

KOMUNIKATOR Z PKI

Nazwa przedmiotu: *Protokoły kryptograficzne*

Student wykonujący: *Bartłomiej Rosa*

Prowadzący projekt: *mgr inż. Mariusz Sepczuk*

Semestr: 13Z

1. Wprowadzenie

1.1. Cel projektu

Celem projektu jest stworzenie bezpiecznej aplikacji służącej do komunikacji między użytkownikami (komunikatora), która będzie umożliwiała rejestrację i logowanie użytkowników, przeprowadzanie rozmów pomiędzy użytkownikami. Bezpieczeństwo aplikacji to zapewnienie zaszyfrowanych połączeń użytkowników oraz gwarancja potwierdzania tożsamości wszystkich użytkowników korzystających z aplikacji. Dzięki zaimplementowaniu Infrastruktury Klucza Publicznego (PKI – Public Key Infrastructure) wymagania te mają zostać spełnione. PKI jest zdefiniowane w dokumencie Internet X.509 Public Key Infrastructure. PKI oparte jest o kryptografię asymetryczną. W modelu tym każda jednostka (końcowa jak użytkownik, serwer, czy podstawowa jak Urząd Certyfikacji) posiada swój klucz prywatny, chroniony przed kradzieżą, oraz klucz publiczny, który jest udostępniany. Najprościej zabezpieczyć komunikację przez szyfrowanie wiadomości kluczem prywatnym i odszyfrowywanie kluczem publicznym (wykorzystanie RSA).

2. Opracowanie teoretyczne

2.1. Pojęcia podstawowe

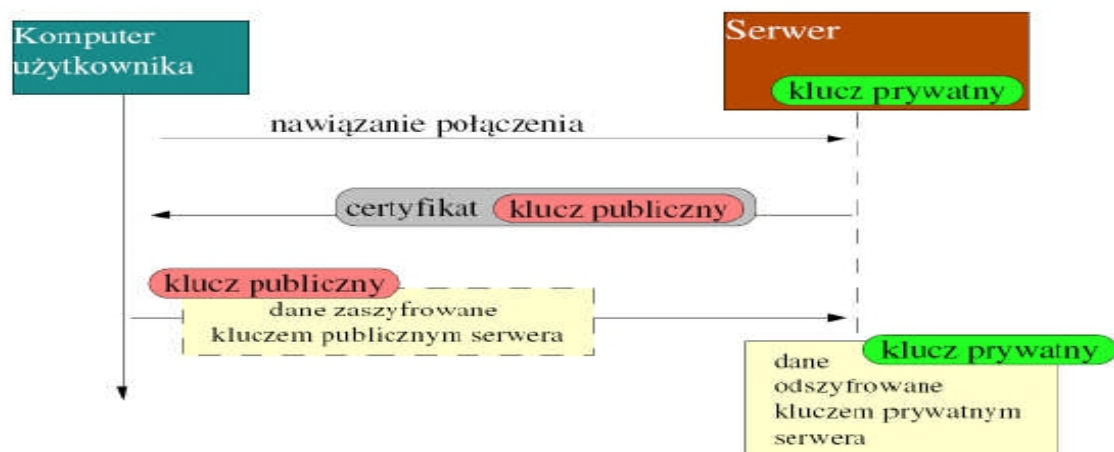
PKI

Infrastruktura Klucza Publicznego umożliwia tworzenie, przechowywanie, zarządzanie i rozprowadzanie certyfikatów klucza publicznego. Dzięki temu możemy prowadzić komunikację z ufnością, że:

- osoba jest tym, za kogo się podaje
- integralność danych nie jest naruszona (to, co wysyłamy i odbieramy jest tym co nadaliśmy i zostało nam nadane)

Kryptografia asymetryczna

Kryptografia asymetryczna znana także jako kryptografia klucza publicznego używa algorytmów, w których Alicja ma klucz prywatny, a Bob (i inni) mają jej klucz publiczny. Klucz publiczny i prywatny jest generowany w tym samym czasie, a dane zaszyfrowane jednym kluczem mogą być odszyfrowane tym drugim. Oznacza to, że każdy może zaszyfrować wiadomość używając znanego klucza publicznego Alicji, ale tylko właścicielka klucza prywatnego, czyli Alicja może wiadomość rozszyfrować. Algorytmy klucza publicznego są wolne w porównaniu z algorytmami symetrycznymi i słabo nadają się do szyfrowania dużych wiadomości. Kryptografia asymetryczna (algorytm RSA) zostanie wykorzystana w projekcie do zaszyfrowanej wymiany wiadomości. Rysunek 1 przedstawia sytuację, gdy wiadomość od użytkownika do serwera ma zostać zaszyfrowana (kluczem publicznym serwera) i odszyfrowana (kluczem prywatnym serwera).



Rys. 1. Przykład szyfrowania asymetrycznego

Kryptografia symetryczna

W kryptografii symetrycznej do szyfrowania i deszyfrowania jest używany ten sam klucz. Standardowym algorytmem jest AES. Szyfrowanie algorytmem AES zostanie użyte w projekcie do tego, aby można było zapisywać zaszyfrowane klucze prywatne na dysku.

Jednokierunkowa funkcja skrótu

Zadaniem jednokierunkowej funkcji skrótu wiadomości jest zapewnienie integralności danych. Argumentem tej funkcji jest wiadomość o dowolnej długości a wartością tej funkcji jest łańcuch bitów o określonej długości.

Uwierzytelnianie

Uwierzytelnianie jest sprawdzeniem, czy osoba jest tą za kogo się podaje. Jednym z przypadków potrzeby potwierdzenia swej tożsamości jest komunikacja między dwoma osobami. Sama dystrybucja klucza publicznego nie wystarcza, ponieważ pozostaje problem z potwierdzeniem tożsamości. Rozwiązaniem jest certyfikat klucza publicznego. W projekcie zostanie zastosowane uwierzytelnianie z wykorzystaniem kryptosystemu klucza publicznego. Przykład takiego uwierzytelnienia:

- host 1 i host 2 wymieniają się certyfikatami
 - host 2 wysyła „Powitanie 2”
 - host 1 odsyła „Powitanie 2” zaszyfrowane swoim kluczem prywatnym
 - host 2 sprawdza to z kluczem publicznym z certyfikatu host 1
- Analogicznie w odwrotną stronę, jeżeli uwierzytelnić się ma host 2 hostowi 1.

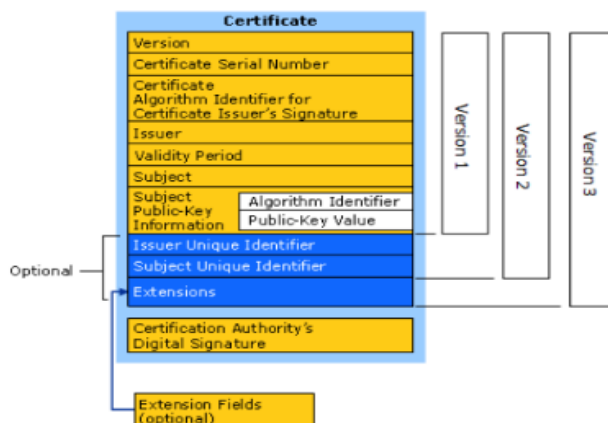
Integralność danych

Pojęcie integralności związane jest z niezmiennością danych. Celem jest, aby dane podczas transmisji od nadawcy do odbiorcy nie zostały zmodyfikowane. Główną metodą weryfikacji integralności danych jest jednokierunkowa funkcja skrótu. Jakakolwiek zmiana w wiadomości powoduje wygenerowanie innego wyniku tej funkcji. Taka funkcja może zapewnić integralność danych, ale aby można było ją stosować trzeba uniknąć sytuacji modyfikacji zarówno wiadomości jak i funkcji skrótu tej wiadomości.

Certyfikat

Certyfikat to dokument, który łączy klucz publiczny oraz dane jednostki, której klucz jest własnością. Aby certyfikat był wiarygodny musi być ustanowiony przez zaufaną przez wszystkich stronę, która

podpisze certyfikat (tzw. Urząd Certyfikacji). Standardem definiującym schemat dla certyfikatów klucza publicznego jest X.509. W projekcie wszystkie klienty, serwer, Urząd Certyfikacji mają swoje certyfikaty klucza publicznego X.509. Certyfikat zobrazowany jest na Rys. 2. a opisany poniżej.



Rys. 2. Certyfikat X.509

Podstawowe elementy certyfikatu X.509 to:

Numer wersji (Version)

Numer wersji formatu certyfikatu. Określa składnię certyfikatu wynikającą z wersji (wersja 1,2,3). Wersja 1 nie obsługuje rozszerzeń.

Numer seryjny (Certificate Serial Number)

Numer przydzielony certyfikatowi przez CA. Unikalny dla danego CA. Jest liczbą całkowitą. Połączenie nazwy CA oraz numeru seryjnego jednoznacznie identyfikują każdy certyfikat.

Identyfikator algorytmu (Certificate Algorithm Identifier for Certificate Issuer's Signature)

Określa algorytm użyty do podpisania certyfikatu i jego parametry

Identyfikator wystawcy (Issuer)

Nazwa CA, który wydał i podpisał certyfikat.

Okres ważności (Validity Period)

Data początku i końca ważności certyfikatu.

Użytkownik certyfikatu (Subject)

Określa użytkownika.

Informacja o kluczu publicznym (Subject Public-Key Information)

Klucz publiczny użytkownika oraz identyfikator algorytmu.

Podpis cyfrowy (Certification Authority Digital Signature)

Uwierzytelnia pochodzenie certyfikatu. Funkcja skrótu jest stosowana do wszystkich pól certyfikatu (oprócz pola podpisu). Wynik haszowania jest szyfrowany kluczem prywatnym CA.

Rozszerzenia

Informacje dodatkowe. To opcjonalne pole pojawia się tylko w wersji trzeciej certyfikatu X.509.

Certyfikat zawiera jedno lub więcej rozszerzeń. Każde rozszerzenie zawiera swój identyfikator, flagę

krytyczną (czy musi być stosowane), wartość rozszerzenia. Identyfikator rozszerzenia określa format wartości występującej w rozszerzeniu. Flaga krytyczna określa, czy dane rozszerzenie musi być używane, czy może być zignorowane. Najważniejsze rozszerzenia i ich omówienie:

- *basic constraints* mówi o ograniczeniach stosowania certyfikatu (czy może on służyć do tworzenia nowych)
- *key usage* ustala do czego może być wykorzystywany klucz publiczny zawarty w certyfikacie (niezaprzeczalność, szyfrowanie danych, podpisywanie list CRL)
- *subject alternative name* pozwala na umieszczenie w certyfikacie dodatkowych nazw określających podmiot korzystający z certyfikatu (na przykład adres e-mail)
- *CRL distribution points* zawiera informacje, gdzie informacja na temat unieważnionych certyfikatów mogą się znajdować (gdzie jest lista CRL)

2.2. Funkcje realizowane przez PKI

Rejestracja

Proces za pomocą którego dana jednostka przedstawia się Urzędowi Certyfikacji. W uproszczeniu Urząd Rejestracji może być powiązany z Urzędem Certyfikacji.

Certyfikowanie

Proces, w którym CA tworzy certyfikat i przekazuje go właścicielowi.

Generowanie kluczy

Para kluczy (symetryczny i asymetryczny) może być generowana lokalnie przez użytkownika lub przez CA. W projekcie każdy element PKI sam generuje parę kluczy.

Unieważnianie

Różne okoliczności mogą spowodować przedterminową utratę ważności certyfikatu. Może to być zmiana powiązania między klientem a Urzędem Certyfikacji, ujawnienie lub podejrzenie ujawnienia klucza prywatnego.

2.3. Elementy PKI

Urząd Certyfikacji

Potwierdza on autentyczność podmiotu jak i samego certyfikatu. Każdy użytkownik PKI musi mieć zaufanie do CA, aby mógł zaufać certyfikatowi i jego zawartości. Jego klucz publiczny jest stosowany bezpośrednio lub pośrednio do podpisywania wszystkich certyfikatów w strukturze PKI.

Wymienione zadania Urzędu Certyfikacji:

- poświadcza swoim autorytetem powiązanie klucza publicznego z danymi użytkownika przez certyfikat
- dokonuje tego na podstawie jasno sformułowanych metod i procedur ogłoszonych publicznie – ściśle stosując się do utworzonej w ten sposób polityki bezpieczeństwa
- publikuje listy certyfikatów unieważnionych – CRL (podpisane przez klucz prywatny CA)

Urząd Rejestracji

Odpowiedzialny jest za rejestrację użytkowników – służy do weryfikacji zawartości żądania wydania certyfikatu. W uproszczonym modelu połączony jest częścią CA.

Repozytoria certyfikatów, lista unieważnionych certyfikatów (CRL)

W repozytorium znajdują się wszystkie podpisane przez Urząd Certyfikacji certyfikaty.

Lista unieważnionych certyfikatów musi być publicznie dostępna dla wszystkich korzystających z danej PKI. Dzięki temu użytkownicy mogą sprawdzać, czy certyfikaty innych są jeszcze ważne. Certyfikat jest dodawany do listy unieważnionych w kilku przypadkach:

- wyciekł klucz prywatny użytkownika
- użytkownik nie jest dłużej certyfikowany przez CA
- certyfikat CA nie jest dłużej ważny

Schemat CRL jest przedstawiony na Rysunku 3. CRL zawiera poniższe pola:

Wersja (Version)

Opcjonalne pole określające składnię CRL (zazwyczaj wersja 2).

Podpis (Signature)

Zawiera identyfikator algorytmu podpisu cyfrowego stosowanego przez CA do podpisania CRL.

Wystawca (Issuer)

Zawiera nazwę CA, które wydało listę CRL.

Ta aktualizacja (This update)

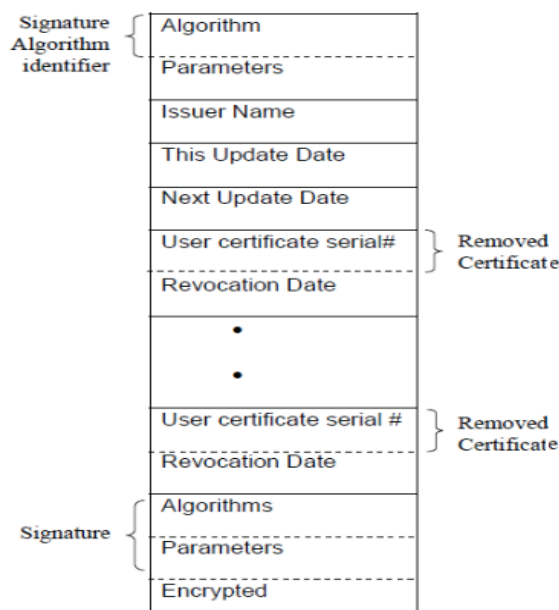
Pole zawiera datę opublikowania aktualnej listy CRL

Następna aktualizacja (Next update)

Pole zawiera datę opublikowania następnej listy CRL

Unieważnione certyfikaty (Revoked certificates)

Struktura jest listą unieważnionych certyfikatów. Każdy element listy zawiera serial number certyfikatu, czas unieważnienia, opcjonalnie rozszerzenia CRL.



Rys. 3. Przedstawienie schematu CRL

Użytkownicy PKI

Użytkownicy PKI to elementy infrastruktury korzystające z certyfikatów, ale nie mogące podpisywać nowych certyfikatów. Zależą od innych komponentów PKI do uzyskiwania, weryfikacji certyfikatów.

2.4. Protokół SSL

Protokół SSL tworzy pomiędzy dwoma aplikacjami prywatne, wiarygodne połączenie umożliwiające

potwierdzanie tożsamości obu komunikujących się stron.

Uwierzytelnianie serwera i klienta – RSA

Ochrona poufności danych (szyfrowanie) – DES/AES

Ochrona autentyczności i integralności danych - HMAC

2.5. Wykorzystane narzędzia

openssl

Oprogramowanie openssl jest ogólnie dostępną biblioteką funkcji kryptograficznych. Zawiera też implementację protokołu SSL. Użytkownicy za pomocą tej biblioteki mogą samodzielnie tworzyć część Infrastruktury Klucza Publicznego poprzez utworzenie własnego Urzędu Certyfikacji. Program openssl to narzędzie wiersza poleceń do używania funkcji biblioteki openssl.

Program będzie używany do:

- Tworzenia kluczy RSA, DH i DSA.
- Wystawiania certyfikatów X.509, CSR oraz CRL.

openssl obsługuje dwa podstawowe formaty plików:

- PEM (Privacy Enhanced Mail) – domyślny, tekstowy, może zawierać wiele certyfikatów w jednym pliku, rozszerzenia: .pem, .crt
 - DER – opcjonalny, binarny, jeden plik zawiera jeden certyfikat, rozszerzenia: .der, .cer
- Domyślnie stosowany jest format PEM.

Serwer FTP

Za pomocą protokołu FTP będą udostępniane certyfikaty i lista unieważnionych certyfikatów.

Pakiet java.net.SSL

W pakiecie zawarte są klasy odpowiedzialne za transmisję z użyciem protokołu SSL. Wykorzystane w aplikacjach napisanych w języku Java.

Pakiet java.security

Konkretnie klasa KeyStore reprezentująca zbiór kluczy i certyfikatów zapisanych w pamięci.

keytool

Program, który pozwala na konwersję certyfikatów do postaci odczytywanej przez domyślne biblioteki Java (pliki formatu PKCS12). Pozwala poza tym zarządzać kluczami, certyfikatami użytkownika. Dodaje też CA do zbioru zaufanych wystawców certyfikatów.

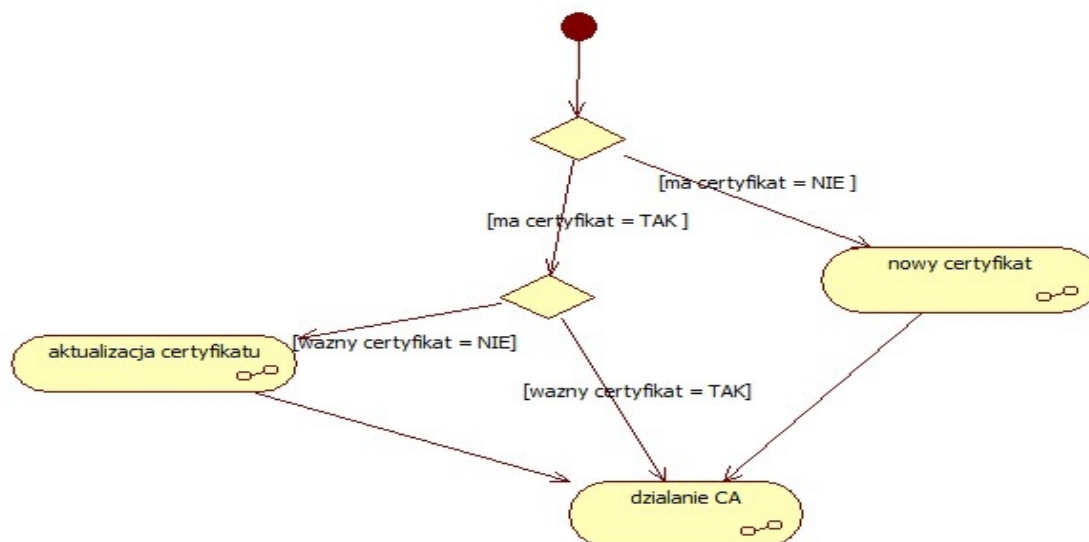
3. Koncepcja rozwiązania problemu

Na projekt składają się:

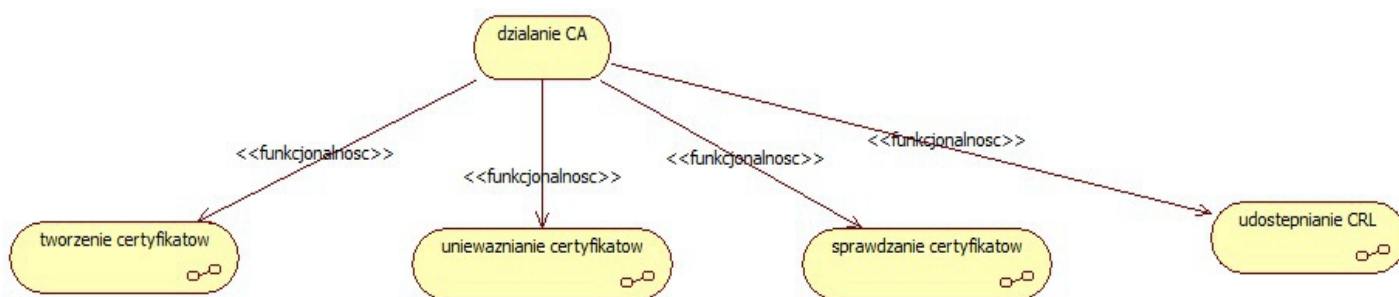
- Urząd Certyfikacji, z którym komunikują się proszące o certyfikaty klienty i serwer
- Serwer obsługujący komunikację klientów
- Klienci, czyli aplikacje służące do komunikacji z innymi użytkownikami w ramach komunikatora

3.1. Aplikacja Urzędu Certyfikacji

Urząd Certyfikacji działać będzie jako aplikacja napisana w Javie, która będzie korzystała z narzędzia openssl. Na Rysunkach 4 i 5 zostało zobrazowane jak ma wyglądać działanie aplikacji Urzędu Certyfikacji.



Rysunek 4. Uruchomienie aplikacji Urzędu Certyfikacji



Rysunek 5. Działalność Urzędu Certyfikacji

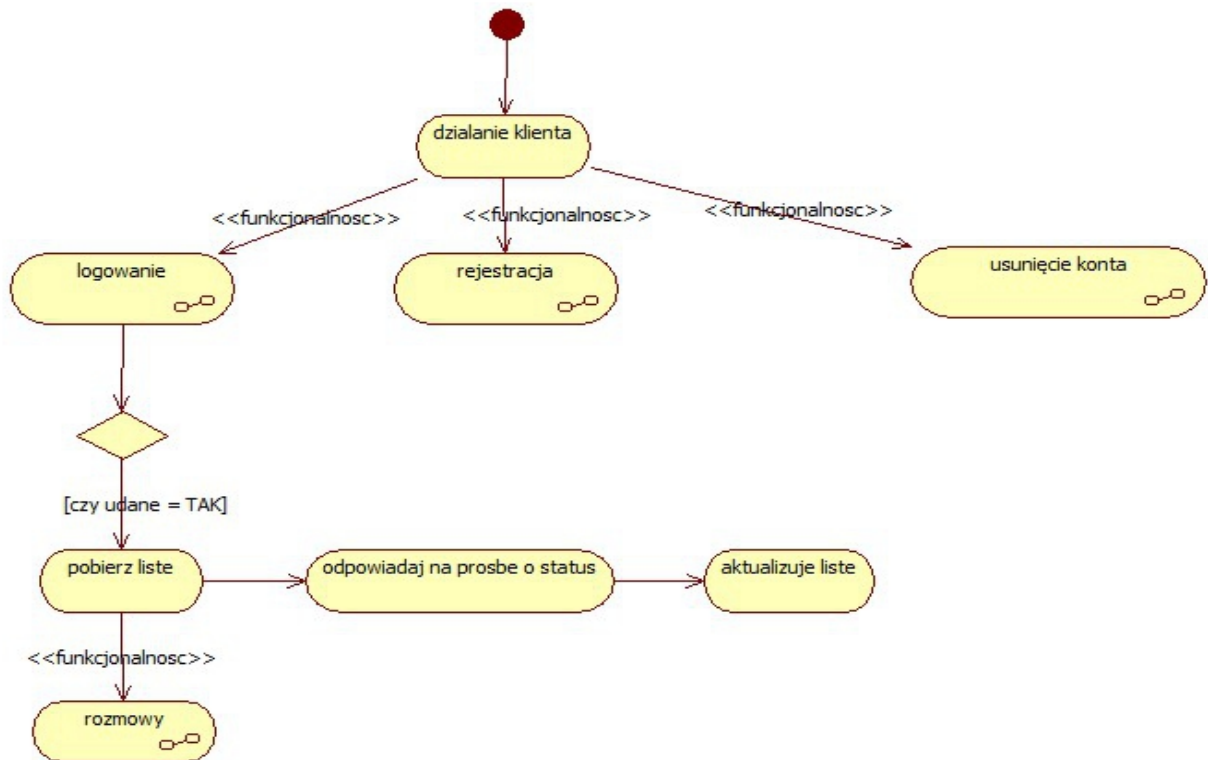
3.2. Klient (komunikator)

Klienta (aplikacja użytkownika) wykorzystuje użytkownik do korzystania z komunikatora. Musi ona realizować pewne funkcjonalności. Zarys działania aplikacji klienta przedstawiony jest na Rysunku 6. Działanie klienta składa się z kilku podstawowych funkcjonalności: możliwość logowania, rejestracji, usunięcia konta (unieważnienie certyfikatu), możliwość przeprowadzania rozmów z innymi użytkownikami.

Rejestracja – użytkownik rejestrując się wypełnia formularz rejestracyjny w aplikacji i tworzy w ten sposób żądanie certyfikacyjne. Klienta zapisuje otrzymany certyfikat i będzie z niego korzystał podczas komunikacji wykorzystującej protokół SSL.

Logowanie – klient musi uwierzytelnić się serwerowi. Gdy już się uwierzytelnia, to serwer dodaje go do listy klientów połączonych.

Rozmowa – inaczej wymiana wiadomości tekstowych pomiędzy użytkownikami komunikatora. Klient wysyła żądanie do serwera, że chce się połączyć z innym klientem. Serwer realizuje to i umożliwia rozmowę między klientami.



Użytkownik klienta przy uruchomieniu aplikacji będzie miał przedstawione w Tabeli 1 funkcjonalności.

Tabela 1. Przy uruchomieniu

Rejestracja
Logowanie
Usunięcie konta

Gdy postanowi zarejestrować nowego użytkownika, pokażą mu się pola przedstawione w Tabeli 2. Hasło nie będzie zapisywane na dysku (stąd nie można go odzyskać). Służyć będzie do szyfrowania algorytmem AES klucza prywatnego.

Tabela 2. Pola do rejestracji

nazwa
mail
hasło

Użytkownik, aby się zalogować musi podać ustaloną podczas rejestracji nazwę oraz hasło. Pola te zostały przedstawione w Tabeli 3.

Tabela 3. Pola do logowania

nazwa
hasło

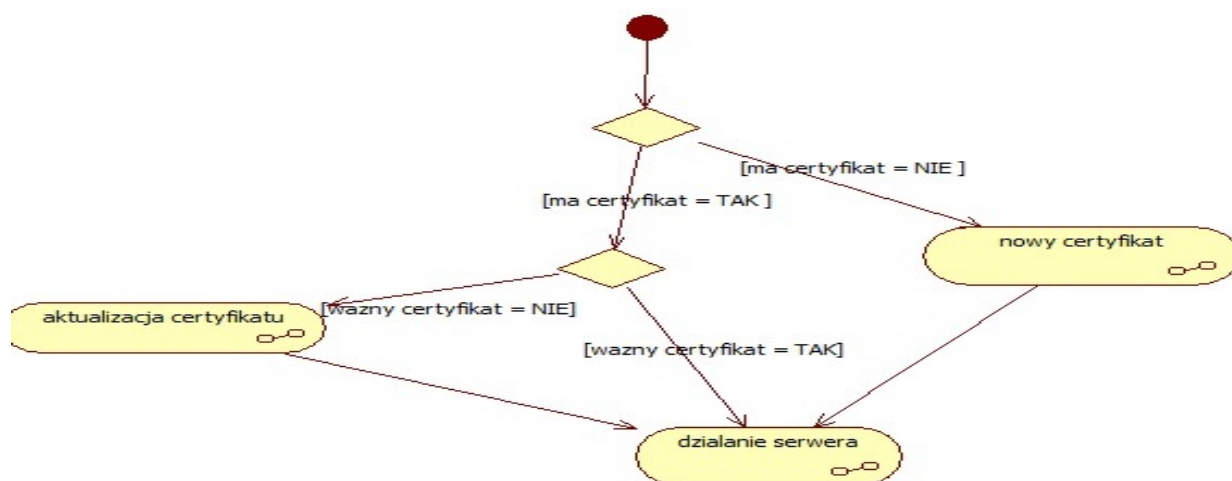
Po udanym zalogowaniu użytkownikowi zostanie przedstawiona lista wszystkich połączonych użytkowników. Przedstawia to Tabela 4. Użytkownik może przeprowadzać rozmowy z innymi, wybranymi użytkownikami.

Tabela 4. Lista użytkowników połączonych

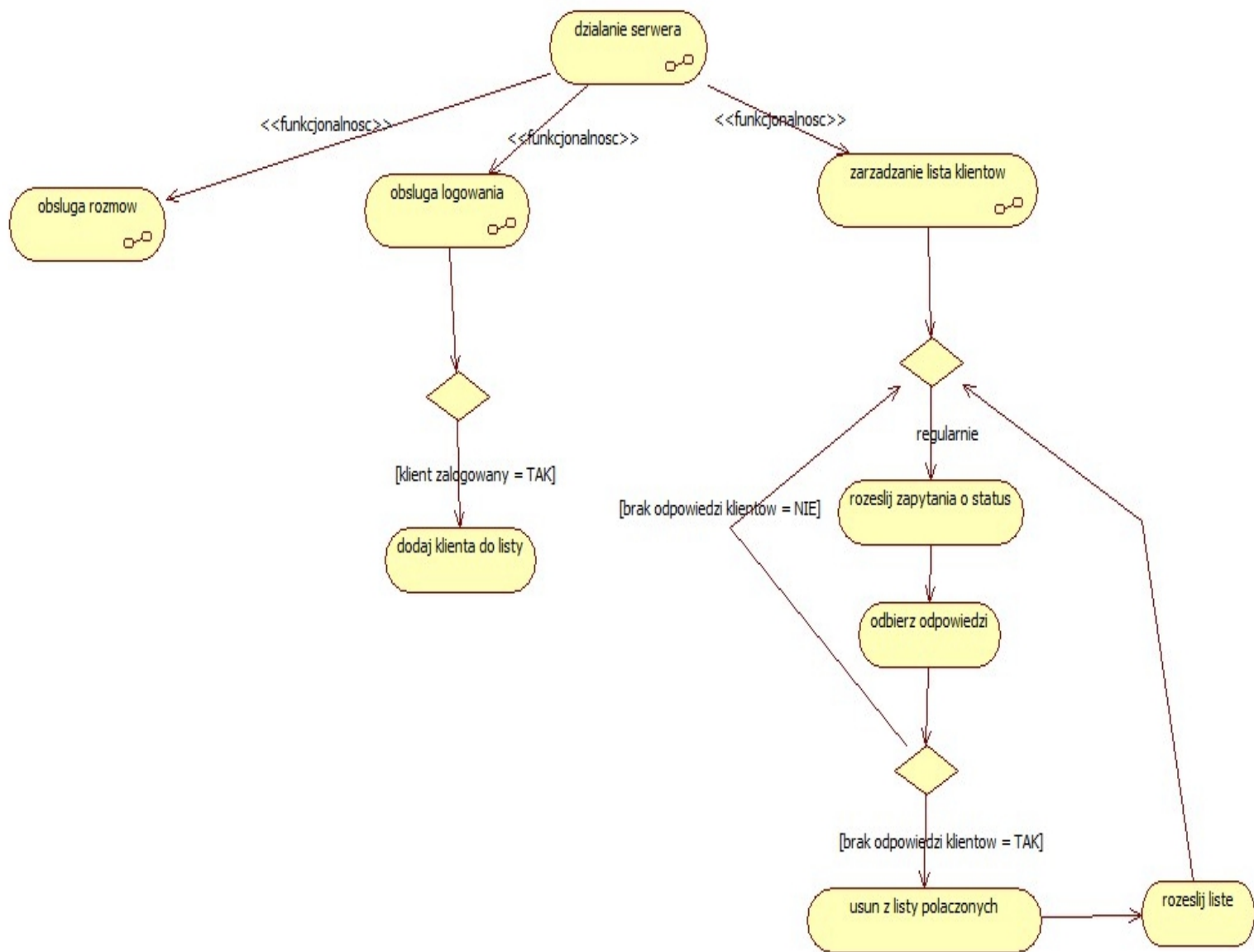
użytkownik1
użytkownik2
..

3.3. Aplikacja serwera

Zadaniem serwera jest obsługa klientów w celu umożliwienia komunikacji pomiędzy nimi.



Rysunek 7. Uruchomienie serwera

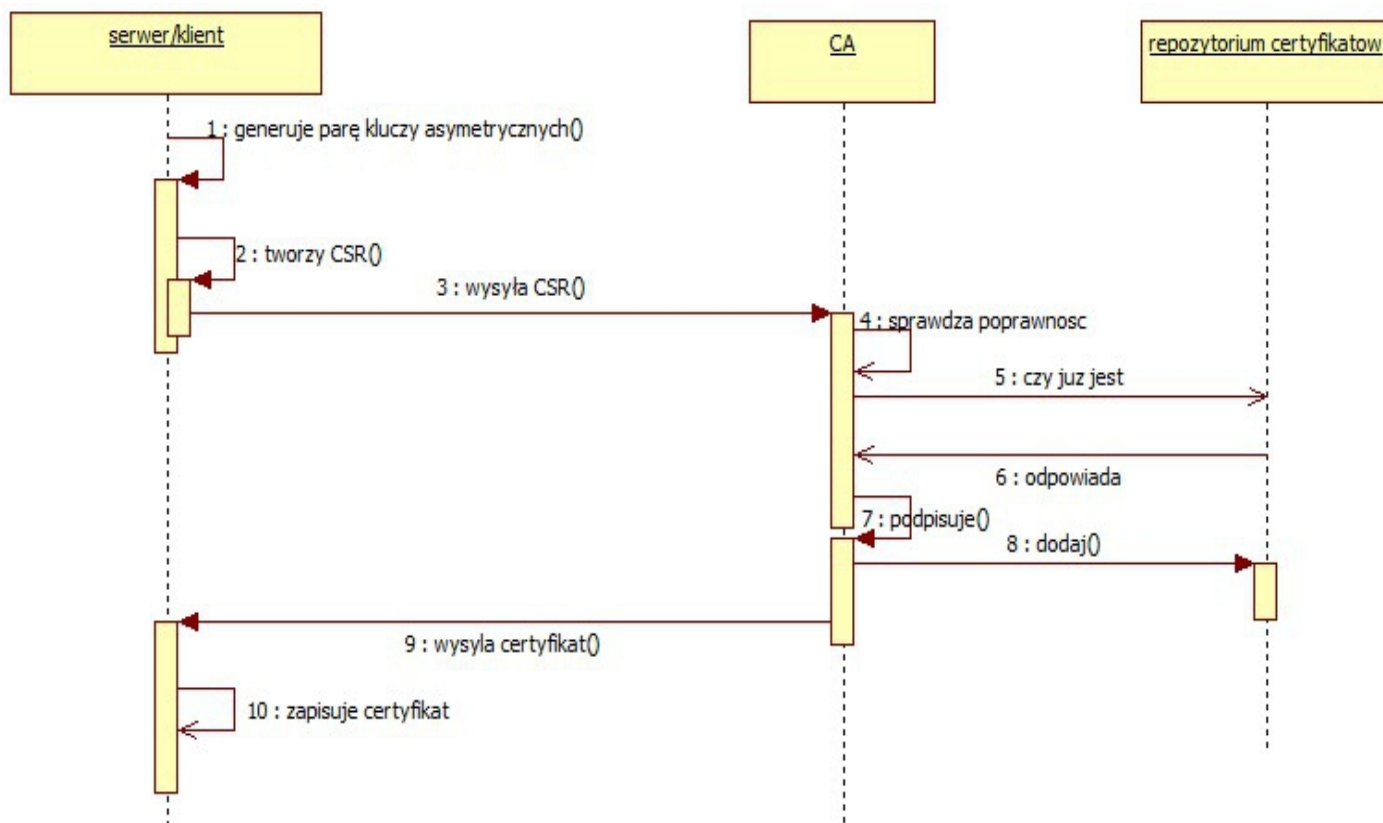


Rysunek 8. Działalność serwera

3.4. Operacje wykonywane w projekcie

Tworzenie certyfikatu

Serwer lub klient mogą starać się o certyfikat w Urzędzie Certyfikacji. W celu otrzymania certyfikatu musi zostać zobrażowana na Rysunku 9 procedura tworzenia certyfikatu. Rysunek przedstawia udaną procedurę, kiedy serwer lub klient otrzymują certyfikat.

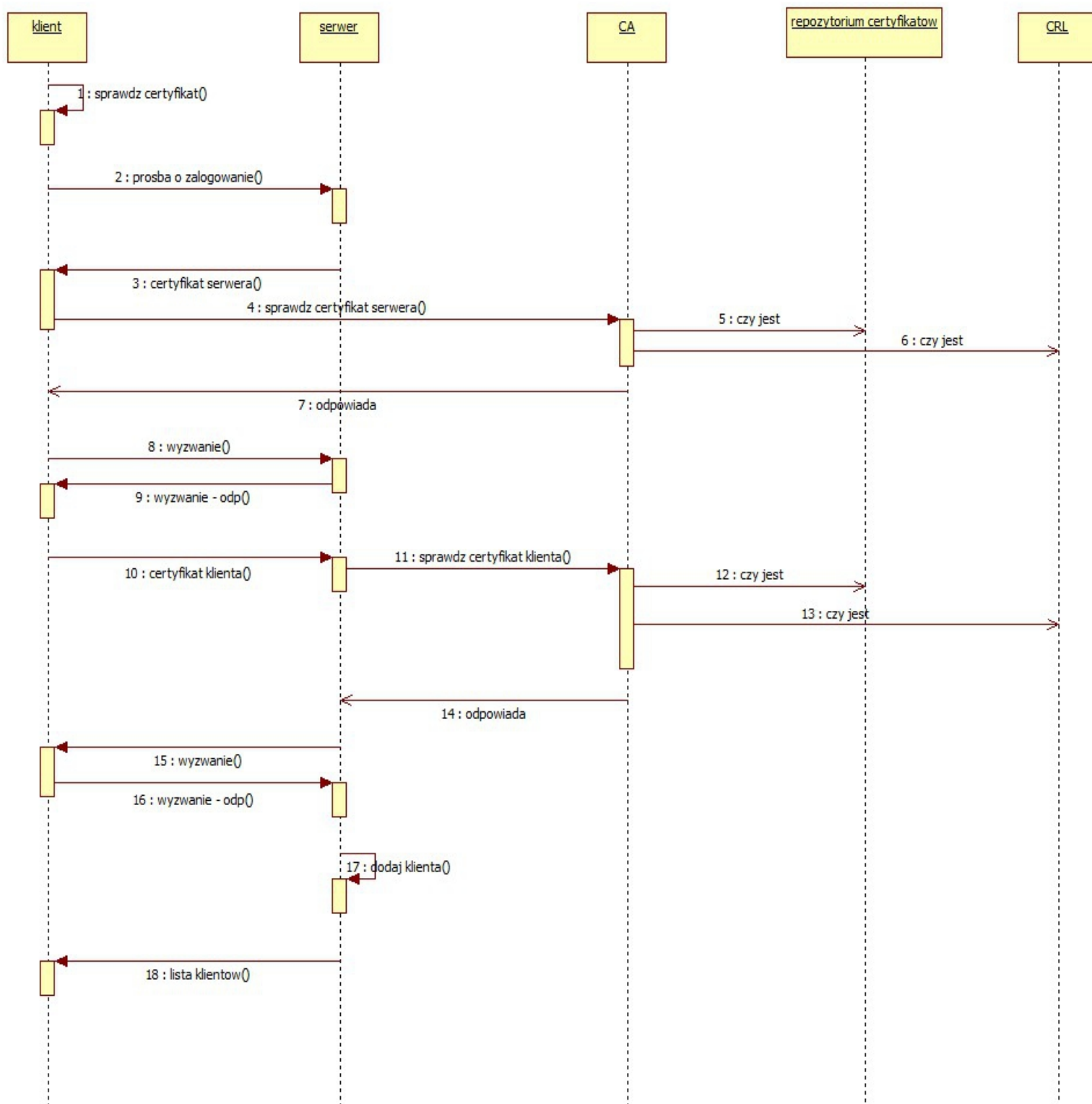


Rysunek 9. Procedura tworzenia certyfikatu

- (1) Generuje parę kluczy asymetrycznych i zapisuje zaszyfrowany klucz prywatny
- (2) Serwer/klient tworzy żądanie certyfikatu (CSR), łącząc:
 - a. klucz publiczny,
 - b. dodatkowe informacje identyfikacyjne: nazwę, adres e-mail
 - c. pożądane rozszerzenia certyfikatu: okres ważności, wykorzystanie (serwer/klient SSL)
- (4) Urzędu Certyfikacji sprawdza żądanie certyfikacyjne:
 - a. sprawdza techniczną poprawność (format) żądania
 - b. sprawdza czy polityka CA pozwala na podpisanie certyfikatu o podanych informacjach identyfikacyjnych
 - c. sprawdza czy żądane przez klienta rozszerzenia certyfikatu są zgodne z dopuszczalnymi
 - d. sprawdza w repozytorium, czy na podane dane nie ma już certyfikatu
- (7) Żądanie certyfikacyjne zostaje podpisane przez Urząd Certyfikacji tworząc certyfikat

Logowanie

Procedura logowania dotyczy klientów chcących korzystać z komunikatora (prowadzenia rozmów z innymi). Rysunek 10 przedstawia udaną procedurę logowania.



Rysunek 10. Procedura logowania

(1) Sprawdza datę ważności certyfikatu. Jeżeli upłynęła, to aktualizuje certyfikat.

(3-9) Na początku serwer przesyła swój certyfikat, później klient sprawdza w Urzędzie Certyfikacji, czy

certifikat jest ważny. Na końcu przesyła serwerowi wyzwanie (jakaś losowa wiadomość), gdzie serwer musi swoim kluczem prywatnym zaszyfrować je i odesłać klientowi.

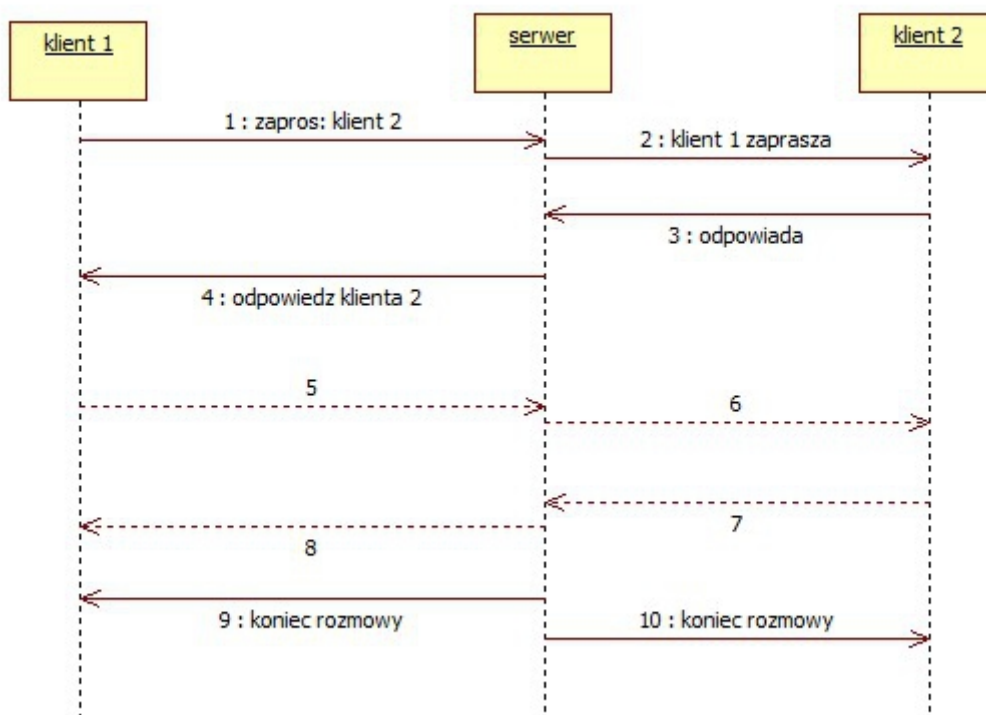
(10-16) Analogicznie do (3-9)

(17) Serwer po udanym uwierzytelnieniu klienta dodaje go do listy połączonych (jego IP, nazwę)

(18) Serwer przesyła klientowi pełną listę połączonych klientów

Rozmowa

Ustanowienie rozmowy pomiędzy dwoma klientami przedstawia Rysunek 11. Procedura przedstawia udane ustanowienie rozmowy, gdzie klient 2 zgadza się na rozmowę.



Rysunek 11. Procedura ustanowienia rozmowy

(5,6, 7, 8) Serwer przekazuje wiadomości klienta 1 do klienta 2 i na odwrót

(9,10) Kiedy jeden z klientów przerwie rozmowę (rozłączy się, zamknie okno), to serwer informuje drugiego klienta o zakończeniu rozmowy

Aktualizacja certyfikatu

Aktualizacja certyfikatu polega na uwierzytelnieniu się serwera/klienta i ponownej procedurze rejestracji.

Unieważnienie certyfikatu

Unieważnienie certyfikatu polega na uwierzytelnieniu się serwera/klienta i przesłaniu polecenia unieważnienia certyfikatu.

4. Instrukcja użytkownika

4.1. Administrator Urzędu Certyfikacji

Urząd Certyfikacji tworzony jest za pomocą narzędzia openssl. W folderze /etc znajdują się pliki konfiguracyjne używane do tworzenia certyfikatów CA, klientów, serwera. W folderze /ca/db są pliki:

ca.crl.srl (zawiera numer seryjny następnego certyfikatu w CRL),
ca.crt.srl (zawiera numer seryjny następnego certyfikatu do wystawienia),
ca.db (jest spisem wystawionych certyfikatów).

W folderze /certs znajdować się będą certyfikaty, żądania i klucze prywatne (każde o nazwie będącej numerem seryjnym). W folderze /crl znajdować się będzie plik ca.crl z informacją o unieważnionych certyfikatach. W folderze /etc znajdują się pliki konfiguracyjne do tworzenia CA, certyfikatów, CRL.

Przygotowanie stanowiska pracy:

1) Instalacja serwera Apache (z dodatkiem openssl)

2) Powiązanie adresu rosa.ca z adresem IP komputera

W pliku *C:\Windows\System32\drivers\etc\hosts* ustawiamy *127.0.0.1 rosa.ca*.

3) Przekopiowanie odpowiednich plików i folderów

W miejsce instalacji serwera Apache wchodzimy do folderu *bin* i tam przekopiuujemy wszystkie pliki i foldery (skrypty *makeCA*, *makeClient*, *makeServer*, *revokeCert*, foldery *ca*, *certs*, *crl*, *etc*).

4) Utworzenie Urzędu Certyfikacji

Uruchamiamy skrypt *makeCA.bat*, który wygeneruje CRL, certyfikat CA.

4) Konfigurujemy SSL w Apache

W tym celu zmieniamy odpowiednio ustawienia w plikach konfiguracyjnych Apache. Potrzeby jest jeszcze certyfikat serwera wraz z kluczem. Musimy uruchomić skrypt *makeServer.bat* i przekopiować do ustawionego w konfiguracji Apache folderu nasz wygenerowany certyfikat i klucz prywatny.

5) Przekopiowanie CRL

Wygenerowany plik *ca.crl* przekopiuujemy do folderu */htdocs/crl* na serwerze Apache. Plik *ca.crt* dodajemy do folderu */htdocs*.

6) Uruchomienie serwera Apache

Skrypt makeCA.bat

Skrypt ten tworzy certyfikat CA.

```
D:\Programy\Apache\bin>makeCA
D:\Programy\Apache\bin>openssl req -new -config etc/ca.conf -out ca/ca.csr -keyout ca/private/ca.key
Loading 'screen' into random state - done
Generating a 2048 bit RSA private key
.....+++
writing new private key to 'ca/private/ca.key'
-----

D:\Programy\Apache\bin>openssl ca -selfsign -config etc/ca.conf -in ca/ca.csr -out ca/ca.crt -extensions root_ca_ext -enddate 310101000000Z
Using configuration from etc/ca.conf
Loading 'screen' into random state - done
Check that the request matches the signature
Signature ok
Certificate Details:
  Serial Number: 7 (0x7)
  Validity
    Not Before: Jan 15 12:54:47 2014 GMT
    Not After : Jan 1 00:00:00 2031 GMT
  Subject:
    organizationName      = ROSA
    commonName            = ROSA CA
  X509v3 extensions:
    X509v3 Key Usage: critical
    X509v3 Certificate Sign, CRL Sign
    X509v3 Basic Constraints: critical
    CA:TRUE
    X509v3 Subject Key Identifier:
    AF:CA:D2:C7:5F:F9:DE:5C:17:89:78:82:1F:D5:06:5E:5C:02:14:DF
    X509v3 Authority Key Identifier:
    keyid:AF:CA:D2:C7:5F:F9:DE:5C:17:89:78:82:1F:D5:06:5E:5C:02:14:DF
Certificate is to be certified until Jan 1 00:00:00 2031 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]:y
Write out database with 1 new entries
Data Base Updated

D:\Programy\Apache\bin>openssl ca -genctrl -config etc/ca.conf -out ctrl/ca.ctrl
Using configuration from etc/ca.conf
Loading 'screen' into random state - done
D:\Programy\Apache\bin>
```

Rys. 12. Przykładowe wykonanie skryptu makeCA

Skrypt makeServer.bat

Skrypt ten tworzy certyfikat serwera.

UWAGA: Jako hasło do pliku .p12 ustawiamy „polska”.

```
D:\Programy\Apache\bin>makeServer
D:\Programy\Apache\bin>openssl req -new -config etc/server.conf -out certs/server.csr -keyout certs/server.key
Loading 'screen' into random state - done
Generating a 2048 bit RSA private key
.....+++
writing new private key to 'certs/server.key'
-----

D:\Programy\Apache\bin>openssl ca -config etc/ca.conf -in certs/server.csr -out certs/server.crt -extensions server_ext
Using configuration from etc/ca.conf
Loading 'screen' into random state - done
Check that the request matches the signature
Signature ok
Certificate Details:
  Serial Number: 8 (0x8)
  Validity
    Not Before: Jan 15 12:57:01 2014 GMT
    Not After : Jan 15 12:57:01 2015 GMT
  Subject:
    organizationName      = ROSA
    commonName            = server
  X509v3 extensions:
    X509v3 Key Usage: critical
    X509v3 Digital Signature, Non Repudiation, Key Encipherment, Data Encipherment, Key Agreement
    X509v3 Basic Constraints:
    CA:FALSE
    X509v3 Extended Key Usage:
    TLS Web Server Authentication, TLS Web Client Authentication
    X509v3 Subject Key Identifier:
    5C:CE:55:0F:AE:B2:16:E6:20:C6:D5:17:B3:76:12:78:12:B0:71:01
    X509v3 Authority Key Identifier:
    keyid:AF:CA:D2:C7:5F:F9:DE:5C:17:89:78:82:1F:D5:06:5E:5C:02:14:DF
    Authority Information Access:
    CA Issuers - URI:http://rosa.ca//ca.crt
    X509v3 CRL Distribution Points:
    URI:http://rosa.ca//ca.crl
    X509v3 Subject Alternative Name:
    DNS:rosa.ca
Certificate is to be certified until Jan 15 12:57:01 2015 GMT (365 days)
Sign the certificate? [y/n]:y

1 out of 1 certificate requests certified, commit? [y/n]:y
Write out database with 1 new entries
Data Base Updated

D:\Programy\Apache\bin>openssl pkcs12 -export -inkey certs/server.key -in certs/server.crt -certfile ca/02.pem -out certs/server.p12
Loading 'screen' into random state - done
Enter Export Password:
Verify - Enter Export Password:
D:\Programy\Apache\bin>
```

Rys. 13. Przykładowe wykonanie skryptu makeServer

Skrypt makeClient.bat

Służy on administratorowi do zrobienia certyfikatu dla nowego klienta. Bezpieczniejszą formą jest, żeby klient sam generował żądanie certyfikacyjne i nie udostępniał klucza prywatnego serwerowi.

UWAGA: Jako hasło do pliku .p12 ustawiamy „polska”.

```
D:\Programy\Apache\bin>makeClient
D:\Programy\Apache\bin>for /F "delims=" %x in (ca/db/ca.crt.srl) do set Build=%x
D:\Programy\Apache\bin>set Build=09
D:\Programy\Apache\bin>echo 09
09
D:\Programy\Apache\bin>openssl req -new -config etc/client.conf -out certs/09.csr -keyout certs/09.key
Loading screen into random state - done
Generating a 2048 bit RSA private key
.....+++
writing new private key to 'certs/09.key'
++++
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value.
If you enter '.', the field will be left blank.
-----
Set: ROSA []:ROSA
Your unique name: []:NowyKlient
Your email: []:nowy@rosa.ca
D:\Programy\Apache\bin>openssl ca -config etc/ca.conf -in certs/09.csr -out certs/09.crt -extensions client_ext
Using configuration from etc/ca.conf
Loading screen into random state - done
Check that the request matches the signature
Signature ok
Certificate Details:
  Serial Number: 9 (0x9)
  Validity
    Not Before: Jan 15 12:58:36 2014 GMT
    Not After : Jan 15 12:58:36 2015 GMT
  Subject:
    organizationName      = ROSA
    commonName            = NowyKlient
  X509v3 extensions:
    X509v3 Key Usage: critical
      Digital Signature, Non Repudiation, Key Encipherment, Data Encipherment, Key Agreement
    X509v3 Basic Constraints:
      CA:FALSE
    X509v3 Extended Key Usage:
      TLS Web Client Authentication
    X509v3 Subject Key Identifier:
      E0:D1:9C:66:55:1E:5C:23:08:AF:21:2D:E7:89:0F:C0:0F:1D:58:EC
    X509v3 Authority Key Identifier:
      keyId:AF:0A:D2:C7:5F:F9:DE:5C:17:89:78:02:1F:D5:06:5E:5C:02:14:DF
  Authority Information Access:
    CA Issuers - URI:http://rosa.ca/ca.crt
  X509v3 CRL Distribution Points:
    URI:http://rosa.ca/ca.crl
  X509v3 Subject Alternative Name:
    email:nowy@rosa.ca
Certificate is to be certified until Jan 15 12:58:36 2015 GMT (365 days)
Sign the certificate? [y/n]:y
```

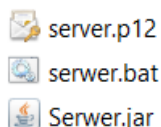
Rys. 14. Przykładowe wykonanie skryptu makeClient

Skrypt revokeCert.bat

Skrypt ten jest odpowiedzialny za unieważnianie certyfikatu – wcześniej trzeba go odpowiednio edytować. Po każdym unieważnieniu administrator.

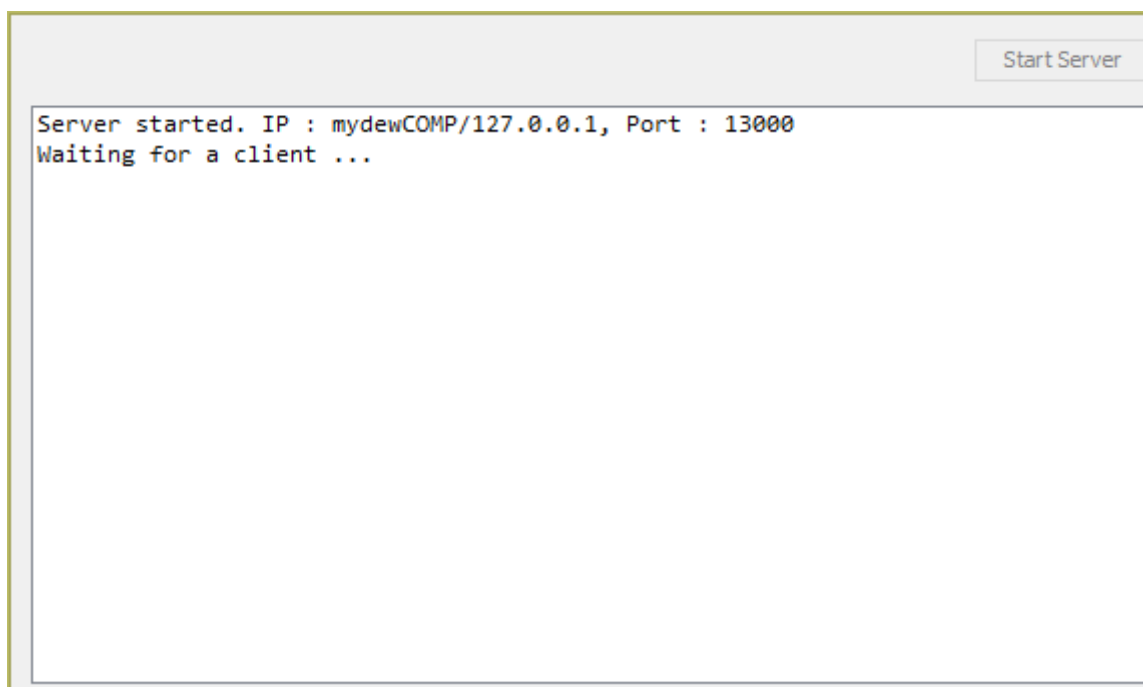
4.1. Administrator serwera

W pliku *hosts* systemu Windows należy zdefiniować IP dla adresu *rosa.ca*. Należy dodać certyfikat CA (pobrany ze strony *rosa.ca/ca.crt*) do zbioru zaufanych wystawców dla środowiska Java (narzędzie *keytool import*). W celu uruchomienia serwera potrzebny jest certyfikat serwera – należy wygenerować żądanie i zgłosić się do Urzędu Certyfikacji z prośbą o podpisanie żądania. Później należy z klucza prywatnego i certyfikatu utworzyć plik PKCS12 – w projekcie wszystko to jest robione po stronie Urzędu Certyfikacji skryptem *makeServer.bat*. Plik *server.p12* należy przekopiować do folderu aplikacji serwera.



Rys. 15. Pliki serwera

Aplikację uruchamiamy poprzez wykonanie skryptu *serwer.bat*. Serwer startujemy przyciskiem Start Server.

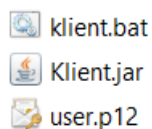


Rys. 16. Okno serwera

4.3. Użytkownik aplikacji klienckiej

Rejestracja

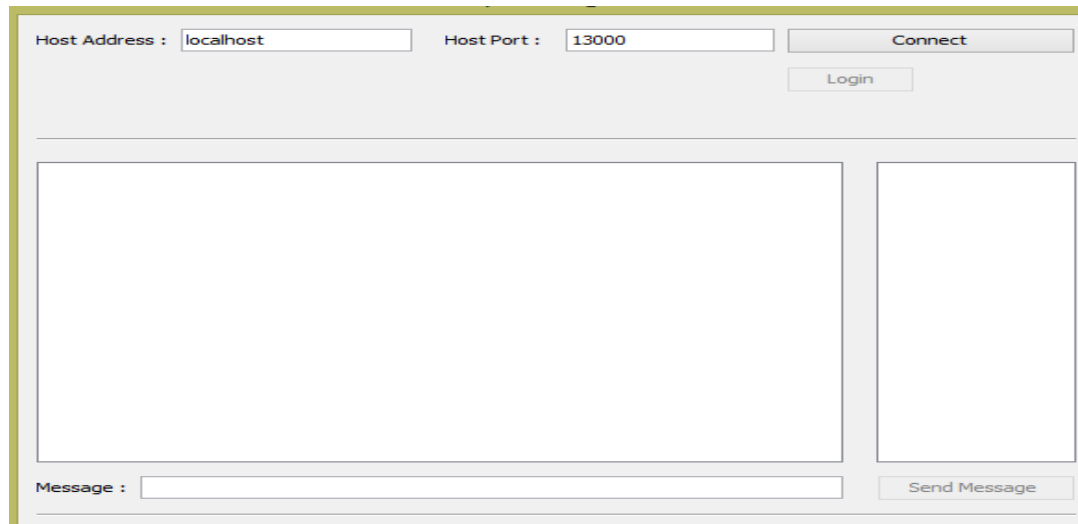
W pliku *hosts* należy zdefiniować IP dla adresu *rosa.ca*. Należy dodać certyfikat CA (pobrany ze strony *rosa.ca/ca.crt*) do zbioru zaufanych wystawców dla środowiska Java (narzędzie *keytool import*). W celu zarejestrowania użytkownika w aplikacji musi on zgłosić się do administratora, który wygeneruje użytkownikowi odpowiedni certyfikat i klucz prywatny. Użytkownik musi zapisać otrzymane pliki w folderze swojej aplikacji. Klucz jest kluczem prywatnym i nie może być udostępniany innym użytkownikom. Po poprawnym przeprowadzeniu tych operacji użytkownik powinien mieć w swoim folderze aplikacji klienckiej pliki przedstawione na Rysunku 17.



Rys. 17. Pliki klienta

Logowanie i rozmowa

W celu zalogowania użytkownik aplikacji klienckiej uruchamia skrypt *klient.bat*. Pokaże mu się okienko aplikacji przedstawione na Rysunku 18.



The screenshot shows a client application window. At the top, there are two input fields: "Host Address" with the value "localhost" and "Host Port" with the value "13000". To the right of these fields are two buttons: "Connect" and "Login". Below the input fields is a large empty rectangular area, likely for displaying chat messages. At the bottom, there is a "Message" input field and a "Send Message" button.

Rys. 18 Aplikacja klienta

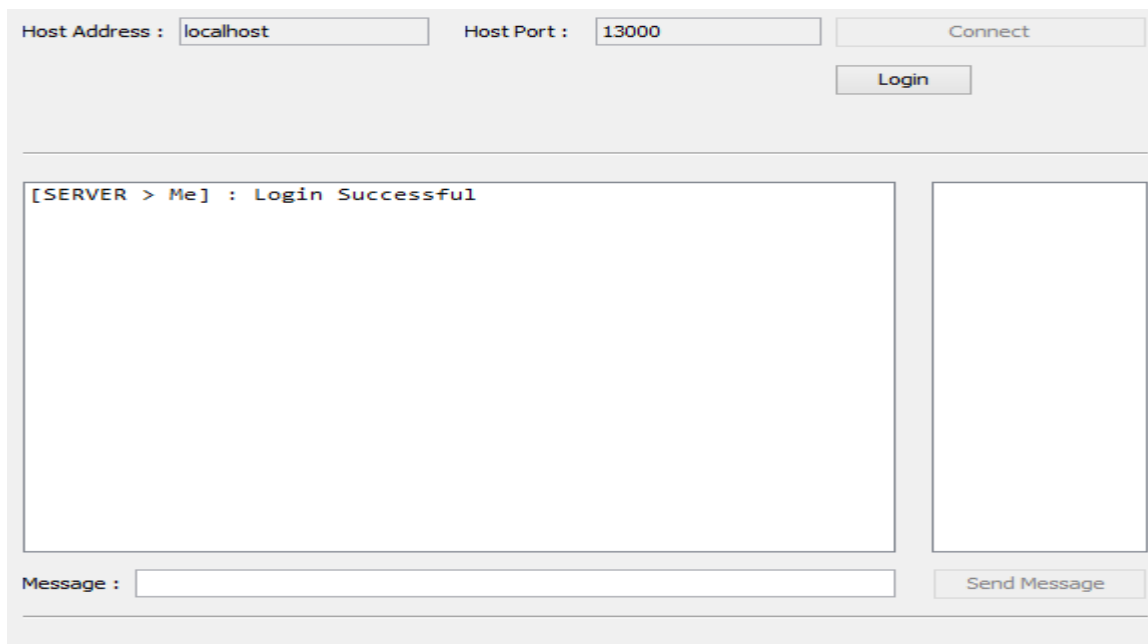
W polu Host Address wpisujemy adres naszego serwera.

W polu Host Port wpisujemy port naszego serwera.

W celu zainicjowania połączenia klikamy na przycisk Connect.

Później w celu zalogowania klikamy Login.

Jeżeli poprawnie się zalogowaliśmy, to ukaże się komunikat z Rysunku 14.



The screenshot shows the same client application window as in Rys. 18, but now the large rectangular area displays the message "[SERVER > Me] : Login Successful". The "Host Address" field still contains "localhost" and the "Host Port" field still contains "13000". The "Connect" and "Login" buttons are still present. The "Message" input field and "Send Message" button are also visible at the bottom.

Rys. 19. Po poprawnym zalogowaniu

Host Address : Host Port :

[SERVER > Me] : Login Successful

michal

Message :

Rys. 20. Wybór rozmówcy

Host Address : Host Port :

[SERVER > Me] : Login Successful
[michal > bartek] : Czesć!

bartek

Message :

Host Address : Host Port :

[SERVER > Me] : Login Successful
[michal > bartek] : Czesć!
[bartek > Me] : Cześć ;)

bartek

Message :

Rys. 21. Przykładowa rozmowa

Unieważnianie konta

W celu unieważnienia konta użytkownik musi zgłosić się ze swoim kluczem prywatnym oraz certyfikatem do administratora, który będzie mógł dodać certyfikat do listy unieważnionych.

Unieważnienie konta oznacza, że certyfikat nie będzie mógł być wykorzystywany do połączeń w komunikatorze.

5. Testy

- 1) Transmisja jest szyfrowana
- 2) Gdy certyfikat jest unieważniony, to użytkownik nie jest w stanie się zalogować na serwer.
- 3) Gdy certyfikat jest podpisany przez niewłaściwe CA lub jest podpisane zbyt słabym kluczem (poniżej 2048 bitów) to użytkownik nie jest w stanie zalogować się na serwer.
- 4) Na serwer nie zaloguje się dwóch użytkowników mających te same nazwy

5) Nazwy użytkowników są dla CA unikalne – na taką samą nazwę CA nie wystawi certyfikatu

```
C:\Apache24\bin>for /F "delims=" %x in (ca/db/ca.crt.srl) do set Build=%x
C:\Apache24\bin>set Build=07
C:\Apache24\bin>echo 07
07
C:\Apache24\bin>openssl req -new -config etc/client.conf -out certs/07.csr -keyout certs/07.key
Loading 'screen' into random state - done
Generating a 2048 bit RSA private key
.....+++
writing new private key to 'certs/07.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Set: ROSA []:ROSA
Your unique name: []:bartek
C:\Apache24\bin>openssl ca -config etc/ca.conf -in certs/07.csr -out certs/07.crt -extensions client_ext
Using configuration from etc/ca.conf
Loading 'screen' into random state - done
Check that the request matches the signature
Signature ok
Certificate Details:
  Serial Number: 7 (0x7)
  Validity
    Not Before: Jan 24 17:29:18 2015 GMT
    Not After : Jan 24 17:29:18 2015 GMT
  Subject:
    organizationName      = ROSA
    commonName            = bartek
  X509v3 extensions:
    X509v3 Key Usage: critical
      Digital Signature, Non Repudiation, Key Encipherment, Data Encipherment, Key Agreement
    X509v3 Basic Constraints:
      CA:FALSE
    X509v3 Extended Key Usage:
      TLS Web Client Authentication
    X509v3 Subject Key Identifier:
      17:8D:C3:D1:6C:42:14:E9:84:F0:EC:A1:78:2A:1D:25:01:4C:FA:E8
    X509v3 Authority Key Identifier:
      key id:02:1B:1E:BC:27:E0:B6:DE:7B:C5:2F:64:DC:6B:56:48:31:53:93:CB
  Authority Information Access:
    CA Issuers - URI:http://rosa.ca/ca.crt
  X509v3 CRL Distribution Points:
    URI:http://rosa.ca/ca.crl
  X509v3 Subject Alternative Name:
    <EMPTY>
Certificate is to be certified until Jan 24 17:29:18 2015 GMT (365 days)
Sign the certificate? [y/n]:y
failed to update database
TXT_DB error number 2
```

6) Certyfikat klienta nie może być użyty jako certyfikat serwera

6. Podsumowanie

Podsumowując, projekt nie został zrealizowany zgodnie z pierwotną koncepcją. Wiele rzeczy zostało uproszczonych, z innych zrezygnowano.

7. Bibliografia

1. "openssl PKI Tutorial"

<https://pki-tutorial.readthedocs.org/en/latest/>

2. "Public Key Infrastructure. Implementation and Design", Suranjan Choudhury

<http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0764548794.html>

3. "PKI. Podstawy i zasady działania", Adams Carlisle, Lloyd Steve

http://helion.pl/ksiazki/pki-podstawy-i-zasady-dzialania-adams-carlisle-lloyd-steve_a_004g.htm

4. "DESIGN AND IMPLEMENTATION OF PUBLIC KEY. INFRASTRUCTURE: A PROPOSED SOLUTION FOR BANGLADESH", Aslam Parvez

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.130.8124&rep=rep1&type=pdf>