Encrypted Messenger

Autorzy: Fatlum Zahiti & Danila Rusak

Przedmiot: Kryptografia

Uczelnia: Akademia Górnośląska

Semestr: 5

Rok: 3

Data: 01/02/2024

Cel realizacji projektu

Projekt "Encrypted Messenger" ma na celu zbudowanie programu typu czat,
którego wiadomości przesyłanie są między dwoma klientami.
Naszym zamierzeniem było stworzenie nie tylko aplikacji zgodnej z wymaganiami akademickimi, ale także narzędzia, które znajdzie praktyczne zastosowanie w codziennej komunikacji online. W dobie rosnącej świadomości dotyczącej prywatności cyfrowej nasz program ma za zadanie stanowić bezpieczną alternatywę dla istniejących rozwiązań komunikacyjnych.

Nasza inicjatywa kieruje się ideą, że bezpieczeństwo i prywatność nie powinny być luksusem, ale podstawowym prawem każdego użytkownika Internetu. Dlatego staramy się, aby nasze oprogramowanie było dostępne dla szerokiego grona odbiorców, niezależnie od ich technicznego doświadczenia. Pragniemy również, aby nasz projekt służył jako edukacyjna platforma dla studentów, profesjonalistów i hobbystów zainteresowanych kryptografią, oferując praktyczne doświadczenie w tworzeniu i zastosowaniu technik szyfrowania.

Wreszcie, "Encrypted Messenger" ma stanowić odpowiedź na dynamicznie zmieniające się wymogi bezpieczeństwa w cyfrowym świecie, dostosowując się do nowych zagrożeń i stale rozwijając się, aby zapewnić najwyższy poziom ochrony dla swoich użytkowników. Naszym celem jest nieustanne doskonalenie i rozbudowywanie funkcjonalności aplikacji, tak aby była ona zawsze krok przed potencjalnymi zagrożeniami i wykorzystywać najnowsze osiągnięcia w dziedzinie kryptografii.

Przykład Działania RSA

Załóżmy, że Alice chce wysłać zaszyfrowaną wiadomość do Boba. Bob generuje parę kluczy: publiczny i prywatny. Klucz publiczny Boba jest wysyłany do Alice. Alice używa klucza publicznego Boba do zaszyfrowania swojej wiadomości, a następnie wysyła zaszyfrowaną wiadomość do Boba. Po otrzymaniu Bob używa swojego klucza prywatnego do odszyfrowania wiadomości.

Matematyczne Podstawy RSA

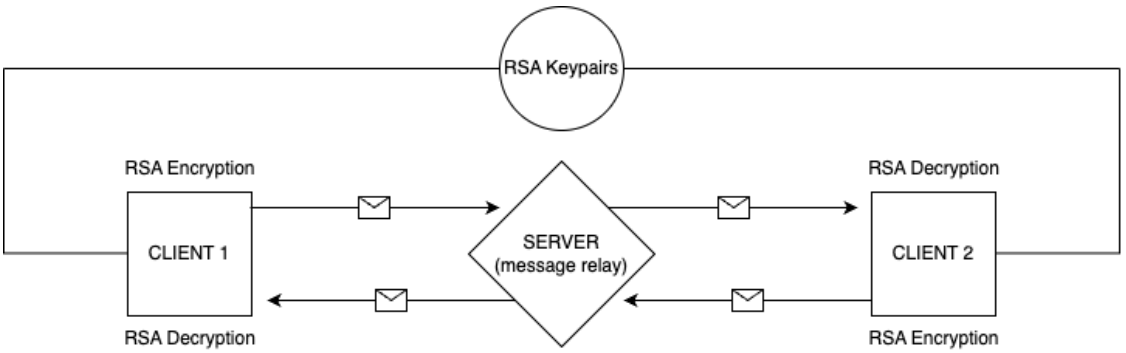
Wspomniana wcześniej trudność rozłożenia liczby na czynniki pierwsze przy dużych wartościach *n* jest kluczowa dla bezpieczeństwa RSA. Matematyczna złożoność tego zadania zapewnia, że nawet przy wykorzystaniu najnowocześniejszych komputerów i algorytmów, rozkładanie takich liczb jest niepraktyczne w rozsądnym czasie. To sprawia, że RSA jest bezpieczny, przy założeniu dostatecznie dużego klucza.

Dlaczego Wybraliśmy RSA?

Wybór algorytmu RSA do naszego projektu "Encrypted Messenger" był podyktowany kilkoma czynnikami. Po pierwsze, jego solidne matematyczne podstawy i udowodniona skuteczność w zapewnianiu bezpiecznej komunikacji. Po drugie, RSA jest szeroko stosowany i rozpoznawalny, co przekłada się na zaufanie w kontekście bezpieczeństwa cyfrowego. Ponadto, używając 2048-bitowego klucza, zapewniamy równowagę między bezpieczeństwem a wydajnością, co jest kluczowe w aplikacjach komunikacyjnych, gdzie czas odpowiedzi i wydajność są tak samo ważne, jak bezpieczeństwo.

Wnioski i Perspektywy

Wykorzystanie RSA w naszym projekcie pozwala nie tylko na zabezpieczenie przesyłanych wiadomości, ale także na szersze zrozumienie zasad kryptografii asymetrycznej. Projekt "Encrypted Messenger" służy więc nie tylko jako narzędzie komunikacyjne, ale również jako platforma edukacyjna, jest bardzo cenną nauką, dzięki której mogliśmy sporo nauczyć się o stosowaniu szyfru RSA szczególnie w kontekście programowania Python.



Step by step

Krok 1: Uruchomienie Serwera

Inicjalizacja: Serwer (server.py) jest uruchamiany pierwszy. Przy starcie inicjalizuje gniazdo sieciowe i zaczyna nasłuchiwać na określonym porcie, oczekując na połączenia od klientów.
Oczekiwanie na połączenia: Serwer pozostaje w stanie oczekiwania, aż dwóch lub więcej klientów nawiąże z nim połączenie.

Krok 2: Uruchomienie Klienta

Generowanie Kluczy: Każdy klient (client.py) przy uruchomieniu generuje parę kluczy RSA - publiczny i prywatny. Klucz publiczny jest przeznaczony do udostępnienia innym, natomiast klucz prywatny służy do dekodowania przychodzących wiadomości.
Połączenie z Serwerem: Klient nawiązuje połączenie z serwerem, przekazując swój klucz publiczny.

Krok 3: Wymiana Kluczy Publicznych

Przesłanie Kluczy do Serwera: Po połączeniu, każdy klient wysyła swój klucz publiczny do serwera.
Dystrybucja Kluczy: Serwer odbiera klucze publiczne i przekazuje każdemu klientowi klucz publiczny drugiego klienta.

Krok 4: Wymiana Wiadomości

Szyfrowanie Wiadomości: Gdy klient chce wysłać wiadomość, używa klucza publicznego odbiorcy (otrzymanego od serwera) do zaszyfrowania wiadomości.
Wysyłanie Wiadomości: Zaszyfrowana wiadomość jest wysyłana do serwera, który następnie przekazuje ją do odpowiedniego klienta.
Odszyfrowywanie Wiadomości: Odbiorca używa swojego klucza prywatnego do odszyfrowania wiadomości.

Wprowadzenie do Architektury Aplikacji

Projekt "Encrypted Messenger" składa się z dwóch głównych komponentów: modułu klienta (*client.py*) i modułu serwera (*server.py*). Ta dwuskładnikowa architektura pozwala na efektywne zarządzanie procesami komunikacji i szyfrowania, jednocześnie zapewniając wysoki poziom bezpieczeństwa i prywatności. Poniżej przedstawiamy szczegółowy opis obu komponentów.

Moduł Klienta (client.py)

Głównym zadaniem modułu klienta jest połączenie się z końcowym użytkownikiem. Odpowiada on za zbieranie, szyfrowanie i wysyłanie wiadomości do modułu serwera, a także odbieranie i deszyfrowanie wiadomości przychodzących. Kluczowym aspektem jest tutaj implementacja mechanizmów kryptograficznych, które umożliwiają bezpieczną wymianę kluczy oraz szyfrowanie i deszyfrowanie treści komunikatów

- Funkcje i Zastosowanie
  - Wysyłanie i odbieranie zaszyfrowanych wiadomości.
  - Zarządzanie kluczami kryptograficznymi.
  - Nawiązywanie bezpiecznego połączenia z serwerem.
- Użyte Biblioteki
  - *socket*: Dla obsługi połączeń sieciowych.
  - *rsa*: Do implementacji algorytmu RSA.
  - *threading*: Do obsługi wielowątkowości.

Moduł Serwera (server.py)

Z kolei moduł serwera pełni rolę centralnej jednostki koordynującej, która zarządza połączeniami od klientów, przekierowuje szyfrowane wiadomości i w niektórych implementacjach może obsługiwać mechanizmy uwierzytelniania i zarządzania sesjami. Jego działanie jest transparentne dla końcowych użytkowników, co oznacza, że cały proces szyfrowania i deszyfrowania odbywa się po stronie klienta. To serwer odpowiada za dostarczanie wiadomości do odpowiednich adresatów oraz za utrzymanie ciągłości i dostępności usługi. Dzięki zastosowaniu nowoczesnych technologii i protokołów komunikacyjnych moduł serwera jest w stanie obsługiwać wysoki wolumen wiadomości, zapewniając przy tym skalowalność i wydajność całego systemu.

- Funkcje i Zastosowanie
  - Koordynacja połączeń między klientami.
  - Zarządzanie przesyłaniem zaszyfrowanych wiadomości.
  - Zapewnienie integralności i ciągłości komunikacji.
- Użyte Biblioteki
  - *socket*: Do zarządzania połączeniami sieciowymi.
  - *logging*: Dla rejestrowania zdarzeń i monitorowania działania serwera.
  - *threading*: Do obsługi wielu połączeń klientów jednocześnie.

Podsumowując

Projekt "Encrypted Messenger" był dla nas niezwykle cennym doświadczeniem edukacyjnym i zawodowym. Uświadomił nam znaczenie projektowania oprogramowania z myślą o ochronie prywatności użytkowników oraz dostarczył solidne fundamenty do dalszej pracy w dziedzinie bezpieczeństwa cyfrowego. Jesteśmy przekonani, że zdobyte umiejętności i wiedza będą cennym atutem w naszych przyszłych przedsięwzięciach zawodowych, a doświadczenie zdobyte podczas pracy nad tym projektem będzie inspiracją do dalszego rozwoju i doskonalenia w dziedzinie informatyki oraz programowania na podstawie bezpieczeństwa.

Z wyrazami szacunku.
Fatlum Zahiti & Danila Rusak