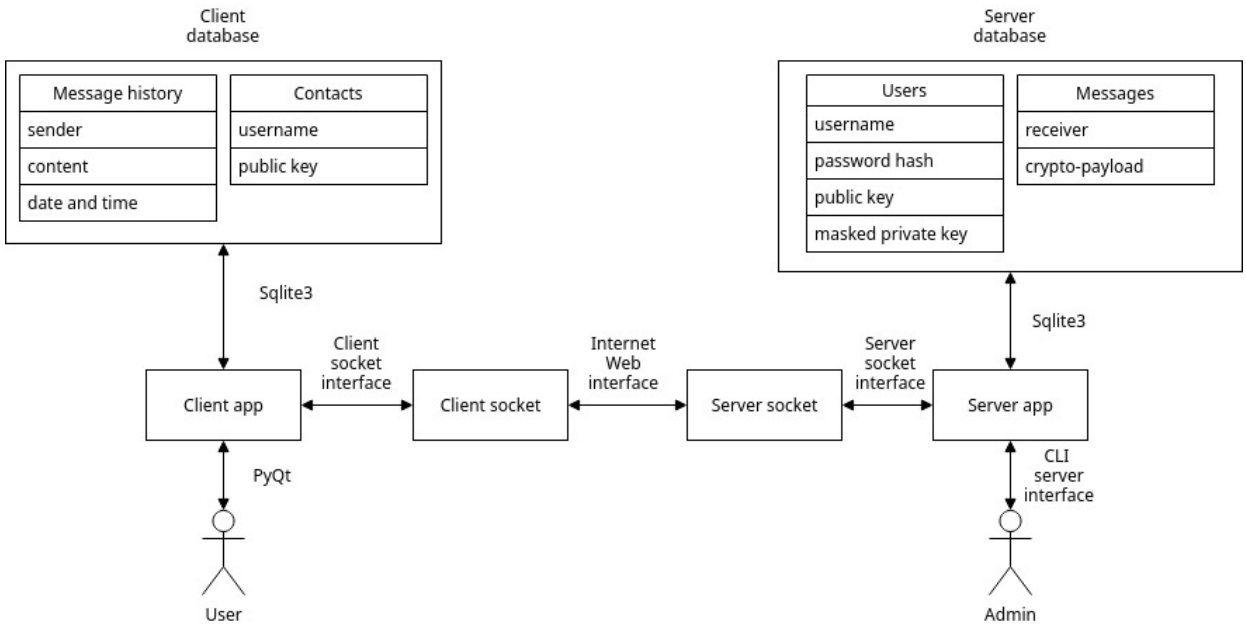
Projekt szyfrowany komunikator desktopowy - kamień milowy 2

Piotr Chlebowski, Kacper Szatan, Łukasz Tomczak, Daniel Zatoń

2 listopada 2021

1 Docelowa architektura systemu

1.1 Podział systemu na komponenty



Rysunek 1: Diagram komponentów i połączeń między nimi (na obrazku pokazano tylko jednego klienta, aplikacja jako całość będzie miała strukturę gwiazdy z serwerem w środku i klientami na ramionach)

Client app
Jest komponentem aplikacji przeznaczonym dla użytkownika aplikacji, z jej poziomu za pomocą PyQt jest w stanie wykonywać wszystkie przewidziane w wymaganiach czynności. Szyfruje wiadomości otrzymane dzięki GUI używając kluczy zaciągniętych z bazy danych, a następnie przekazuje je swojemu socketowi i zapisuje w bazie jeśli wiadomość dotarła. Odszyfrowuje otrzymane wiadomości oraz ma możliwość zakodowania i odkodowania swojej bazy danych.

Client database
Przechowuje historię wiadomości i kontakty danego użytkownika. Jako część aplikacji ewidentnie odmienna od reszty i wymagająca specjalnego interfejsu do kontaktu została wydzielona do osobnego komponentu. Dane z bazy Client app będzie w stanie uzyskać za pomocą pythonowego interfejsu Sqlite3. Przechowuje dane do kontaktu z innymi użytkownikami i historię wszystkich konwersacji.

Client socket + Server socket
Jest to część aplikacji mająca na celu połączenie użytkownika z serwerem. Została wydzielona ze względu na jej ważność w funkcjonowaniu całego projektu. Socket klienta potrafi skutecznie wysłać na serwer napisaną przez użytkownika wiadomość, socket serwera po przetworzeniu otrzymanej wiadomości ze strony aplikacji serwera potrafi skutecznie rozesłać wiadomości do odbiorców. Sockety odpowiadają również za potwierdzanie otrzymania wiadomości. Znajdują się stale w stanie nasłuchiwania przychodzących komunikatów. Będą potrafiły, jeśli to konieczne sklejać poszatkowaną

otrzymywaną wiadomość z kilku paczek.

Server app
Jest komponentem aplikacji przeznaczonym do przetwarzania przechodzących wiadomości. Połączony jest z bazą danych w której odkłada wiadomości i socketem od którego je otrzymuje / wysyła je przez niego. Przychodzące wiadomości odszyfrowuje swoim kluczem prywatnym, weryfikuje, zapisuje w bazie i daje socketowi do przesłania odpowiedniemu odbiorcy. Dostęp do aplikacji serwera jest możliwy dzięki prostemu CLI server interface z poziomu którego administrator jest w stanie wykonywać proste czynności zarzadzające.

Server database
Baza danych serwera przechowująca wszystkie wiadomości jakie przez niego przeszły (zaszyfrowane w sposób uniemożliwiający odczytanie ich z poziomu serwera) oraz dane użytkowników przechowywane potrzebne do prawidłowego działania całej aplikacji. Server app ma do nich dostęp dzięki interfejsowi Sqlite3.

1.2 Podstawowe wersje interfejsów pomiedzy komponentami

Sqlite3
Gotowa biblioteka w pythonie umożliwiająca kontakt programu z bazą danych za pomocą klasycznych zapytań SQL. Jeśli interfejs ten okaże się zbyt ograniczony dla pisanej aplikacji zostanie prawdopodobnie wymieniony na analogiczne bardziej zaawansowane narzędzie.

Client socket interface / Server socket interface
Interfejs służący do kontaktu aplikacji z własnym soketem. Służy do bezpośredniej, bezkonfliktowej wymiany wiadomości między dwoma odpowiednimi komponentami. Wersja klienta i serwera zapewne będą różnić się szczegółami, będą jednak to analogiczne sobie interfejsy.

PyQt
Gotowa biblioteka w pythonie umożliwiająca łatwe tworzenie przyjemnego wizualnie, prostego GUI. Za pomocą wygenerowanego interfejsu użytkownika klient będzie w stanie obsługiwać swoją aplikację. Jeśli w trakcie tworzenia apliacji okarze się, że nie posiada pewnych specyficznych wymagań, prawdopodobnie zostanie zastąpiona analogiczną bardziej zaawansowaną biblioteką. Ponadto prawdopodobnie zostanie opakowana w specjalnie naspisaną prostą bibliotekę, aby intuicyjniej się jej używało.

CLI server interface
Bardzo prosty interfejs tekstowy umożliwiający adminowi wykonywanie podstawowych operacji na serwerze.

1.3 Biblioteka do szyfrowania i deszyfrowania oraz przykłady komunikatów

Dla aplikacji zostanie napisana specjalna biblioteka potrafiąca w schludny sposób przeprowadzać szyfrowania i deszyfrowania dla wszystkich potrzebnych danych. Będzie potrafiła ona między innymi:

- po przekazaniu pliku i hasła zaszyfrować go XORując hashem ostzymanego hasła i go w razie potrzeby odszyfrować
- zaszyfrowywać i odszyfrować dowolne wiadomości podanym kluczem
- obliczać hashe dowolnego ciągu znaków / pliku
- inne wymagane w scenariuszach formy utajniania danych

Z biblioteki tej będzie mógł korzystać każdy komponent aplikacji, zarówno po stronie klienta i serwera.

Przykłady zaszyfrowanych komunikatów
Struktura wiadomości od klienta do serwera:

- opakowane w klucz publiczny serwera:
  - adresat wiadomości
  - zaszyfrowane kluczem publicznym odbiorcy:
    - nadawca
    - wiadomosc
    - data i godzina wiadomosci

Struktura wiadomości od servera do klienta:

- zaszyfrowane kluczem publicznym odbiorcy:
  - nadawca
  - wiadomosc
  - data i godzina wiadomosci

Struktura komunikatu zmiany klucza publicznego

- zapakowane kluczem publicznym serwera:
  - nazwa użytkownika
  - nowy klucz publiczny
  - nowy zaszyfrowany XORem klucz prywatny
  - hash hasła w celu weryfikacji

2 Harmonogram implementacji wymagań użytkownika w kolejnych wersjach

2.1 Alfa

Wersja ta ma prezentować produkt działający w okrojonej wersji. Przewidywane jest istnienie aplikacji klienta i servera, które na tym etapie będą łączyły się ze sobą przez localhost oraz będą potrafiły wysyłać między sobą zaszyfrowane wiadomości. Z punktu widzenia aplikacji klienta będzie można utworzyć czat z innym użytkownikiem i wysyłać do niego wiadomości, jak i je otrzymywać, wszystko to przy użyciu GUI. Wiadomości te będą składowane w lokalnej bazie danych z której będzie można odtworzyć historię konwersacji. Z punktu widzenia serwera będzie można odbierać wiadomości i przekazywać je dalej odpowiednim odbiorcom.

2.2 Beta

W wersji tej zostanie dodana możliwość komunikacji przez sieć internet. Dla serwera zostanie dodana możliwość przechowywania wszystkich wiadomości i ich wysłania na rządanie klienta. Uogólniając będzie możliwa pełna komunikacja między klientami w sposób bezproblemowy.

2.3 Finalna

W wersji finalnej zostaną dodane wszystkie elementy pominiete w poprzednich wersjach np. zmiana własnego klucza szyfrującego i związane z tym procedury powiadamiania kontaktów użytkownika, importowanie i eksportowanie lokalnej bazy danych klienta, czy zmiana hasła użytkownika.