Faculdade de Ciências da Universidade de Lisboa

Departamento de Informática

Mestrado em Engenharia Informática

RELATÓRIO DE PROJETO

Privacidade e Segurança de Dados

Secure P2P Messaging App (Phase 2)

Rodrigo Craveiro Rodrigues (Nº64370)

Diogo Serrano Sargaço (Nº58252)

André Filipe Diniz Belo (Nº58211)

Professor: **Doutor Bernardo Ferreira**

1º Semestre Letivo 2024/2025

dezembro 2024

Índice

1. Introdução 3

2. Funcionalidades Adicionadas e Implementações 3

2.1 Conversas em Grupo 3

2.1.1 Implementação Técnica 3

2.1.2 Desafios e Soluções 4

2.2 Armazenamento de Longo Prazo e Alta Disponibilidade 4

2.3 Pesquisa de Mensagens Preservando a Privacidade 4

2.3.1 Arquitetura da Implementação 4

2.3.2 Desafios e Soluções 4

2.4 Sistema de Recomendação Preservando a Privacidade 5

3. Comparação com a Primeira Fase 5

4. Desafios e Soluções Implementadas 5

5. Referências 5

1. Introdução

Nesta segunda fase do projeto, a aplicação foi melhorada com a adição de novas funcionalidades que aumentam a complexidade e a utilidade do sistema. Além de suportar **comunicação segura ponto-a-ponto** de forma **descentralizada**, foram adicionadas novas funcionalidades de forma a possibilitar **conversas em grupo** aplicando o método **ABE** (com base em tópicos de interesse), **armazenamento persistente** e **replicado das mensagens**, funcionalidades de **pesquisa eficiente de palavras-chaves**, e um sistema de **recomendações** recorrendo a ***machine learning*** consoante o perfil do cliente. As melhorias foram feitas com uma atenção especial à **segurança**, **privacidade** e **desempenho**, utilizando conceitos avançados de criptografia e estratégias de armazenamento seguro abordados nas aulas.

2. Funcionalidades Adicionadas e Implementações

2.1 Conversas em Grupo

O sistema permite a criação e a gestão de grupos por tópicos de interesse. Conversas em grupo em sistemas de mensagens seguras são desafiadoras porque requerem que uma única chave de sessão seja compartilhada entre múltiplos membros, garantindo que apenas os participantes autorizados possam desencriptar as mensagens. O objetivo é proporcionar **confidencialidade**, **integridade** e **autenticação** das mensagens.

2.1.1 Implementação Técnica

Utilizamos ***"getrandbits(256)"*** para gerar uma **chave de grupo partilhada** por todos os membros do grupo durante a criação do grupo. Implementámos ***Shamir Secret Sharing*** para dividirmos a chave em diferentes shares, com o mínimo de duas, com o `split_secret` da biblioteca sslib. Isto fez com que cada share tivesse uma parte da chave de grupo, e que se precisasse de um ***threshold*** para reconstruir a chave e poder continuar a encriptação e desencriptação de chaves. Adicionamos este grupo à base de dados ***realtime database*** da **Firebase** e à **Amazon AWS S3**, em que os grupos têm os parâmetros: "***topic***", "***members***", "***shares***", "***threshold***" e "***prime_mod***". Para o envio de mensagens, vamos buscar as shares todas do grupo na Firebase, e tentamos reconstruir a chave de grupo usando o `recover_secret` da biblioteca sslib. Depois de buscarmos a chave de grupo, encriptamos com a classe *Cipher*, utilizando o algoritmo **AES** e um ***nonce*** **GCM**, tal como a *tag*. Isto tudo é junto numa variável `encrypted_message`, que é mandada quando se faz o `send_message` na função de mandar mensagens, quando a entidade é um grupo. A desencriptação segue o processo reverso da encriptação. Durante a conexão de grupos, verificamos a existência do grupo nas réplicas nas duas bases de dados diferentes, depois verificamos se há o número mínimo de shares possível para reconstruir a chave do grupo, se for possível então verifica se o utilizador já faz parte da lista dos membros e vê se a *length* da

lista dos membros é maior que o número de shares, se for reconstrói o grupo todo. No final cria uma entidade *group* para futuras comunicações entre membros.

2.1.2 Desafios e Soluções

Durante o processo de teste da comunicação em grupos com 2 pessoas diferentes viu-se que as mensagens enviadas eram replicadas 4 vezes na interface das outras pessoas. Teve de se remover as mensagens duplicadas, visto que se iam buscar todas as mensagens das 4 réplicas.

2.2 Armazenamento de Longo Prazo e Alta Disponibilidade

As mensagens são armazenadas de forma persistente na base de dados **Firebase** e *buckets* da **AWS S3**. A escolha do Firebase foi baseada na sua **escalabilidade** e **facilidade** de integração com Python, tal como AWS S3. Para garantir a **alta disponibilidade**, recorremos à **replicação das mensagens** em 4 diferentes localizações dentro do Firebase, e *buckets* s3. Esta estratégia permite que os dados sejam recuperados mesmo em caso de comprometimento de um único ponto de armazenamento. Os utilizadores quando entram na aplicação são registados na pasta dos ***users*** nas bases de dados. Para garantir o ***Shamir Secret Sharing*** nesta parte do programa, dividimos a chave pública criada com ***diffie-hellman elyptic curve*** em várias shares, espalhando-as pelos *buckets*/réplicas diferentes, criando o utilizador nas bases de dados com "***topic***", "***prime***", "**threshold**" e "***public_key_share***". Caso o número de réplicas existentes seja menor que o ***threshold*** ou o utilizador não exista, cria-se uma **chave pública**, repetindo este processo de dividir *shares* e dividi-las pelas réplicas. Como os dados são encriptados antes de serem enviados para a nuvem, mesmo um ataque ao provedor não comprometerá a **privacidade das mensagens**, assim este método promove o critério de **autorização** com a proteção contra acessos não autorizados.

2.3 Pesquisa de Mensagens Preservando a Privacidade

2.3.1 Arquitetura da Implementação

Técnicas como **encriptação homomórfica**, índices encriptados, ou pesquisa baseada em ***tokens*** seguros são utilizadas para realizar buscas sem expor o conteúdo. Foram criados **índices encriptados** para cada mensagem, facilitando e **otimizando a pesquisa sem revelar o conteúdo**. A pesquisa é feita **localmente**, comparando palavras-chave fornecidas pelo utilizador com os índices armazenados. A interface permite ao utilizador **pesquisar por palavras-chave** e retorna mensagens relevantes, mantendo a simplicidade de uso e a segurança.

2.3.2 Desafios e Soluções

A pesquisa de mensagens em sistemas criptografados representa um desafio, pois não se pode comprometer a privacidade das mensagens. Implementámos uma indexação eficiente para

melhorar o **desempenho** na pesquisa de mensagens. Garantimos **confidencialidade** onde os resultados da pesquisa não exponham informações sensíveis, mesmo em caso de acesso não autorizado.

2.4 Sistema de Recomendação Preservando a Privacidade

Utilizamos técnicas de **anonimização** e **análise segura** de dados para construir um sistema que não compromete a **privacidade**. Nenhum dado sensível é enviado ou armazenado sem encriptação. As mensagens são **analisadas localmente** para extrair **palavras-chave relevantes**. O sistema gera recomendações com base em palavras como "desporto", “música” ou “viagem”. O sistema gera **recomendações de forma descentralizado** com base em ***machine learning***. As recomendações são armazenadas no Firebase, mas sem associar diretamente o utilizador às suas mensagens. Isso garante que o sistema seja seguro e preserva o anonimato. Para garantir a **privacidade**, os identificadores dos utilizadores são substituídos por **pseudónimos** antes de serem enviados para a nuvem, promovendo **anonimização**. As mensagens são processadas e analisadas em **ambiente seguro**, garantindo que nenhum dado sensível é exposto.

3. Comparação com a Primeira Fase

Na primeira fase, a **troca de chaves** era feita de forma ponto-a-ponto. Agora, expandimos para suportar **grupos de conversa**. Na primeira fase, o **armazenamento** das mensagens era localmente, enquanto na segunda fase, introduzimos **armazenamento persistente** em nuvem. **A interface gráfica** foi aprimorada com suporte a grupos, pesquisa de mensagens e recomendações.

4. Desafios e Soluções Implementadas

Descentralização vs. Performance: Implementar um sistema que seja tanto descentralizado quanto eficiente foi um desafio significativo. Utilizámos um modelo híbrido que aproveita o melhor de ambos os mundos. **Segurança e Privacidade:** Proteger os dados dos utilizadores contra acesso não autorizado enquanto se mantém a usabilidade do sistema exigiu o uso de técnicas avançadas de criptografia. **Gerir Múltiplas Conexões:** A introdução de grupos e a necessidade de armazenamento em nuvem aumentaram a complexidade da gestão de conexões, exigindo uma gestão otimizada de *threads*.

5. Referências

[1] Bernardo Ferreira (2024). https://moodle.ciencias.ulisboa.pt/course/view.php?id=5540.