

Jakub Bańka, Radosław Kowalski | mgr inż. Mariusz Sepczuk | 3 grudnia 2015 , semestr 15Z

Komunikator z zapewnieniem usługi poufności i niezaprzeczalności.

Protokoły kryptograficzne

Spis treści

[Opis tematu projektu 4](#_Toc436905067)

[Opracowanie teoretyczne 4](#_Toc436905068)

[Definicje używanych pojęć [1] 4](#_Toc436905069)

[Poufność 4](#_Toc436905070)

[Niezaprzeczalność 4](#_Toc436905071)

[Symetryczny algorytm szyfrowania 4](#_Toc436905072)

[Klucz symetryczny (tajny) 4](#_Toc436905073)

[Szyfr blokowy 4](#_Toc436905074)

[Asymetryczny algorytm szyfrowania 5](#_Toc436905075)

[Klucz publiczny 5](#_Toc436905076)

[Klucz prywatny 5](#_Toc436905077)

[Funkcja haszująca (skrótu) 5](#_Toc436905078)

[Opis problemu 5](#_Toc436905079)

[Koncepcja rozwiązania problemu [3] [4] 5](#_Toc436905080)

[Sposób generacji kluczy w algorytmie RSA [2] 6](#_Toc436905081)

[Sposób działania algorytmu AES [5][6][7] 6](#_Toc436905082)

[Szyfrowanie i odszyfrowywanie algorytmem RSA 9](#_Toc436905083)

[Zapewnienie poufności 9](#_Toc436905084)

[Zapewnienie niezaprzeczalności 10](#_Toc436905085)

[Scenariusz wysyłania wiadomości w implementowanym rozwiązaniu 11](#_Toc436905086)

[Koncepcja implementacji przedstawionego rozwiązania 11](#_Toc436905087)

[Współpraca aplikacji klienta z aplikacją serwera 11](#_Toc436905088)

[Aplikacja serwera – projekt architektury trójwarstwowej 11](#_Toc436905089)

[Warstwa prezentacji 11](#_Toc436905090)

[Warstwa logiki biznesowej 11](#_Toc436905091)

[Warstwa danych 11](#_Toc436905092)

[Aplikacja klienta 12](#_Toc436905093)

[Warstwa prezentacji 12](#_Toc436905094)

[Warstwa logiki biznesowej 12](#_Toc436905095)

[Warstwa danych 12](#_Toc436905096)

[Bibliografia 12](#_Toc436905097)

# Opis tematu projektu

Tematem projektu jest stworzenie komunikatora z zapewnieniem usług poufności i niezaprzeczalności. Komunikator ten będzie w stanie przesyłać oraz odbierać wiadomości tekstowe pomiędzy dwojgiem użytkowników. Zaproponowane rozwiązanie będzie składać się z dwóch modułów:

* Aplikacji klienckiej
* Aplikacji serwera

W celu zapewnienia usługi niezaprzeczalności skorzystamy z algorytmu RSA, natomiast poufność zapewni algorytm symetryczny AES. Obydwie aplikacje zostaną napisane w języku C#. Wybraliśmy ten język, ponieważ ma on obszerną bibliotekę kryptograficzną i uważamy, że znamy go na tyle dobrze, aby napisać aplikację, która wyczerpuje temat projektu. Więcej informacji na temat aplikacji znajduje się w rozdziale „Koncepcja implementacji przedstawionego rozwiązania”.

# Opracowanie teoretyczne

## Definicje używanych pojęć [[1]](#_Bibliografia)

### Poufność

Jest to ochrona informacji przed ujawnieniem jej osobom innym niż uprawnione. Istnieje wiele metod zapewniania poufności, od fizycznej ochrony, do algorytmów matematycznych, których zastosowanie powoduje, że dane są pozbawione sensu (zaszyfrowane).

### Niezaprzeczalność

Jest to ochrona przed wyparciem się zobowiązań lub działań, które miały miejsce w chwili powstania sporu. W sytuacji, gdy podmiot odmawia przyznania się do podjęcia określonego działania, konieczne jest posiadanie środków umożliwiających rozwiązanie tego sporu. Przykładowo, podmiot może autoryzować kupno rzeczy przez drugi podmiot, ale później zaprzeczyć, że dokonał takiej autoryzacji. Aby rozstrzygnąć ten spór wymagane jest zaangażowanie zaufanej trzeciej strony.

### Symetryczny algorytm szyfrowania

Algorytm szyfrowania opierający się na dwóch przekształceniach – jedno szyfrujące (inicjatora), drugie deszyfrujące (odbiorcy) – które używają tego samego klucza tajnego (symetrycznego) lub pary kluczy charakteryzujących się tym, że łatwo obliczyć jeden na podstawie drugiego. Ujawnienie jednego z kluczy umożliwia odtworzenie zaszyfrowanej wiadomości.

### Klucz symetryczny (tajny)

Klucz stanowiący podstawę przekształceń w symetrycznym algorytmie szyfrowania. Nie może być on publicznie znany. Jego przekazanie odbywa się za pomocą zaufanego medium.

### Szyfr blokowy

Rodzaj szyfru symetrycznego polegający na szyfrowaniu bloku wejściowego na podstawie klucza tajnego, przekształcając go na blok wyjściowy o takiej samej długości tak, że odwrócenie wykorzystanego przekształcenia jest niemożliwe. Klucz w tym typie szyfrów musi mieć długość co najmniej 128 bitów, aby zagwarantować bezpieczeństwo.

### Asymetryczny algorytm szyfrowania

Algorytm szyfrowania opierający się na dwóch przekształceniach, z których jedno definiowane jest przez klucz publiczny (przekształcenie publiczne), a drugie definiowane przez klucz prywatny (przekształcenie prywatne). Własnością systemu z takim algorytmem jest to, że określenie przekształcenia prywatnego na podstawie przekształcenia publicznego jest obliczeniowo niewykonalne.

### Klucz publiczny

Jeden z dwóch kluczy występujących w asymetrycznych algorytmach szyfrowania. Jest on publicznie dostępny. Przesyłany jest za pomocą dowolnego, niezabezpieczonego kanału.

### Klucz prywatny

Jeden z dwóch kluczy występujących w asymetrycznych algorytmach szyfrowania. Jest on trzymany w tajemnicy (zna go tylko użytkownik). Odtworzenie klucza prywatnego na podstawie klucza publicznego musi być obliczeniowo trudne.

### Funkcja haszująca (skrótu)

Funkcja skrótu przekształca wiadomość (ciąg bitów) o dowolnej długości L do ciągu bitów o ustalonej (niedużej) długości l, czyli skrótu wiadomości. Funkcja skrótu jest odporna na kolizję, jeśli znalezienie dwóch dowolnych wiadomości dających ten sam skrót jest obliczeniowo niemożliwe.

## Opis problemu

Komunikatory tekstowe stają się coraz bardziej popularne. Nic więc dziwnego, że stają się one coraz częstszym celem ataków. W związku z tym coraz częściej komunikatory te wyposaża się w mechanizmy zapewniające poufność danych. Mechanizmy te są zazwyczaj reprezentowane poprzez szyfrowanie danych za pomocą szyfrów blokowych lub z infrastrukturą klucza publicznego. Popularnością zaczyna cieszyć się nawiązywanie różnego rodzaju umów, dokonywanie istotnych obrad, itp., za pomocą tego tyu komunikatorów. W takim przypadku duże znaczenie ma niezaprzeczalność informacji. Jest ona osiągana przede wszystkim dzięki mechanizmowi podpisu cyfrowego.

## Koncepcja rozwiązania problemu [[3]](#_Bibliografia) [[4]](#_Bibliografia)

Do rozwiązania problemu niezaprzeczalności użyjemy asymetrycznego algorytmu RSA. Jego nazwa pochodzi od pierwszych liter nazwisk jego twórców (Rivest, Shamir, Adleman). Został opracowany w roku 1977. Jest to jedna z najbezpieczniejszych metod kryptograficznych ze względu na trudność w rozkładaniu na czynniki pierwsze dużych liczb złożonych.

Do rozwiązania problemu poufności użyjemy algorytmu AES. Jest to symetryczny szyfr blokowy. Powstał w wyniku konkursu ogłoszonego w 1997 roku. Konkurs ten ogłoszono, ponieważ siła używanego ówcześnie algorytmu DES została uznana za niewystarczającą. Osoby prowadzące rozmowę muszą znać ten sam tajny klucz tajny. Klucz ten będzie różny od kluczy prywatnych i publicznych używanych przy problemie niezaprzeczalności. Zostanie on przesłany jako wiadomość zaszyfrowana algorytmem RSA kluczem publicznym użytkownika do którego zostanie wysłany. Po otrzymaniu takiego klucza użytkownicy będą mogli rozpocząć rozmowę, w której wiadomości szyfrowane będą algorytmem AES.

### Sposób generacji kluczy w algorytmie RSA [[2]](#_Bibliografia)

Dane wejściowe:

* Przesyłana wiadomość **m**
* Dwie duże liczby pierwsze **p** i **q**

Dane wyjściowe:

* Klucz publiczny – pary liczb **(n,e).**
* Klucz prywatny – pary liczb **(n,d).**

1. Oblicz iloczyn **n = p \* q.**
2. Oblicz funkcję Eulera **ф(n) = (p-1)(q-1).**
3. Wybierz liczbę **e** taką, że **1 < e < ф(n)** oraz względnie pierwszą z  **ф(n).**
4. Oblicz liczbę **d**, która jest odwrotnością liczby **e mod(ф(n)) (e\*d mod(ф(n)) = 1).**

### Sposób działania algorytmu AES [[5][6][7]](#_Bibliografia)

Z racji, że algorytm ten jest bardzo skomplikowany poszczególne operacje są wyjaśnione w kolejnych podpunktach.

Dane wejściowe:

* Tekst jawny **m** w postaci macierzy 4x4 bajty szeregowanej kolumnami (128 bitowy)
* Klucz tajny o długości 128 bitów

Dane wyjściowe:

* Zaszyfrowany tekst **c** w postaci bloku 128 bitów.

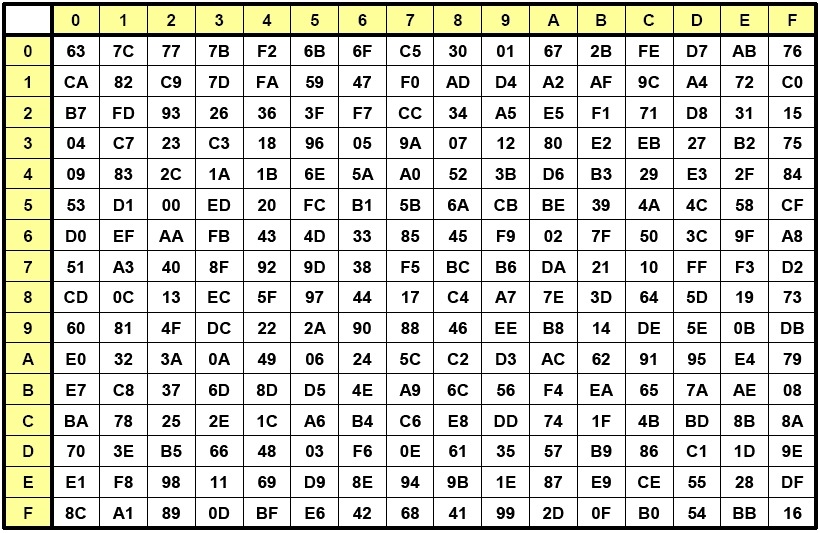
1. Przygotuj podklucze na podstawie tajnego klucza (po jednym dla każdej rundy szyfrującej w tym wypadku 10)
2. Dodaj każdy bajt w bloku danych do odpowiadającego mu bajtowi w kluczu tajnym.
3. Wykonaj 9 rund składających się z następujących operacji:
   1. *Substitute Bytes,*
   2. *Shift Rows,*
   3. *Mix Columns,*
   4. *Add Round Key,*
4. Wykonaj rundę kończącą składającą się z tych samych operacji z wyjątkiem operacji *Mix Columns*.

Odszyfrowanie tekstu wymaga odwrócenia tych operacji. Podklucze brane są w odwrotnej kolejności.

#### Operacja Substitute Bytes

Dane wejściowe:

* Macierz A 4x4 bajty szeregowana kolumnami (128 bitowa)
* Tablica podstawień (S-box)



Rysunek 1 S-box w algorytmie AES

Dane wyjściowe:

* Macierz B 4x4 bajty szeregowana kolumnami (128 bitowa) z zamienionymi bajtami

1. Weź pierwszy bajt z tabeli.
2. Weź 4 starsze bity tego bajtu i zamień je na postać szesnastkową – wskazuje ona wiersz w S-boxie.
3. Weź 4 młodsze bity tego bajtu i zamień je na postać szesnastkową – wskazuje ona kolumnę w S-boxie.
4. Zamień wartość bajtu na wartość z tablicy S-box.
5. Powtórz to dla wszystkich bajtów.

#### Opercja Shift Rows

Dane wejściowe:

* Macierz A 4x4 bajty szeregowana kolumnami (128 bitowa)

Dane wyjściowe:

* Macierz B 4x4 bajty szeregowana kolumnami (128 bitowa) z przesuniętymi wierszami

1. Przesuń bajty w drugim wierszu o jeden w prawo (skrajny prawy bajt staje się pierwszy).
2. Przesuń bajty w trzecim wierszu o dwa w prawo.
3. Przesuń bajty w czwartym wierszu o cztery w prawo.

#### Operacja Mix Columns

Dane wejściowe:

* Macierz A 4x4 bajty szeregowana kolumnami (128 bitowa)
* Macierz przekształceń:

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | 3 | 1 | 1 |
| 1 | 2 | 3 | 1 |
| 1 | 1 | 2 | 3 |
| 3 | 1 | 1 | 2 |

Dane wyjściowe:

* Macierz B 4x4 bajty szeregowana kolumnami (128 bitowa) ze zmienioną wartością bajtów

1. Pomnóż macierz przekształceń przez każdą kolumnę macierzy A.
2. Z wyników mnożenia stwórz macierz B.

#### Operacja Add Round Key

Dane wejściowe:

* Macierz A 4x4 bajty szeregowana kolumnami (128 bitowa)
* Podklucz danej rundy o długości 128 bitów

Dane wyjściowe:

* Macierz B 4x4 bajty szeregowana kolumnami (128 bitowa) ze zmienioną wartością bajtów

1. Dodaj każdy bajt w bloku danych do odpowiadającego mu bajtowi w podkluczu trwającej rundy.

#### Tworzenie podklucza rundy

Dane wejściowe:

* Podklucz poprzedniej rundy o długości 128 bitów (jeżeli pierwsza runda to klucz tajny)

Dane wyjściowe:

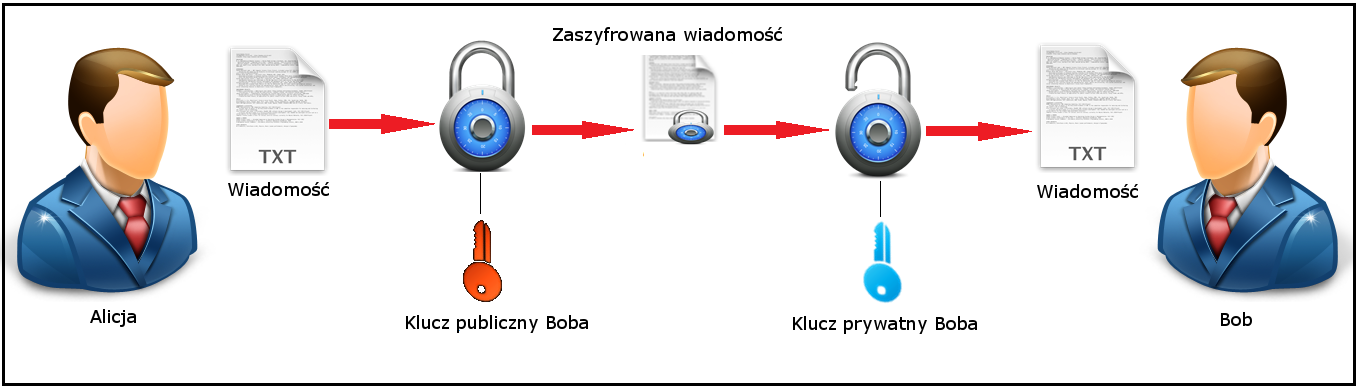
* Podklucz danej rundy o długości 128 bitów

1. Przepisz pierwsze 4 bajty klucza.
2. Wykonaj rotację bajtów w wektorze o jeden w lewo (skrajnie lewy bajt przepisz na skrajnie prawą pozycję).
3. Wykonaj operację *Substitute Bytes*.
4. Za pomocą funkcji XOR zsumuj najbardziej lewy bajt z dwójką do potęgi „numer iteracji – 1”(operacja Rcon).
5. Za pomocą funkcji XOR zsumuj otrzymany wektor z 4-bajtowym wektorem odpowiadającym mu w kluczu z poprzedniej rundy.
6. Dla pozostałych 12 bajtów powtórz czynności 1,3 i 4 3 razy przepisując kolejne bity.

### Szyfrowanie i odszyfrowywanie algorytmem RSA

Założenia:

* Nadawca – **Alicja**
* Odbiorca – **Bob**
* Obydwie strony mają wygenerowane pary kluczy prywatnych i publicznych; klucze powszechnie publiczne są znane



Rysunek 2 Schemat szyfrowania algorytmem RSA

Rysunek 2 przedstawia scenariusz przesyłania wiadomości przy użyciu algorytmu RSA jako algorytmu szyfrującego.

Dane wejściowe:

* Wiadomość jawna **m**
* Klucz publiczny Boba **(n,e)**
* Klucz prywatny Boba **(n,d)**

Dane wyjściowe:

* Zaszyfrowana wiadomość **c**

Wykonywane czynności: [[2]](#_Bibliografia)

1. Alicja generuje wiadomość **m.**
2. Alicja za pomocąklucza publicznego Boba (**n,e**) tworzy zaszyfrowaną wiadomość **c = me mod(n)** i wysyła ją do Boba.
3. Bob odbiera zaszyfrowaną wiadomość **c** i deszyfruje ją swoim kluczem prywatnym (**n,d**), tworząc wiadomość **m = cd mod(n).**

### Zapewnienie poufności

Dane wejściowe:

* Wiadomość jawna **m**
* Klucz tajny **k**

Dane wyjściowe:

* Zaszyfrowana wiadomość **c**

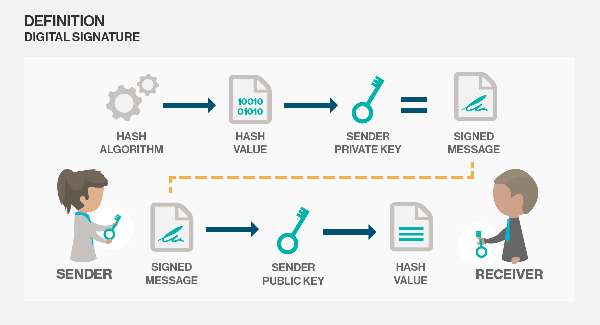
Wykonywane czynności: [[2]](#_Bibliografia)

1. Alicja generuje wiadomość **m.**
2. Alicja za pomocąklucza tajnego **k** tworzy zaszyfrowaną wiadomość **c** korzystając z algorytmu AESi wysyła ją do Boba.
3. Bob odbiera zaszyfrowaną wiadomość **c** i deszyfruje ją swoim kluczem tajnym **k**, tworząc wiadomość **m.**

### Zapewnienie niezaprzeczalności

Założenia:

* Nadawca – **Alicja**
* Odbiorca – **Bob**
* Obydwie strony mają wygenerowane pary kluczy prywatnych i publicznych; klucze publiczne są powszechnie znane



Rysunek 3 Schemat podpisu cyfrowego przy użyciu algorytmu RSA

Rysunek 3 przedstawia scenariusz podpisu cyfrowego przy użyciu algorytmu RSA.

Wykonywane czynności: [[2]](#_Bibliografia)

1. Alicja chce wysłać podpisaną wiadomość **m** do Boba.
2. Alicja szyfruje funkcję skrótu wiadomości **m**, otrzymując **h**, swoim kluczem prywatnym **(n,d)**, tworząc podpisany dokument **s = md mod(n).**
3. Alicja przesyła do Boba dokument **s** oraz wiadomość **m.**
4. Bob odbiera oba dokumenty oraz deszyfruje wiadomość **s** kluczem publicznym Alicji **(n,e)**, otrzymując **h.**
5. Bob porównuje wartość **h** z wartością funkcji skrótu otrzymanej wiadomości **m.**

### Scenariusz wysyłania wiadomości w implementowanym rozwiązaniu

W naszym programie będziemy wykonywać obydwie opisane wcześniej czynności. Wysyłana wiadomość będzie zaszyfrowana, co zapewni poufność, a także będzie zawierała podpis cyfrowy, co zapewni niezaprzeczalność.

# Koncepcja implementacji przedstawionego rozwiązania

## Współpraca aplikacji klienta z aplikacją serwera

Aplikacja klienta będzie łączyć się z aplikacją serwera za pomocą protokołu TCP IP. Po połączeniu klient będzie musiał podać nazwę użytkownika i hasło. Informacje te będą przesyłane jako wiadomość zaszyfrowana za pomocą RSA. Aplikacja serwera sprawdzi czy są one poprawne i poinformuje o wyniku aplikację klienta. Jeżeli logowanie przebiegło pomyślnie klient będzie mógł prosić o nawiązanie połączenia z innym klientem. Jeżeli klient ten istnieje serwer odpowie przesłaniem tajnego klucza do obu klientów szyfrując go algorytmem RSA. Jeżeli nie ma takiego klienta powiadomi o błędzie. Po zakończeniu rozmowy klienci powiadamiają o tym serwer i rozłączają się.

## Aplikacja serwera – projekt architektury trójwarstwowej

### Warstwa prezentacji

Warstwa odpowiedzialna za interfejs graficzny. Będzie powiadamiana przez warstwę logiki biznesowej o przyjściu nowego połączenia w celu wyświetlenia odpowiedniego powiadomienia. Będzie powiadamiać warstwę logiki biznesowej o porcie wybranym przez użytkownika.

### Warstwa logiki biznesowej

Warstwa odpowiedzialna za obsługę połączeń z klientami, przekierowywanie wiadomości użytkowników oraz generację kluczy tajnych. Będzie powiadamiana przez warstwę prezentacji o wybranym przez użytkownika porcie nasłuchiwania. Od warstwy danych będzie pobierać przechowywane wartości kluczy i dane logowania. Do warstwy prezentacji będzie przekazywać informacje o nowym połączeniu, a do warstwy danych dane nowych użytkowników w celu ich zapisania.

### Warstwa danych

Warstwa składująca dane. Składać się będzie z systemu plików – jawnego z kluczami jawnymi i tajnego z danymi logowania. Do warstwy logiki biznesowej przekazywać będzie wartości kluczy i danych logowania w niezmienionym stanie. Od tej samej warstwy będzie otrzymywać informacje o nowych użytkownikach w celu ich zapisania.

## Aplikacja klienta

### Warstwa prezentacji

Warstwa odpowiedzialna za interfejs graficzny. Będzie powiadamiana przez warstwę logiki biznesowej o przyjściu nowej wiadomości. Od warstwy logiki dostawać będzie również wiadomości w formie jawnej w celu ich wyświetlenia. Do warstwy logiki będzie dostarczać wiadomości w formie jawnej w celu zaszyfrowania i wysłania ich do adresata.

### Warstwa logiki biznesowej

Warstwa odpowiedzialna za obsługę połączeń z serwerem, szyfrowanie i wysyłanie wiadomości użytkowników oraz generację kluczy . Od warstwy prezentacji dostawać będzie wiadomości w formie jawnej do zaszyfrowania i wysłania. Od warstwy danych będzie pobierać przechowywane wartości kluczy i dzienniki rozmów. Do warstwy prezentacji i do warstwy danych będzie wysyłać wiadomości w formie odpowiednio jawnej i tajnej. Do warstwy danych będzie również wysyłać klucz prywatny w formie tajnej oraz publiczny w formie jawnej.

### Warstwa danych

Warstwa składująca dane. Składać się będzie z systemu plików tajnych z dziennikami rozmów (jeden plik to jeden dziennik) i kluczami. Do warstwy logiki biznesowej przekazywać będzie wartość kluczy i dzienników rozmów w niezmienionym stanie. Od tej samej warstwy będzie otrzymywać informacje o nowych kluczach i wiadomościach w celu ich zapisania.

# Praktyczne zastosowanie implementowanego zagadnienia

# Zaimplementowane przez nas narzędzie zostało stworzone w celu zapewnienia możliwości komunikacji między użytkownikami. Służy ono do przesyłania wiadomości tekstowych, które są szyfrowane i podpisywanie podpisem cyfrowym. Przez to oferowana usługa komunikacji zapewnia bezpieczeństwo i niezaprzeczalność. W związku z rozwojem w dziedzinie technologii internetowych, a także nieustannym wzrostem popularności komunikacji mobilnej mamy do czynienia z coraz to większa ilością ataków na bezpieczeństwo i prywatność użytkowników. Dlatego też implementowany przez nas komunikator w szczególności służy zapewnieniu bezpieczeństwa dla osób z niego korzystających.

# Instrukcja używania programu

## aplikacja serwera

Przed uruchomieniem upewnij się, że plik *usernames.xml* znajduje się w tym samym katalogu, co uruchamiany program. Uruchom program wybierając plik *TestApp.exe*.

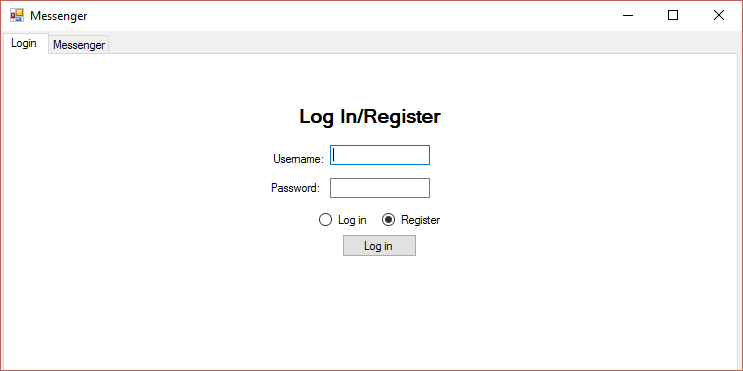
## aplikacja klienta

### Uruchamianie

Uruchom program wybierając plik *Client.exe.* Otworzy się okienko z rysunku 4.

### Logowanie/rejestracja

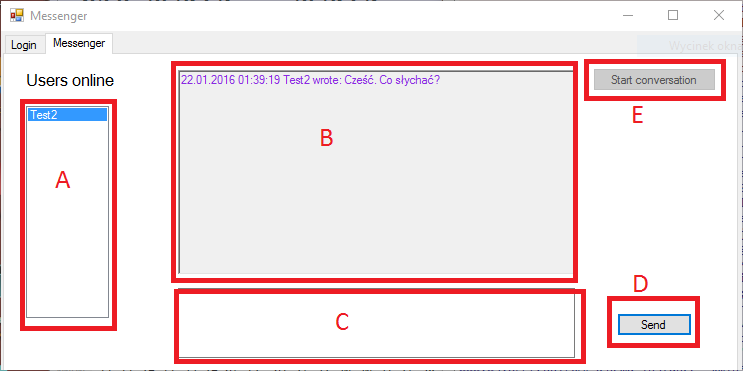
Wprowadź nazwę użytkownika i hasło w odpowiednie pola, wybierz czy chcesz się zalogować czy zarejestrować i naciśnij przycisk zaloguj. Jeżeli logowanie się powiodło przycisk zaloguj będzie nieaktywny. Możesz teraz przejść na zakładkę *Messenger*.



Rysunek 4. Główne okno programu.

### Wysyłanie wiadomości

Zakładka *Messenger* wygląda jak na rysunku 5. Składa się ona z listy użytkowników online (A), pola, w którym pokazywane będą rozmowy (B), pola do wpisywania wiadomości (C), przycisku wyślij (D) oraz przycisku *Start conversation*(E). Aby aktywować pole C należy wybrać użytkownika z pola A, nacisnąć przycisk E i poczekać na komunikat potwierdzający możliwość rozmowy. Jeżeli pole C pozostanie nieaktywne należy ponownie wybrać użytkownika w polu A. Wpisaną wiadomość wysyłamy poprzez naciśnięcie przycisku D bądź naciśnięcie klawisza Enter w polu C.



Rysunek 5. Okno komunikatora.

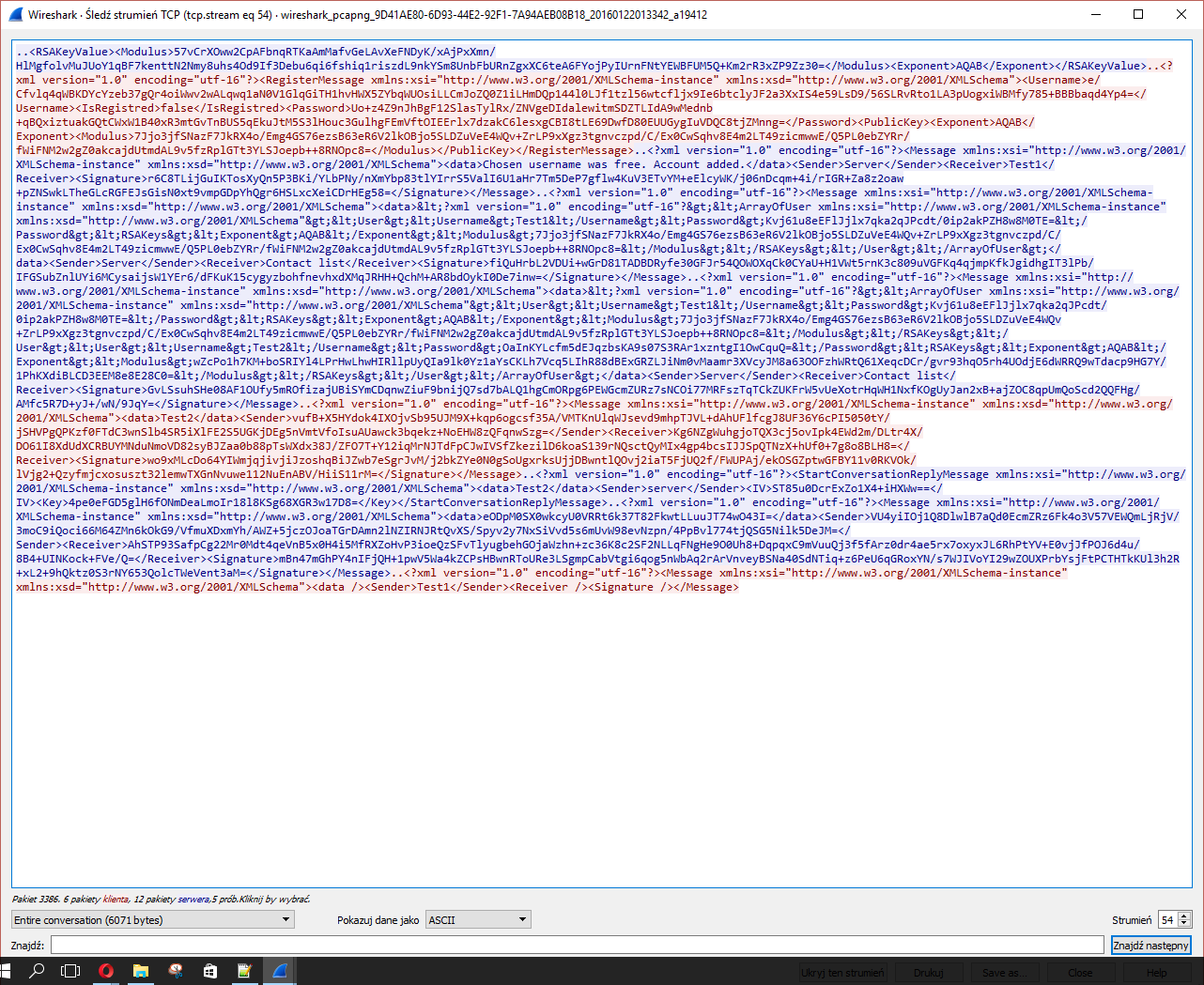
### Kończenie pracy

Aby zakończyć pracę należy nacisnąć przycisk krzyżyka znajdujący się w prawym górnym rogu.

# Raport z testów aplikacji

## sprawdzenie czy zapewniono usługę poufności

Aby sprawdzić czy nasz komunikator zapewnia usługę poufności włączyliśmy aplikację serwera oraz dwie instancje aplikacji klienta. Po zarejestrowaniu kont „Test1” i „Test2” z tego drugiego wysłaliśmy wiadomość „Cześć. Co słychać?”. Drugi klient odebrał tą samą wiadomość. Aby sprawdzić czy była ona przesłana w sposób bezpieczny użyliśmy programu Wireshark. Zapis z tego programu został dołączony do projektu w pliku *potwierdzenieTestu.pcapng*. Na rysunku 6 widać, że wiadomości, oraz ich nadawcy i odbiorcy są przesyłani w sposób tajny.



Rysunek 6. Zrzut ekranu z programu Wireshark

## sprawdzenie czy zapewniono usługę niezaprzeczalności

W naszym komunikatorze usługę niezaprzeczalności zapewniliśmy używając podpisu cyfrowego. Każda wiadomość jest weryfikowana przed jej wyświetleniem. Jeżeli podpis nie zgadza się wiadomość nie jest wyświetlana. Aby przetestować ten moduł zmodyfikowaliśmy metodę przekazującą wiadomości w serwerze. Zmodyfikowana metoda zmieniała wartości jednego z bitów podpisu cyfrowego. Zgodnie z przewidywaniami wiadomość docierała do odbiorcy, jednak nie była ona wyświetlana.

# Bibliografia

1. A. Menezes, P. van Oorschot, S. Vanstone: “Kryptografia stosowana”,
2. Z. Kotulski: Plansze do wykładów z przedmiotu PKRY,
3. <https://developer.mozilla.org/en-US/docs/Archive/Security/Digital_Signatures> ,
4. <https://developer.mozilla.org/en-US/docs/Archive/Security/Introduction_to_SSL>,
5. <http://e-handel.mm.com.pl/crypto/aes.htm>,
6. <http://e-handel.mm.com.pl/crypto/rijndael_ingles2004.swf>,
7. <http://www.crypto-it.net/pl/symetryczne/aes.html?tab=0>.