Dokumentacja Projektu

Szyfrowana Komunikacja RSA-OAEP

1. ZADANIE 1A: Wys■anie zaszyfrowanej wiadomo■ci do deadbeef

Cel: Wys■a■ poprawnie zaszyfrowan■ wiadomo■■ do u■ytkownika *deadbeef* i otrzyma■ odszyfrowan■ wiadomo■■ w odpowiedzi.

Jak to dzia■a:

1. Klient pobiera klucz publiczny deadbeef z serwera
2. Klient szyfruje wiadomo■■ algorytmem RSA-OAEP
3. Klient koduje wiadomo■■ w base64 i wysy■a do serwera
4. Serwer odbiera zaszyfrowan■ wiadomo■■
5. Serwer odszyfrowuje wiadomo■■ kluczem prywatnym
6. Serwer zwraca odszyfrowan■ wiadomo■■ w JSON

Uruchomienie:

```
Terminal 1 - SERWER:
cd "d:\stud\sem 5\OchronaDanych\6\rsa_server_client_with_keys\server"
$env:DEADBEEF_KEY_1 = "key.1"
$env:DEADBEEF_KEY_2 = "key.2"
python app.py
```

```
Terminal 2 - KLIENT:
cd "d:\stud\sem 5\OchronaDanych\6"
python send_message.py
```

Rezultat: Klient wysy■a zaszyfrowan■ wiadomo■■, serwer j■ odszyfrowuje i zwraca. ■ SUKCES potwierdza poprawne wykonanie zadania.

2. ZADANIE 2A: Dwukierunkowa szyfrowana komunikacja

Cel: Przeprowadzi■ szyfrowan■ komunikacj■ w obu kierunkach mi■dzy u■ytkownikami *john_snow* i *bob_bob*.

Fazy wykonania:

Faza 1 - Generowanie kluczy:
Skrypt generate_keys.py generuje pary kluczy RSA (2048-bit) dla obu u■ytkowników. Ka■dy otrzymuje klucz publiczny i prywatny.

Faza 2 - Za■adowanie kluczy:
Serwer za■adowuje klucze dla trzech u■ytkowników: deadbeef, john_snow, bob_bob.

Faza 3 - Komunikacja:
John szyfruje wiadomo■■ → wysy■a do Bob → Bob odszyfrowuje → Bob wysy■a odpowied■ → John odszyfrowuje

Uruchomienie (3 terminale):

```
Krok 1 - Generowanie kluczy:
cd "d:\stud\sem 5\OchronaDanych\6"
python generate_keys.py
```

```
Terminal 1 - SERWER:
cd "d:\stud\sem 5\OchronaDanych\6\rsa_server_client_with_keys\server"
$env:DEADBEEF_KEY_1 = "key.1"
$env:DEADBEEF_KEY_2 = "key.2"
python app.py
```

```
Terminal 2 - BOB (czeka na wiadomo■■):
cd "d:\stud\sem 5\OchronaDanych\6"
python bob_bob_client.py
```

```
Terminal 3 - JOHN (wysy■a wiadomo■■):
cd "d:\stud\sem 5\OchronaDanych\6"
python john_snow_client.py
```

Rezultat: Na stronie http://127.0.0.1:5555/ wida■ dwie zaszyfrowane wiadomo■ci. John otrzymuje odszyfrowan■ odpowied■ od Bob. ■ SUKCES - dwukierunkowa komunikacja dzia■a.

3. ARCHITEKTURA SYSTEMU

Technologie:
• Python 3.x
• Flask - serwer HTTP
• PyCryptodome - biblioteka szyfrowania RSA
• Base64 - kodowanie wiadomo■ci
• JSON - format komunikacji

Algorytm szyfrowania:
• RSA-OAEP (Optimal Asymmetric Encryption Padding)
• Rozmiar klucza: 2048 bitów
• Funkcja haszowania: SHA-1 (domy■lna)

Bezpiecze■stwo:
■ Wiadomo■ci szyfrowane RSA-OAEP
■ Ka■dy u■ytkownik ma unikaln■ par■ kluczy
■ Klucze prywatne nigdy nie s■ wysy■ane
■ Tylko posiadacz klucza prywatnego mo■e odszyfowa■
■■ To projekt edukacyjny - brak szyfrowania transportu (HTTP)

4. PODSUMOWANIE

Zadanie 1A - Status: ■ ZREALIZOWANE
Wys■anie zaszyfrowanej wiadomo■ci do deadbeef i otrzymanie odszyfrowanej odpowiedzi.

Zadanie 2A - Status: ■ ZREALIZOWANE
Dwukierunkowa szyfrowana komunikacja mi■dzy john_snow i bob_bob.

Metody weryfikacji:
• Otwórz http://127.0.0.1:5555/ aby zobaczy■ zaszyfrowane wiadomo■ci
• Sprawd■ logi w terminalach (stdout) pokazuj■ce kroki: pobieranie klucza, szyfrowanie, wysy■anie
• Wiadomo■ci s■ przechowywane na serwerze w formie zaszyfrowanej

Projekt wykonany: 23.11.2025
Przedmiot: Ochrona Danych
Temat: RSA-OAEP Szyfrowana Komunikacja