

Dokumentacja Projektu

Temat : Elektroniczna gotówka

Spis treści

1. Opis algorytmu	2
2. Bezpieczeństwo algorytmu	3
3. Moduły aplikacji	4
4. Wygląd aplikacji.....	5
5. Działanie aplikacji	7
6. Wygląd aplikacji po wykonaniu algorytmu.....	12

Opis algorytmu

Algorytm polega na tym, że dwie strony (osoba płacąca i sprzedawca) próbują z pomocą osoby trzeciej (bank) dokonać transakcji wirtualnymi pieniędzmi. Bazuje on na algorytmie klucza publicznego RSA.

Dla opisu algorytmu przyjmujemy:

- osoba płacąca : Alice
- sprzedawca : Bob
- bank : Bank

Alice posiada konto bankowe o numerze u , a Bank przechowuje wartość v związaną z tym kontem oraz w przyjętym przez nas rozwiązaniu sklep, również posiada konto w tym samym banku.

Kroki algorytmu

1. Alice łączy się z Bank
2. Bank tworzy klucze
 - Bank losuje dwie duże liczby pierwsze p i q
 - Bank oblicza tzw. funkcję Eulera zgodnie ze wzorem $\phi = (p - 1) \times (q - 1)$ oraz liczbę $n = p \times q$
 - Bank wykorzystując odpowiednio algorytm Euklidesa znajduje liczbę e , która jest względnie pierwsza z wyliczoną wartością funkcji Eulera ϕ (tzn. $\text{NWD}(e, \phi) = 1$). Liczba ta powinna również spełniać nierówność $1 < e < n$
 - Bank oblicz liczbę odwrotną modulo ϕ do liczby e , czyli spełniającą równanie $d \times e \bmod \phi = 1$
 - Klucz publiczny jest parą liczb (e, n) , gdzie e nazywa się publicznym wykładnikiem, który można przekazać każdemu zainteresowanemu
 - Klucz tajny to (d, n) , gdzie d nazywa się prywatnym wykładnikiem. Klucz ten jest przechowywany pod ścisłym nadzorem banku.
3. Bank wysyła klucz publiczny do Alice
4. Alice losuje liczby S oraz R
5. Alice tworzy monetę zgodnie ze wzorem
$$M = S^{ec} \bmod n$$
6. Alice „pakuje” monetę do koperty
$$M * R^e \bmod n$$
7. Zapakowana moneta jest wysyłana do Banku wraz z wartością monety, która ma być odjęta z konta Alice
8. Bank podpisuje monetę
$$(M * R^e)^d \bmod n = M^d * R \bmod n$$
9. Bank wysyła podpisaną monetę do Alice
10. Alice sprawdza czy moneta została właściwie podpisana
 - Rozpakowuje monetę z koperty
$$(M^d * R) * R^{-1} = M^d$$
 - Sprawdza czy
$$M == (M^d)^e \bmod n ?$$

11. Jeśli moneta została właściwie podpisana, Alice łączy się z Bob i wysyła do Boba podpisaną monetę bez koperty
 M^d
12. Bob wysyła to co otrzymał do Banku, i Bank sprawdza czy moneta nie została już kiedyś użyta, jeśli nie wysyła żądanie do Boba o liczbę S
13. Bob wysyła żądanie do Alice o S , Alice wysyła S do Boba, a Bob następnie do Bank
14. Bank sprawdza czy:
 $S^{ec} \bmod n == (M^d)^e \bmod n$?
15. Jeśli, powyższa operacja się zgodziła, dodaje wartość monety do konta Boba i wysyła do Boba informacje o „transakcji zaakceptowanej”, którą Bob później przekazuje do Alice.

Bezpieczeństwo algorytmu

Aby złamać szyfr **RSA** należy rozbić klucz publiczny na dwie liczby pierwsze będące jego dzielnikami. Znajomość tych liczb pozwala rozszyfrować każdą informację zakodowaną kluczem prywatnym i publicznym.

Brzmi dosyć prosto. Jednakże nie ma prostej metody rozbijania dużych liczb na czynniki pierwsze. Nie istnieje żaden wzór, do którego podstawiamy daną liczbę i w wyniku otrzymujemy wartości jej czynników pierwszych. Należy je znaleźć testując podzielność kolejnych liczb.

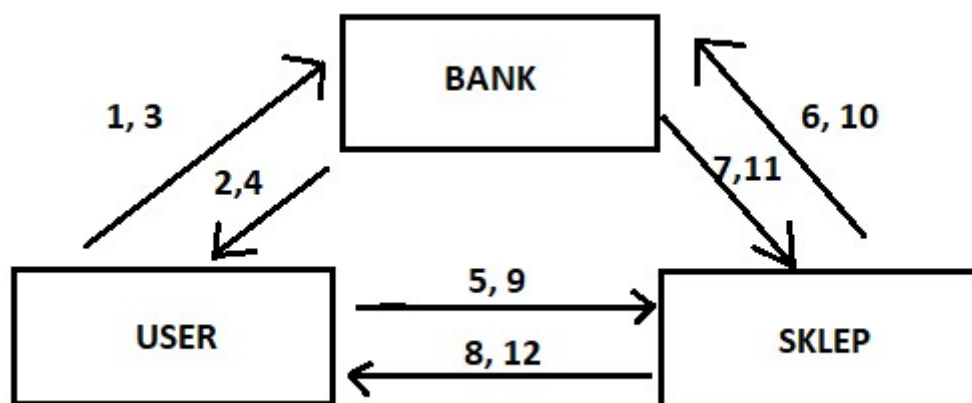
Moduły aplikacji

Projekt został napisany w języku Python. Wybraliśmy taki język, ponieważ jest on dosyć prostym językiem oraz potrzebowaliśmy zestawić połączenia między trzema modułami aplikacji.

Aplikacja składa się z trzech modułów sieciowych:

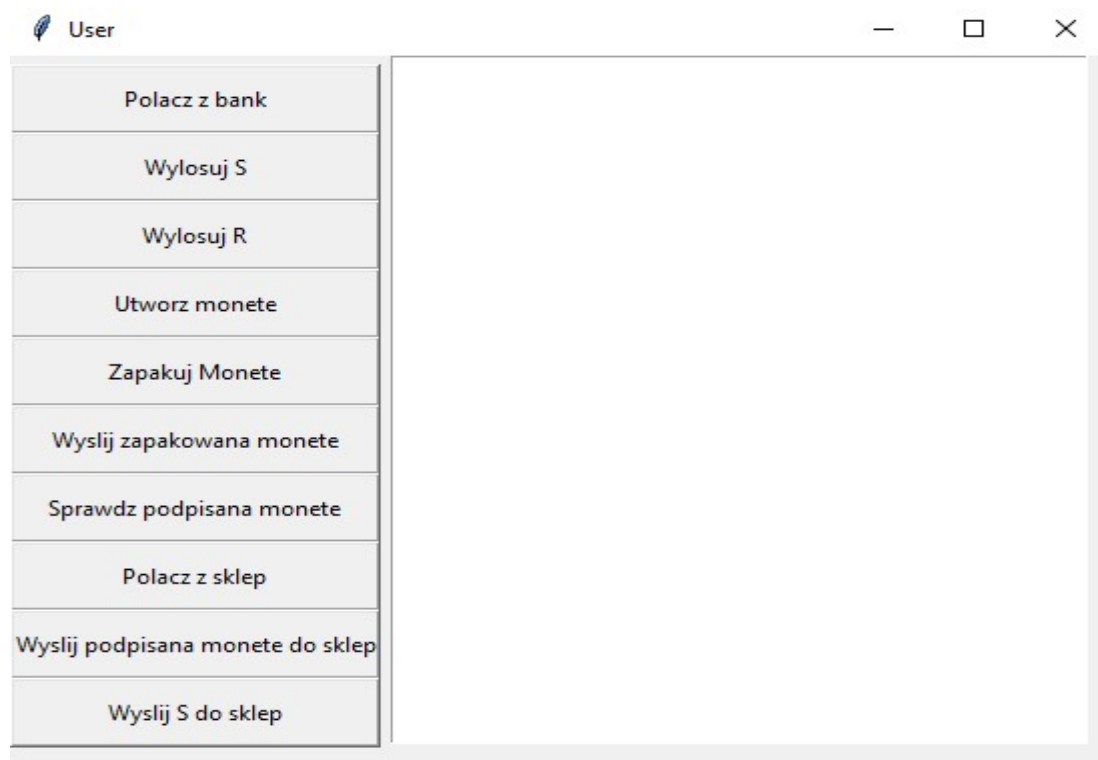
- User – klient, który realizuje funkcje przeznaczone w algorytmie dla Usera
- Bank – serwer, który realizuje funkcje przeznaczone w algorytmie dla Banku
- Sklep – serwer, który realizuje funkcje przeznaczone w algorytmie dla Sklepu

Graficzny schemat działania aplikacji

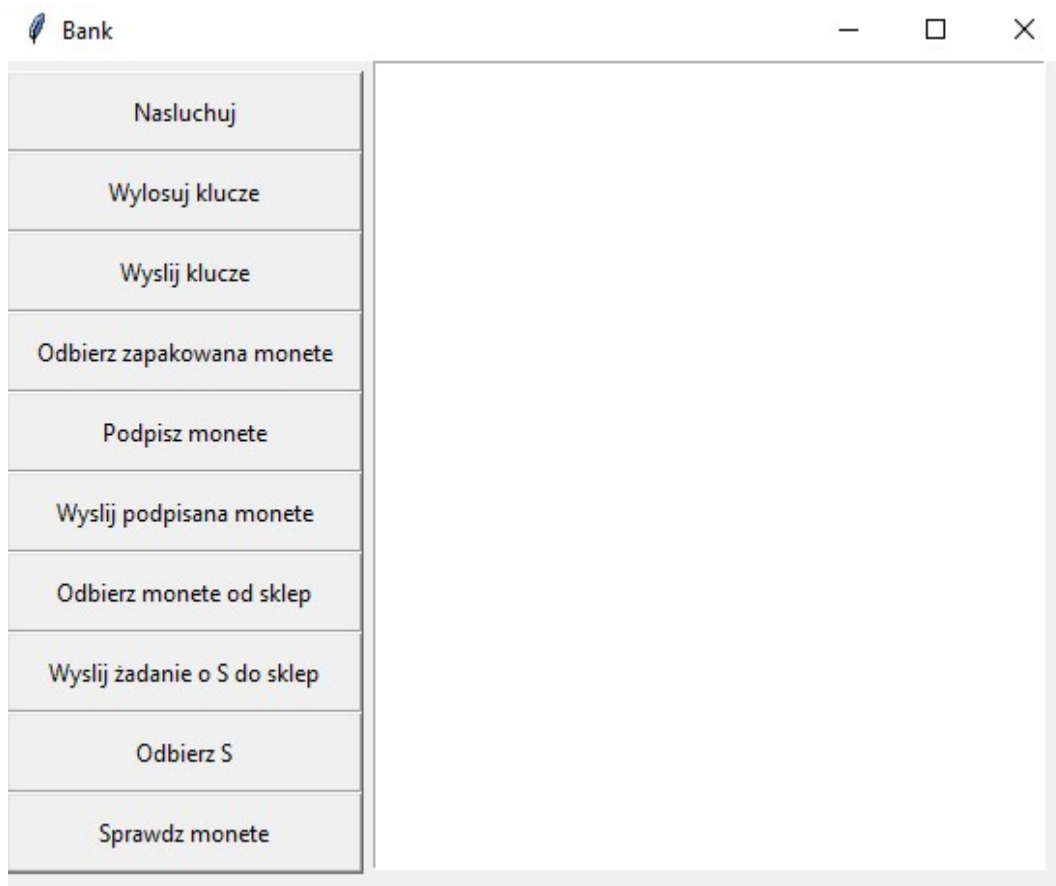


Wygląd aplikacji

➤ Aplikacja Usera



➤ Aplikacja Banku



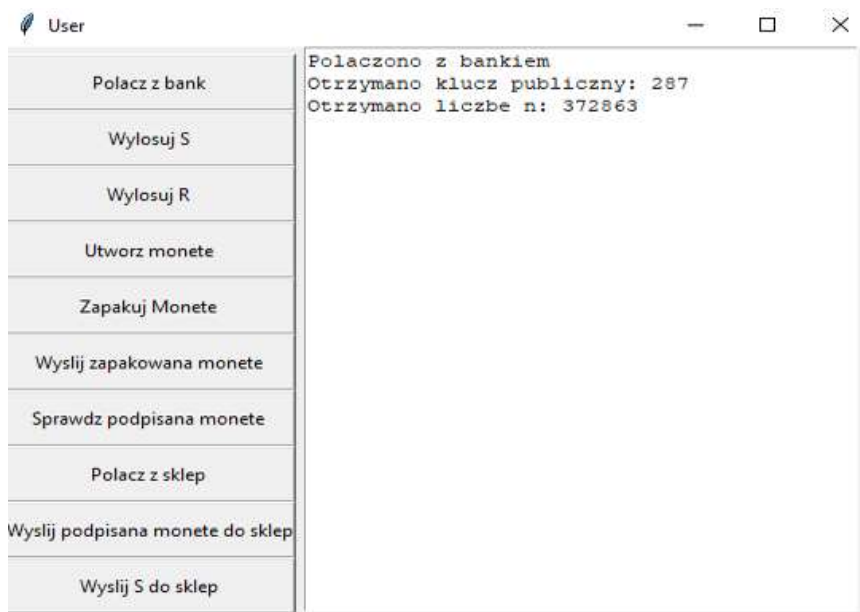
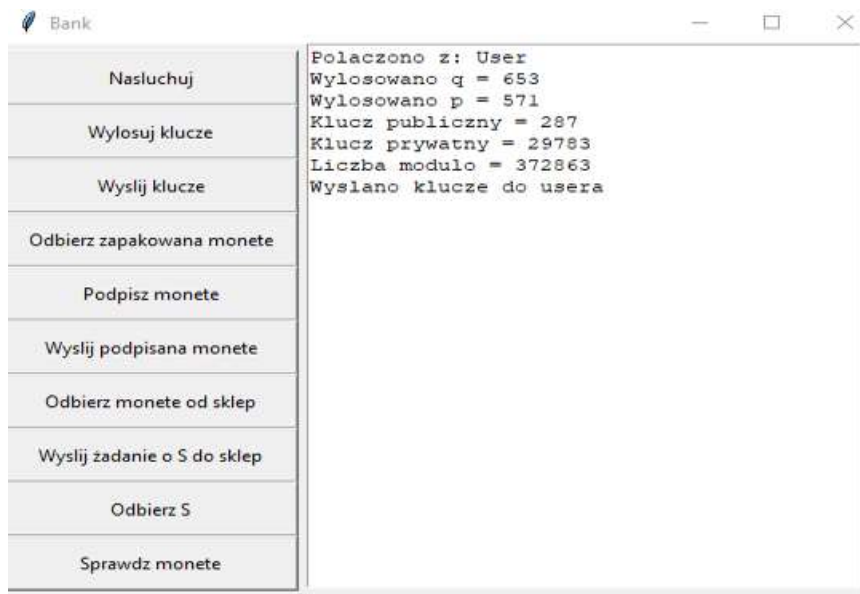
➤ Aplikacja Sklepu



Po wciśnięciu odpowiedniego klawisza (przejdzie do następnego etapu algorytmu). W prawym polu, możemy zobaczyć na bieżąco, co zostało zrobione, co zostało wysłano oraz co zostało odebrane.

Działanie aplikacji

Na początku bank nasłuchuje, czeka na zgłoszenie Usera. Gdy, User zgłosi się do banku, Bank losuje klucze, a następnie wysyła do Usera klucz publiczny p oraz liczbę modulo n .



Następnie User losuje liczbę S, a później R. Tworzy monetę, którą „pakuje w kopertę” i wysyła zapakowaną monetę do Banku, w celu podpisania, która po wykonaniu tej operacji odsyła do Usera.

User	
Polacz z bank	Polaczono z bankiem
Wylosuj S	Otrzymano klucz publiczny: 287
Wylosuj R	Otrzymano liczbe n: 372863
Utworz monete	Wylosowane S: 30
Zapakuj Monete	Wylosowane R: 25
Wyslij zapakowana monete	Utworzono monete: 93598
Sprawdz podpisana monete	Zapakowana Moneta: 194511
Polacz z sklep	Wyslano zapakowana monete do bank
Wyslij podpisana monete do sklep	
Wyslij S do sklep	

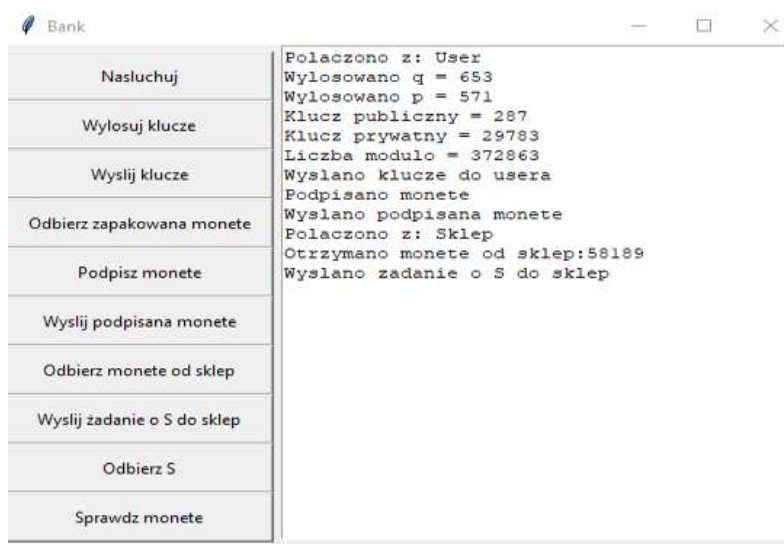
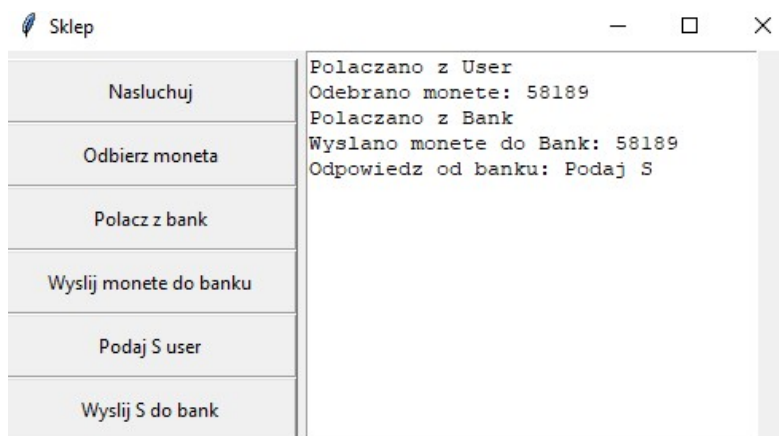
Bank	
Nasluchuj	Polaczono z: User
Wylosuj klucze	Wylosowano q = 653
Wyslij klucze	Wylosowano p = 571
Odbierz zapakowana monete	Klucz publiczny = 287
Podpisz monete	Klucz prywatny = 29783
Wyslij podpisana monete	Liczba modulo = 372863
Odbierz monete od sklep	Wyslano klucze do usera
Wyslij zadanie o S do sklep	Podpisano monete
Odbierz S	Wyslano podpisana monete
Sprawdz monete	

W kolejnym etapie, po otrzymaniu przez Usera podpisanej monety, sprawdza, czy została właściwie podpisana. Jeśli tak, łączy się ze sklepem i wysyła rozpakowaną, podpisaną monetę do sklepu.

User	
Polacz z bank	Polaczono z bankiem Otrzymano klucz publiczny: 287
Wylosuj S	Otrzymano liczbe n: 372863 Wylosowane S: 30
Wylosuj R	Wylosowane R: 25 Utworzono monete: 93598
Utworz monete	Zapakowana Moneta: 194511 Wyslano zapakowana monete do bank
Zapakuj Monete	Otrzymano podpisana monete od bank: 336136 Moneta zostala wlasciwie podpisana
Wyslij zapakowana monete	Polaczono z sklep Wyslano podpisana monete do sklep
Sprawdz podpisana monete	
Polacz z sklep	
Wyslij podpisana monete do sklep	
Wyslij S do sklep	

Sklep	
Nasluchuj	Polaczano z User Odebrano monete: 58189
Odbierz moneta	
Polacz z bank	
Wyslij monete do banku	
Podaj S user	
Wyslij S do bank	

Następnie Sklep łączy się z Bankiem i wysyła otrzymaną przez Usera monetę. Bank sprawdza czy nie została już owa moneta wykorzystana, jeśli nie, wysyła, żądanie do Sklepu o S.



Następnie Sklep wysyła, żądanie do Usera o S. User wysyła S, a po otrzymaniu przez Sklep tej liczby, Sklep wysyła S do Bank.

User	
Polacz z bank	Polaczono z bankiem Otrzymano klucz publiczny: 287 Otrzymano liczbe n: 372863
Wylosuj S	Wylosowane S: 30 Wylosowane R: 25
Wylosuj R	Utworzono monete: 93598 Zapakowana Moneta: 194511
Utworz monete	Wyslano zapakowana monete do bank Otrzymano podpisana monete od bank: 336136 Moneta zostala wlasciwie podpisana
Zapakuj Monete	Polaczono z sklep Wyslano podpisana monete do sklep Otrzymano od sklep: Podaj S
Wyslaj zapakowana monete	Wyslano S do sklep
Sprawdz podpisana monete	
Polacz z sklep	
Wyslaj podpisana monete do sklep	
Wyslaj S do sklep	

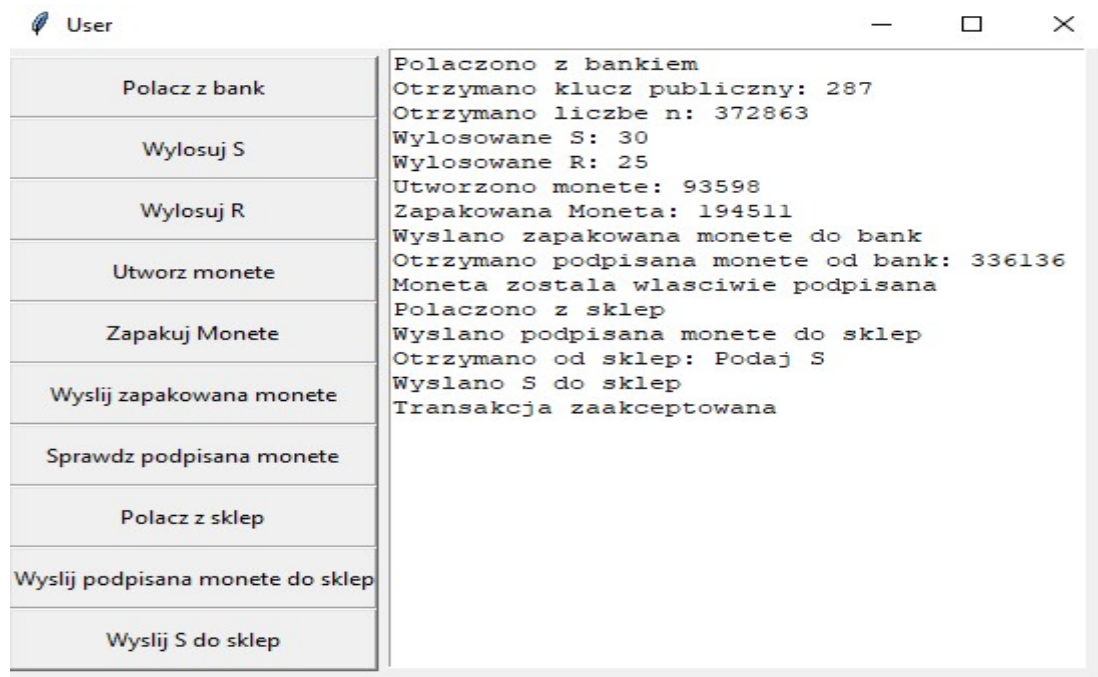
Sklep	
Nasluchuj	Polaczano z User Odebrano monete: 58189 Polaczano z Bank
Odbierz moneta	Wyslano monete do Bank: 58189 Odpowiedz od banku: Podaj S
Polacz z bank	Wyslano do User: Podaj S S od User: 30
Wyslaj monete do banku	Wyslano do Bank S: 30
Podaj S user	
Wyslaj S do bank	

Bank	
Nasluchuj	Polaczono z: User Wylosowano q = 653 Wylosowano p = 571
Wylosuj klucze	Klucz publiczny = 287 Klucz prywatny = 29783 Liczba modulo = 372863
Wyslaj klucze	Wyslano klucze do usera Podpisano monete
Odbierz zapakowana monete	Wyslano podpisana monete Polaczono z: Sklep
Podpisz monete	Otrzymano monete od sklep: 58189 Wyslano zadanie o S do sklep Odebrano S: 30
Wyslaj podpisana monete	
Odbierz monete od sklep	
Wyslaj zadanie o S do sklep	
Odbierz S	
Sprawdz monete	

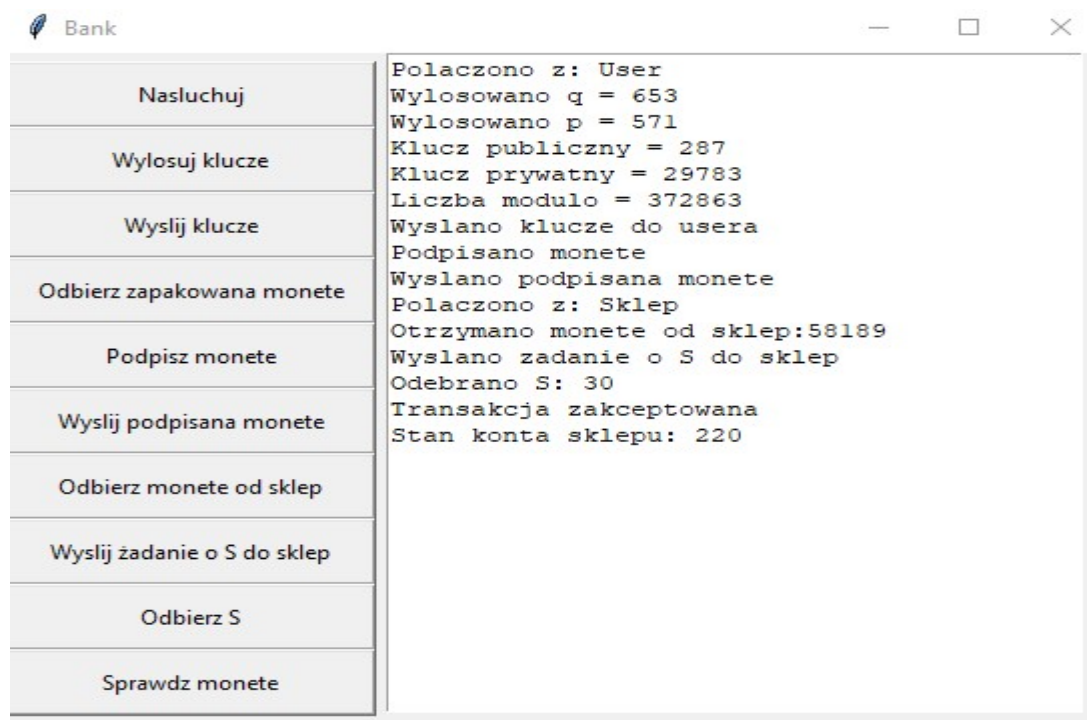
Gdy, Bank otrzyma S sprawdza czy $S^{ec} \bmod n == (M^d)^e \bmod n$. Jeśli wszystko się zgadza dodaje pieniądze do stanu konta sklepu oraz odsyła wiadomość „Ok” do sklepu, a Sklep przesyła do Usera „Transakcja zaakceptowana”

Wygląd aplikacji po wykonaniu algorytmu

➤ Aplikacja Usera



➤ Aplikacja Banku



➤ Aplikacja Sklepu

