Systemy Kryptograficzne - sprawozdanie

Created	@June 8, 2022 8:34 AM
Class	I-SKR-DP
Type	Laboratories
Materials	
Edited	@June 9, 2022 2:18 PM
Semester	6

Zadanie 1
- Protokół Diffiego-Helmana
- Szyfrowanie/Deszyfrowanie RSA
- Generacja/weryfikacja podpisu cyfrowego
- Wnioski

Zadanie 2
- Wygenerować 2048 bitowe klucze asymetryczne dla Alicji i Boba
- Utworzyć plik „testRSA1.txt” i wpisać wiadomość
- Zaszyfruj ww. wiadomość, którą Alicja chce przesłać do Boba
- Deszyfruj otrzymaną przez Boba wiadomość
- Przeprowadź ponownie eksperyment używając innych kombinacji kluczy
- Wnioski

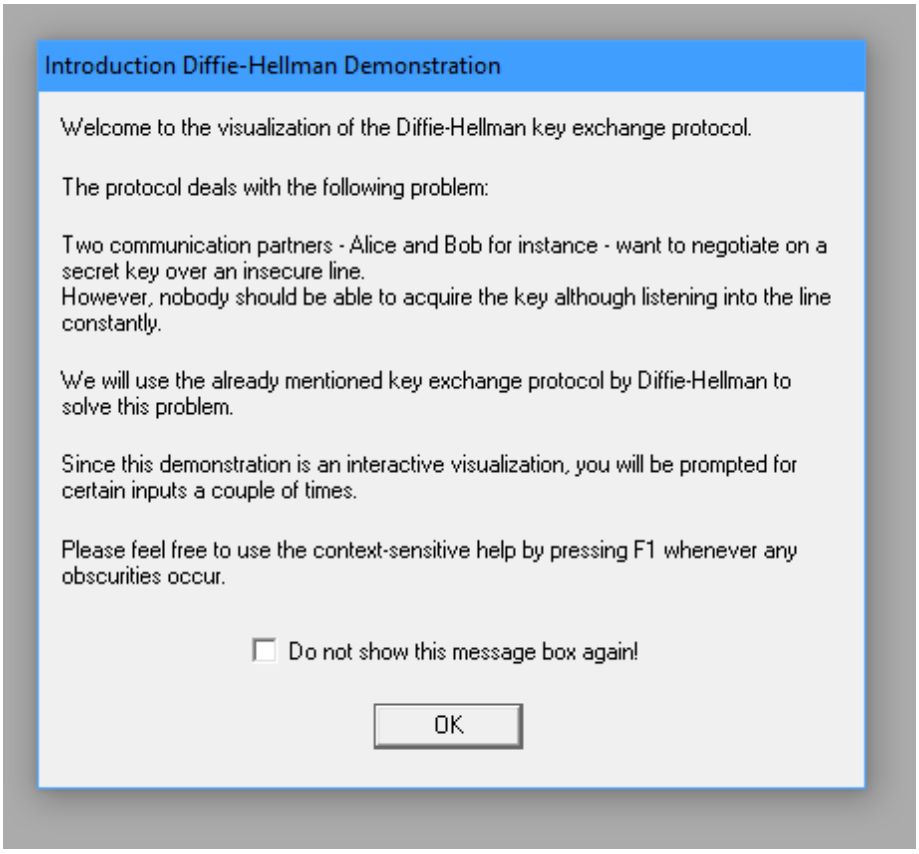
Zadanie 3
- Wygenerować 2048 bitowe klucze asymetryczne dla Alicji i Boba
- Utworzyć plik wiadomości „testRSA2.txt”
- Podpisz przez Alicję ww. wiadomość
- Zweryfikuj przez Boba otrzymany przez Boba podpis cyfrowy
- Wyciągnąć wnioski
- Przeprowadź ponownie eksperyment

Zadanie 4
- Zasymuluj w Cryptool 2
- Opis kryptosystemu, wnioski

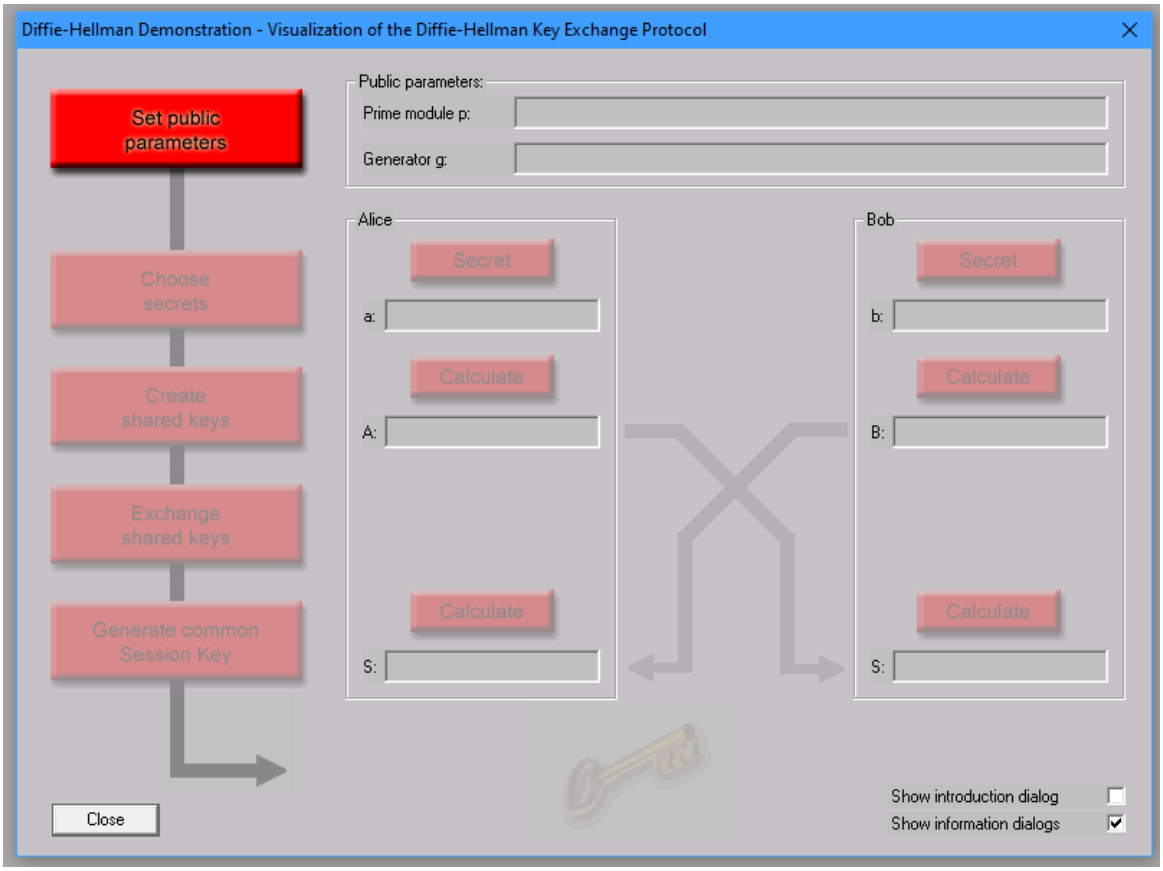
Tomasz Michalski 19012 - ID06IO1

Zadanie 1

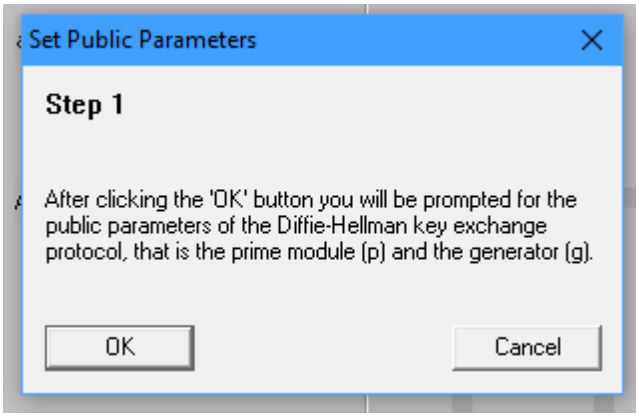
▼ Protokół Diffiego-Helmana



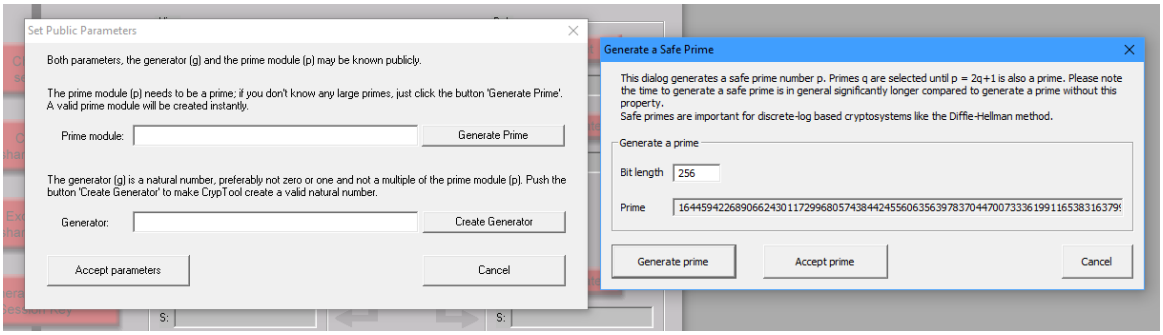
Klikam na instrukcje i akceptuje.



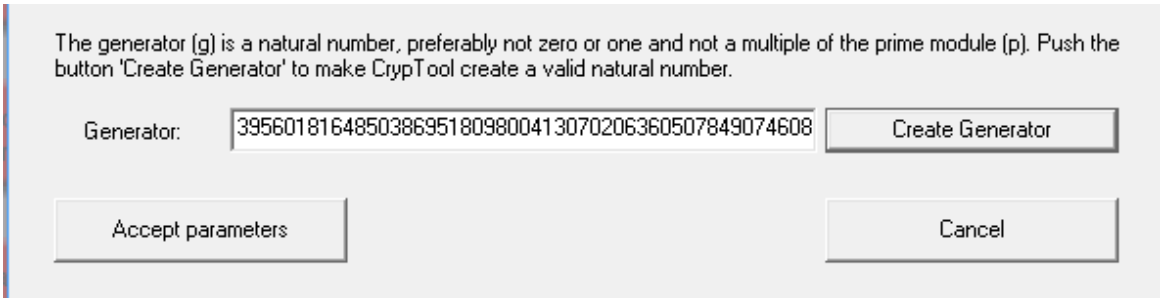
Pojawia mi się takie okienko.



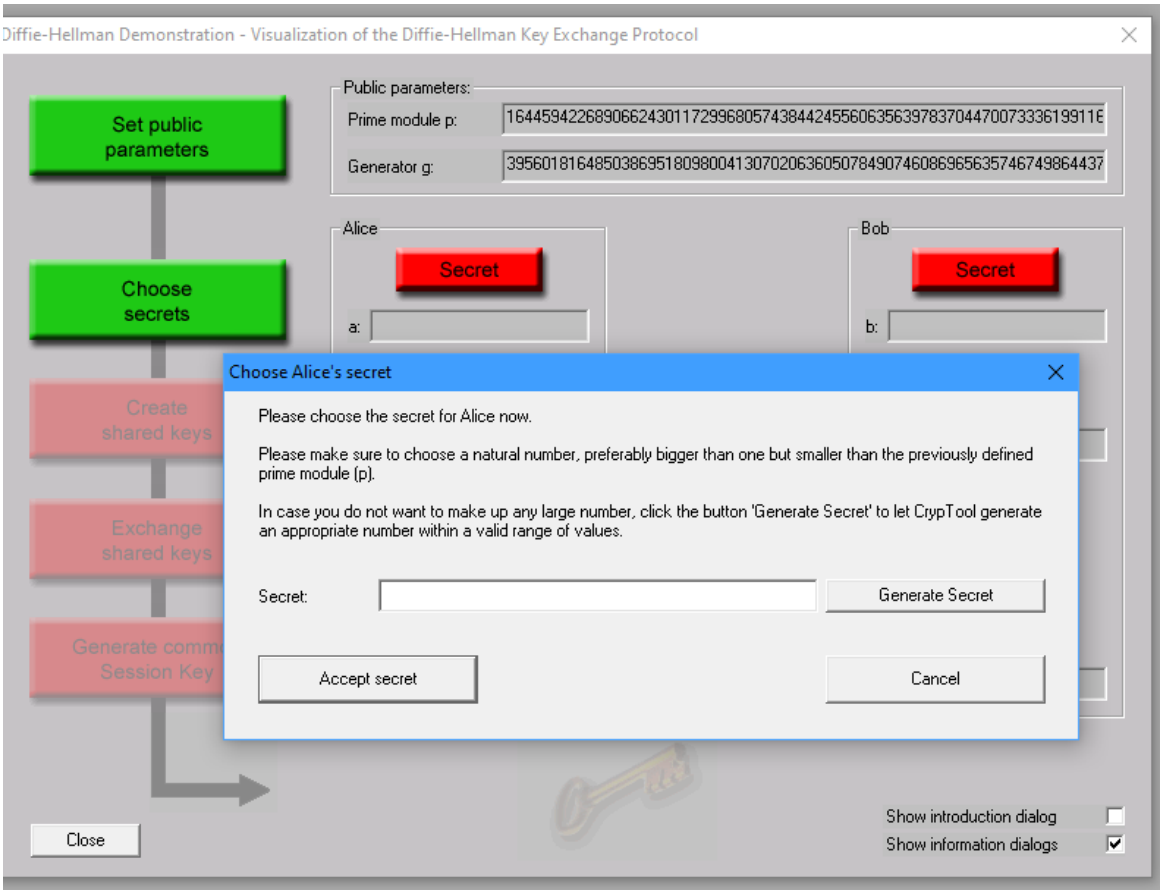
Zaczynamy od wprowadzenia parametrów publicznych. Te informacje są jawne i są uzgodnione przed wymianą informacji.



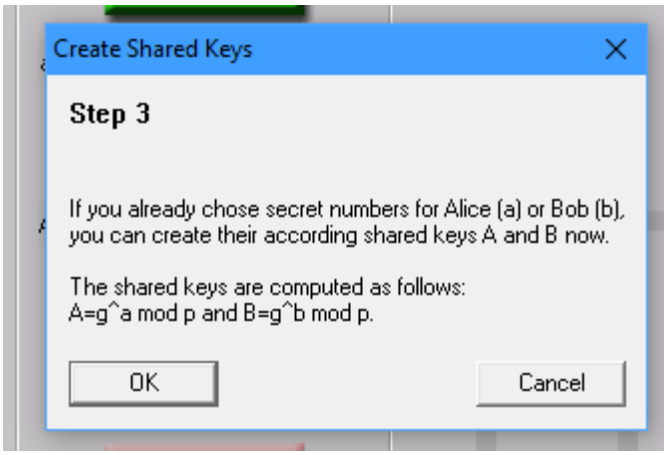
Generuję liczbę P.



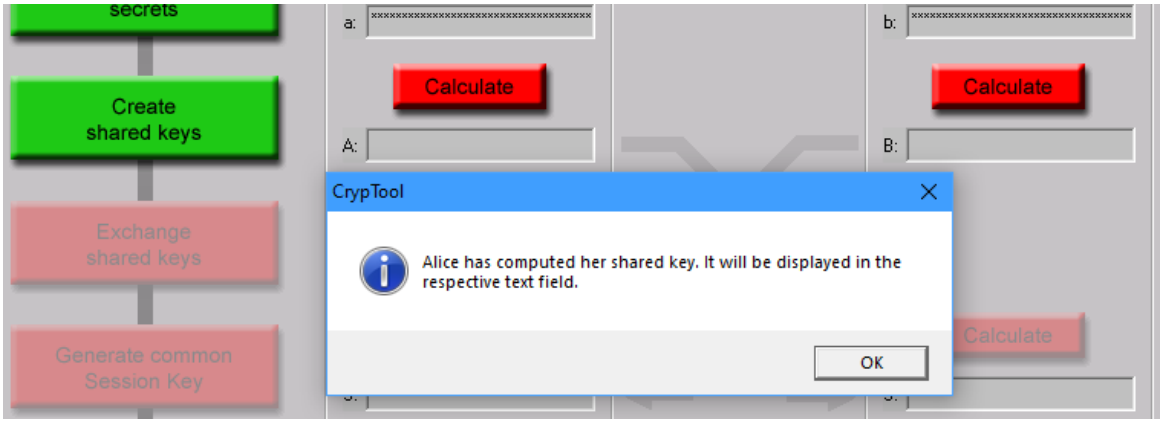
Oraz generator i akceptuję.

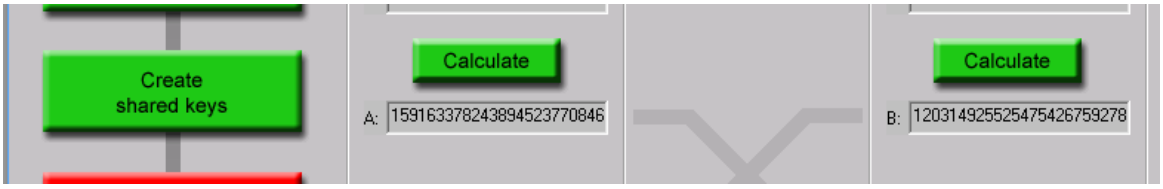


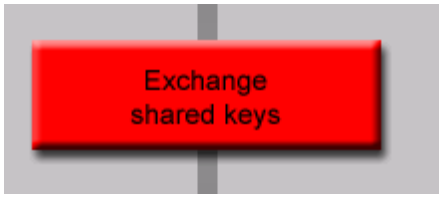
Następnie generuję sekrety dla Alicji oraz Boba.



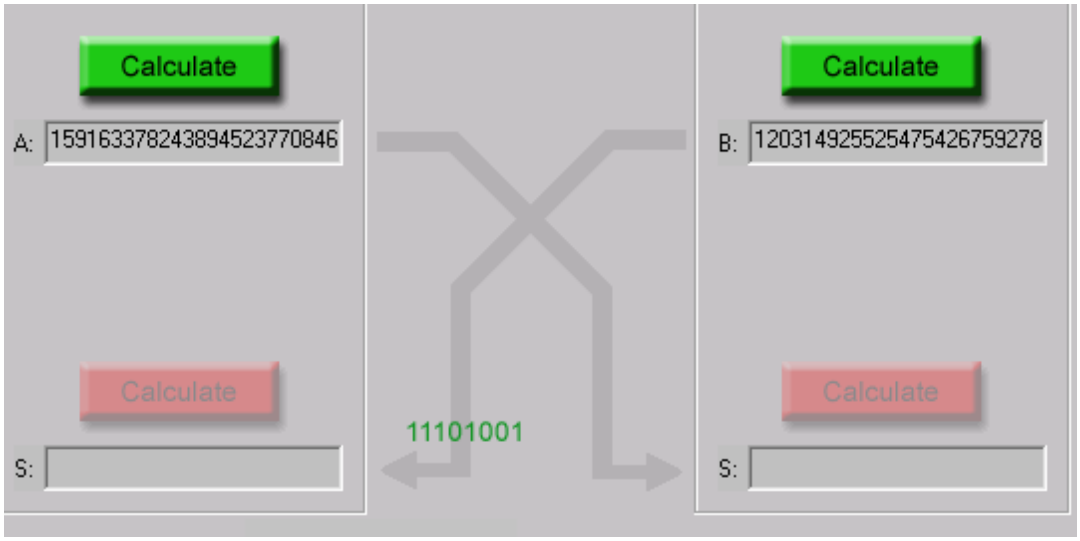
Teraz będziemy generować dzielone klucze. Jest to robione za pomocą wyżej pokazanej formuły.





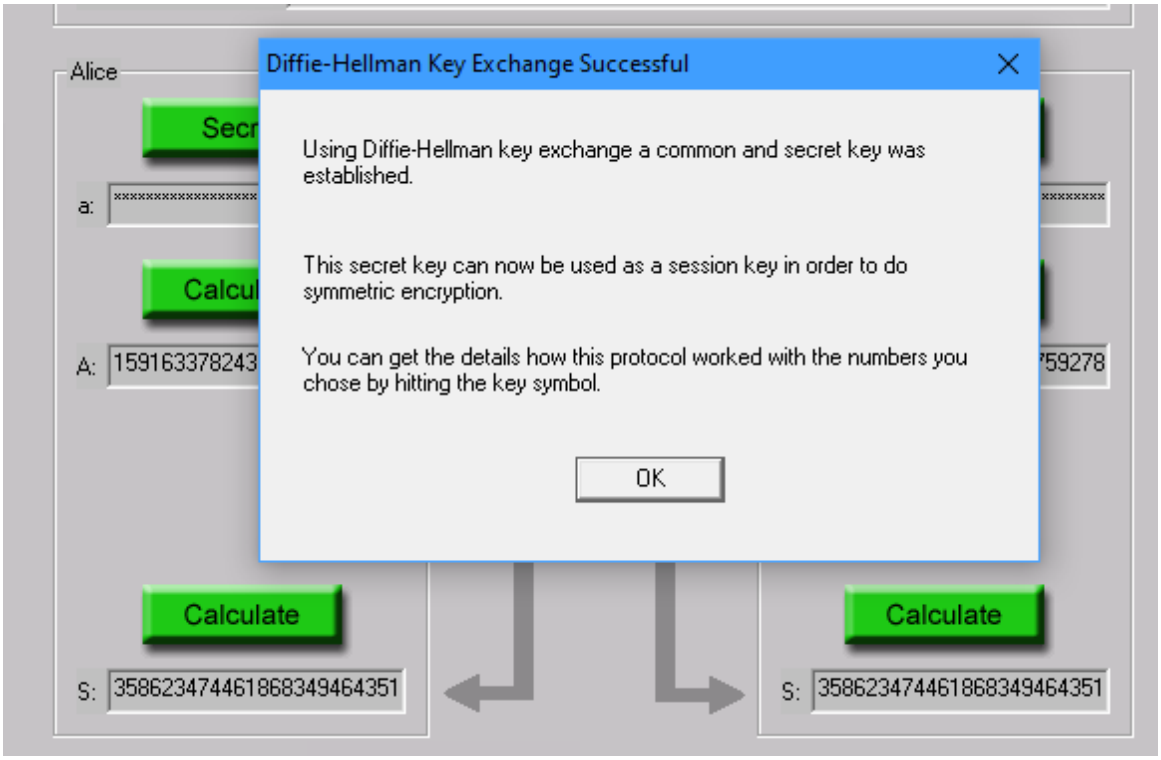


Po wygenerowaniu kluczy dzielonych możemy zająć się ich wymianą między Bobem i Alicją. Do momentu ich wspólnego użycia pozostają elementami bezpiecznymi.

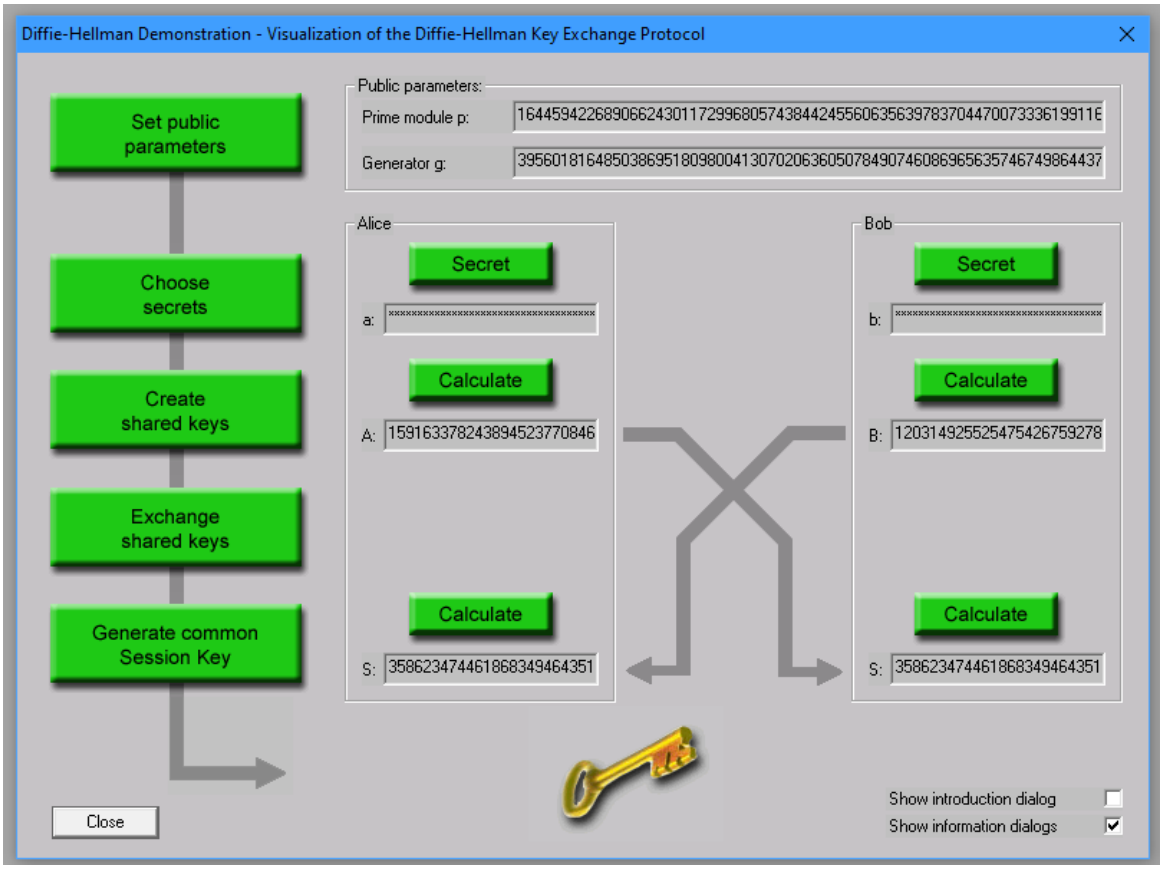


Wymianie kluczy towarzyszy animacja opisująca do którego użytkownika dany klucz wędruje. A do Boba. B do Alicji.

Następnie generujemy wspólny klucz sesji.
Robimy to za pomocą formuły zapisanej powyżej.
Klucz wspólny sesji pozwala nam na deszyfracje wiadomości od drugiej strony i jest on ten sam dla obu stron (w sytuacji gdy algorytm działa poprawnie.)



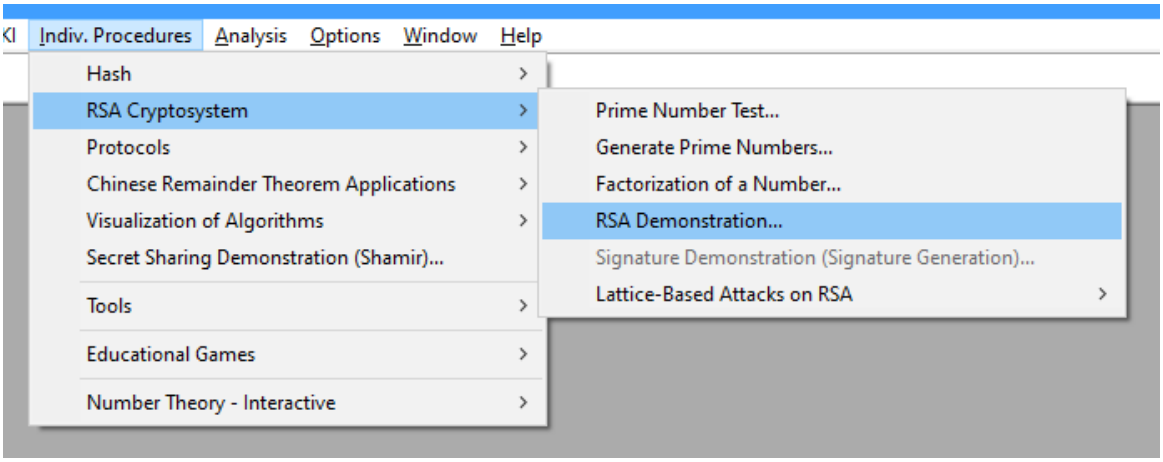
Klucze zostały wygenerowane i zauważamy, że mają tą samą wartość. Jesteśmy poinformowani, że protokół ukończył się sukcesem. Teraz taki klucz S możemy użyć do wymiany informacji pomiędzy stronami.



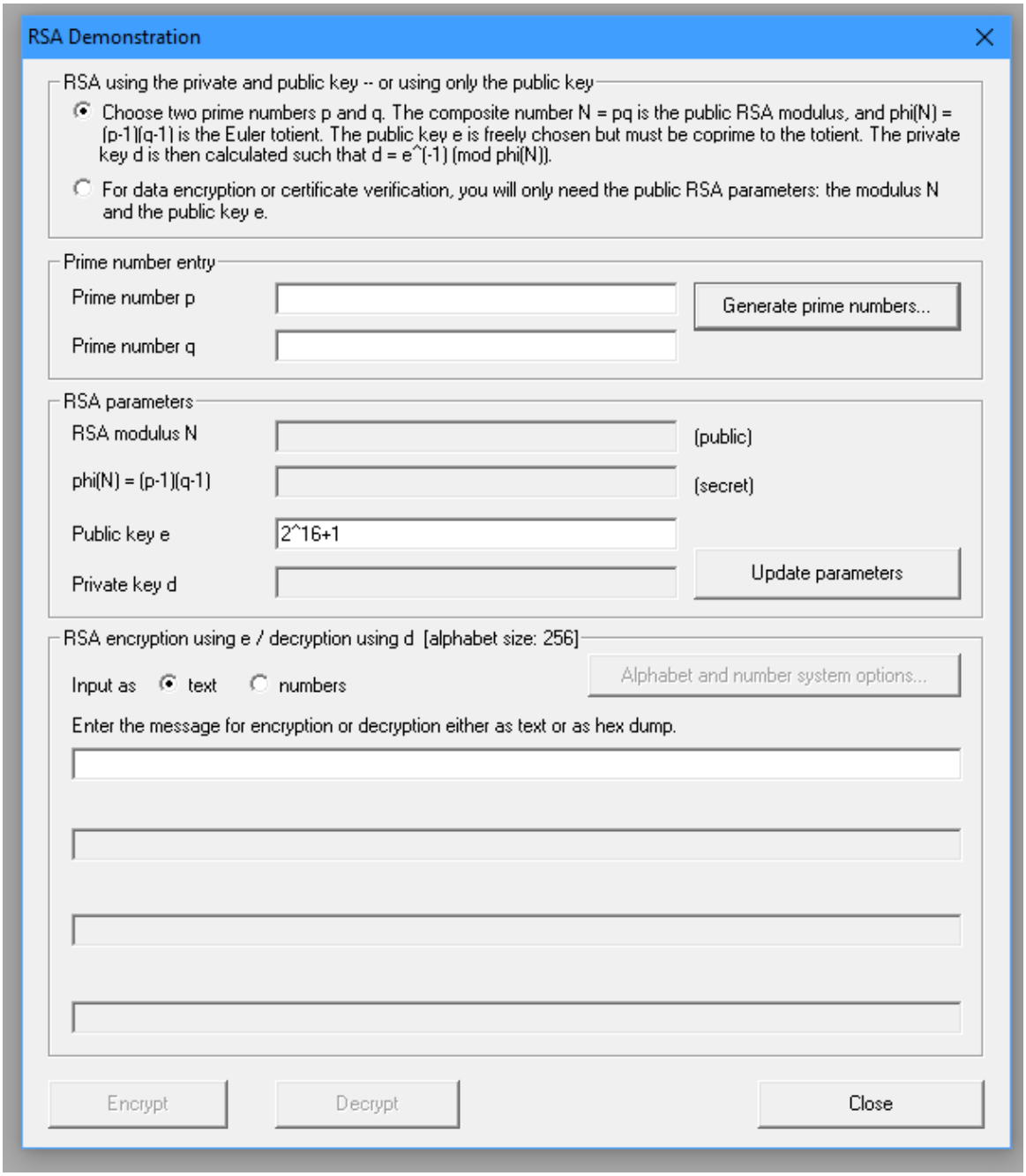
Jeszcze raz grafika ogólna pokazująca proces wymiany kluczy w protokole.

1. Generowanie publicznych wartości p i g.
2. Generowanie sekretów a i b.
3. Utworzenie wymienialnych kluczy A i B.
4. Wymiana wymienialnych kluczy A i B.
5. Generowanie klucza publicznego sesji.

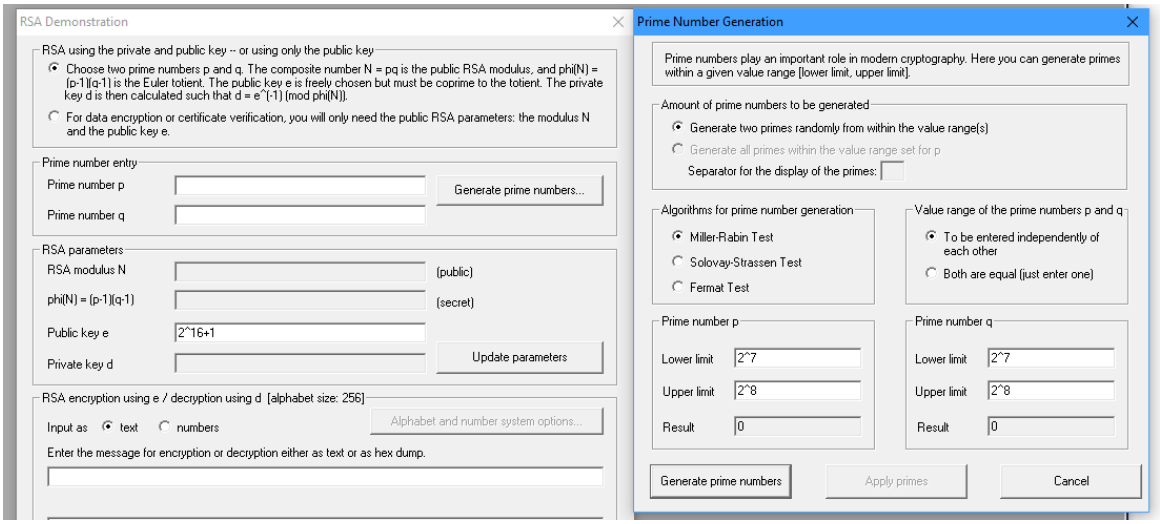
▼ Szyfrowanie/Deszyfrowanie RSA



Znajduję opcję RSA Demonstration.

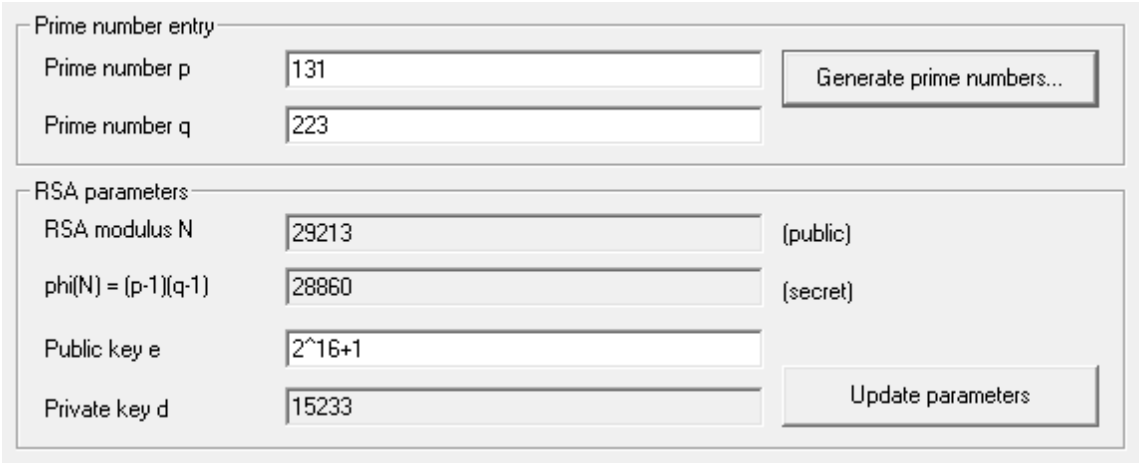


Pojawia nam się taki modal. Mamy dwie opcję, możemy wygenerować prywatny oraz publiczny klucz lub wygenerować tylko klucz publiczny. My wybierzemy opcję pierwszą.
By wygenerować oba typy kluczy potrzebujemy dwóch liczb pierwszych - p i q.

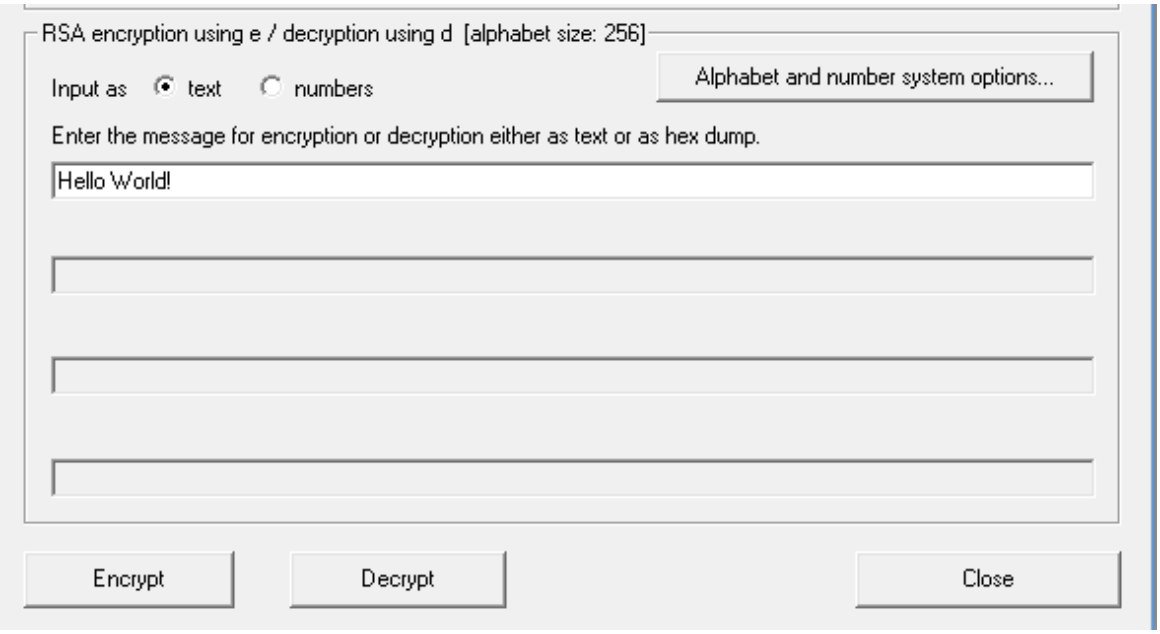


Mamy bardzo dużo opcji generacji liczb pierwszych. Algorytm generowania zostawiam domyślnie wybrany na Miller-Rabin oraz chcemy dwie niezależne liczby. Po wybraniu opcji dla p i q klikamy przycisk do wygenerowania.

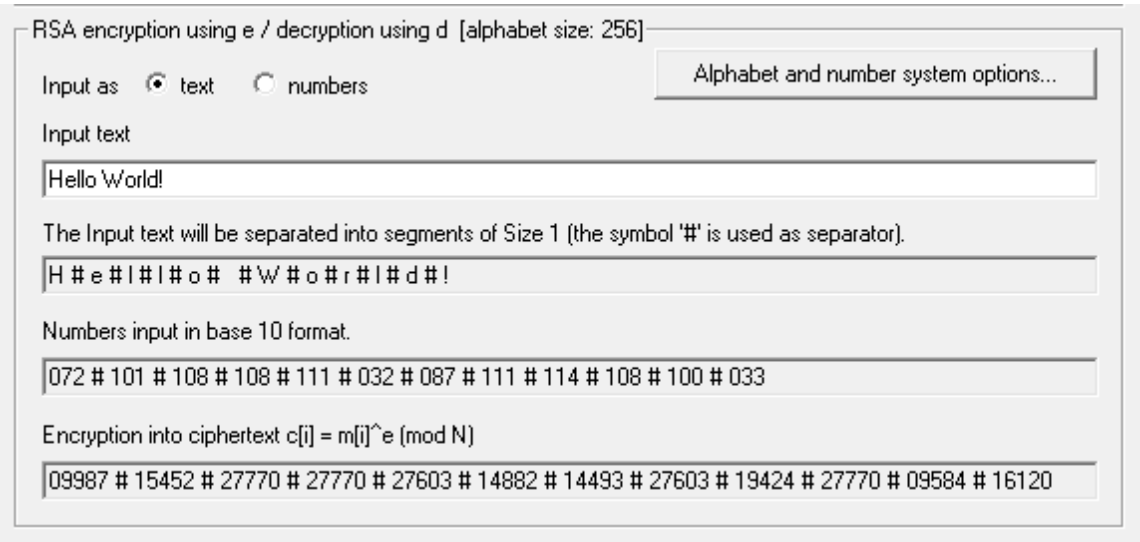
Jeżeli liczby pierwsze nam pasują klikam Apply.

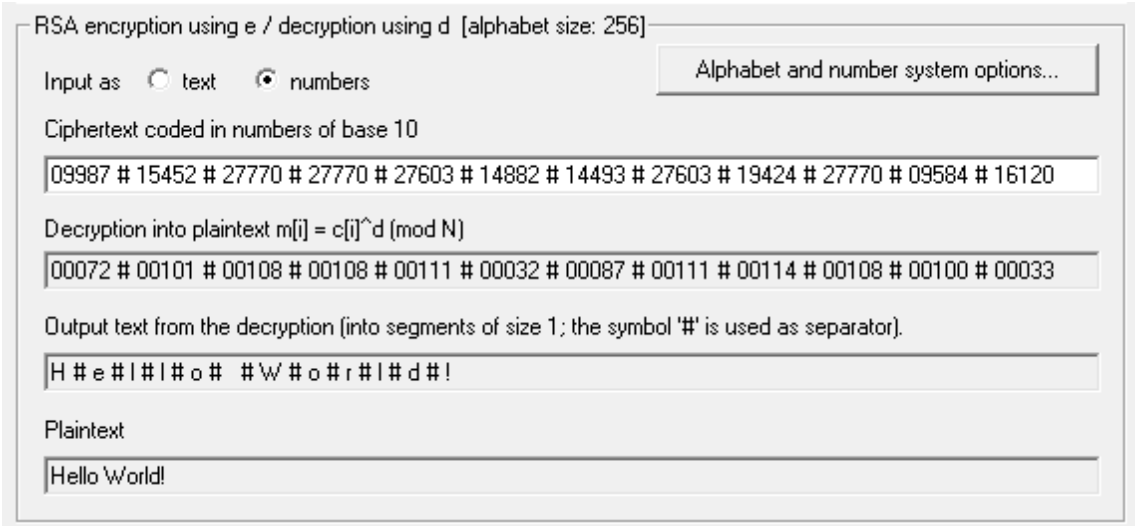


Po zaaplikowaniu liczb pierwszych program automatycznie wygenerował nam klucz prywatny i publiczny.



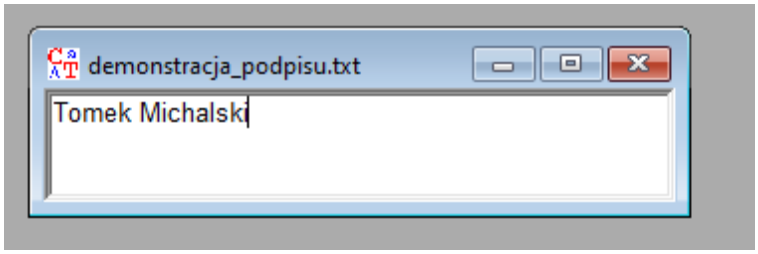
By zademonstrować poprawne działanie algorytmu możemy zaszyfrować i odszyfrować dane hasło.



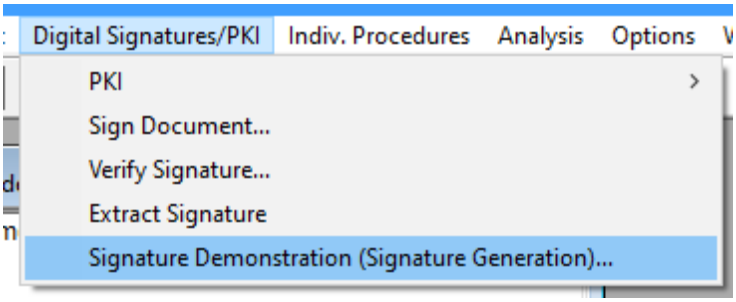


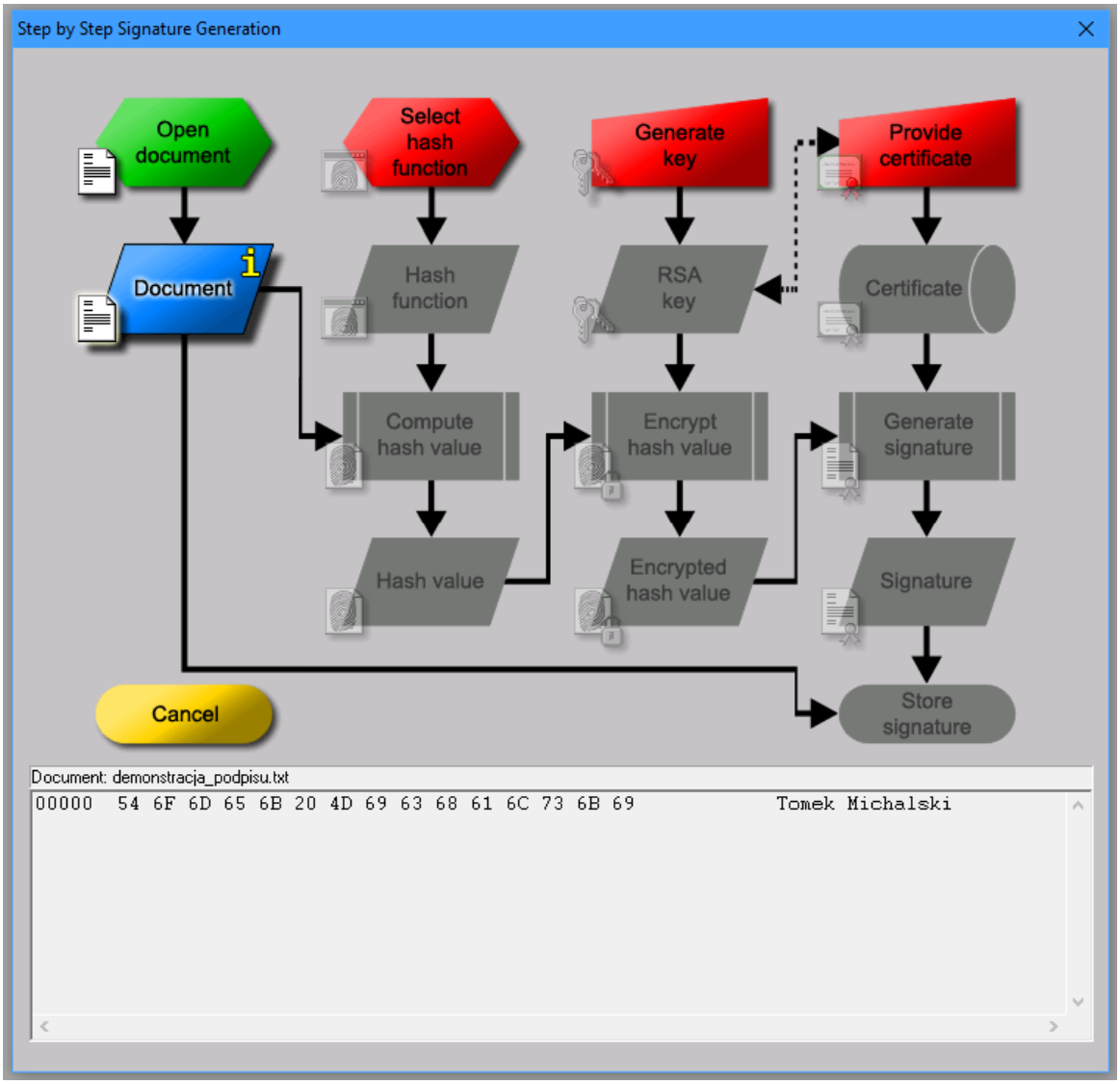
Widzimy, że algorytm działa poprawnie.

▼ Generacja/weryfikacja podpisu cyfrowego

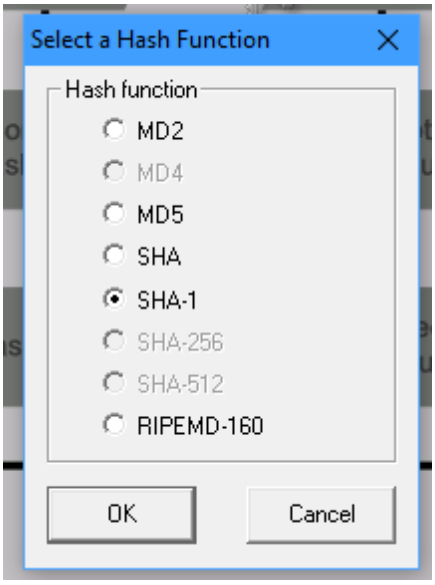


Przykład pliku tekstowego.

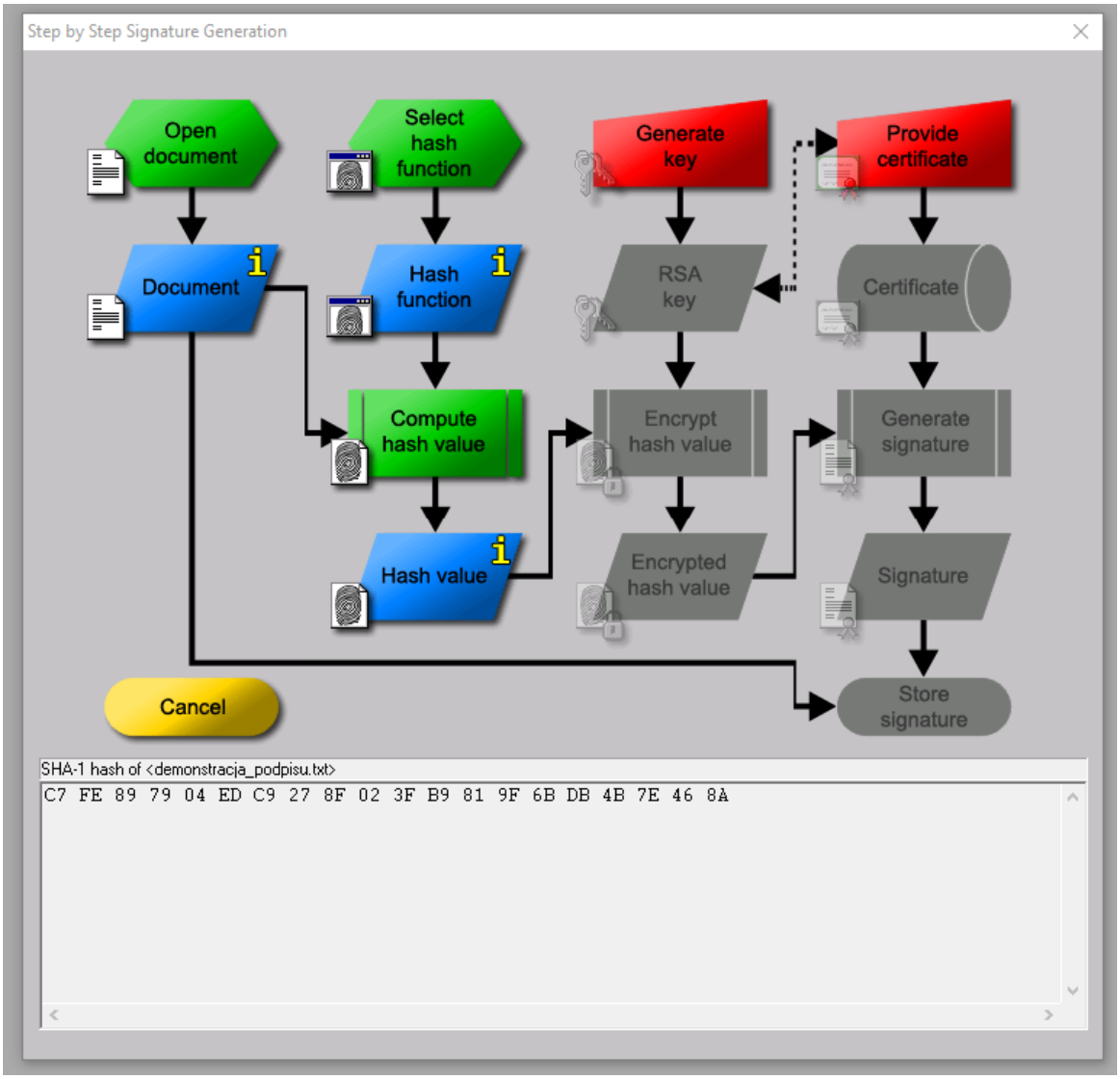




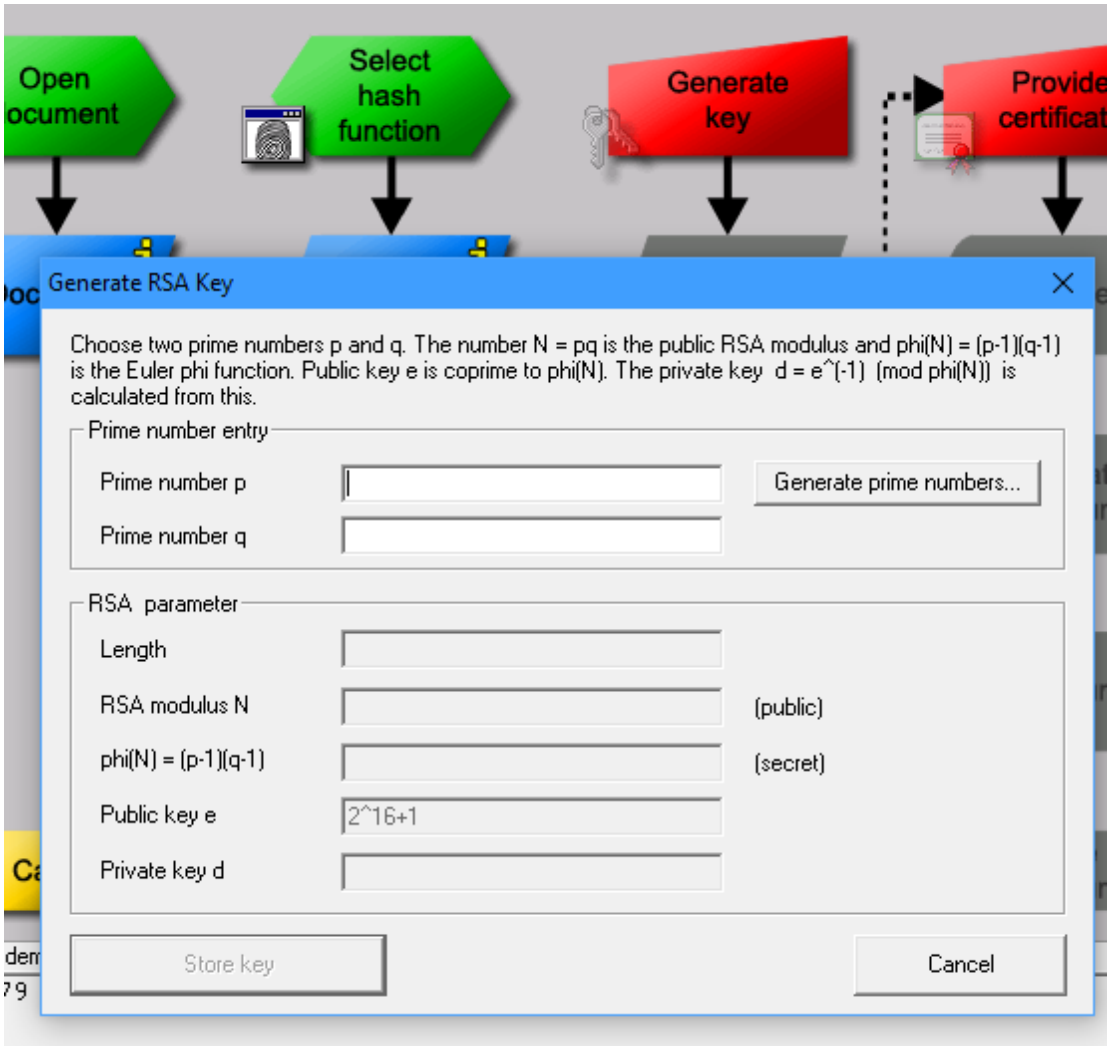
Wyskakuje nam okno “krok po kroku”. Do wygenerowania podpisu mamy kilka rzeczy do zrobienia. Musimy wybrać funkcje haszującą, obliczyć wartość hasza.



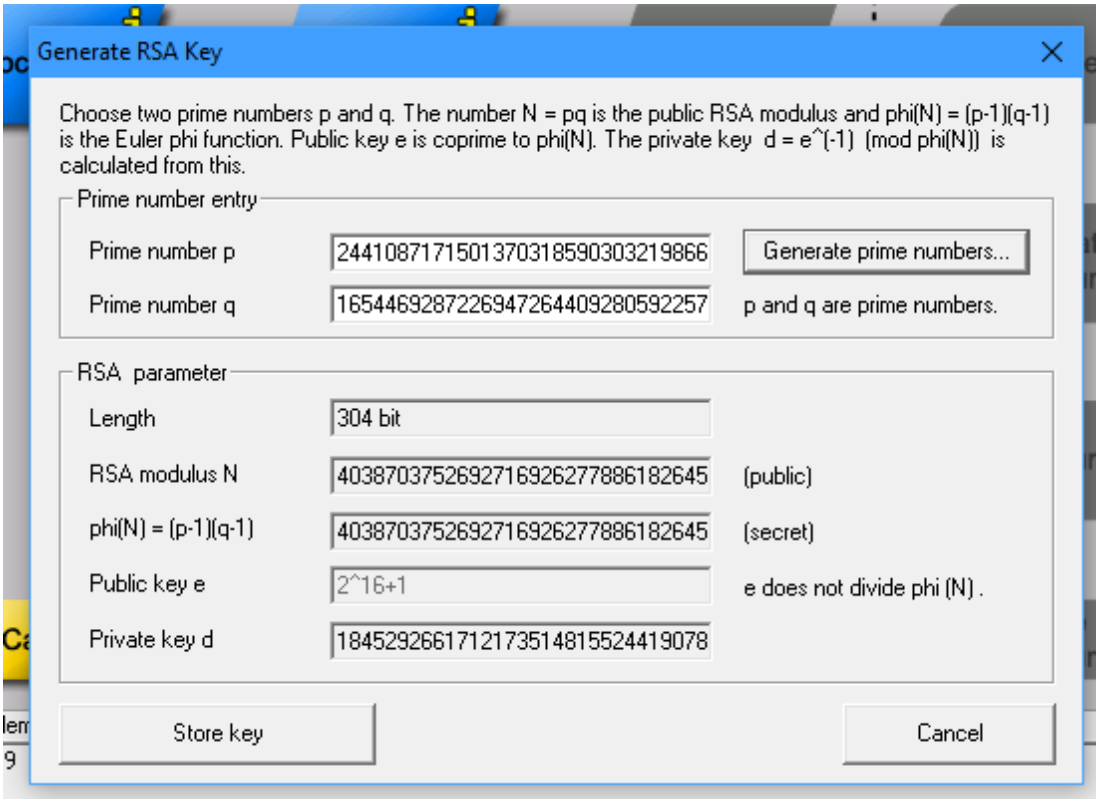
Wybiorę podstawową funkcję SHA-1.

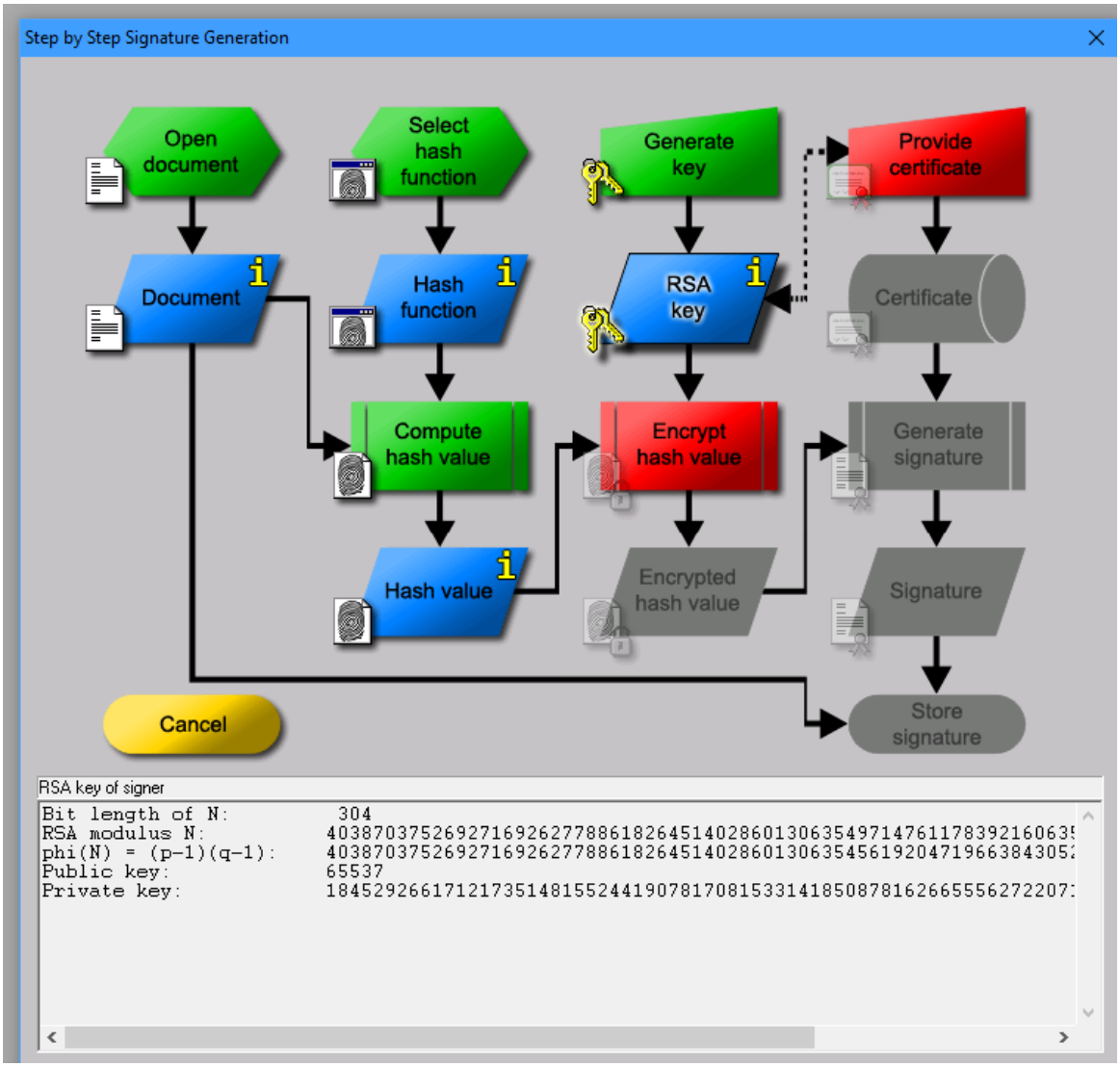


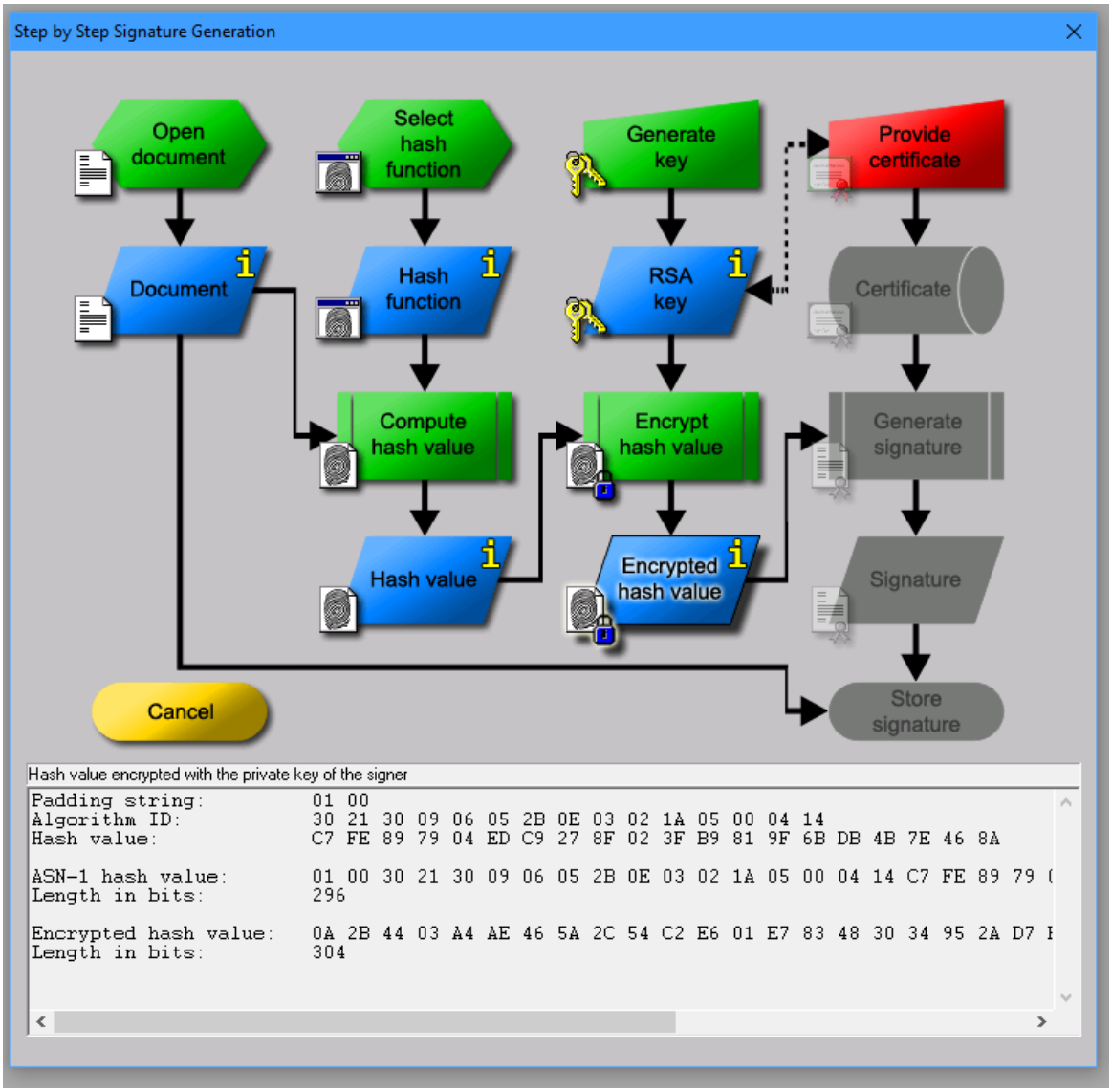
Wygenerowana wartość hasz.



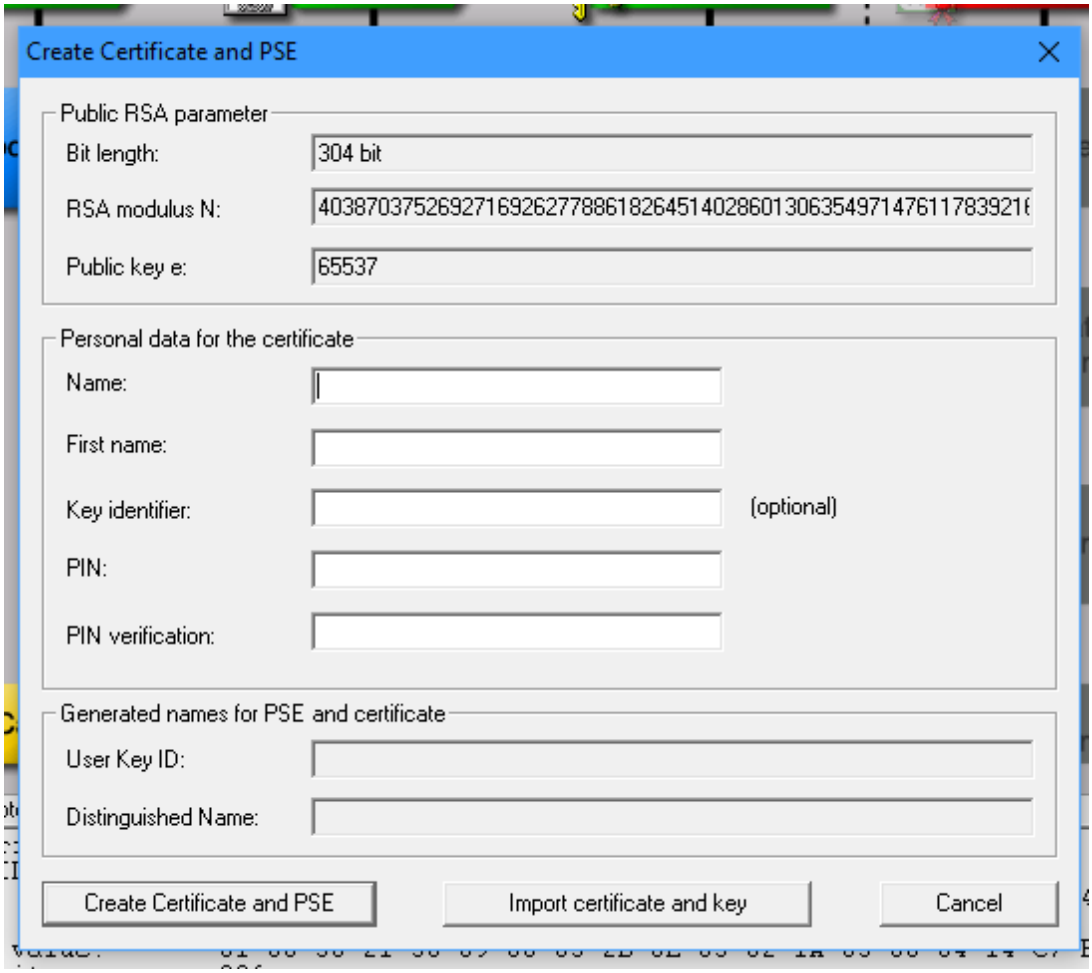
Tworzymy nasz klucz RSA. Działanie tego omawialiśmy w poprzednim zadaniu.


Generate RSA Key
Choose two prime numbers p and q. The number N = pq is the public RSA modulus and phi(N) = (p-1)(q-1) is the Euler phi function. Public key e is coprime to phi(N). The private key d = e^(-1) (mod phi(N)) is calculated from this.
Prime number entry
Prime number p
24410871715013703185903032198 66
Generate prime numbers...
Prime number q
16544692872269472644092805922 57
p and q are prime numbers.
RSA parameter
Length
304 bit
RSA modulus N
40387037526927169262778861826 45
(public)
phi(N) = (p-1)(q-1)
40387037526927169262778861826 45
(secret)
Public key e
2^16+1
e does not divide phi (N) .
Private key d
18452926617121735148155244190 78
Store key
Cancel

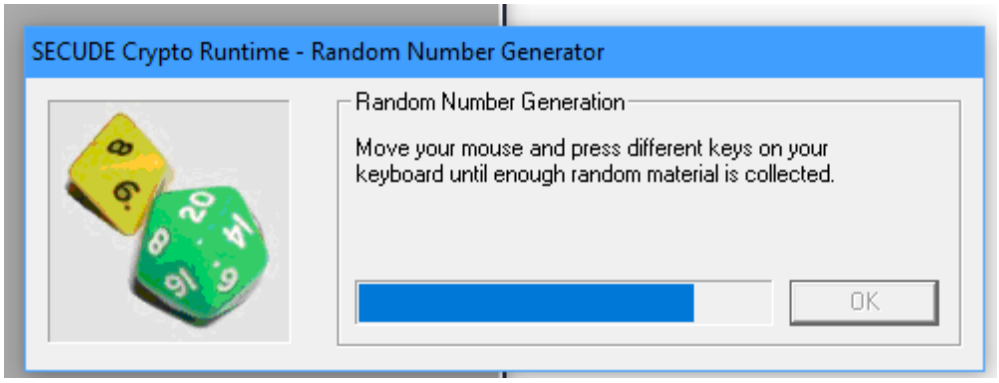

Step by Step Signature Generation
Open document
Select hash function
Generate key
Provide certificate
Document
Hash function
RSA key
Certificate
Compute hash value
Encrypt hash value
Generate signature
Hash value
Encrypted hash value
Signature
Cancel
Store signature
RSA key of signer
Bit length of N: 304
RSA modulus N:
phi(N) = (p-1)(q-1):
Public key: 65537
Private key:



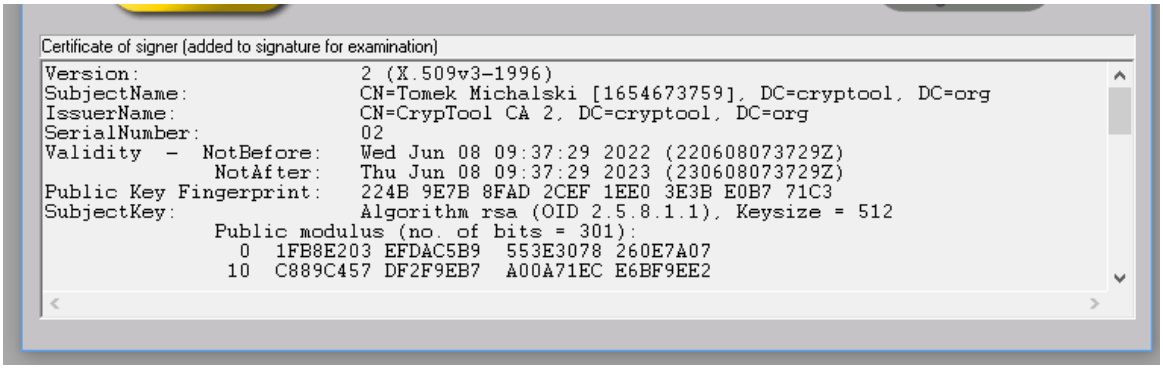
Po wygenerowaniu klucza możemy zaszyfrować naszą wartość hasz.



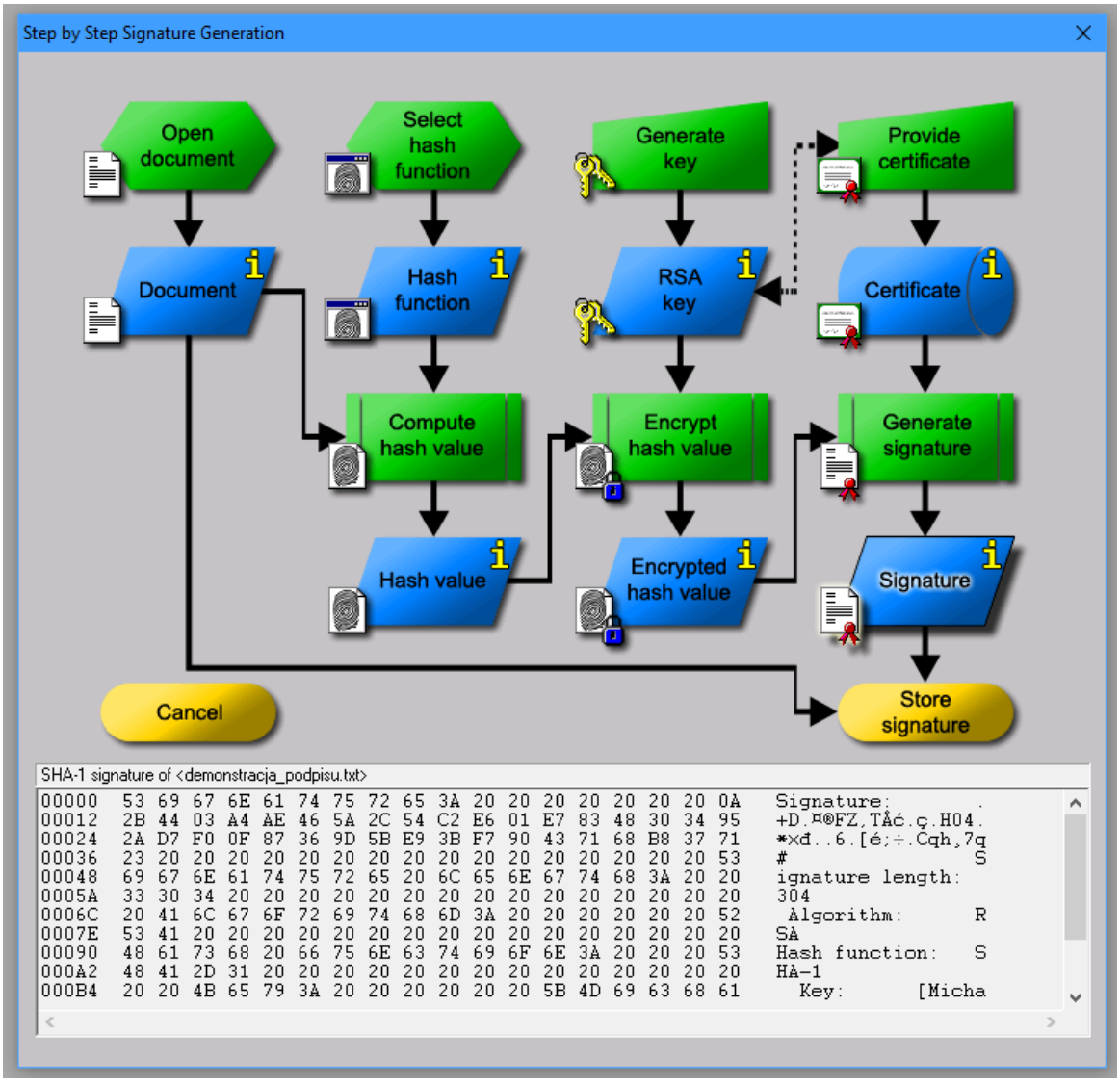
Do wygenerowania naszego certyfikatu potrzebujemy wprowadzić kilka danych. Wprowadzę swoje imię oraz losowy PIN 1111.



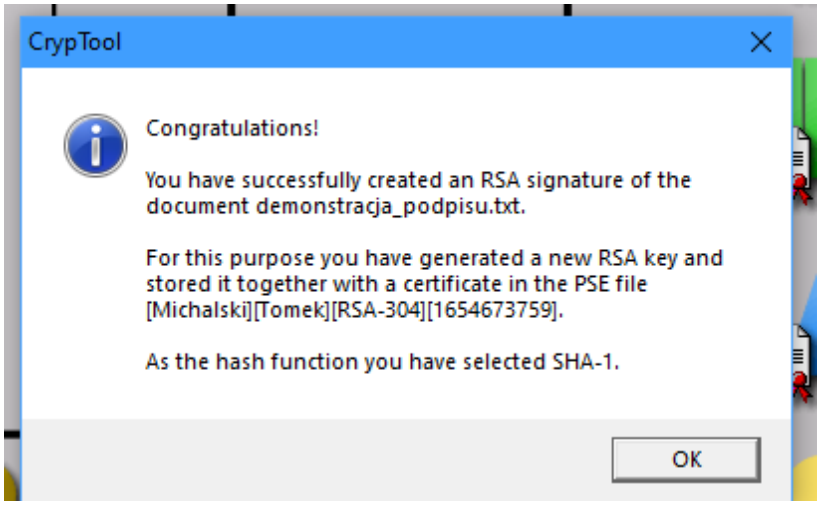
Certyfikat zostaje wygenerowany za pomocą liczb losowych.

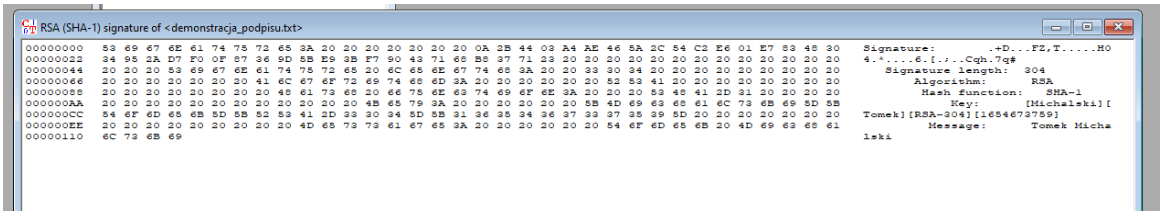


Część certyfikatu możemy zobaczyć w oknie podglądu.

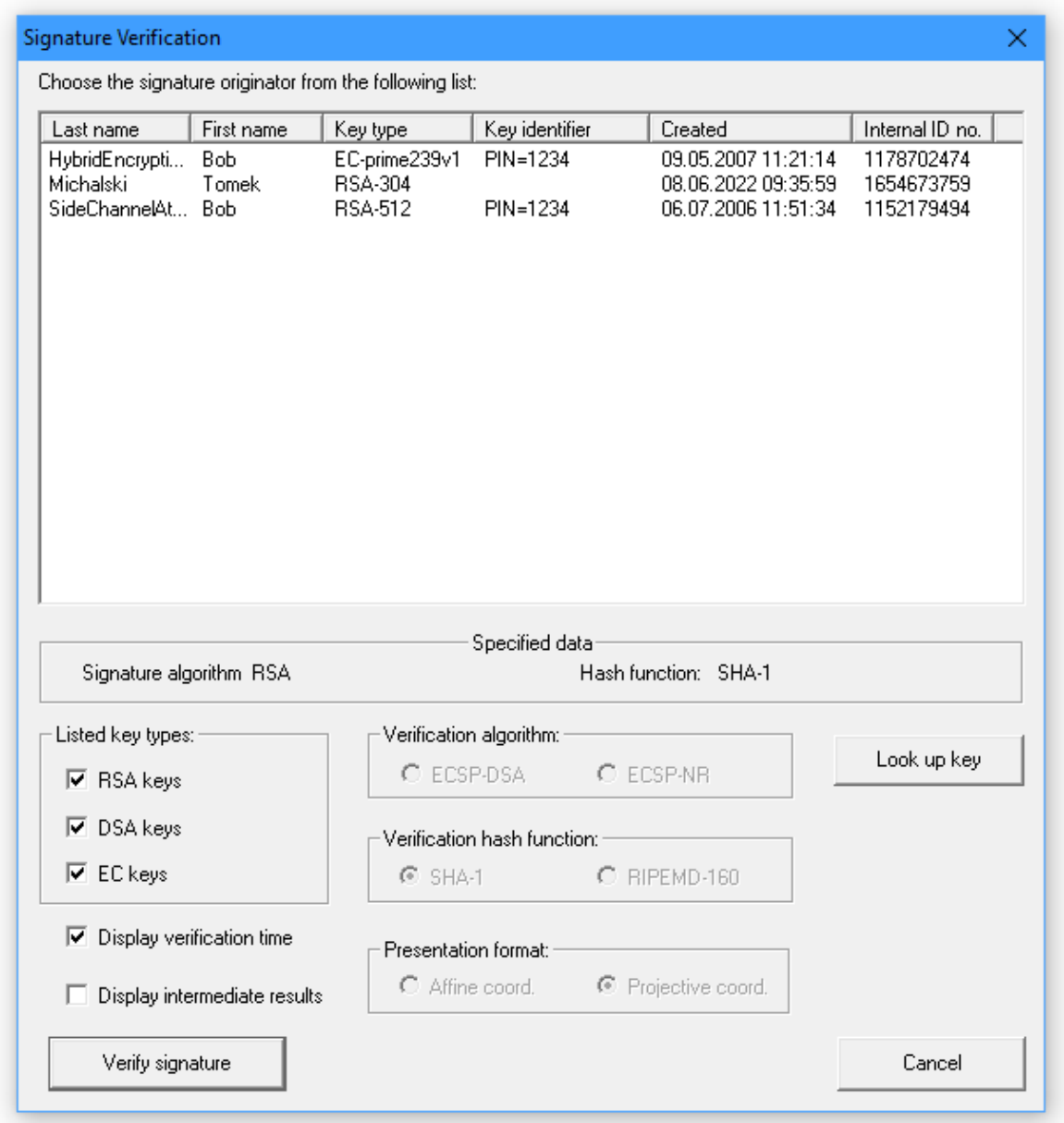


Następnie generujemy nasz podpis i go zapisujemy.





Nasz gotowy podpis.



Dla upewnienia się możemy zweryfikować nasz podpis.

▼ Wnioski

Diffie-Hellman jest odporny na podsłuchiwanie pod warunkiem, że jego dane wejściowe są wystarczająco duże i nie da się ich “odgadnąć” w łatwy sposób.

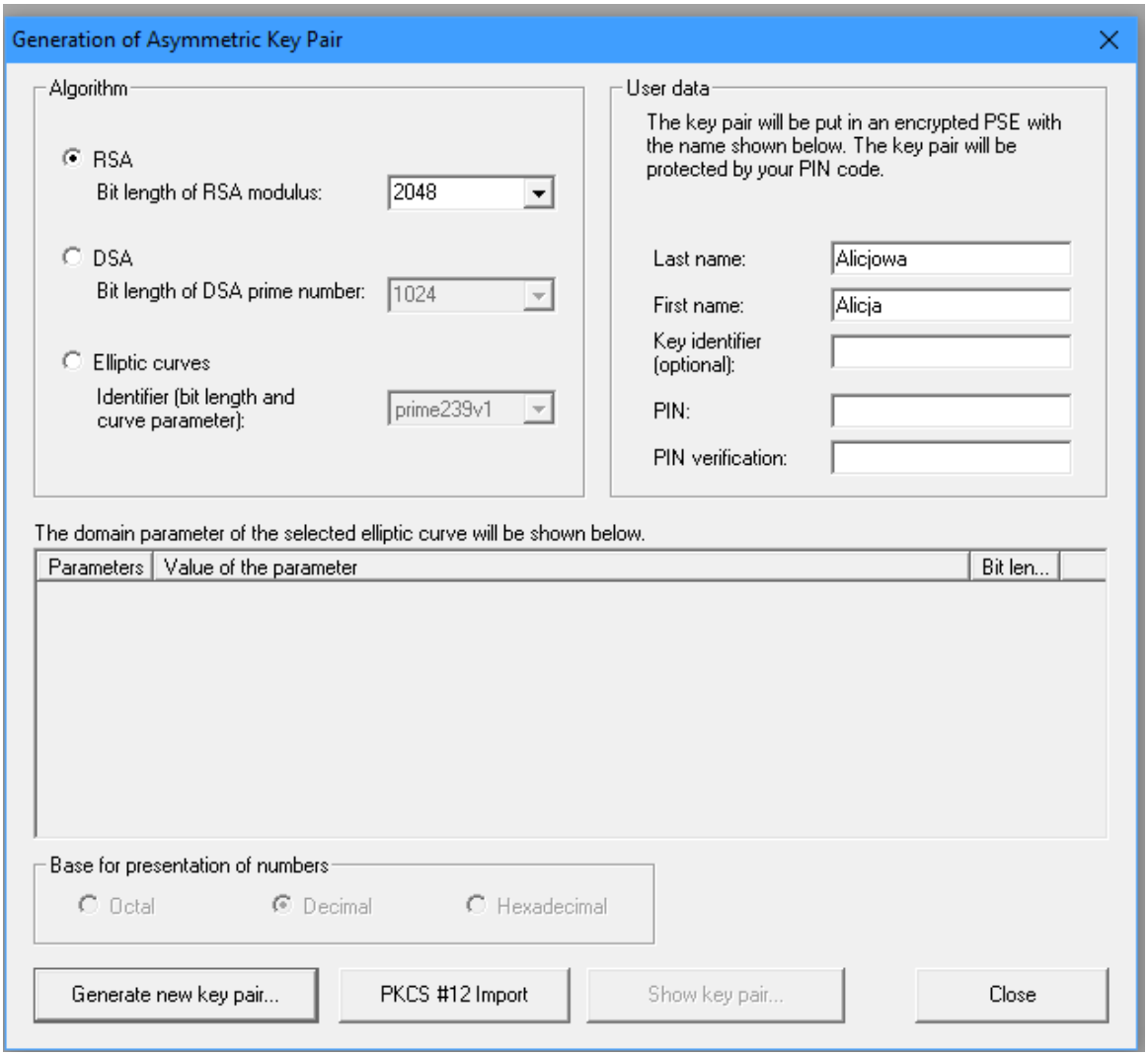
Jeżeli generacja danych jest w pewien sposób przewidywalna może to stanowić pewne problemy w zabezpieczeniach.

RSA - Tutaj tak samo jak w DH - bardzo ważna jest poprawna losowość i wielkość danych wejściowych. Przy braku mocnych liczb losowych podsłuchiwacz może bezpośrednio próbować się włamać i zgadnąć klucze symetryczne. Warto dodać, że popularność algorytmu również stanowi względne zagrożenie - więcej użytkowników - więcej potencjalnych prób złamania.

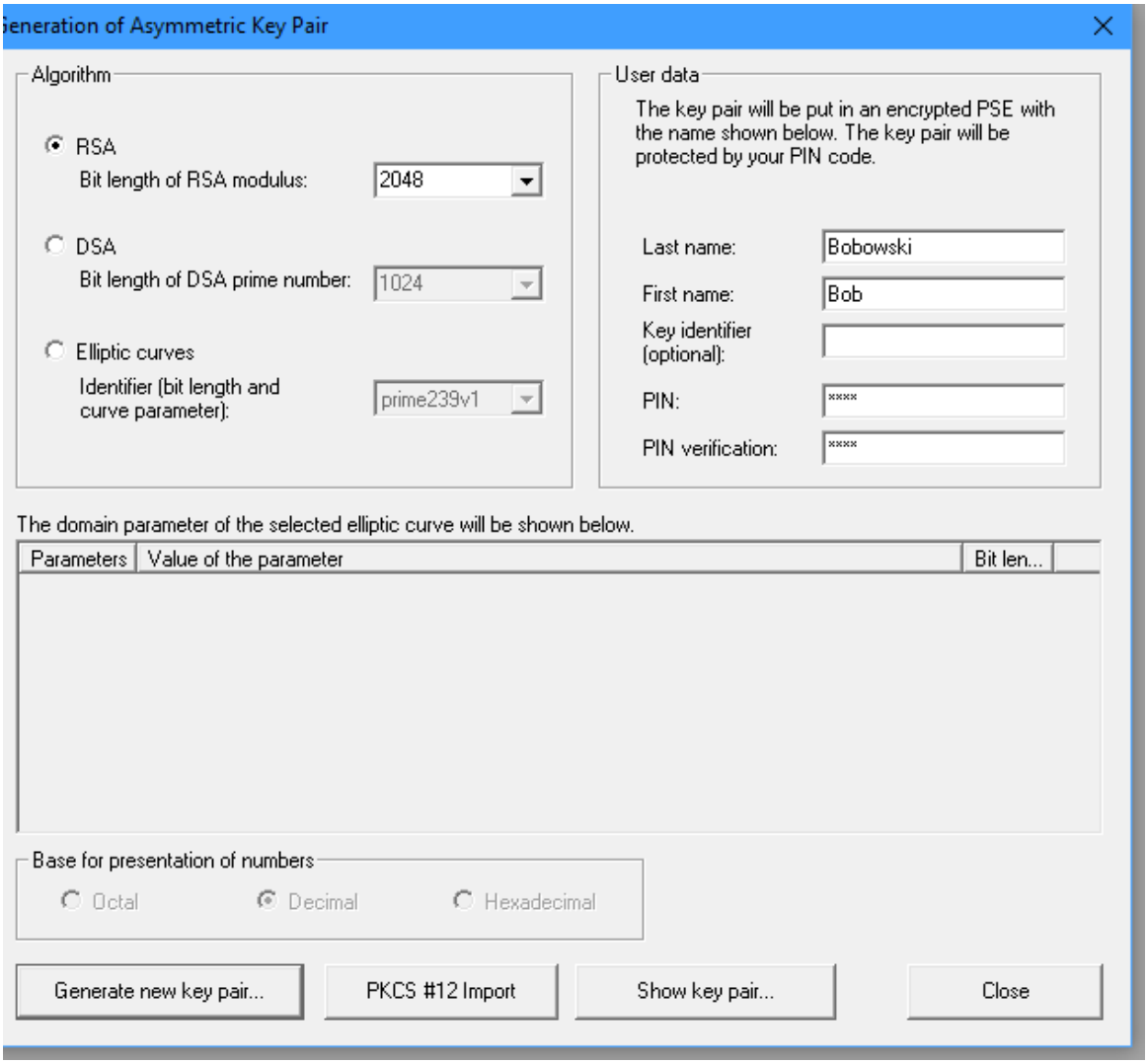
Złamanie szyfru generowanym za pomocą podpisu jest relatywnie trudniejsze w porównaniu do RSA gdyż składa się z większej ilości komponentów.

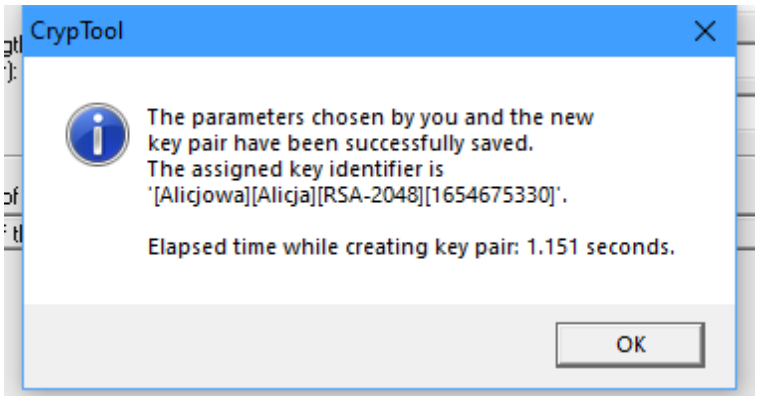
Zadanie 2

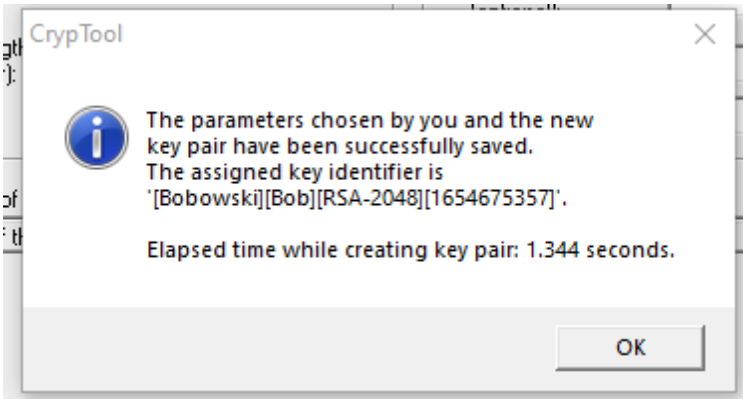
▼ Wygenerować 2048 bitowe klucze asymetryczne dla Alicji i Boba

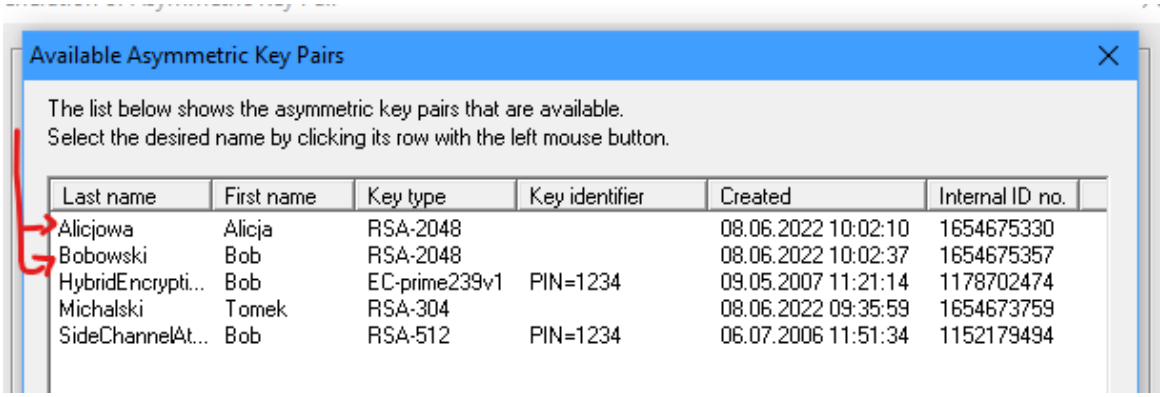


Generowanie kluczy - pin w obu przypadkach - 1111


eneration of Asymmetric Key Pair
Algorithm
RSA
Bit length of RSA modulus:
2048
DSA
Bit length of DSA prime number:
1024
Elliptic curves
Identifier (bit length and curve parameter):
prime239v1
User data
The key pair will be put in an encrypted PSE with the name shown below. The key pair will be protected by your PIN code.
Last name:
Bobowski
First name:
Bob
Key identifier (optional):
PIN:
****
PIN verification:
****
The domain parameter of the selected elliptic curve will be shown below.
Parameters
Value of the parameter
Bit len...
Base for presentation of numbers
Octal
Decimal
Hexadecimal
Generate new key pair...
PKCS #12 Import
Show key pair...
Close

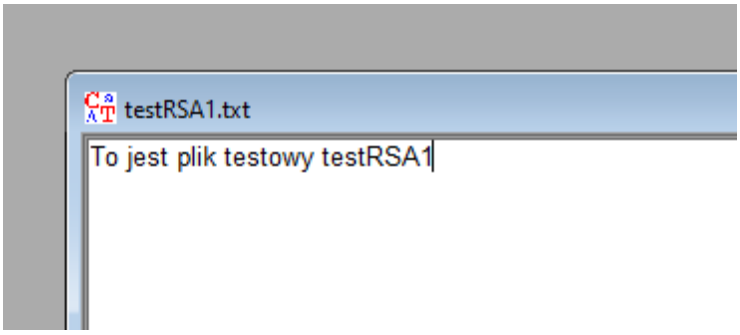

CrypTool
The parameters chosen by you and the new key pair have been successfully saved.
The assigned key identifier is '[Alicjowa][Alicja][RSA-2048][1654675330]'.
Elapsed time while creating key pair: 1.151 seconds.
OK





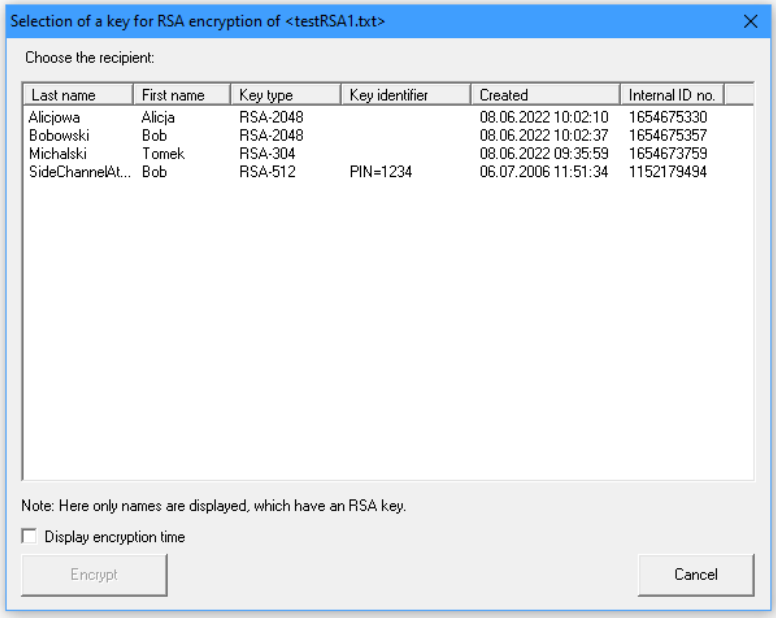
Widzimy wygenerowane klucze w tabeli.

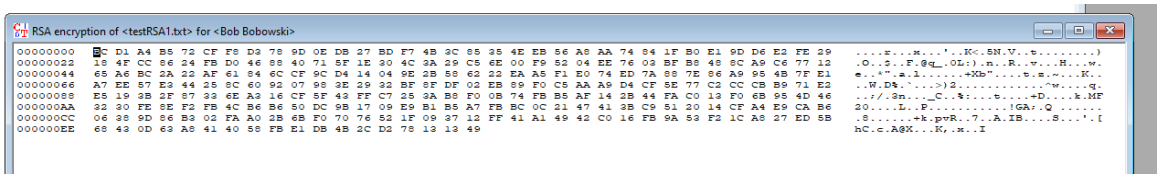
▼ Utworzyć plik „testRSA1.txt” i wpisać wiadomość



▼ Zaszyfruj ww. wiadomość, którą Alicja chce przesłać do Boba

Skoro Alicja chce wysłać coś do Boba, to musi zaszyfrować to jego kluczem publicznym.

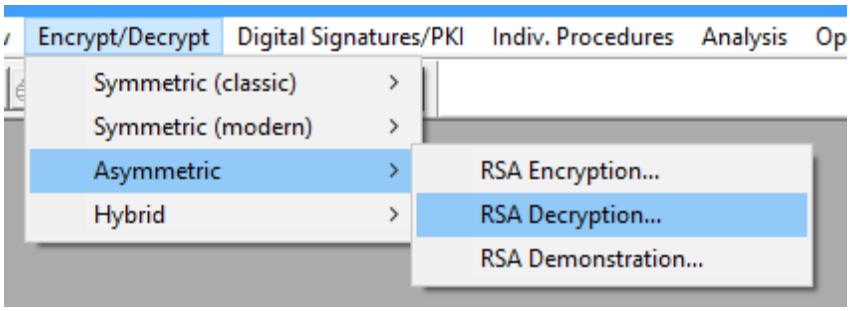


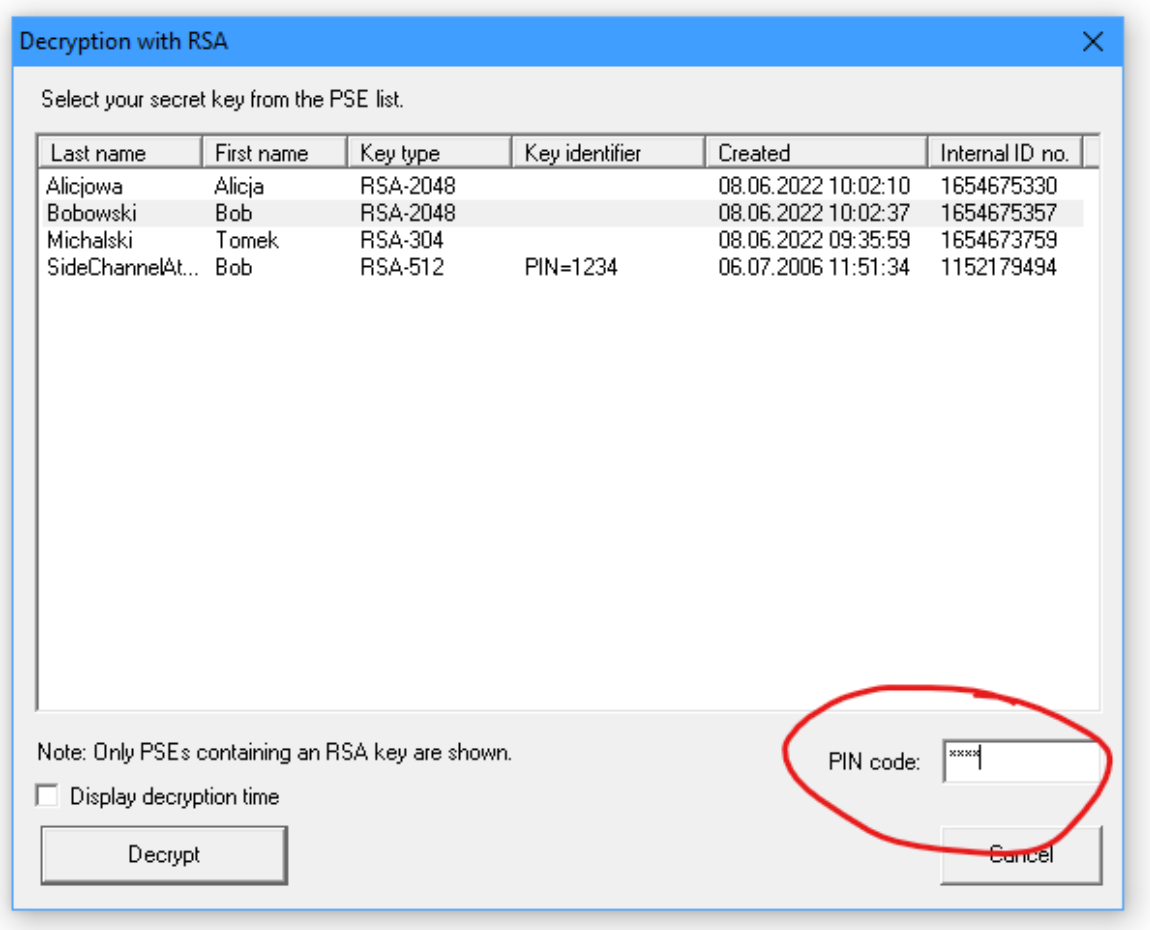


W wyniku szyfrowania kluczem Boba dostajemy taki plik.

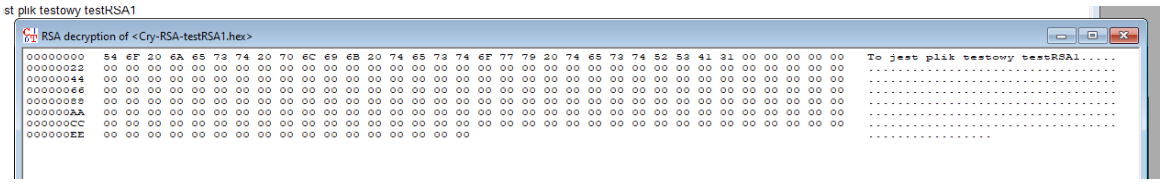
▼ Deszyfruj otrzymaną przez Boba wiadomość

Potrzebujemy teraz klucza prywatnego Boba.





Tym razem upewniamy się, że podajemy poprawny pin - 1111

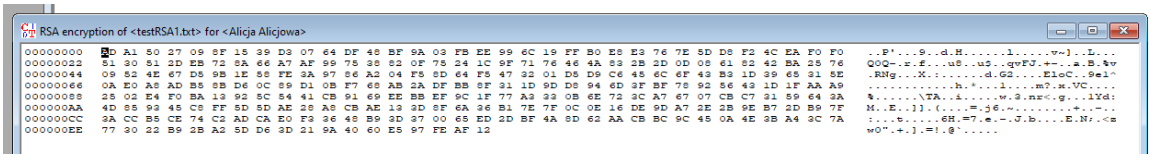


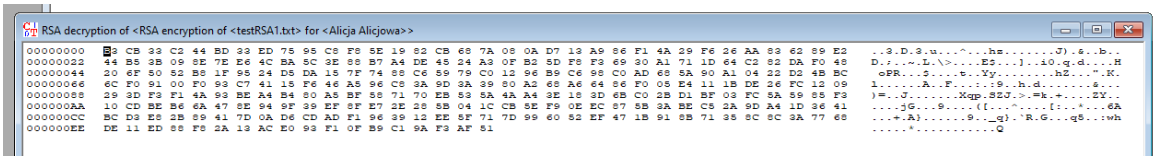
Po odszyfrowaniu dostajemy naszą oryginalną wiadomość.

▼ Przeprowadź ponownie eksperyment używając innych kombinacji kluczy

Nasz scenariusz jest taki, że użytkownik nie umie użyć poprawnej kombinacji kluczy.

Szyfrujemy wiadomość za pomocą klucza publicznego Alicji i próbujemy odszyfrować kluczem prywatnym Boba.





Próba odszyfrowania nie powiodła się.

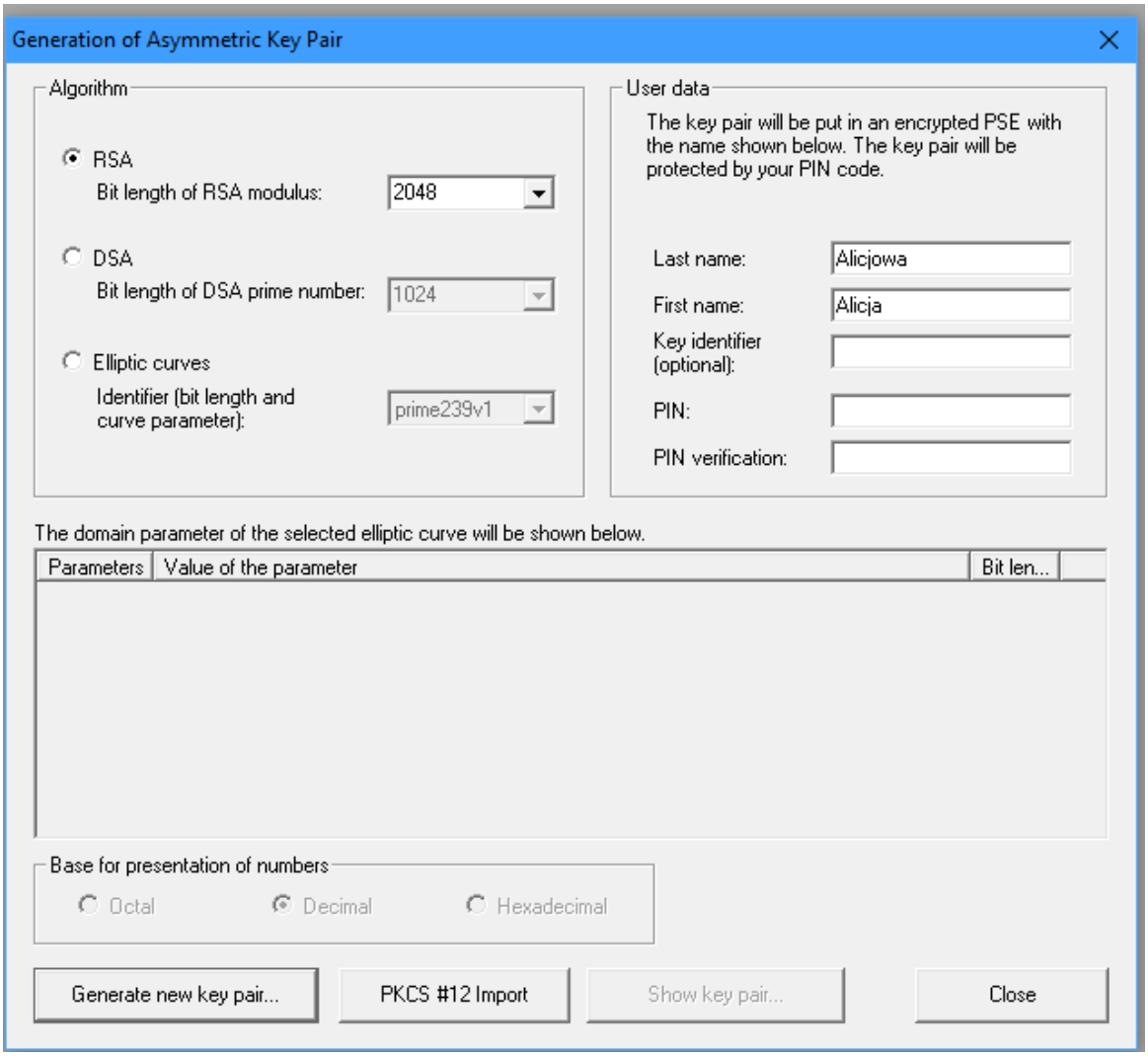
▼ Wnioski

Poprawność działania algorytmu ma podstawy w poprawnym zaszyfrowaniu i odszyfrowaniu wiadomości, w tym kolejności kombinacji kluczy. Im wcześniej zostanie popełniony błąd w cyklu przesyłania wiadomości, tym większy i bardziej niebezpieczny skutek może być.

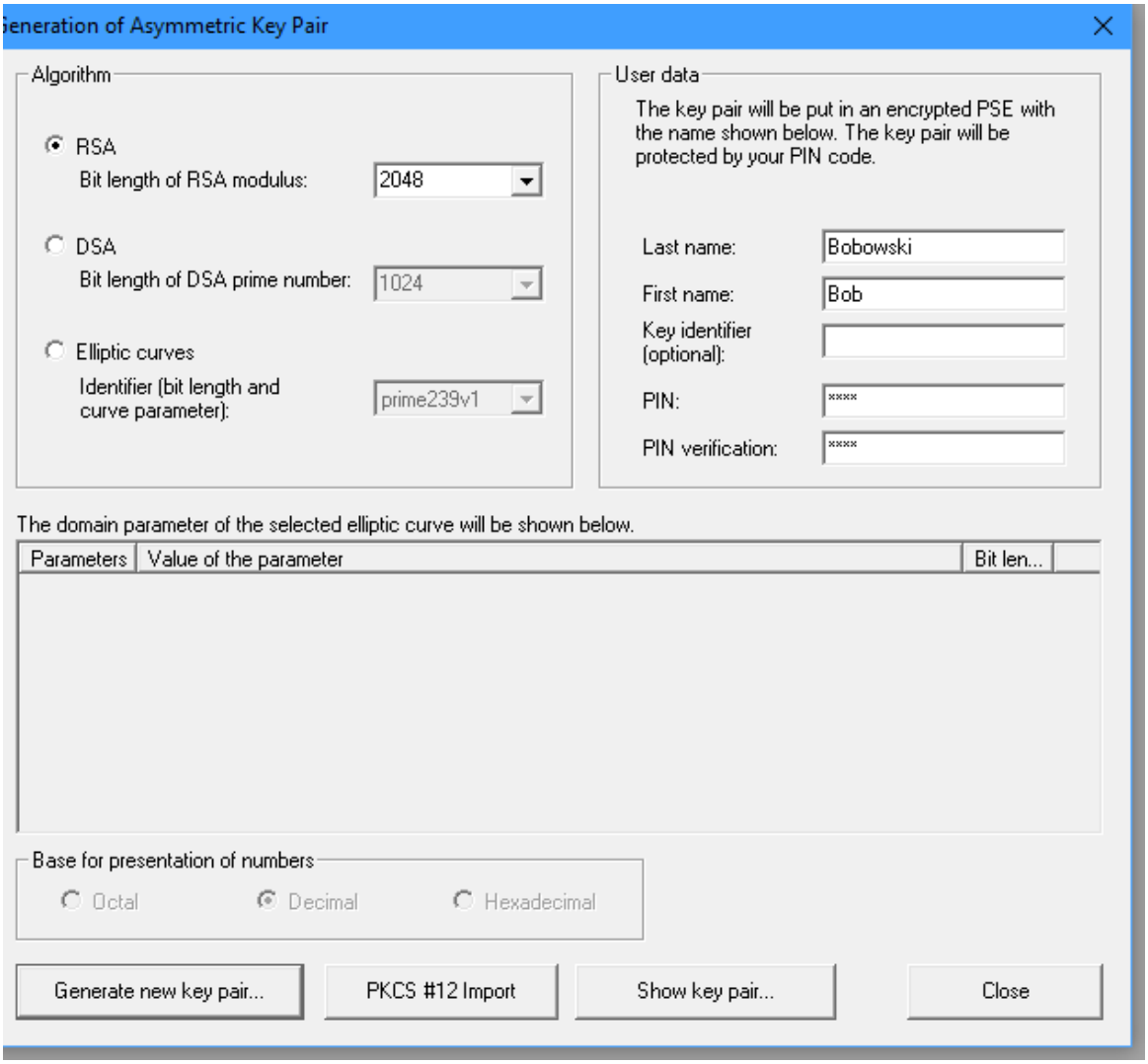
Osoby trzecie mogą próbować odgadnąć klucze prywatne w przypadku zaszyfrowania złym kluczem. Wiadomość może zostać utracona poprzez adresata w wyniku złego początkowego zaszyfrowania.

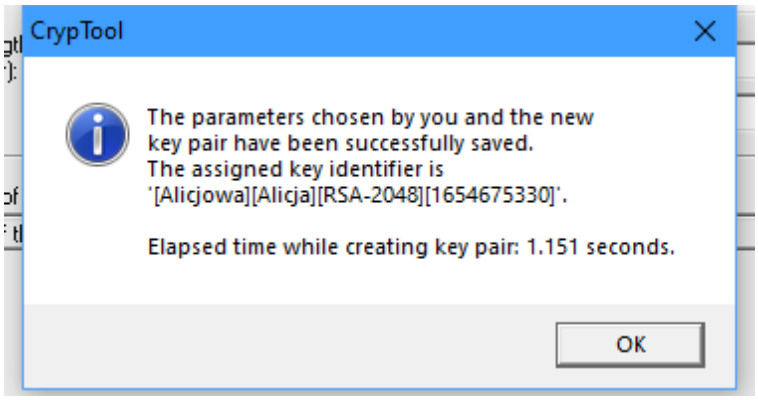
Zadanie 3

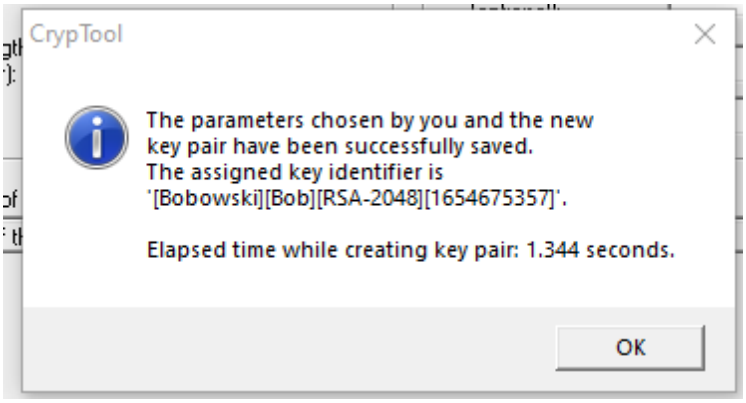
▼ Wygenerować 2048 bitowe klucze asymetryczne dla Alicji i Boba

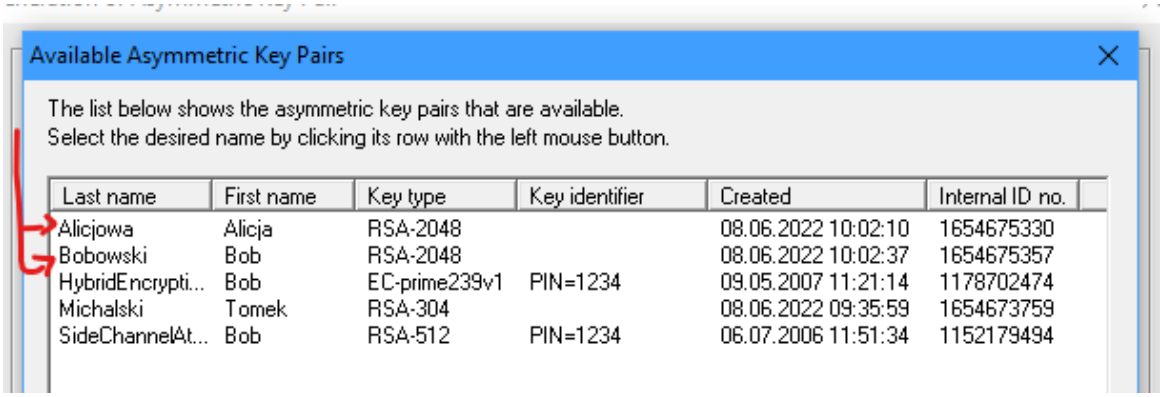


Generowanie kluczy - pin w obu przypadkach - 1111


eneration of Asymmetric Key Pair
Algorithm
RSA
Bit length of RSA modulus:
2048
DSA
Bit length of DSA prime number:
1024
Elliptic curves
Identifier (bit length and curve parameter):
prime239v1
User data
The key pair will be put in an encrypted PSE with the name shown below. The key pair will be protected by your PIN code.
Last name:
Bobowski
First name:
Bob
Key identifier (optional):
PIN:
****
PIN verification:
****
The domain parameter of the selected elliptic curve will be shown below.
Parameters
Value of the parameter
Bit len...
Base for presentation of numbers
Octal
Decimal
Hexadecimal
Generate new key pair...
PKCS #12 Import
Show key pair...
Close

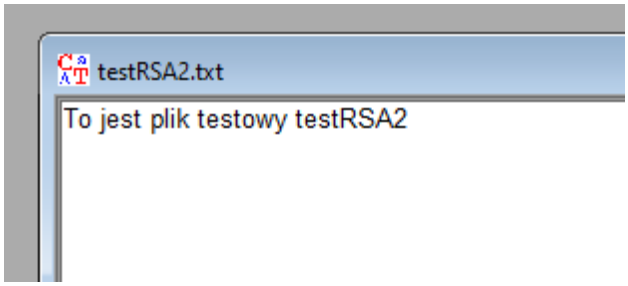

CrypTool
The parameters chosen by you and the new key pair have been successfully saved.
The assigned key identifier is '[Alicjowa][Alicja][RSA-2048][1654675330]'.
Elapsed time while creating key pair: 1.151 seconds.
OK





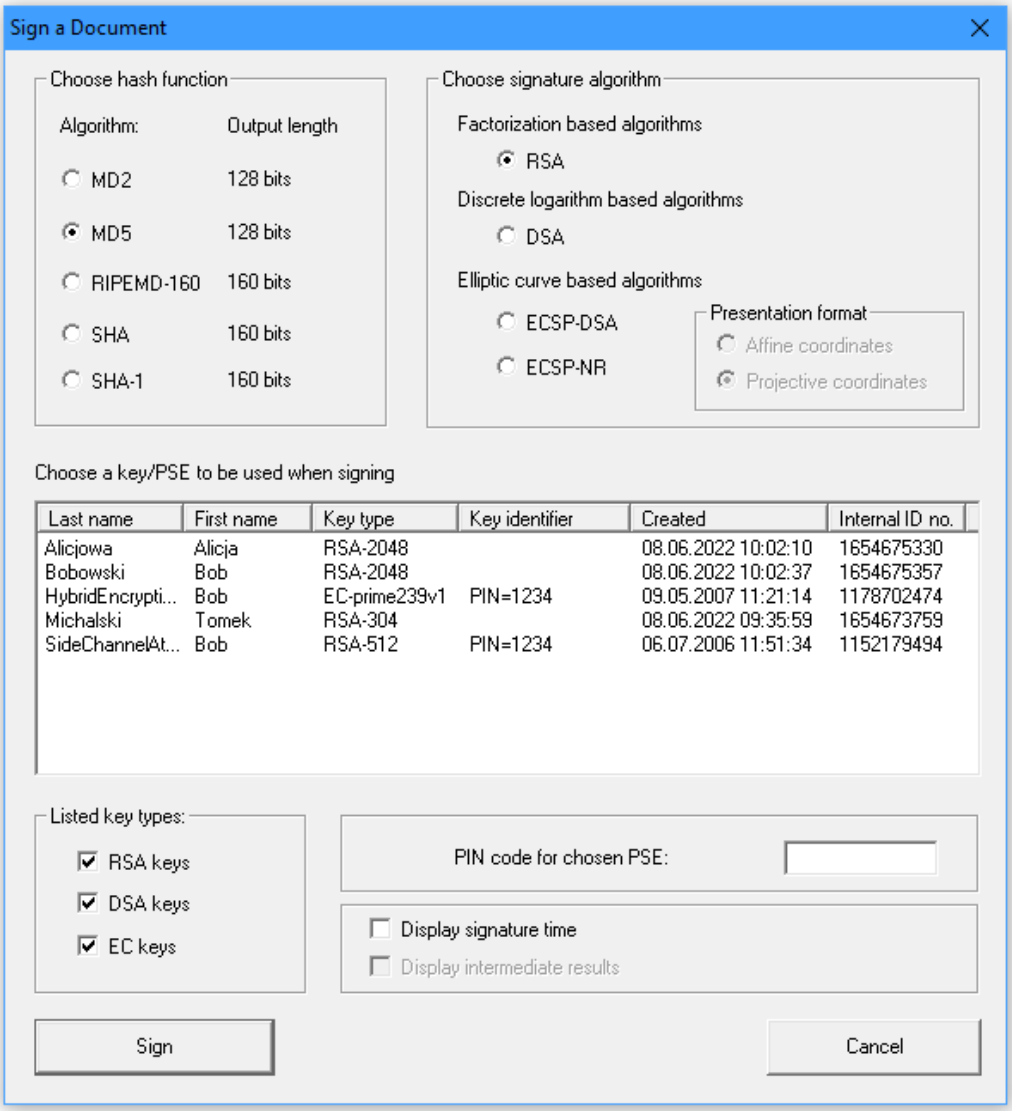
Widzimy wygenerowane klucze w tabeli.

▼ Utworzyć plik wiadomości „testRSA2.txt”

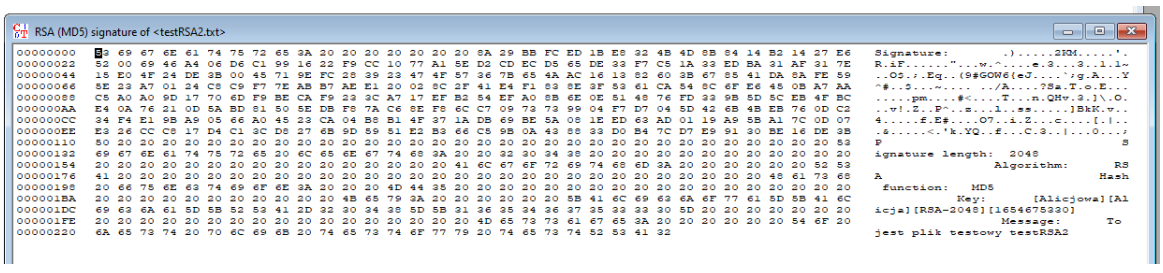


Utworzenie pliku

▼ Podpisz przez Alicję ww. wiadomość

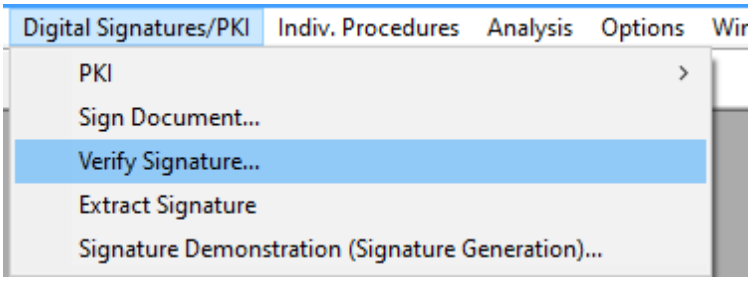


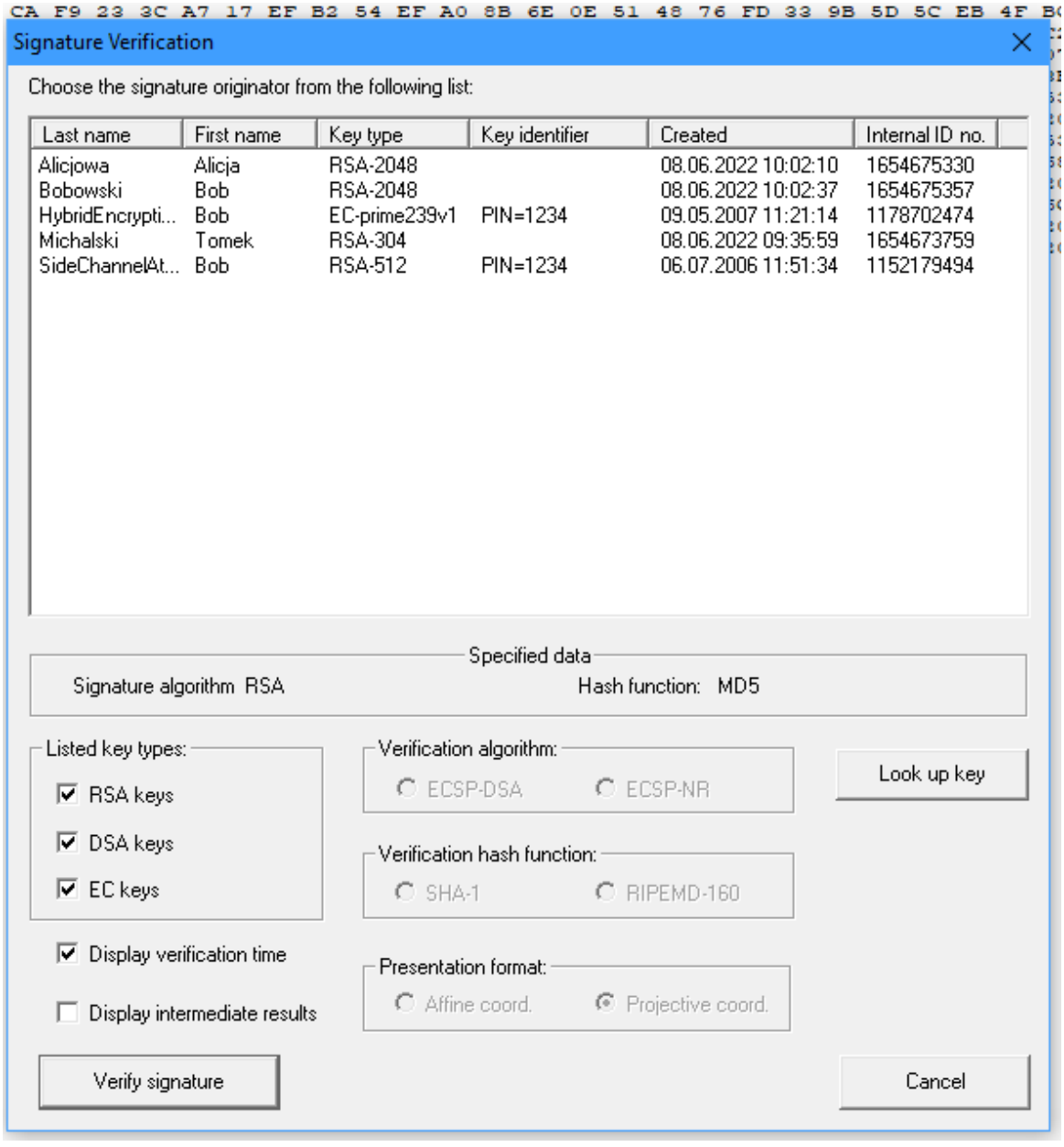
Wybieramy Alicję i wpisujemy 1111 jako kod pin.



W wyniku czego mamy podpis cyfrowy Alicji.

▼ Zweryfikuj przez Boba otrzymany przez Boba podpis cyfrowy





Wybieramy Alicję jako osobę od której dostaliśmy wiadomość.

Weryfikacja jest poprawna.

▼ Wyciągnąć wnioski

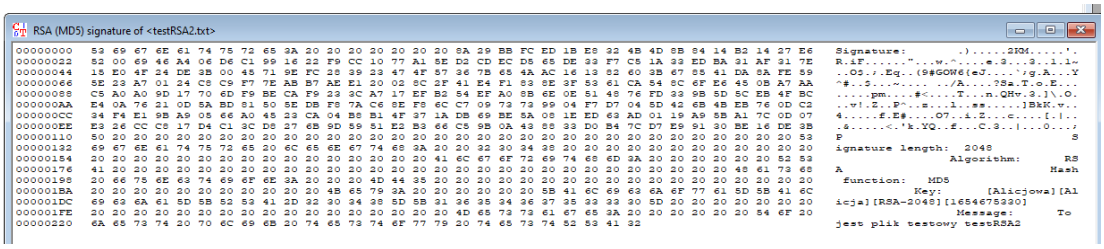
Podpis cyfrowy pozwala nam na zweryfikowanie, czy wiadomość jest od prawdziwego adresata lub czy jest podrobiona. Alicja podpisuje się kluczem prywatnym do którego tylko ona ma dostęp, następnie adresat może potwierdzić wiadomość by wyeliminować wątpliwości za pomocą klucza publicznego.

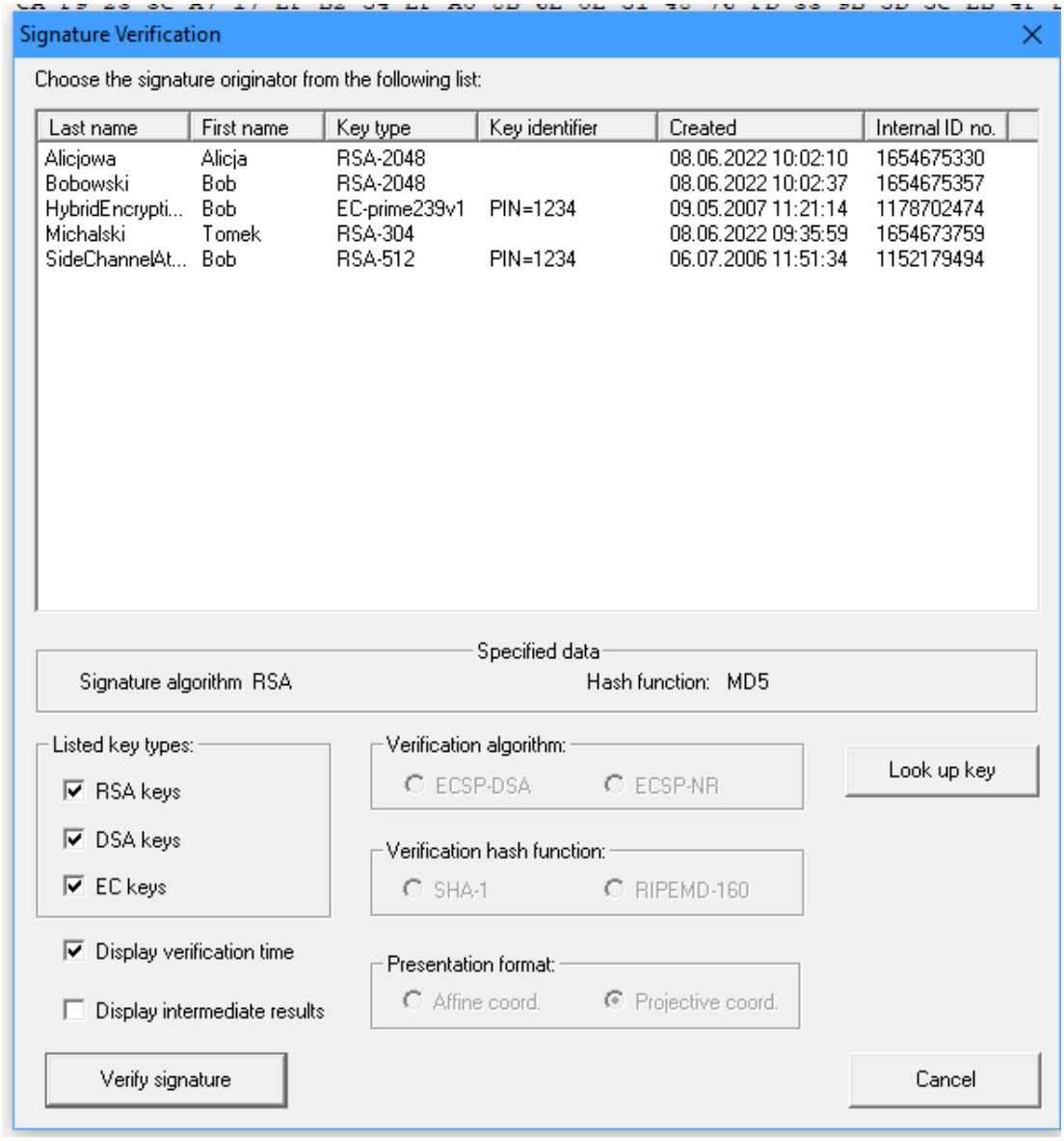
Nie udostępniamy nikomu klucza prywatnego.

▼ Przeprowadź ponownie eksperyment

▼ Używając niewłaściwych kluczy

Podpiszmy się kluczem prywatnym Alicji i spróbujmy zweryfikować kluczem publicznym Boba. Podpisany plik mamy już z poprzednich punktów.





Wybieramy tym razem Boba.

I jak mogliśmy się spodziewać, weryfikacja jest nieudana.

▼ **Naruszając integralność wiadomości**

Zmieńmy teraz wartości w pliku .hex.

```
5 3A 20 :
3 41 32
```

Zmienię 2 ostatnie wartości heksadecymalne miejscami.

```
55 3A 20 20
53 33 41
```

Spróbujmy zweryfikować za pomocą poprawnego klucza publicznego Alicji.

Tutaj również widzimy, że weryfikacja została nie udana.

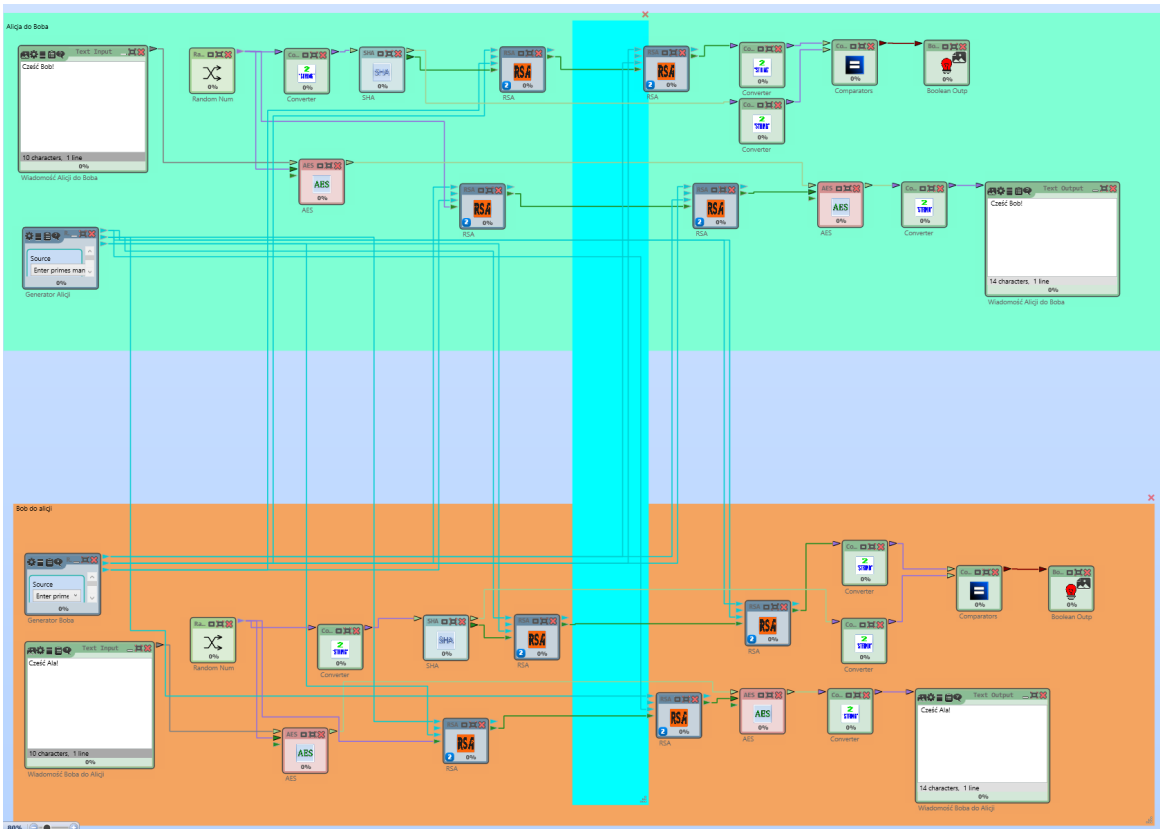
▼ Wnioski

Weryfikacja podpisu pomaga nam również w przypadku, gdy wiadomość została zmodyfikowana w trakcie podróży do adresata.

I oczywiście użycie niewłaściwych kluczy kończy się niepowodzeniem weryfikacji.

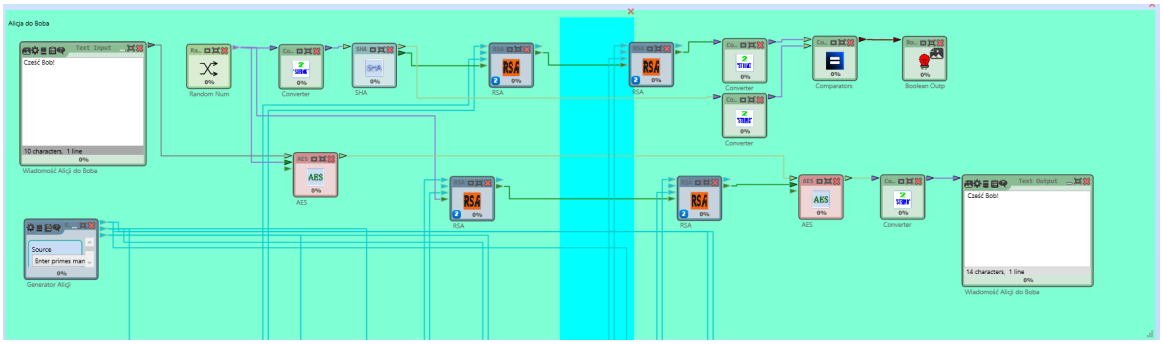
Zadanie 4

▼ Zasymuluj w Cryptool 2



▼ Opis kryptosystemu, wnioski

Zacznijmy od początku od lewej strony. Bob na dole ma to samo więc prezentacja górnej części w zupełności wystarczy.



Mamy trzy rzeczy - inputy na starcie projektu.

- Wiadomość jawna
- Klucze RSA - potrzebne nam do szyfru RSA
- Liczba 16 bajtowa z RGN - potrzebna do szyfru AES

Główna wiadomość

1. Szyfrowana jest za pomocą szyfru AES (+ liczba losowa)
2. Dalej jest wysłana do adresata w zaszyfrowanej formie
3. Jest odszyfrowana za pomocą klucza (o tym później)
4. Konwertowana jest do postaci tekstowej i wyświetlana.

Klucz do AES

1. Jest losowo generowaną liczbą 16-bajtową.
2. Jest on również konwertowany na bajty i haszowany (o czym później)
3. Klucz jest użyty do szyfrowania jawnej wiadomości oraz sam jest przepuszczany przez RSA z danymi publicznymi Boba.
4. Klucz jest odszyfrowany przy pomocy klucza prywatnego Boba i jest użyty przy odszyfrowywaniu wiadomości jawnej.

Wartość haszowana

1. Jest przesyłana na dwa sposoby:
    a. oryginalny
    b. przepuszczony przez RSA Boba (tak samo jak wyżej)
2. wartości są porównywane po stronie Boba
3. Wartość true potwierdza integralność przesyłania danych

Jedyne do czego mają dostęp obie strony do klucze publiczne adresata co zapewnia o poufności. Użycie kluczy RSA przy haszu również uwierzytelnia dane, że są od danej osoby.