# Relatório da primeira entrega do projeto final de Redes de Computadores

Nome: Arthur Fonseca Vale
Matrícula: 221031120

## 1. Introdução

Este relatório refere-se à primeira entrega do projeto "Sistema de Chat P2P com Salas em Rede Local". Nesta etapa, foi desenvolvido o servidor tracker, responsável pela autenticação de usuários e manutenção da lista de peers ativos e salas de chat. O tracker atua como um ponto central para a descoberta de peers na rede local, permitindo que os usuários se autentiquem, registrem-se e obtenham informações sobre outros peers e salas disponíveis.

O sistema utiliza criptografia RSA para proteger as senhas durante a transmissão entre o peer e o tracker. O tracker possui uma chave pública que é utilizada pelos peers para criptografar as senhas antes de enviá-las.

## 2. Estrutura do Tracker

O servidor **tracker.py** atua como o núcleo central do sistema, responsável por:

- **Autenticação**: Valida usuários via senhas criptografadas com RSA.
- **Gerenciamento de Peers**: Mantém registro dos peers ativos (usuário, IP, porta).
- **Controle de Salas**: Gerencia salas de chat e seus membros.
- **Segurança**: Opera com chaves RSA (pública/privada) para criptografia.

**Principais componentes:**

- **Inicialização**:
    - Gera par de chaves RSA ao iniciar.
    - Carrega usuários do arquivo users.json.
    - Inicia socket TCP na porta 5000.

- **Estrutura de dados:**

```
# Estruturas de dados
self.users = {}  # {username: {'password_hash': hash, 'salt': salt}}
self.active_peers = {}  # {username: (ip, port)}
self.rooms = {}  # {room_name: set(members)}
```

- **Threads**: Usa threading para lidar com múltiplos clientes simultaneamente.

```
Arthur@ArthurBook4 /c/VScode Workshop/tracker_redes > python tracker.py
[*] Carregados 2 usuários
[*] Tracker ouvindo em 0.0.0.0:5000
[*] Chave pública:
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAyvVOtA32DaHk2z62Rsem
N1XbDAvzC6PYZip1A2lhi/dxkmsuMk5FS+l7jg7w8EpSKb92F23ss/u2MOmvVvOU
XujQIncMAraaabuw+cl5+up6j6tTSMHqfuRQ+kGYiDhYj52rNTFxJpjvWiSMuip8
7BIF/T23cluX/J+X+LtE2n0y6LwHacr8rDWCtCv3nVu8QkMyg058D3M3LOla9m5G
pUS1uiQLrvfxyfQO0P56JDtxiIr2LsTejLwSYzVwNFRZh8tCB0hfEPdw2QJ0Z/n9
e0ZntrxveQsPhPtPV4JpzQ+YD+t42rSYlGZpnhHptwjcbWNAR5idp9vvIK/DY6sm
GQIDAQAB
-----END PUBLIC KEY-----

[+] Conexão estabelecida com 127.0.0.1
[+] Conexões ativas: 1
[2025-06-03 19:57:58.689651] REQ: {'cmd': 'GET_PUBKEY'}
```

Log do tracker mostrando chave pública e conexões ativas

## 3. Protocolo de autenticação

A autenticação ocorre em 3 etapas:

1. **Obtenção da Chave Pública**:
    a. Peer envia {"cmd": "GET_PUBKEY"}.
    b. Tracker responde com sua chave pública em formato PEM.
2. **Criptografia da Senha (Peer)**:
    a. Peer criptografa a senha com a chave pública do tracker:

```
encrypted = public_key.encrypt(password, padding.OAEP(...))
password_b64 = base64.b64encode(encrypted).decode()
```

**Login/Registro**:

- **Registro**: tracker gera um *salt*, faz *hash* SHA-256 da senha, e salva em users.json.

```
{"cmd": "REGISTER", "user": "alice", "password": "senha_criptografada"}
```

- **Login**: tracker descriptografa a senha, verifica o *hash*, e registra o peer em active_peers.

```
{"cmd": "LOGIN", "user": "alice", "password": "senha_criptografada",
"peer_port": 6000}
```


Captura de pacotes mostrando troca de comandos de autenticação

## 4. Formato de mensagens e dados.

**Mensagens Tracker ↔ Peer (TCP/JSON):**

| Comando | Exemplo de Requisição | Resposta de Sucesso |
|---|---|---|
| LOGIN | {"cmd":"LOGIN","user":"alice","password":"...","peer_port":6000} | {"status":"success","message":"Login realizado"} |
| REGISTER | {"cmd":"REGISTER","user":"bob","password":"..."} | {"status":"success","message":"Registro bem-sucedido"} |
| LIST_PEERS | {"cmd":"LIST_PEERS"} | {"status":"success","peers":[{"username":"alice","ip":"192.168.1.5","port":6000}]} |

Dados armazenados:

- **user.json:**

```
{
  "alice": {
    "password_hash": "a1b2c3...",
    "salt": "f1e2d3..."
  }
}
```

- **Dados em Memória (Tracker):**
  - Peers ativos: {"alice": ("192.168.1.5", 6000)}
  - Salas: {"dev": {"alice"}}

## 5. Funcionamento do Sistema

**Fluxo Típico:**

1. Peer obtém chave pública do tracker via GET_PUBKEY.
2. Usuário registra-se ou autentica-se com senha criptografada.
3. Após login, o peer:
   a. Inicia um socket na porta 6000 para conexões P2P.
   b. Pode listar peers/salas ou criar salas via comandos CLI.

**Exemplo de Sessão:**

```
# Terminal do Tracker:
[*] Tracker ouvindo em 0.0.0.0:5000
[*] Chave pública: ...
[+] Conexão estabelecida com 192.168.1.5

# Terminal do Peer:
1. Login
```

```
> Username: alice

> Password: ******

Resposta do tracker: {"status": "success", "message": "Login

realizado"}
```

```
=== SISTEMA DE CHAT P2P ===
1. Login
2. Registrar novo usuário
3. Sair
> 1
Username: teste
Password: teste
Resposta do tracker: {'status': 'success', 'message': 'Login realizado'}

Bem-vindo(a), teste!
1. Listar peers ativos
2. Listar salas
3. Criar sala
4. Logout
[*] Ouvindo conexões P2P na porta 6000
> 1

Peers ativos:
 - teste (127.0.0.1:6000)

Bem-vindo(a), teste!
1. Listar peers ativos
2. Listar salas
3. Criar sala
4. Logout
```

CLI do peer mostrando menu e listagem de peers

## 6. Conclusão

O tracker opera como um serviço centralizado para:

- Autenticação segura com RSA.
- Descoberta de peers.
- Gerenciamento de sessões em rede local.