UNIWERSYTET WSB MERITO W POZNANIU

WYDZIAŁ FINANSÓW I BANKOWOŚCI

Zdecentralizowany System Komunikacji P2P z Szyfrowaniem End-to-End w języku Rust

PROJEKT DYPLOMOWY

UNIWERSYTET WSB MERITO W POZNANIU
WYDZIAŁ FINANSÓW I BANKOWOŚCI

Poznań 2025

UNIWERSYTET WSB MERITO W POZNANIU
WYDZIAŁ FINANSÓW I BANKOWOŚCI

DANE PARTNERÓW

A1. Dane Promotora

Imię i nazwisko	Rafał Brodziak
Stopień / Tytuł naukowy	doktor inżynier
Data i podpis	

A2. Dane członków Zespołu projektu

Imię i nazwisko	Lukrecja Pleskaczyńska
Kierunek studiów	Informatyka
Tryb studiów	Stacjonarne
Data i podpis	

UNIWERSYTET WSB MERITO W POZNANIU
WYDZIAŁ FINANSÓW I BANKOWOŚCI

ZAŁOŻENIA PROJEKTU

B1. Opis projektu

1. Uzasadnienie wyboru tematu

Współczesna komunikacja internetowa często opiera się na scentralizowanych architekturach, gdzie dane i metadane użytkowników są przechowywane na serwerach dostawców usług. Taki model stwarza ryzyko nadzoru, cenzury oraz wycieku danych podczas naruszeń bezpieczeństwa. Centralizacja prowadzi również do punktów awaryjnych, gdzie awaria jednego serwera może uniemożliwić komunikację dla wielu użytkowników.

Projekt stanowi alternatywę do tradycyjnych komunikatorów internetowych, oferując bezpieczną komunikację peer-to-peer bez konieczności korzystania z centralnych serwerów. Dzięki temu system umożliwia anonimową, odporną na cenzurę i bezpieczną wymianę wiadomości. W przeciwieństwie do rozwiązań wykorzystujących skomplikowane mechanizmy NAT traversal, w niniejszym podejściu skupiono się na wdrożeniu systemu działającego w lokalnej sieci, co pozwala na uproszczenie architektury, lepszą czytelność kodu oraz skoncentrowanie się na kluczowych funkcjonalnościach, takich jak szyfrowanie i interfejs webowy.

Implementacja projektu opiera się na wykorzystaniu technik szyfrowania end-to-end, przechowywaniu wiadomości w rozproszonej tablicy haszującej (DHT) oraz automatycznym wykrywaniu węzłów w sieci lokalnej za pomocą protokołu mDNS. Wybór języka programowania Rust wynika z jego wysokiej efektywności, bezpieczeństwa pamięci oraz rozwiniętego ekosystemu bibliotek kryptograficznych i sieciowych. W projekcie wykorzystano m.in. bibliotekę **libp2p** – modułowy stos protokołów umożliwiający tworzenie aplikacji peer-to-peer, którego dokumentację można znaleźć na stronie: `https://docs.libp2p.io/`.

2. Problem badawczy

W jaki sposób zaprojektować i zaimplementować w pełni zdecentralizowany system komunikacji peer-to-peer, który zapewni prywatność użytkowników poprzez zastosowanie szyfrowania end-to-end, umożliwi asynchroniczną wymianę wiadomości (w tym

dostarczanie wiadomości do nieaktywnych użytkowników), a jednocześnie umożliwi efektywne wykrywanie oraz łączenie się z innymi urządzeniami w obrębie sieci lokalnej przy użyciu mDNS?

Problemy szczegółowe:

- Jak zapewnić efektywne wykrywanie i łączenie się z innymi węzłami sieci w warunkach lokalnych?
- W jaki sposób zaimplementować bezpieczne przechowywanie wiadomości offline w rozproszonej tablicy haszującej (DHT) przy zachowaniu pełnej poufności treści?
- Jak zaprojektować mechanizm weryfikacji dostarczenia wiadomości umożliwiający bezpieczne usuwanie wiadomości z DHT po ich odczytaniu?
- W jaki sposób zapewnić autentyczność i integralność przesyłanych danych przy jednoczesnym zachowaniu anonimowości użytkowników?

3. Cel główny i cele szczegółowe projektu

Cel główny: Stworzenie bezpiecznego, zdecentralizowanego komunikatora peer-to-peer, zaimplementowanego w języku Rust, wykorzystującego biblioteki takie jak **libp2p** oraz mechanizmy DHT do przechowywania historii wiadomości, a także zapewniającego intuicyjny interfejs użytkownika oparty na frameworku Vue.js.

Cele szczegółowe:

- Opracowanie architektury systemu peer-to-peer (P2P) opartej na bibliotece **libp2p**;
- Implementacja szyfrowania end-to-end (X25519 + ChaCha20-Poly1305) gwarantującego poufność komunikacji;
- Wdrożenie mechanizmu wykrywania węzłów w sieci lokalnej (mDNS) z możliwością ustawienia pseudonimów;
- Stworzenie funkcjonalności wysyłania wiadomości oraz przechowywania historii czatu;
- Wstępna integracja rozproszonego przechowywania wiadomości offline z wykorzystaniem DHT;
- Opracowanie mechanizmu weryfikacji dostarczenia wiadomości i autoryzowanego

usuwania danych;

- Zapewnienie skalowalności systemu i mechanizmów przeciwdziałających przeciążeniom sieci;
- Projekt i implementacja interfejsu użytkownika do komunikacji oraz zarządzania kluczami kryptograficznymi;
- Przeprowadzenie testów bezpieczeństwa oraz optymalizacji wydajności systemu.

4. Zakres podmiotowy, przedmiotowy, czasowy i przestrzenny
   - **Zakres podmiotowy**: Projekt skierowany jest do użytkowników wymagających bezpiecznych narzędzi komunikacyjnych, w szczególności:
     - osób ceniących wysoką prywatność w komunikacji,
     - aktywistów i dziennikarzy działających w warunkach ograniczonej wolności słowa,
     - organizacji przetwarzających wrażliwe dane,
     - społeczności technologicznych zainteresowanych rozwiązaniami zdecentralizowanymi,
     - badaczy i entuzjastów P2P oraz kryptografii.
   - **Zakres przedmiotowy**: Projekt obejmuje:
     - analizę istniejących rozwiązań P2P i E2EE,
     - projekt architektury systemu z wykorzystaniem **libp2p**,
     - implementację szyfrowania end-to-end oraz mDNS,
     - funkcje wysyłania wiadomości i przechowywania historii czatu,
     - wstępną integrację DHT do przechowywania wiadomości offline,
     - mechanizmy weryfikacji dostarczenia i autoryzowanego usuwania danych,
     - projekt interfejsu użytkownika i zarządzanie kluczami,
     - testy bezpieczeństwa i wydajności.
   - **Zakres czasowy**: Projekt trwa 12 miesięcy od grudnia 2024:
     - **Miesiące 1–2 (grudzień 2024 – styczeń 2025)**: analiza wymagań i projekt architektury;

  - **Miesiące 3–5 (luty – kwiecień 2025)**: mDNS, wysyłanie wiadomości, historia czatu oraz E2EE;
  - **Miesiące 6–8 (maj – lipiec 2025)**: integracja DHT do przechowywania offline i weryfikacja dostarczenia;
  - **Miesiące 9–10 (sierpień – wrzesień 2025)**: projekt i wdrożenie interfejsu użytkownika;
  - **Miesiące 11–12 (październik – listopad 2025)**: testy bezpieczeństwa, audyt i optymalizacja wydajności.
- **Zakres przestrzenny**: Wdrożenie w sieci lokalnej (LAN) w celu łatwego prototypowania i testowania.

5. Metody i techniki badawcze

W projekcie zastosowane zostaną następujące metody i techniki badawcze:

- Analiza literatury oraz istniejących rozwiązań,
- Prototypowanie iteracyjne,
- Weryfikacja poprawności implementacji algorytmów kryptograficznych,
- Ewaluacja użyteczności interfejsu użytkownika,
- Testy wydajnościowe przeprowadzane w różnych warunkach sieciowych.

B2. Zadania w projekcie

Cele szczegółowe projektu	**Zadania i termin realizacji**	**Osoby zaangażowane**
Cel 1: Analiza technologii i narzędzi (libp2p, szyfrowanie)	Zadanie 1: Analiza rozwiązań P2P i biblioteki **libp2p** (grudzień 2024)	Lukrecja Pleskaczyńska
	Zadanie 2: Analiza kryptografii (X25519, ChaCha20-Poly1305) (styczeń 2025)	Lukrecja Pleskaczyńska

Cele szczegółowe projektu	**Zadania i termin realizacji**	**Osoby zaangażowane**
Cel 2: Implementacja podstawowych funkcji (mDNS, wysyłanie wiadomości, historia, szyfrowanie)	Zadanie 1: Wdrożenie wykrywania węzłów w sieci lokalnej (mDNS) i przesyłania wiadomości (luty 2025)	Lukrecja Pleskaczyńska
	Zadanie 2: Implementacja historii czatu i zarządzania kluczami (marzec 2025)	Lukrecja Pleskaczyńska
	Zadanie 3: Szyfrowanie end-to-end (X25519 + ChaCha20-Poly1305) oraz testy (kwiecień–maj 2025)	Lukrecja Pleskaczyńska
Cel 3: Analiza wykonywalności DHT do przechowywania offline	Zadanie 1: Projekt mechanizmu DHT i podstawowa integracja (maj–czerwiec 2025, ograniczony zakres)	Lukrecja Pleskaczyńska
	Zadanie 2: Testy funkcjonalności DHT i weryfikacja dostarczenia (lipiec 2025)	Lukrecja Pleskaczyńska
Cel 4: Projekt i implementacja interfejsu użytkownika	Zadanie 1: Projekt UI i makiety (sierpień 2025)	Lukrecja Pleskaczyńska
	Zadanie 2: Implementacja frontendowa (Vue.js) (wrzesień 2025)	Lukrecja Pleskaczyńska

Cele szczegółowe projektu	**Zadania i termin realizacji**	**Osoby zaangażowane**
	Zadanie 3: Integracja UI z backendem (październik 2025)	Lukrecja Pleskaczyńska
Cel 5: Testowanie i optymalizacja	Zadanie 1: Testy penetracyjne i audyt bezpieczeństwa (październik 2025)	Lukrecja Pleskaczyńska
	Zadanie 2: Testy wydajnościowe i symulacje przeciążeń (październik 2025)	Lukrecja Pleskaczyńska
	Zadanie 3: Wprowadzenie poprawek i optymalizacja (listopad 2025)	Lukrecja Pleskaczyńska

C1. Opracowanie projektu

1. Założenia teoretyczne

W dobie powszechnej cyfryzacji i globalnej komunikacji coraz większego znaczenia nabiera kwestia prywatności, bezpieczeństwa oraz niezależności użytkowników internetu. W tradycyjnych komunikatorach internetowych użytkownicy muszą polegać na scentralizowanych usługodawcach, którzy przechowują zarówno treści komunikatów, jak i metadane. Takie rozwiązanie, choć wygodne, wiąże się z ryzykiem naruszeń prywatności, cenzury oraz podatności na awarie pojedynczych punktów (single point of failure).

Projektowany system komunikacji peer-to-peer (P2P) odpowiada na powyższe zagrożenia, stawiając na pełną decentralizację, bezpieczeństwo i anonimowość. Główną ideą rozwiązania jest stworzenie aplikacji umożliwiającej bezpieczną, szyfrowaną komunikację między użytkownikami bez pośrednictwa centralnych serwerów, opierającą się wyłącznie na zdecentralizowanej architekturze sieci lokalnej. Umożliwia to bezpośrednią wymianę wiadomości, wykrywanie urządzeń w sieci lokalnej oraz przechowywanie historii konwersacji w rozproszonym rejestrze.

Jednym z kluczowych aspektów systemu jest wykorzystanie kryptografii krzywych eliptycznych (X25519) do uzgadniania kluczy oraz szyfrowania symetrycznego (ChaCha20-Poly1305) zapewniającego poufność i integralność przesyłanych wiadomości. Architektura oparta jest na bibliotece `libp2p`, która umożliwia wdrożenie zaawansowanych protokołów sieciowych w duchu modularności i rozszerzalności.

Istotne znaczenie w projekcie odgrywa protokół mDNS (Multicast DNS), umożliwiający automatyczne wykrywanie aktywnych uczestników komunikacji w sieci lokalnej. Dzięki temu użytkownik nie musi znać adresów innych uczestników ani konfigurować ręcznie połączeń. Kolejnym elementem jest mechanizm rozproszonej tablicy haszującej (DHT), wykorzystywany do przechowywania wiadomości adresowanych do użytkowników nieobecnych w danej chwili online – co umożliwia asynchroniczną komunikację i przesyłanie wiadomości offline.

Projekt zakłada wysoką użyteczność rozwiązania poprzez implementację prostego

interfejsu tekstowego (REPL), umożliwiającego wysyłanie wiadomości, podgląd historii oraz zarządzanie kontaktami. W przyszłości planowana jest integracja z interfejsem webowym opartym o framework Vue.js, zapewniająca wygodę obsługi i nowoczesną prezentację danych.

Podstawowe założenia systemu:

- Pełna decentralizacja: brak centralnych serwerów i punktów awarii.
- Prywatność komunikacji: szyfrowanie end-to-end, unikanie przechowywania metadanych.
- Łatwość obsługi: automatyczne wykrywanie urządzeń, możliwość nadawania pseudonimów.
- Możliwość komunikacji offline: przechowywanie i przekazywanie wiadomości przez DHT.
- Otwarta architektura: możliwość dalszej rozbudowy i integracji.

2. Opis sytuacji faktycznej

Obecny etap realizacji projektu obejmuje funkcjonalny prototyp zdecentralizowanego komunikatora P2P działającego w sieci lokalnej, zrealizowany w języku Rust. Aplikacja umożliwia uruchomienie klienta na dowolnej liczbie urządzeń, które automatycznie odnajdują się nawzajem w obrębie tej samej podsieci lokalnej. Każdy użytkownik po uruchomieniu programu wybiera swój pseudonim, który jest widoczny dla pozostałych uczestników.

Podczas testów i wdrożenia rozwiązania zidentyfikowano następujących aktorów i przypadki użycia:

Aktorzy:

- **Użytkownik końcowy** – inicjuje połączenie z siecią, wysyła oraz odbiera zaszyfrowane wiadomości tekstowe, zarządza kontaktami i przegląda historię rozmów.
- **System** – automatycznie wykrywa obecność innych węzłów, zarządza procesami szyfrowania i deszyfrowania, odpowiada za przechowywanie i transmisję wiadomości.

Przypadki użycia:

1. **Automatyczne wykrywanie użytkowników:** Po uruchomieniu programu każdy klient emituje w sieci LAN komunikat powitalny, zawierający pseudonim i klucz publiczny użytkownika. Pozostałe aktywne węzły rejestrują nowego uczestnika i automatycznie dodają go do lokalnej „książki adresowej".

2. **Wysyłanie wiadomości:** Użytkownik może wybrać z listy dowolny dostępny kontakt (po pseudonimie lub PeerId) i wysłać mu wiadomość tekstową. Przed transmisją wiadomość jest szyfrowana kluczem uzgodnionym za pomocą algorytmu X25519, a następnie przekazywana przez protokół `libp2p-request-response`.

3. **Odbieranie wiadomości:** Każda otrzymana wiadomość zostaje automatycznie odszyfrowana i zapisana w lokalnej bazie (sled) wraz z informacją o nadawcy i kierunku transmisji.

4. **Przeglądanie historii:** Użytkownik może przeglądać całą historię komunikacji z wybranym rozmówcą lub globalną historię wszystkich wiadomości. Dzięki przechowywaniu szyfrogramów w lokalnej bazie, możliwy jest bezpieczny dostęp do archiwum rozmów.

5. **Przechowywanie wiadomości offline (częściowo zaimplementowane):** Prototyp zawiera wstępną integrację mechanizmu DHT, umożliwiającego przechowywanie wiadomości adresowanych do nieobecnych użytkowników, tak aby odebrali je po ponownym połączeniu z siecią.

6. **Usuwanie peerów i kluczy:** W przypadku wykrycia wygaśnięcia obecności uczestnika (np. utrata połączenia) system automatycznie usuwa powiązane dane kryptograficzne i aktualizuje książkę adresową.

W toku realizacji projektu wykonano szereg testów funkcjonalnych, sprawdzających poprawność wykrywania użytkowników, niezawodność transmisji wiadomości, pra-

widłowość uzgadniania kluczy kryptograficznych oraz integralność przechowywanych danych. Przeprowadzono symulacje pracy w różnych warunkach sieciowych, w tym testy w środowisku wielourządzeniowym, z użytkownikami na różnych systemach operacyjnych.

Opis typowego przebiegu komunikacji:

- Uruchomienie programu z wybranym pseudonimem przez użytkownika;
- Automatyczne wykrycie obecnych w sieci lokalnej użytkowników (peerów) poprzez protokół mDNS;
- Wymiana kluczy publicznych i zapisanie ich w lokalnej bazie;
- Wysłanie pierwszej wiadomości do wybranego kontaktu – automatyczne uzgodnienie klucza symetrycznego i szyfrowanie wiadomości;
- Odbiór wiadomości przez adresata, jej deszyfracja oraz potwierdzenie odbioru;
- Zapisanie wiadomości w lokalnej historii komunikacji.

Obecny stan projektu umożliwia podstawową komunikację tekstową między dowolnymi użytkownikami tej samej sieci lokalnej bez ryzyka podsłuchu lub przechwycenia wiadomości przez osoby trzecie. System został zaprojektowany modularnie, co umożliwia dalszą rozbudowę o nowe funkcjonalności (np. przesyłanie plików, integracja z przeglądarką, interfejs webowy).

3. Badania własne / opis metod, technik i narzędzi badawczych / aparatura / oprogramowanie

Metodologia pracy nad projektem

Projekt realizowany był w duchu iteracyjnego prototypowania – kolejne funkcje wdrażane były w krótkich cyklach, z regularnym testowaniem stabilności i bezpieczeństwa systemu. Każdy moduł powstawał w wyniku analizy wymagań funkcjonalnych, konsultacji z literaturą techniczną oraz porównania istniejących rozwiązań open-source. Przeprowadzano testy w środowisku zbliżonym do rzeczywistych warunków użytkowania: z udziałem kilku instancji programu uruchamianych na różnych systemach operacyjnych, zarówno w sieciach przewodowych, jak i bezprzewodowych.

Wybór języka programowania Rust i jego przewagi

Decyzja o wyborze języka Rust nie była przypadkowa. Rust, będący obecnie jednym z najbardziej nowoczesnych języków systemowych, oferuje nie tylko wydajność porównywalną z C/C++, ale także unikatowe mechanizmy bezpieczeństwa pamięci (ownership, borrow checker, brak garbage collectora), które minimalizują klasę błędów typowych dla programów niskopoziomowych – takich jak wycieki pamięci czy race conditions. Pozwala to pisać aplikacje sieciowe o wysokiej niezawodności, bez obawy o podatności na typowe exploity (use-after-free, double free, buffer overflow).

Rust zyskał uznanie społeczności bezpieczeństwa, czego dowodem jest szeroka adopcja w projektach związanych z blockchainem, kryptografią oraz programowaniem sieciowym. Dodatkowo, Rust oferuje bogaty ekosystem bibliotek (*crates.io*), nowoczesny system zarządzania zależnościami (Cargo) oraz bardzo dobre wsparcie dla programowania asynchronicznego i wielowątkowego.

Biblioteka `libp2p` jako fundament komunikacji P2P

Sercem projektu jest biblioteka `libp2p`, zaprojektowana do obsługi komunikacji peer-to-peer w duchu decentralizacji i modularności. Libp2p pozwala na łatwe komponowanie własnego stosu sieciowego – użytkownik może dobierać protokoły (TCP, QUIC, WebRTC), systemy wymiany wiadomości (request-response, pubsub), oraz mechanizmy odkrywania peerów (mDNS, Kademlia, custom DHT).

W projekcie wykorzystano m.in.:

- **libp2p-mdns** – automatyczne wykrywanie peerów w sieci lokalnej przez Multicast DNS, eliminujące konieczność ręcznej konfiguracji adresów IP czy portów.
- **libp2p-request-response** – bezstanowy protokół wymiany wiadomości, zapewniający prostą semantykę żądanie–odpowiedź, co doskonale pasuje do architektury komunikatora.
- **libp2p-noise** – zestawienie bezpiecznych połączeń przez protokół Noise (oparty na krzywych eliptycznych), gwarantujący poufność transmisji już na poziomie trans-

portu.

Libp2p obsługuje także mechanizmy kadencyjnego przechowywania danych (Kademlia DHT), co docelowo umożliwia przechowywanie wiadomości offline oraz realizację zaawansowanych scenariuszy (np. zdecentralizowanej historii czatu czy reputacji peerów).

Metody badawcze:

- **Analiza porównawcza:** Na początku projektu przeprowadzono analizę istniejących rozwiązań komunikacji P2P oraz dostępnych protokołów kryptograficznych (m.in. X25519, ChaCha20-Poly1305), co pozwoliło na wybór najbezpieczniejszych i najwydajniejszych algorytmów do zastosowania w komunikatorze.
- **Prototypowanie iteracyjne:** Implementacja przebiegała w cyklu krótkich iteracji, pozwalających na szybkie weryfikowanie nowych rozwiązań oraz szybkie wykrywanie i usuwanie błędów.
- **Testy funkcjonalne:** Każda kolejna funkcjonalność była testowana przez uruchamianie wielu instancji programu na różnych urządzeniach, a następnie weryfikację poprawności wykrywania peerów, przesyłania wiadomości oraz zachowania w warunkach nieobecności niektórych uczestników.
- **Ewaluacja bezpieczeństwa:** Przeprowadzono testy odporności komunikacji na podsłuch i manipulacje, weryfikując, że treści wiadomości pozostają nieczytelne bez znajomości kluczy kryptograficznych.
- **Analiza wydajności:** Mierzono czas wymiany wiadomości, opóźnienia w wykrywaniu peerów oraz efektywność działania systemu w środowisku wielourządzeniowym.

Techniki i narzędzia programistyczne:

- **Język programowania:** Rust – ze względu na wysoką wydajność, bezpieczeństwo pamięci oraz bogaty ekosystem bibliotek kryptograficznych i sieciowych.
- **Biblioteki:**
    - **sled** – nowoczesna, wydajna baza danych key-value, napisana w Rust, użyta

do lokalnego przechowywania historii rozmów i stanu aplikacji.

- **Kryptografia:** `x25519-dalek`, `chacha20poly1305`, `hkdf`, `sha2` – pozwalająca na realizację E2EE, uzgadnianie kluczy oraz bezpieczne generowanie losowych wartości. Algorytmy te są obecnie standardem w nowych protokołach internetowych (np. Signal, WireGuard).
- **Programowanie asynchroniczne:** `tokio`, `futures` – pozwalające na obsługę wielu zadań i zdarzeń równocześnie (networking, komunikacja z użytkownikiem, operacje na bazie), bez blokowania głównego wątku programu.
- **REPL, parsery argumentów:** `clap`, `shell-words` – pozwalające na wygodne i elastyczne przetwarzanie poleceń użytkownika z linii komend.
- **Inne biblioteki:** `base64`, `hex`, `once-cell` (zarządzanie stanem globalnym), `tracing` (logowanie zdarzeń i diagnostyka).

- **Aparatura i środowisko testowe:**
  - Laptopy i komputery PC z systemami Linux i Windows, podłączone do tej samej sieci lokalnej;
  - Sieć Wi-Fi oraz przewodowa (Ethernet) do testów pracy w różnych warunkach;
  - Narzędzia do analizy ruchu sieciowego (np. Wireshark), pozwalające na monitorowanie transmisji i weryfikację szyfrowania.

Opis kluczowych modułów aplikacji:

1. **Moduł `main.rs`:** Odpowiada za inicjalizację systemu, ładowanie tożsamości użytkownika, przygotowanie bazy danych, uruchomienie stosu sieciowego i obsługę głównej pętli zdarzeń.

2. **Moduł `identity.rs`:** Zapewnia generowanie oraz odczyt kluczy Ed25519 wykorzystywanych przez `libp2p` do identyfikacji peerów, a także generuje unikalne pseudonimy.

3. **Moduł `encryption.rs`:** Implementuje mechanizmy kryptograficzne, zarządza klu-

czami X25519 oraz procesem szyfrowania i deszyfrowania wiadomości.

4. **Moduł `database.rs`:** Zawiera funkcje do zapisywania i odczytu wiadomości z lokalnej bazy danych oraz zarządza historią komunikacji.

5. **Moduł `commands.rs`:** Implementuje interfejs tekstowy REPL, obsługuje polecenia użytkownika oraz przekazuje żądania do odpowiednich funkcji systemu.

6. **Moduł `protocol.rs`:** Definiuje protokoły komunikacji, kodery/decoder wiadomości oraz łączy funkcje protokołu `libp2p` z mechanizmem mDNS.

Obecna wersja systemu zapewnia realizację podstawowych celów projektu: automatyczne wykrywanie użytkowników, uzgadnianie kluczy kryptograficznych, bezpieczne przesyłanie i przechowywanie wiadomości, podstawową obsługę komunikacji offline oraz intuicyjny interfejs tekstowy. Dalsze etapy rozwoju przewidują rozszerzenie funkcjonalności o obsługę DHT, interfejs webowy oraz integrację zaawansowanych testów bezpieczeństwa i wydajności.

C2. Efekty realizacji projektu

C3. Użyteczność projektu

C4. Autoewaluacja zespołu projektowego

C5. Użyte materiały i bibliografia związana z realizacją projektu

C6. Spis załączników