USMTP

Dokumentacja protokołu Ultra Simple Mail Transfer Protocol

Autorzy protokołu: Michał Jakubowski, Mateusz Kopczan

Spis treści

1. Protokół USMTP

Protokół USMTP jest to protokół wysyłki maila z dodatkową możliwością dodania załączników do przesyłanej wiadomości. Protokół ten został stworzony ze zwróceniem szczególnej uwagi na bezpieczeństwo i niezawodność komunikacji. Całość połączenia jest w pełni szyfrowana, dzięki czemu korzystanie z protokołu jest w pełni bezpieczne i maksymalnie uproszczone.

1.1 Opis funkcjonalności

Główną funkcjonalnością protokołu jest możliwość wysyłki maili. Użytkownik korzystając z protokołu ma możliwość zalogowania się na swoje konto wyznaczając tym samym nadawcę wiadomości. Dodatkowo może on wskazać odbiorcę wiadomości i inne kluczowe informację o wiadomości takie jak temat, treść i załączniki. Serwer protokołu zrywa połączenie dopiero, gdy otrzyma taki sygnał od użytkownika lub w momencie wystąpienia błędu krytycznego, który automatycznie zerwie połączenie z klientem. Oznacza to, że użytkownik ma możliwość wysyłki wielu wiadomości jednej po drugiej bez konieczności ciągłego logowania.

1.2 Kody odpowiedzi

Aplikacja ze względu na wykonywane funkcję zwraca wiele kodów odpowiedzi, które można podzielić na kilka głównych grup:

- 1xy – informacyjne,
- 2xy – odpowiedź pozytywna,
- 3xy – błędy składniowe,
- 4xy – błędy po których wymagane jest ponowne przesłanie danych,
- 5xy – błędy po których połączenie jest zamykane.

Informacyjne kody odpowiedzi:

- 100 – informację startowe zawierające aktualnie wspieraną wersję protokołu,
- 111 – prośba o przesłanie adresu e-mail w procesie logowania,
- 112 – prośba o przesłanie hasła w procesie logowania,
- 121 – prośba o przesłanie adresu e-mail nadawcy,
- 122 – prośba o przesłanie adresu e-mail odbiorcy,
- 123 – prośba o przesłanie tematu,

- 124 – prośba o przesłanie treści,
- 125 – prośba o przesłanie liczby załączników do przesłania,
- 126 – prośba o przesłanie załącznika,
- 130 – informację procesu logowania.

Kody odpowiedzi pozytywnych:

- 200 – kod potwierdzający poprawne odebranie danych,
- 201 – poprawny SSL,
- 202 – poprawny algorytm szyfrowania,
- 203 – poprawne deszyfrowanie klucza,
- 210 – logowanie poprawne,
- 220 – wiadomość została wysłana.

Kody przesyłane przy wykryciu błędów składniowych:

- 300 – nieobsługiwana komenda,
- 301 – błąd składni adresu e-mail.

Kody błędów wymagające ponownego przesłania danych:

- 401 – nieprawidłowe hasło,
- 402 – nieznany odbiorca wiadomości.

Kody błędów krytycznych kończących połączenie z klientem:

- 500 – błąd nawiązywania połączenia,
- 501 – niepoprawna wersja protokołu USMTP,
- 502 – niewspierany algorytm szyfrujący,
- 503 – niepoprawna wersja SSL,
- 504 – niepoprawne deszyfrowanie klucza,
- 505 – niepoprawna długość klucza klienta,
- 510 – wielokrotne niepoprawne logowanie,
- 520 – nieautoryzowana próba wysyłki wiadomości.
- 521 – zła liczba załączników

1.3 Format wiadomości

Każda z przesyłanych wiadomości przez serwer rozpoczyna się od kodu odpowiedzi, po którym znajduje się stosowny komunikat kończący się podwójnym CRLF. Każda z

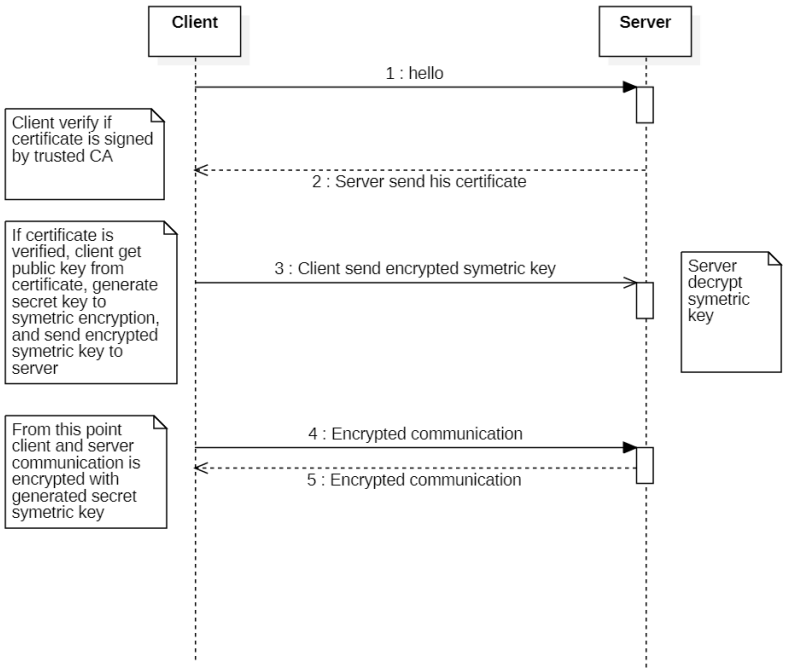
przesyłanych wiadomości jest dodatkowo szyfrowana uniemożliwiając jakiekolwiek podsłuchanie wiadomości.

1.4 Wersje protokołu

Obecnie jedyną obsługiwaną wersją protokołu USMTP jest wersja 1.0. Protokół obsługuję połączenie od klientów używających IPv4 jak i IPv6.

1.5 Szyfrowanie

Protokół zapewnia poufność komunikacji w sieci poprzez szyfrowanie danych przesyłanych między podmiotami. Schemat komunikacji protokołu został zaprezentowany na rysunku nr 1.



Rysunek 1 Schemat komunikacji protokołu SSL

Klient wysyła wiadomość *hello* do serwera w celu zainicjowania komunikacji. Serwer w odpowiedzi odsyła swój certyfikat. Klient weryfikuje czy certyfikat został podpisany przez znany mu podmiot autoryzujący, jeśli tak to generuje klucz do szyfrowania symetrycznego. Następnie pobiera klucz publiczny serwera z certyfikatu i szyfruje nim swój klucz do komunikacji symetrycznej. Serwer odszyfrowuje z pomocą swojego klucza prywatnego klucz wysłany przez klienta. W ten sposób klient i serwer wymienili między sobą klucze niezbędne do bezpiecznej komunikacji.

1.5.1 Schemat poprawnej komunikacji SSL

Do szyfrowania asymetrycznego jest wymagany klucz RSA w wersji 2048 bitowej, natomiast do szyfrowania symetrycznego jest wymagany algorytm AES 256 bitowy w trybie ECB.

Format przesyłanych wiadomości w ramach protokołu został zaprezentowany poniżej:

1.Inicjacja komunikacji(Klient):

Hello ssl1.0?**\r\n\r\n**

2.Odpowiedź na wersje(Serwer):

201 ssl valid**\r\n\r\n**

3.Inicjacja szyfrowania(Klient):

support asymetric-rsa2048 symetric-aes-256-ecb?**\r\n\r\n**

4.Przesłanie certyfikatu(Serwer):

202 algos valid Treść_certyfikatu**.\r\n\r\n**

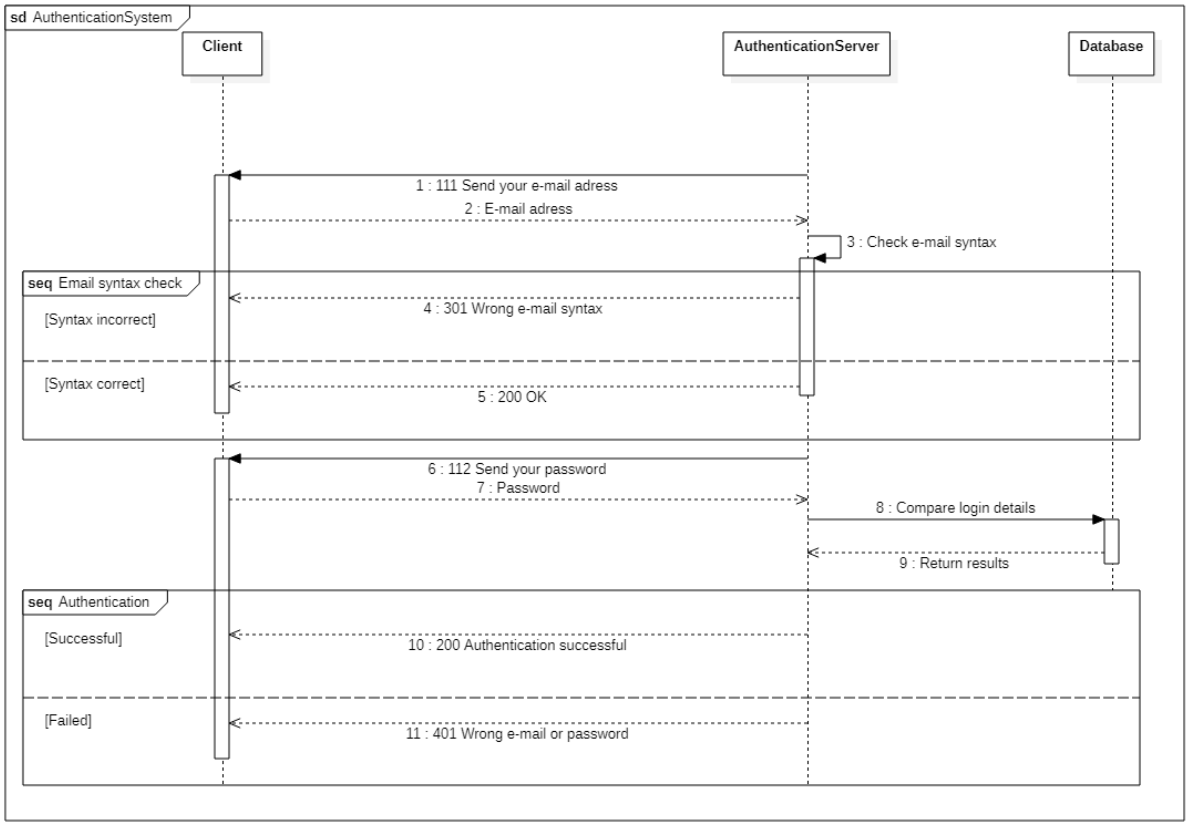
5.Przesłanie zaszyfrowanego klucza do komunikacji jeśli ufa certyfikatowi(Klient):

Zaszyfrowana_wiadomość(klucz)**\r\n\r\n**

Jak widać wszystkie komunikaty oprócz komunikatu z przesłanym certyfikatem kończą się podwójnym <crlfem>. Do komunikatu z wysyłanym certyfikatem przed <crlfem> zostaje dodana kropka, aby uniknąć problemów związanych z nowymi liniami w pliku. Powyżej został zaprezentowany schemat w wypadku poprawnej komunikacji, koniec komunikatów błędów wyznaczają również te same znaki, aby klient mógł w ten sam sposób odczytać błędne komunikaty jak i te prawidłowe.

1.6 Uwierzytelnianie klienta

Logowanie rozpoczyna się od przesłanie przez klienta wiadomości o treści *LOGIN*. Serwer odbierając tę informację rozpoczyna proces logowania, który został przedstawiony na poniższym diagramie.



Rysunek 2 Schemat komunikacji procesu logowania

Na początku komunikacji serwer przesyła informację o gotowości przyjęcia wiadomości z adresem e-mail użytkownika. Następnie klient przesyła wymagane dane, które serwer waliduje pod względem poprawności składni. Jeśli składnia nie jest poprawna zwraca komunikat błędu i oczekuje na ponowne przesłanie adresu. Jeśli składnia jest poprawna serwer odsyła wiadomość o pozytywnej walidacji adresu i oczekuje na przesłanie hasła. Po przesłaniu hasła przez użytkownika serwer porównuje otrzymane informację z tymi znajdującymi się w bazie danych. W przypadku, gdy adres e-mail i hasło pasują do siebie serwer zwraca informację o poprawnym zakończeniu procesu logowania. W przeciwnym przypadku serwer odsyła informację o niepoprawnym adresie e-mail lub haśle i komunikacja procesu logowania rozpoczyna się od początku. Dodatkowo serwer zlicza wszystkie błędy, które wystąpiły w procesie logowania i w przypadku, gdy będzie ich więcej niż pięć prześle do klienta komunikat o wielokrotnym pojawieniu się błędu i zakończy połączenie.

1.6.1 Schemat poprawnej komunikacji dla uwierzytelniania klienta:

Każda przesyłana wiadomość jest szyfrowana za pomocą wymienionego między stronami klucza do szyfrowania symetrycznego. Na końcu każdej zaszyfrowanej wiadomości jest umieszczany jest ciąg znaków *\r\n\r\n*

1.Komunikat oznaczający chęć zalogowania(Klient):

LOGIN

2.Prośba o email (Serwer):

111 send your email address

3.Podanie emaila(Klient):

example@example.com

4.Prośba o hasło(Serwer):

112 send your password

5.Podanie hasła(Klient):

Password123

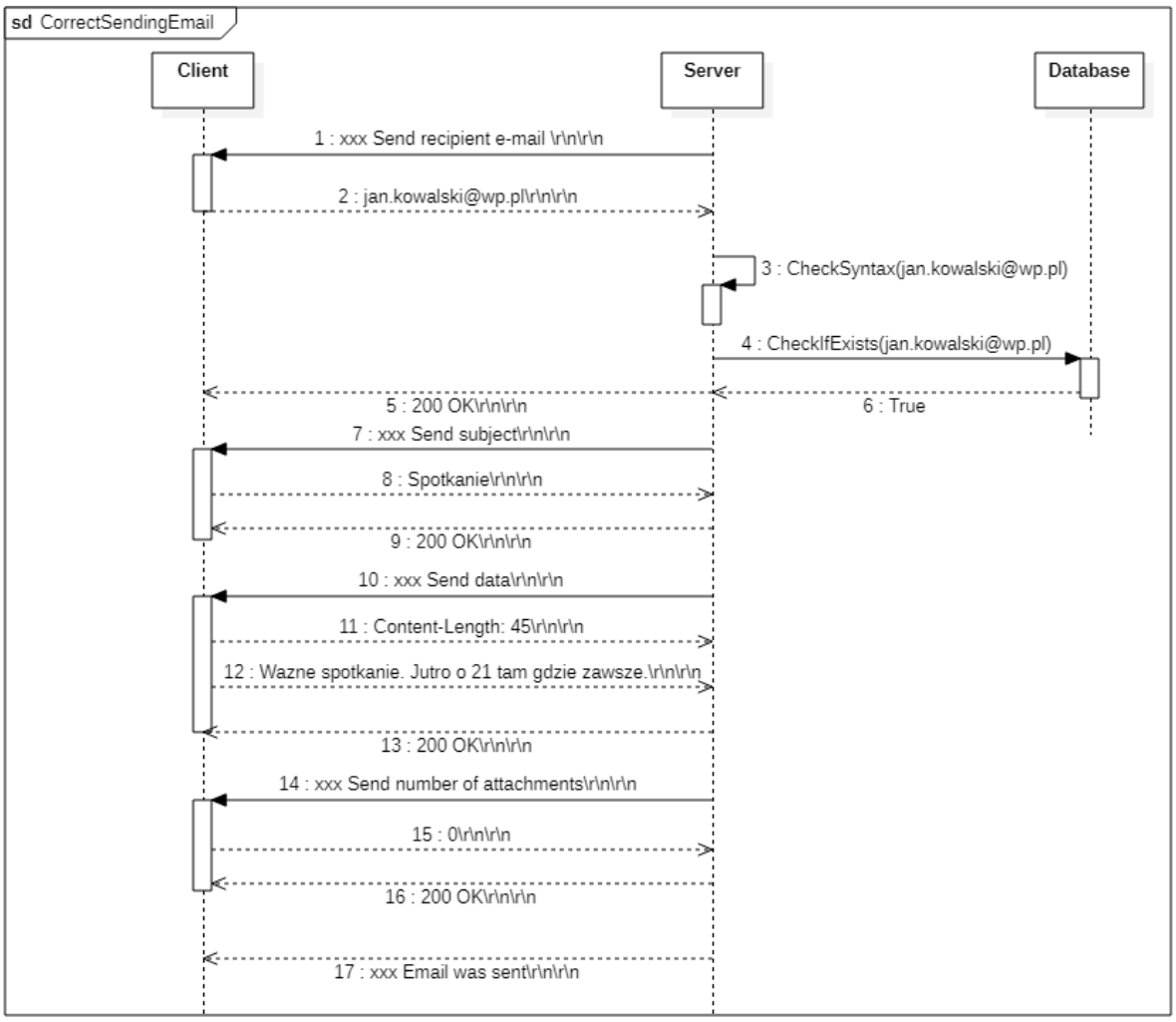
6.Informacja na temat udanego logowania(Serwer):

210 Authentication successful

Powyżej można dostrzec prosty schemat uwierzytelniania. Klient podaje dane logowania. Serwer w przypadku podanie poprawnego emaila oraz hasła daje mu dostęp do wysyłania maila.

1.7 Wysyłka maila

Proces wysyłki maila rozpoczyna się od przesłania przez klienta wiadomości o treści *SEND MAIL*. Serwer otrzymując tę informację sprawdza, czy klient od którego otrzymał tę wiadomość przeszedł pomyślnie proces logowania. Jeśli klient nie jest zalogowany serwer zwraca stosowny komunikat i prosi o zalogowanie się. Jeśli klient przeszedł logowanie rozpoczyna się praktyczna komunikacja przedstawiona na poniższym diagramie.



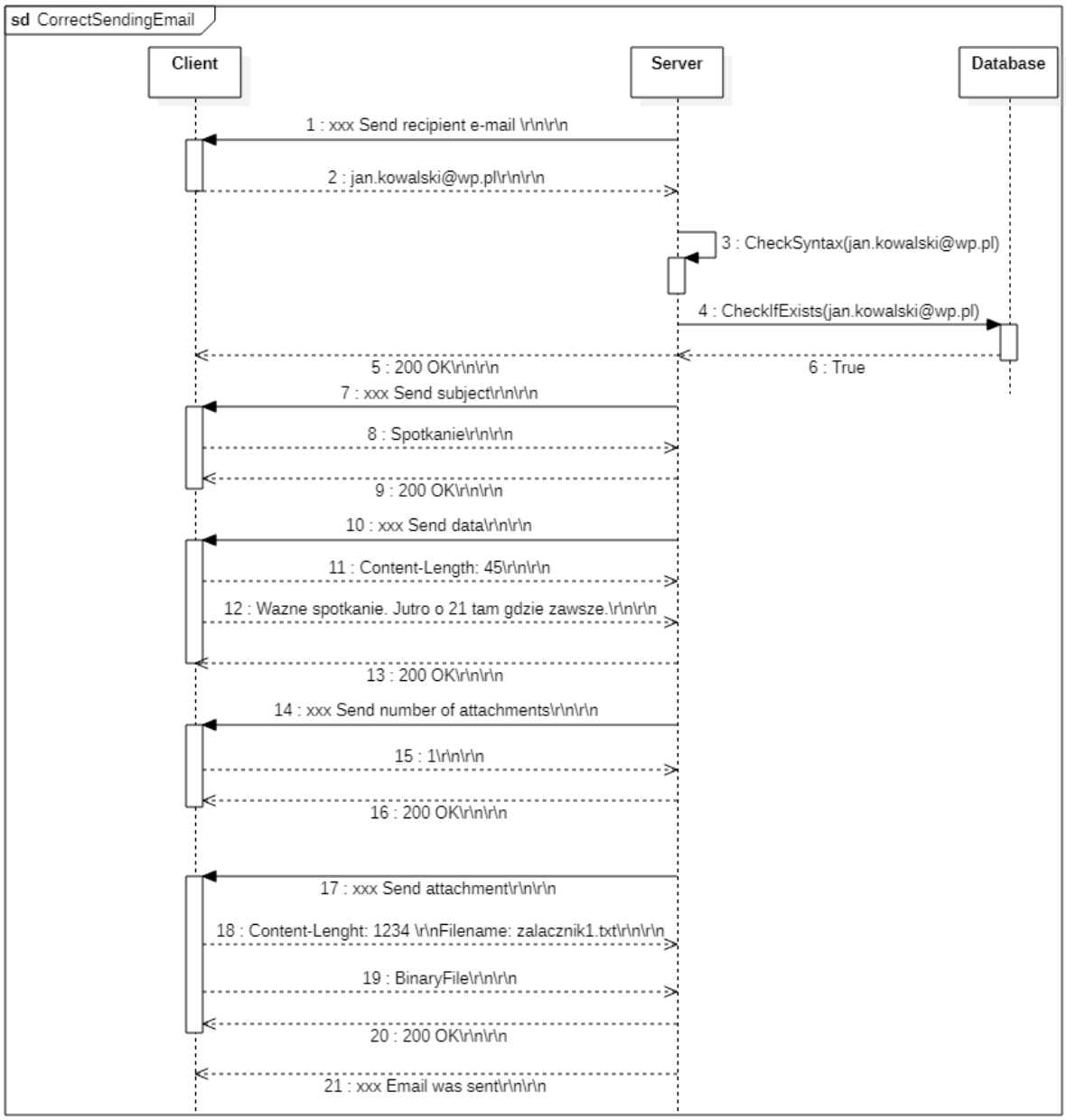
Rysunek 3 Schemat komunikacji procesu wysyłki maila

Komunikacja rozpoczyna się od automatycznego przesłania na serwer adresu e-mail nadawcy, który został wprowadzony w procesie logowania. Następnie serwer czeka na otrzymanie adresu odbiorcy. Po odebraniu wiadomości serwer waliduje otrzymany adres składniowo i pod kątem występowania w bazie danych. W przypadku poprawnej walidacji przechodzi dalej, jeśli pojawią się błędy prosi o ponowne przesłanie adresu. Następnie klient przesyła temat wiadomości, a w kolejnej wiadomości dane. Po odebraniu danych serwer dopytuje klienta o ilość załączników, które chce dołączyć do wiadomości. W przypadku wybrania innej liczby niż 0 odbywa się dalsza komunikacja w celu odebrania załączników (proces ten został dokładnie opisany w kolejnym rozdziale). Jeśli klient nie chce załączyć

żadnych plików serwer przesyła wiadomość e-mail i zwraca użytkownikowi informację o poprawnej wysyłce wiadomości.

1.8 Załączniki

Proces wysyłki załączników to ostatni etap wysyłki wiadomości przedstawionej w poprzednim rozmiarze. W momencie gdy klient prześlę informację iż chce załączyć do wysyłanej wiadomości plik serwer kontynuuje komunikację w celu uzyskania niezbędnych informacji. Proces wysyłki maila dokładnie ilustruje poniższy diagram.



Rysunek 4 Schemat komunikacji wysyłki maila z załącznikiem

Proces dołączania plików do wiadomości rozpoczyna się od przesłania do programu klienta zapytania o liczbę plików do załączenia. Serwer po odebraniu danych pracuje w pętli do momentu odebrania wszystkich załączników. Program klienta odpytuje użytkownika o nazwę pliku do przesłania, następnie sprawdza czy podany plik istnieje i przesyła na serwer informację o jego nazwie i długości po czym przesyła plik w formie binarnej. Serwer po odebraniu długości pliku odbiera cały plik, który dołącza do wysyłanej wiadomości. Po odebraniu pliku serwer przesyła do programu klienckiego komunikat o poprawnym odebraniu załącznika. W momencie odebrania wszystkich plików serwer przesyła wiadomość e-mail i przesyła do klienta informację o poprawnym wysłaniu wiadomości.

1.8.1 Schemat poprawnej komunikacji dla wysyłania maila z załącznikiem

Każda przesyłana wiadomość jest szyfrowana za pomocą wymienionego między stronami klucza do szyfrowania symetrycznego. Na końcu każdej zaszyfrowanej wiadomości oprócz wiadomości z treścią maila jest umieszczany jest ciąg znaków *\r\n\r\n*, który wyznacza koniec wczytywania informacji. Dla treści wiadomości jej koniec wyznacza ciąg znaków *.\r\n\r\n*

1.Inicjacja wysyłki maila(Klient):

SEND MAIL

2. Prośba o adres nadawcy(Serwer):

121 send sender address

3.Adres nadawcy:

mail from: example@example.com

4.Odpowiedź(Serwer):

200 ok

5.Prośba o adres odbiorcy(Serwer):

122 send recipient address

6.Adres obiorcy(Klient):

mail to: example2@example.com

7.Odpowiedź(Serwer):

200 ok

8.Prośba o temat maila(Serwer):

123 send subject

9.Wysłanie tematu(Klient):

Some example subject

10.Odpowiedź(Serwer):

200 ok

11.Prośba o treść maila(Serwer):

124 send data

12.Wysłanie danych(Klient):

Some example data

13.Odpowiedź(Serwer):

200 ok

14.Prośba o liczbę załączników(Serwer):

125 send number of attachments

15.Liczba załączników(Klient):

1

16.Odpowiedź(Serwer):

200 ok

17.Prośba o załącznik – powtarzana określoną ilość razy (Serwer):

126 send attachment number x

18.Długość załącznika(Klient):

500

19.Odpowiedź(Serwer):

200 ok

20.Treść załącznika(Klient):

Example file content

21.Odpowiedź(Serwer):

200 ok

22.Zakończenie(Klient):

BYE

1.9 Zakończenie połączenia

Połączenie pomiędzy serwerem, a klientem może być zakończone w następujących sytuacjach:

- Wysłania przez klienta komunikatu *BYE*,
- Wielokrotne niepoprawne wpisanie danych logowania,
- Niepoprawne nawiązanie połączenia SSL,
- Nieautoryzowana próba wysłania maila.

1.10 Ograniczenia protokołu

Protokół USTMP posiada kilka ograniczeń:

- Odbiorcą wiadomości może być tylko jedna osoba,
- Nie ma możliwości wysłania większej liczby załączników niż pięć.

1.11 Bezpieczeństwo

Komunikacja między serwerem, a programem klienta jest w pełni szyfrowana. Nie istnieje żadna możliwość odszyfrowania komunikacji bez posiadania odpowiedniego klucza, którym wiadomości zostały zaszyfrowane. Do szyfrowania zastosowano klucz o długości 256 bitów co jest wystarczającą długością. Dodatkowo protokół uniemożliwia wykonanie jakichkolwiek operacji bez poprawnego logowania. Uniemożliwia więc to wysyłanie wiadomości od anonimowych nadawców.

1.12 Informacje dodatkowe

W ramach projektu protokół został zaimplementowany w języku Python, który można znaleźć pod linkiem: https://github.com/mjakubowski99/pas-smtp-project

1.13 Uwagi do implementacji

Proces wprowadzania danych w programie klienta trwa tak długo dopóki użytkownik nie wprowadzi pustej linii(ENTER) i słowa kluczowego END. Jest to przykładowy sposób radzenia sobie z nowymi liniami przy wczytywaniu danych od użytkownika w programie napisanym w języku python. Sposób odczytywania całości danych treści wiadomości przez program klienta

jest dowolny. Ważne aby wiadomość przesłana do serwera kończyła się ciągiem znaków *.\r\n\r\n*

W implementacji można też znaleźć plik do zarządzania certyfikatami o nazwie **certManager.py** Udostępnia on 3 możliwości: wygenerowanie podpisanego swoim kluczem prywatnym certyfikatu. Z pomocą tego generatora został wygenerowany certyfikat stworzonego przez nas na potrzeby projektu, któremu ufa program klienta znajduje się on pod ścieżką *ssl/rootCert/cert.pem*. Kolejną możliwością jest wygenerowanie żądania podpisania certyfikatu. Poprzez ten generator zostało stworzone żądanie podpisania certyfikatu, który został podpisany z pomocą certyfikatu, któremu ufa program klienta. Certyfikat ten używany jest jako certyfikat przedstawiany przez serwer. Znajduje się on pod ścieżką *ssl/certs/cert1.pem.* Podpisanie certyfikatu to trzecia opcja, którą udostępnia program certManager.py z pomocą tego narzędzia został podpisany wyżej wspomniany certyfikat.

Serwer zapisuje logi komunikacji do pliku ServerLogs.txt znajdującego się w głównym katalogu projektu. Można tam znaleźć informacje o próbach łączenia z serwerem oraz o tym jak komunikacja przebiegała, czyli np czy komunikacja ssl przebiegła pomyślnie, czy uwierzytelnianie przebiegło pomyślnie, kiedy klient się rozłączył.

Spis rysunków