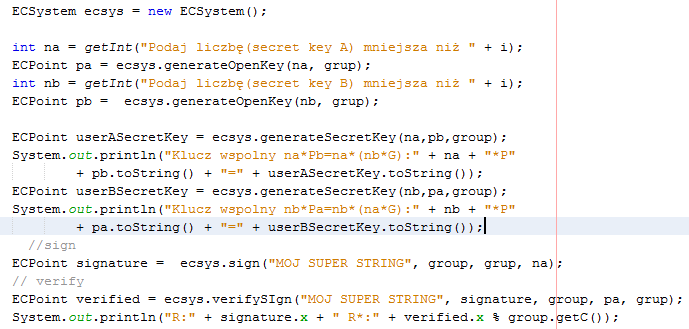
* Użytkownik oblicza wynik funkcji hashującej na wiadomości M, m=H(M)
* Sprawdzane są liczby podpisu czy należą do przedziału (r,s) należą do [1,c-1]
* Obliczana jest wartość w=
* Obliczana jest wartość oraz
* Obliczana jest wartość gdzie r\* = mod c
* Jeżeli r\* = r podpis jest prawidłowy.

# Rozwiązanie

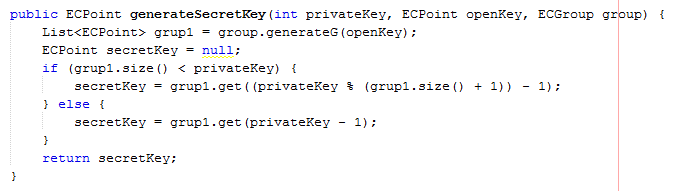
Logika aplikacji do drugiej częsci sprawozdania skłąda się z 4 metod:

* generateOpenKey(int privateKey, List<ECPoint> group)
* generateSecretKey(int privateKey, ECPoint openKey, ECGroup group)
* sign(String message, ECGroup group, List<ECPoint> generatedGPoints, int privateKey)
* verifySIgn(String message, ECPoint signature, ECGroup group,ECPoint openPaKey, List<ECPoint> generatedGPoints)

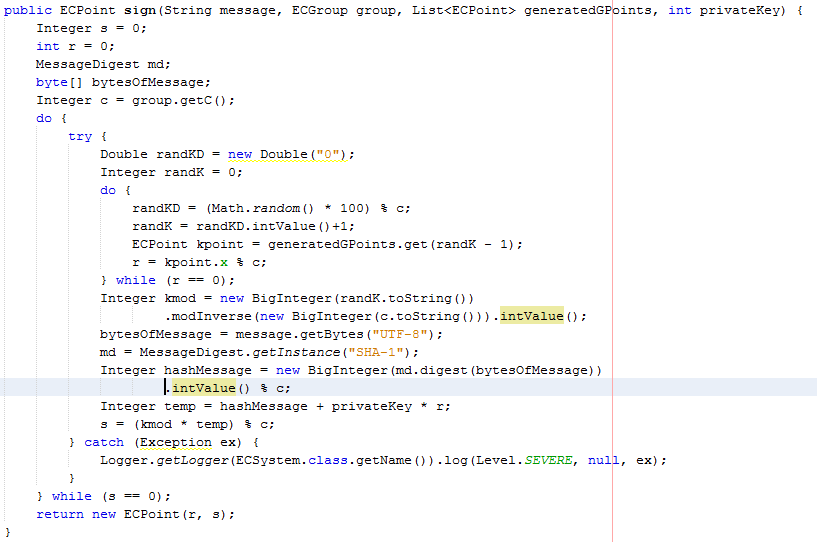
Przy wymianie kluczy dla każdego z użytkowników po wyborze swoich liczb ktore są kluczami prywatnymi wywoływana jest metoda generateOpenKey która zwraca klucz publiczny w postaci punktu.



Następnie po wymianie kluczy publicznych rozpoczyna się generowanie wspólnego klucza prywatnego dla każdego z użytkowników za pomocą metody generateSecretKey



Drugą częścią zadania było wykonanie podpisu odbywa się to za pomocą metody sign :



Zgodnie z algorytmem początkowo wybierana jest wartość 1<k<c-1 z na podstawie której pobierany jest punkt z pomocą którego obliczamy niezerową dodatnią wartość r. Kolejno wykonywane jest działanie którego wynik wraz z wynikiem funkcji hashującej wykonanej na wiadomości oraz kluczem praywatnym podpisującego daje nam niezerową dodatnią wartość s stanowiąca razem z wartością r parę, która jest podpisem.

Weryfikacja odbywa się za pomocą metody verifySign gdzie podpis otrzymany jest sprawdzany czy jego wartości należą do zakresu [1,c-1], oraz wykonywana jest funkcja hashujaca na wiadomości m. Na wartości s z otrzymanego podpisu wykonywane jest działąnie z otrzymanych wartości wyznacza ne są kolejne u1 i u2. Które następnie używane sa do wyznaczenia punktu weryfikującego, i sprawdza czy wartość x % c punktu jest równa r.

