



## Segurança e Aplicações de Hardware Confiável

---

### *Secure Data Lakes*

---

Maria Carreira up202408787  
Matilde Simões up202108782  
Ricardo Amorim up202107843

Maio 2025

# Conteúdo

<b>1</b>	<b>Descrição do Problema</b>	<b>2</b>
<b>2</b>	<b><i>Data Lakes</i> [1]</b>	<b>2</b>
2.1	Arquitetura . . . . .	2
2.2	Desafios e Oportunidades . . . . .	3
<b>3</b>	<b>Abordagem e <i>Hardware</i> Confiável Adotado</b>	<b>3</b>
3.1	Abordagem . . . . .	3
3.2	<i>Hardware</i> Confiável Adotado . . . . .	4
<b>4</b>	<b>Requisitos e Solução do Projeto</b>	<b>5</b>
4.1	Requisitos Funcionais . . . . .	5
4.2	Requisitos de Segurança . . . . .	6
4.3	Solução Desenvolvida . . . . .	6
<b>5</b>	<b><i>Datasets</i> Utilizados</b>	<b>8</b>
5.1	Dados Hospitalares . . . . .	8
5.2	Dados Laboratoriais . . . . .	8
<b>6</b>	<b>Descrição da Implementação</b>	<b>9</b>
6.1	Ambiente de Desenvolvimento . . . . .	9
6.2	Bibliotecas e Ferramentas Utilizadas . . . . .	10
6.3	Estrutura da Implementação . . . . .	10
6.4	App/ . . . . .	11
6.4.1	Atestação Remota . . . . .	11
6.4.2	Geração e Persistência da Chave Mestra . . . . .	12
6.4.3	Execução de Operações . . . . .	13
6.5	Enclave/ . . . . .	14
6.6	Comunicação entre o Cliente, App e o Enclave . . . . .	15
6.7	Ficheiros Auxiliares . . . . .	16
<b>7</b>	<b>Fluxo de Execução</b>	<b>17</b>
<b>8</b>	<b>Validação dos Requisitos</b>	<b>19</b>
<b>9</b>	<b>Conclusão</b>	<b>19</b>

# 1 Descrição do Problema

O armazenamento e processamento de grandes volumes de dados em sistemas de *data lakes* tem se tornado cada vez mais comum, especialmente em plataformas *cloud-based*. Estes repositórios permitem armazenar dados em diversos formatos e podem ser utilizados de modo a tornar possível a análise e o acesso à informação por múltiplas partes. No entanto, este trabalho colaborativo em ambientes de *data lakes* apresenta inúmeros desafios relacionados à segurança e à proteção dos dados armazenados.

O objetivo deste projeto passa por explorar soluções em *hardware* confiável para o armazenamento e processamento seguro de dados, com destaque para dados médicos.

Como *proof-of-concept*, propõe-se o desenvolvimento de uma base de dados remota, gerida por várias partes, onde dados sensíveis possam ser armazenados de forma segura. O acesso a essa base de dados irá proceder-se de modo a garantir que todos os resultados obtidos sejam agregações de dados (por exemplo, médias), mantendo assim a privacidade e a segurança de todos os envolvidos.

## 2 Data Lakes [1]

Os *data lakes* são grandes repositórios de dados diversos, armazenados em diferentes formatos e, frequentemente, sem metadados.

Podem ser utilizados para desacoplar produtores e consumidores de dados, por exemplo, quando estes produtores correspondem a possíveis sistemas *legacy*. Também na área da ciência de dados, estes repositórios constituem um local de armazenamento de dados experimentais muito conveniente.

A criação e o uso de dados podem ser realizados de forma autónoma, sem necessidade de coordenação com outras partes. Para além disso, o armazenamento compartilhado de um *data lake*, aliado a uma *framework* computacional, fornece a infraestrutura básica necessária para ambientes colaborativos e reutilização de conjuntos de dados de grande dimensão.

### 2.1 Arquitetura

*Data lakes* agregam dados de diversas fontes, incluindo sistemas *legacy*, redes sociais ou *web* e *data brokers*. Estes repositórios utilizam *frameworks* como *Hadoop* e *Apache Spark*, para o armazenamento de *datasets*, e várias ferramentas, para a sua gestão e para várias tarefas da próxima subsecção 2.2.

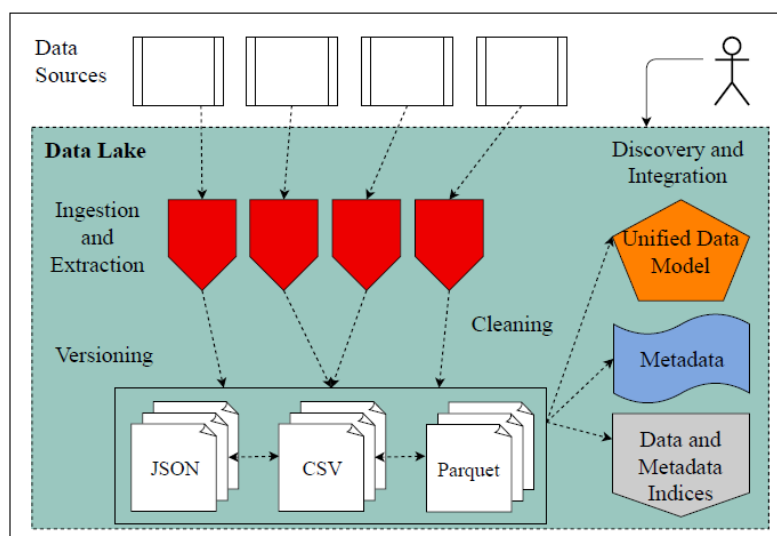


Figura 1: Exemplo de um sistema de gestão de um *data lake*. [1]

## 2.2 Desafios e Oportunidades

Contudo, a gestão de *data lakes* apresenta desafios significativos. Seguem-se alguns tópicos, abordados no artigo [1], que levantam verdadeiros problemas, ainda por solucionar.

- **Ingestão:** Processo de importação de dados de diversas fontes.  
Desafio: Necessidade de paralelismo e baixa latência para suportar a ingestão de dados de alta velocidade, para além de técnicas de indexação mais eficientes.
- **Extração:** Conversão de dados brutos (*raw*) para formatos estruturados ou semi-estruturados.  
Desafio: Os métodos atuais transformam um ficheiro de cada vez. No entanto, deseja-se otimizar extrações futuras utilizando conhecimento anterior.
- **Limpeza:** Identificação e correção de erros e inconsistências nos dados.  
Desafio: A ausência de esquemas unificados dificulta a deteção destes erros. Estratégias de *data cleaning* coletivas podem melhorar a qualidade dos dados.
- **Descoberta:** Processo de busca e identificação de *datasets* relevantes no *data lake*.  
Desafio: Uma busca eficiente seria, provavelmente, solucionada por uma boa catalogação dos dados e um bom mecanismo de navegação. Este mecanismo poderia, por exemplo, permitir a navegação através de grafos ou estruturas em hierarquia.
- **Gestão de Metadados:** Desenvolvimento e organização de informações descritivas sobre os dados armazenados (metadados).  
Desafio: Sem metadados, os *data lakes* tornam-se "*data swamps*". Sistemas como o *Google Dataset Search* e o *Constance* propõem abordagens para a extração e o desenvolvimento de metadados.
- **Integração:** Combinação de dados de diferentes fontes para criar um modelo unificado.  
Desafio: Para atingir integração de dados *on-demand*, é necessário lidar com a heterogeneidade dos *data lakes* e, possivelmente, executar extração e limpeza como parte da integração.
- **Versionamento:** Controlo das múltiplas versões dos dados ao longo do tempo.  
Desafio: *Data lakes* são dinâmicos, dado que exigem mecanismos eficientes para gerir versões e a evoluções dos dados sem comprometer o seu armazenamento e a sua recuperação.

## 3 Abordagem e *Hardware* Confiável Adotado

### 3.1 Abordagem

A computação colaborativa refere-se ao processamento de dados realizado por múltiplas entidades, permitindo a partilha de informação sem que os participantes tenham acesso direto aos dados sensíveis uns dos outros. No contexto de *data lakes* seguros, este paradigma torna-se especialmente relevante em áreas críticas como a saúde, onde hospitais, laboratórios e centros de investigação precisam de aceder e analisar dados de forma colaborativa, respeitando sempre a privacidade dos pacientes.

Neste contexto, o *Intel SGX (Software Guard Extensions)* é apresentado como uma solução eficaz para viabilizar a computação colaborativa segura, ao permitir que os dados sejam processados dentro de enclaves protegidos. Estes enclaves asseguram que a informação permaneça isolada e cifrada durante todo o processo de computação e garante que apenas estatísticas, como

médias, contagens ou distribuições, sejam revelados, sem nunca expor os registos individuais dos participantes.

Desta forma, torna-se possível implementar uma base de dados distribuída e segura, onde cada entidade contribui com os seus próprios dados para um processo de análise colaborativa, sem abdicar do controlo sob a confidencialidade da sua informação. Os acessos sensíveis aos dados decifrados são executados diretamente dentro dos enclaves *SGX*, pois só assim se assegura que os dados nunca são expostos em texto legível fora do ambiente de execução confiável.

Adicionalmente, o mecanismo de atestação remota do *SGX* desempenha um papel crucial neste modelo, pois permite que cada participante verifique a integridade e a autenticidade do enclave antes de partilhar os seus dados. Esta funcionalidade reforça a confiança mútua entre os intervenientes, garantindo que o ambiente de execução é legítimo e íntegro.

Assim, a integração do *SGX* em contextos de computação colaborativa permite conciliar segurança, privacidade e colaboração, tornando possível a análise partilhada de dados sensíveis (como os de natureza médica) sem comprometer os princípios de confidencialidade e autonomia sobre a gestão da informação de cada entidade envolvida. Esta abordagem alinha-se com os objetivos do projeto, promovendo um sistema robusto, confiável e escalável para o processamento seguro de informação médica.

### 3.2 *Hardware* Confiável Adotado

A utilização de *hardware* confiável é essencial para garantir *data lakes* seguros, especialmente quando se trata de dados sensíveis, como informações médicas. É fundamental garantir a segurança nestes ambientes, onde múltiplos utilizadores precisam do acesso aos dados agregados sem comprometer a privacidade individual. Uma das soluções mais eficazes para este tipo de problema é a utilização do *Intel SGX*, que é um ambiente de execução confiável (*TEE*) que permite o processamento seguro dos dados e protege contra diversos ataques, incluindo ataques de componentes com altos privilégios dentro do sistema [2].

O *Intel SGX* é uma tecnologia baseada em *hardware* que possibilita a execução de código e o processamento de dados dentro dos enclaves, o que impede acessos não autorizados e modificações indevidas. A escolha do *SGX* justifica-se pelas seguintes vantagens:

- **Confidencialidade:** Os dados dentro dos enclaves *SGX* permanecem cifrados e inacessíveis, mesmo para administradores do sistema, prestadores de serviço *cloud* ou atacantes com acesso privilegiado.
- **Integridade:** O ambiente de execução garante que apenas código confiável é executado dentro dos enclaves, prevenindo alterações não autorizadas.
- **Atestação Remota:** O *SGX* fornece mecanismos para verificar a integridade e a autenticidade do código em execução dentro de um enclave. Este processo permite que um utilizador verifique a integridade do enclave antes de interagir com ele, assegurando que os dados processados não foram adulterados.
- **Minimização da Superfície de Ataque:** Ao isolar cálculos sensíveis do sistema operativo, o *SGX* reduz significativamente o risco de ataques, como a manipulação de memória, injeção de código e explorações ao nível do *kernel*.

Assim, o *SGX* assegura que os enclaves são isolados e protegidos, permitindo que dados sensíveis sejam processados com segurança dentro do *CPU*, sem ficarem expostos ao software privilegiado [2].

Os dados processados dentro dos enclaves *SGX* estão protegidos contra acessos e modificações não autorizadas devido à estrutura da arquitetura que isola os enclaves em zonas protegidas da

memória. Apenas o código autorizado dentro do enclave pode aceder e operar sobre os dados, impedindo que mesmo utilizadores com privilégios administrativos ou atacantes que comprometam o sistema operativo tenham acesso a informações confidenciais. Além disso, as chaves criptográficas utilizadas para proteger os dados nunca saem do enclave, eliminando riscos associados a ataques externos.

A confidencialidade dos dados dentro dos enclaves é garantida pelo *Memory Encryption Engine (MEE)*, que cifra automaticamente toda a informação na *Processor Reserved Memory (PRM)*. Isto impede que processos externos, incluindo o próprio sistema operativo, tenham acesso direto aos dados [2]. Os dados que se encontram fora do *CPU* permanecem cifrados, garantindo que qualquer tentativa de leitura, por partes não autorizadas, resulte apenas em informação ilegível. Apenas o *CPU*, através do *MEE*, é capaz de realizar a decifração em tempo real dos dados, garantindo que o único local onde podem ser lidos em texto legível é dentro do enclave, enquanto estão a ser processados.

A integridade dos enclaves é assegurada pelo mecanismo de atestação remota, que previne *rollback attacks*, em que o atacante tenta restaurar um enclave para um estado anterior vulnerável, que pode conter falhas de segurança já corrigidas. Assim, é assegurado que apenas a versão mais recente e confiável do enclave seja executada, evitando a reutilização de estados comprometidos. Além disso, a criptografia aplicada pelo *MEE* protege os dados contra ataques de extração de dados sensíveis diretamente da memória *RAM*, como os *cold boot attacks*. Os dados guardados na *DRAM* não desaparecem imediatamente após se desligar o sistema, logo um atacante pode tentar recuperá-los. Assim, o *MEE* impede que informações sensíveis sejam recuperadas mesmo que um atacante tenha acesso físico ao hardware.

Apesar da segurança avançada oferecida pelo *SGX*, existem potenciais vulnerabilidades que podem ser exploradas para comprometer a sua proteção. Um dos principais desafios são os ataques de *side channels* [3], que podem extrair informações sensíveis ao analisar padrões de acesso à memória, medições de tempo de execução ou consumo de energia. Além disso, falhas no próprio código do enclave podem ser exploradas para executar instruções maliciosas e comprometer a segurança do sistema [4]. Outras ameaças incluem enclaves maliciosos [5], que podem enganar aplicações legítimas e obter dados confidenciais, e ainda falhas na implementação do *SGX*, como demonstrado em ataques como *Foreshadow* [6] <https://www.intel.com/content/www/us/en/security-center/advisory/intel-sa-00161.html>, que explora vulnerabilidades de microarquitetura para aceder a informações protegidas.

## 4 Requisitos e Solução do Projeto

### 4.1 Requisitos Funcionais

Os principais requisitos funcionais definidos para o sistema são os seguintes:

- **Armazenamento seguro de dados sensíveis:** O sistema deve permitir que cada entidade armazene os seus dados de forma remota (*cloud*), assegurando que estes estão protegidos por mecanismos de criptografia e nunca são expostos fora dos enclaves *SGX*.
- **Cálculo de estatísticas:** O sistema deve suportar consultas sobre os dados de múltiplos participantes, retornando apenas resultados estatísticos (por exemplo: médias, somas, contagens) e nunca dados individuais.
- **Colaboração entre entidades:** Deve ser possível permitir que múltiplas entidades colaborem no processamento e análise de dados, mantendo controlo sob a própria informação.

- **Verificação da integridade do enclave (Atestação remota):** Cada participante deve ser capaz de verificar, antes da partilha dos dados, se o enclave *SGX* é legítimo e está em conformidade com os parâmetros de integridade e segurança.
- **Autenticação dos participantes:** O sistema deve incluir um processo de autenticação que assegure que apenas entidades autorizadas possam aceder ao sistema e participar na computação colaborativa.
- **Solução escalável:** A solução deve ser escalável, permitindo a integração de novas entidades e o aumento do volume de dados sem comprometer o desempenho ou a segurança do sistema.

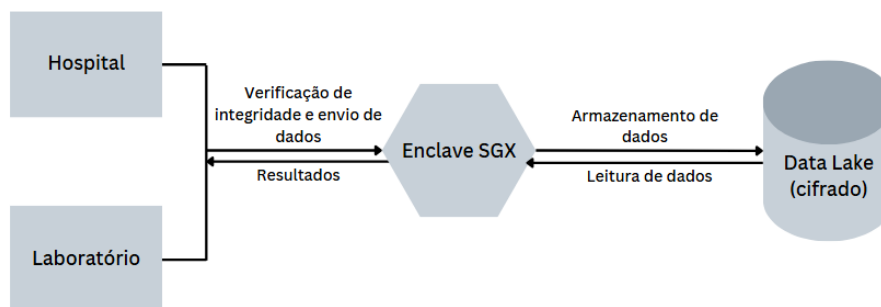
## 4.2 Requisitos de Segurança

Tendo em consideração a natureza sensível dos dados, são definidos os seguintes requisitos de segurança:

- **Confidencialidade dos dados:** Os dados devem estar protegidos contra acessos não autorizados, mesmo por parte de administradores de sistema ou fornecedores de serviços.
- **Integridade do código e dos dados:** É essencial garantir que o código em execução dentro do enclave não foi adulterado, bem como assegurar a integridade dos dados durante todo o processo.
- **Isolamento na execução (Enclaves *SGX*):** A computação sensível deve ocorrer exclusivamente dentro de enclaves protegidos.
- **Resistência a ataques internos:** A implementação deve ser concebida para resistir a ataques de utilizadores/componentes maliciosos com acesso privilegiado ao sistema.

## 4.3 Solução Desenvolvida

A solução proposta, ilustrada na Fig.2, utiliza enclaves *SGX* para garantir a computação colaborativa segura entre um hospital e um laboratório, no contexto de *data lakes*, com dados sensíveis.



**Figura 2:** Solução proposta.

Neste modelo, é utilizado um servidor para verificar a *quote* e depois é retornado um *token* para as entidades – atestação remota – para garantir que o ambiente de execução é legítimo e não foi comprometido. Para além disso, as chaves públicas de cada entidade estarão incorporadas diretamente no código, *hardcoded*, permitindo verificar a autorização das entidades e se a informação guardada corresponde efetivamente a essas entidades autorizadas a aceder ao sistema. Após esta verificação, os dados são enviados por um canal seguro (SSH) para o enclave *SGX*, para serem processados de forma isolada e protegida.

O enclave atua como um núcleo de confiança no sistema, sendo responsável por processar os dados recebidos e realizar os cálculos pedidos de forma segura, sem os expor ao sistema operativo ou a outros participantes. Após o processamento, apenas os resultados são devolvidos às entidades (por SSH na mesma), garantindo a confidencialidade dos dados individuais. O enclave também é capaz de aceder e ler dados previamente guardados.

Para que o enclave não tenha apenas uma única operação e para permitir que possa executar múltiplos cálculos de forma segura, serão definidas diversas operações matemáticas específicas que abrangem diversas estatísticas. O objetivo é aumentar as funções do sistema e ao mesmo tempo manter o controlo e a confidencialidade dos dados processados. Para garantir que nenhuma entidade realize operações de forma isolada ou aceda a informação sensível, será exigido que todas as entidades assinem cada operação. Apenas quando todas as assinaturas forem validadas, a operação será autorizada e executada pelo enclave.

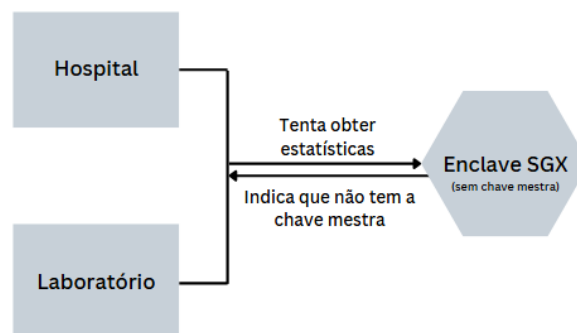
Este fluxo de interação assegura que os dados sensíveis nunca são expostos fora do enclave, os participantes mantêm controlo sobre os seus dados e a confidencialidade e a integridade são preservadas ao longo de todo o ciclo do processamento e do armazenamento no *data lake*.

Para garantir a robustez e continuidade do sistema face a eventuais falhas do enclave *SGX* ou do hardware subjacente, foi adotada uma estratégia baseada em *key wrapping*. Este mecanismo permite a recuperação segura da chave mestra sem que esta permaneça armazenada de forma persistente dentro do enclave, respeitando os princípios de segurança e confidencialidade exigidos.

Na solução implementada, a chave mestra encontra-se armazenada na *cloud*, de forma cifrada. A cifração é realizada utilizando chaves privadas individuais das entidades participantes, como hospitais ou laboratórios. Em caso de reinicialização do enclave ou de migração para um novo *hardware*, as entidades não enviam a chave mestra diretamente, mas sim apenas a sua própria chave privada. De seguida o enclave acede à versão cifrada da chave mestra armazenada na *cloud* e realiza o processo de decifração dentro do ambiente seguro do enclave. Este processo permite recuperar a chave mestra sem comprometer a sua confidencialidade.

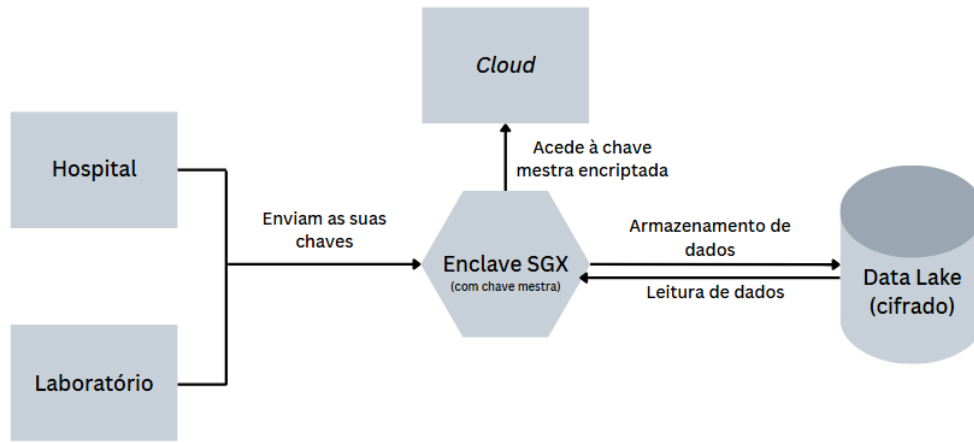
Esta abordagem apresenta várias vantagens significativas. Por um lado, evita o armazenamento direto de informação sensíveis dentro do enclave, reduzindo a superfície de ataque. Por outro lado, garante que mesmo após falhas técnicas ou reinicializações, o enclave pode restabelecer o seu estado criptográfico e continuar a operar sem grandes intervenção ou degradação da segurança. Além disso, trata-se de uma solução distribuída e escalável, uma vez que qualquer enclave autorizado pode recuperar a chave mestra desde que obtenha as chaves das entidades participantes.

A Fig.4 ilustra a solução descrita.



**Figura 3:** O enclave comunica que não tem acesso à chave mestra.





**Figura 4:** As entidades partilham as suas chaves privadas.

## 5 *Datasets* Utilizados

Para validar a solução desenvolvida e simular um cenário realista de partilha e processamento colaborativo de dados sensíveis, foram utilizados dois conjuntos de dados sintéticos representativos de um hospital e de um laboratório clínico. Estes dados permitiram testar funcionalidades como autenticação, atestação remota, cifração, controlo de acessos e cálculo de estatísticas num ambiente protegido.

### 5.1 Dados Hospitalares

O primeiro conjunto de dados – `hospital.csv` – representa registos clínicos provenientes de um hospital. Cada linha corresponde a um paciente, contendo os seguintes atributos:

- `healthcare_number`: identificador único do paciente;
- `name`: nome do paciente;
- `age`: idade do paciente;
- `gender`: género (M ou F);
- `admission_date`: data de admissão hospitalar;
- `discharge_date`: data de alta;
- `diagnosis`: diagnóstico clínico.

Este ficheiro inclui diversos registos com diagnósticos como gripe, pneumonia, anemia, diabetes tipo 2, insuficiência renal e fratura do fémur, permitindo testar diferentes cenários clínicos.

### 5.2 Dados Laboratoriais

O segundo conjunto de dados – `laboratory.csv` – contém os resultados dos exames laboratoriais associados aos mesmos pacientes identificados no `hospital.csv`, através da correspondência do campo `healthcare_number`. Cada registo representa um exame realizado e inclui os seguintes atributos:

- `healthcare_number`: identificador único do utente;

- **exam\_type**: tipo de exame realizado (por exemplo, colesterol, glicemia, TSH, leucócitos, entre outros);
- **exam\_date**: data de realização do exame;
- **result\_value**: valor numérico do resultado do exame.

Este conjunto de dados complementa os registos hospitalares com dados laboratoriais relevantes para análises estatísticas e validação dos diagnósticos colaborativa entre instituições.

Ambos os conjuntos de dados foram utilizados para validar a funcionalidade da solução proposta em ambiente *SGX*, demonstrando a capacidade do sistema de tratar dados reais com confidencialidade, integridade e controlo de acessos. Além disso, permitiram testar operações matemáticas/estatísticas, cifração, e, ainda, o mecanismo de *key wrapping* quando existem múltiplos participantes.

## 6 Descrição da Implementação

O primeiro passo, na preparação do ambiente para desenvolver este projeto, consistiu na instalação do *Intel SGX SDK*, que é o *kit* de desenvolvimento oficial disponibilizado pela *Intel* para criar aplicações que utilizam esta tecnologia. Para tal, seguiram-se os procedimentos descritos no repositório oficial da *Intel*, que orientam a compilação e instalação do *SDK* [7]. Com isso, foi possível concluir a instalação e preparar o ambiente necessário para o desenvolvimento com *SGX* [8].

### 6.1 Ambiente de Desenvolvimento

O enclave *SGX* desenvolvido neste projeto é executado numa máquina virtual (VM) da **Microsoft Azure**, especificamente configurada para suportar o *Intel SGX* em **modo hardware** (*HW mode*). Esta configuração garante que o enclave beneficia do isolamento e proteção oferecidos pelo *SGX* em ambiente real, ao invés de simulado.

Para isso, foi utilizada uma VM da série **Standard DC2s v3**, que oferece suporte nativo a enclaves *SGX* e permite a ativação do *SGX* no momento da criação da instância. Esta escolha assegura que:

- O enclave corre com as garantias de segurança reais do *SGX*, com proteção contra ataques de software, mesmo por processos com privilégios elevados no sistema operativo.
- A geração de *reports* e *quotes* para atestação remota é suportada em modo hardware, permitindo integração com serviços externos como o *Azure Attestation*.
- As operações criptográficas internas ao enclave (como geração de chaves, cifração e decifração) são aceleradas por hardware e autenticadas por mecanismos de raiz de confiança.

A instância utilizada neste projeto apresenta as seguintes características:

- **Sistema Operativo:** Ubuntu 22.04 (Linux)
- **Tipo de Instância:** Standard DC2s v3 (2 vCPUs, 16 GiB RAM)
- **Localização:** East US – Zona de disponibilidade 1

A execução em modo HW também foi essencial para a validação da *quote* gerada pelo enclave, uma vez que o *Azure Attestation Service* apenas reconhece plataformas *SGX* genuínas para esse fim.

## 6.2 Bibliotecas e Ferramentas Utilizadas

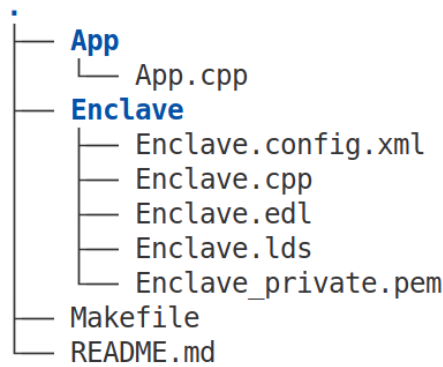
Durante o desenvolvimento do projeto, foram utilizadas várias bibliotecas e ferramentas, tanto para suportar o funcionamento do enclave *SGX* como para permitir funcionalidades adicionais relacionadas com criptografia, estatística, comunicação segura e integração com a cloud. Abaixo são descritas as principais:

- **Intel SGX SDK** – Versão 2.25.100.3: Ferramenta essencial para desenvolvimento de enclaves *SGX*, fornecendo as bibliotecas `sgx_trts.h`, `sgx_tcrypto.h`, `sgx_report.h`, `sgx_utils.h`, entre outras. Estas suportam operações como geração de relatórios de atestação, cifração com AES-GCM, geração de chaves aleatórias, verificação de assinaturas ECDSA e controle de acesso seguro dentro do enclave.
- **OpenSSL** – Versão: 3.0.2: Utilizado na aplicação não-confiável (cliente e aplicação *SGX*) para:
  - Gerar e carregar pares de chaves ECC (curva NIST P-256).
  - Assinar mensagens com ECDSA.
  - Codificar e decodificar dados em Base64.
- **DCAP (*Data Center Attestation Primitives*)** – Versão 1.22.100.3: Conjunto de bibliotecas da Intel para suportar a **atestação remota**, incluindo as funções `sgx_qe_get_quote`, `sgx_qv_verify_quote` e estruturas associadas. Foi usado em conjunto com o **Azure Attestation Service** como verificador externo de *quotes*.
- **Google Cloud SDK (gsutil)** – Versão: 522.0.0: Utilizado na aplicação para fazer **upload** e **download** de ficheiros para o Google Cloud Storage (GCS). Essa integração permite persistir dados cifrados e a chave mestra envolvida (`wrapped_key.bin`) na cloud, mantendo a arquitetura distribuída e segura.
- **POSIX (Linux)** – Bibliotecas padrão como `<unistd.h>`, `<sys/stat.h>`, `<fcntl.h>` e funções como `mkfifo`, `stat` e `time` são usadas para operações com pipes, gestão de tempo, leitura de ficheiros e comunicação entre processos locais.
- **C++ STL** – Versão: 11.4.0: Utilizou-se amplamente a STL para manipulação de dados: `std::vector`, `std::string`, `std::map`, `std::sort`, `std::accumulate`, entre outras, tanto dentro como fora do enclave.
- **Make** – Versão: 4.3: O projeto é compilado com um Makefile personalizado, que organiza a construção do enclave, geração do ficheiro `Enclave.signed.so`, chamada ao `sgx_edger8r`, e permite alternar entre modos HW e SIM.

Estas bibliotecas e ferramentas foram fundamentais para garantir a segurança, portabilidade e escalabilidade do sistema, bem como a sua integração com serviços externos de atestação e armazenamento.

## 6.3 Estrutura da Implementação

Posto isto, apresenta-se na Fig.5 a estrutura modular do programa em C++ implementado. Esta separação observável entre a aplicação e o enclave é recomendada e suportada pela própria *Intel*, permitindo uma melhor organização e minimizando a ocorrência de erros durante o desenvolvimento.



**Figura 5:** Estrutura do projeto.

É importante realçar a existência de funções *ECALL* - chamadas dentro do enclave, e funções *OCALL* - localizadas fora do enclave, mas invocadas a partir do seu interior.

## 6.4 App/

O ficheiro `App.cpp` corresponde à parte da aplicação que é executada fora do enclave, ou seja, no ambiente não confiável. Ele é responsável por controlar o fluxo principal do programa, incluindo a inicialização do enclave, a gestão das interações com o utilizador (como a leitura de comandos ou dados), e o encaminhamento de chamadas para dentro do enclave. Também trata da receção de resultados do enclave e da exibição ou tratamento desses dados fora da zona segura. Resumidamente, serve de ponte entre a lógica segura executada no enclave e o restante ambiente da aplicação.

Inicialmente, o programa **cria e inicializa o enclave** ao utilizar a função `sgx_create_enclave`. Este processo envolve a biblioteca segura do enclave, contida em `Enclave.signed.so`. A lógica da aplicação segue uma sequência bem definida de passos cruciais, que garantem a segurança e funcionalidade do sistema. Estes passos são:

1. **Atestação Remota** – Verifica a legitimidade e integridade do enclave antes de qualquer operação sensível.
2. **Geração e Persistência da Chave Mestra** – Gera uma chave mestra simétrica dentro do enclave e protege-a através de *key wrapping*, permitindo a sua recuperação futura.
3. **Execução de Operações** – Após validada e restaurada a chave mestra, o enclave pode realizar operações seguras, como cifração de dados e cálculo de estatísticas.

Nas próximas secções, cada um destes passos é descrito em detalhe.

### 6.4.1 Atestação Remota

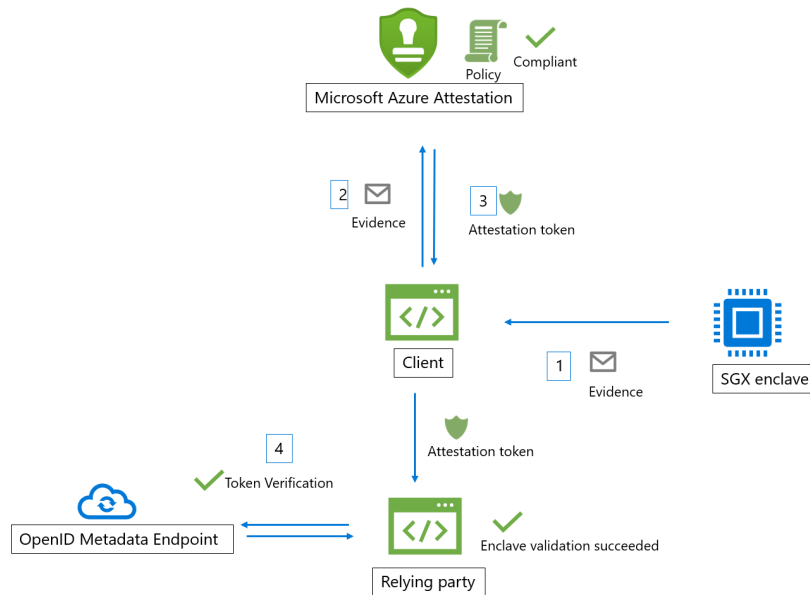
Para garantir que o enclave está a ser executado num ambiente legítimo e confiável, foi implementado um processo de **atestação remota** com recurso ao *Azure Attestation Service*. Este processo permite validar externamente a integridade e autenticidade do enclave, assegurando que o código em execução corresponde exatamente à versão esperada e não foi adulterado.

O processo tem início com a criação de um relatório seguro dentro do enclave, através da função `ecall_create_report`, que invoca `sgx_create_report`. Este relatório contém dados de identificação do enclave, incluindo o `MRENCLAVE` (medida do código carregado) e o `MRSIGNER` (identidade do assinante).

Esse relatório é convertido numa *quote* autenticada, com recurso à função `sgx_qe_get_quote`. A aplicação (`App.cpp`) envia essa *quote* para o **Azure Attestation Service**, que realiza a verificação de forma remota. Este serviço valida a assinatura da *quote*, confirma a autenticidade da plataforma SGX e devolve um resultado que indica se a *quote* é confiável ou não.

No caso desta implementação, **a decisão de confiança é tomada localmente com base na resposta do Azure**, sem emissão nem utilização de *tokens JWT*. No entanto, o *Azure Attestation Service* também disponibiliza a opção de emitir um **token de atestação** (formato *JWT*), que pode ser utilizado noutros cenários. Esse *token*, assinado criptograficamente, pode ser enviado a terceiros (como hospital ou laboratório), permitindo-lhes validar autonomamente a legitimidade do enclave com base nas chaves públicas fornecidas pela *Microsoft* via *OpenID Metadata Endpoint*.

Embora essa abordagem com *token* não tenha sido adotada nesta solução, ela constitui uma alternativa viável para sistemas que requerem uma prova de confiança transportável. A Fig. 6 ilustra o fluxo completo da atestação com o *Azure Attestation Service*.



**Figura 6:** Fluxo de atestação com *Azure Attestation Service*. [9]

#### 6.4.2 Geração e Persistência da Chave Mestra

Após a verificação bem-sucedida da atestação remota, o enclave procede à **geração da chave mestra** (`sk_m`). Esta chave `sk_m` é gerada internamente, invocando uma função implementada no enclave chamada de `ecall_generate_master_key`. Esta função recorre a `sgx_read_rand` para produzir 256 *bits* (32 *bytes*) de entropia criptograficamente segura, correspondentes à chave mestra simétrica.

Para garantir a robustez e continuidade do sistema face a eventuais falhas do enclave *SGX* ou do *hardware* subjacente, foi adotada uma estratégia baseada em *key wrapping*. Este mecanismo permite a recuperação segura da chave mestra sem que esta permaneça armazenada de forma persistente dentro do enclave, respeitando os princípios de segurança e confidencialidade exigidos.

Existem diversas abordagens possíveis para garantir a persistência de uma chave mestra (`sk_m`) em enclaves *SGX*:

- **Selagem (sealing):** técnica suportada pelo *SGX* que permite cifrar dados utilizando uma chave derivada do hardware local. Embora segura, esta abordagem não é portátil entre diferentes dispositivos *SGX*.

- **Servidores de chaves externos:** consistem em delegar o armazenamento da chave a um serviço de terceiros. Contudo, introduzem dependência externa e representam um ponto único de falha.
- **Key wrapping de múltiplas entidades:** a chave mestra é cifrada recursivamente utilizando chaves simétricas (*sk\_i*) fornecidas por diferentes entidades autorizadas.

A solução implementada adota esta última abordagem (*key wrapping* recursivo), por oferecer um equilíbrio entre segurança, resiliência e descentralização. Este mecanismo funciona da seguinte forma:

- O enclave gera internamente a chave mestra *sk\_m*.
- A *sk\_m* é sucessivamente cifrada com cada *sk\_i*, numa ordem definida, criando múltiplas camadas de proteção.
- O resultado final é armazenado na *cloud* (nomeadamente, no *Google Cloud Storage*) em formato binário cifrado.
- Em caso de falha ou reinicialização, o enclave descarrega a versão cifrada e aguarda que as entidades reenviem as suas *sk\_i* para que o processo de decifração (*unwrapping*) ocorra de forma segura dentro do enclave.

Esta abordagem garante que a chave mestra só pode ser restaurada se todas as entidades participantes cooperarem, evitando a necessidade de armazenamento permanente dentro do enclave e promovendo um modelo de confiança distribuída.

### 6.4.3 Execução de Operações

Existem **três operações** principais disponíveis: a **cifração de dados**, o **cálculo de estatísticas** sobre dados cifrados e o **envio da chave secreta**.

- No caso da cifração (comando *"encrypt"*), são esperados 6 *tokens*, que incluem o assinante da mensagem (*signer*), o ficheiro a cifrar (*filename*), o caminho para o *bucket* de destino no *Google Cloud Storage* (*gcs\_path*), um *timestamp* e a assinatura da mensagem (*signature\_b64*). A função *ecall\_process\_encrypt*, dentro do enclave, valida a assinatura e o *timestamp* para garantir a autenticidade e a frescura dos dados.

Após a validação, o ficheiro especificado é lido e o enclave gera um vetor de inicialização (IV) único, através da função *ecall\_generate\_iv*. A utilização de um IV aleatório em cada operação de cifração é fundamental para assegurar a confidencialidade dos dados cifrados: impede que padrões presentes nos dados originais sejam revelados e protege contra ataques do tipo *chosen-plaintext*. Desta forma, mesmo que os mesmos dados sejam cifrados múltiplas vezes com a mesma chave, os resultados serão sempre distintos.

Seguidamente, é utilizada a função *ecall\_encrypt\_data* para cifrar os dados com a chave mestra (*sk\_m*). O ficheiro resultante, contendo o IV, o MAC e o conteúdo cifrado, é então enviado para o GCS através da função *upload\_to\_gcs*.

- No cálculo de estatísticas, (comando *"stat"*), são esperados 8 *tokens*, que incluem, mais uma vez, o assinante da mensagem (*signer*), a coluna do *dataset* a utilizar (*column*), a operação estatística desejada (*operation*), o caminho para o *bucket* dos dados no GCS (*gcs\_path*), um *timestamp* e as assinaturas da mensagem pelas partes envolvidas (*signature1\_b64* e *signature2\_b64*). O programa faz *download* do ficheiro cifrado no GCS (com a função *download\_from\_gcs*) e lê o seu conteúdo. Neste momento, há uma validação para identificar se a operação é sobre dados categóricos, para que sejam tratados de forma diferente.

De seguida, chama a função `ecall_process_stats` dentro do enclave, que verifica a assinatura da entidade que enviou a mensagem (`signature1_b64`) e confere a autorização concedida por outras partes (`signature2_b64`) e, caso se verifique, decifra o conteúdo, realiza o cálculo estatístico pedido e retorna o resultado.

- No envio da chave secreta (comando `"addkey"`), o cliente (hospital ou laboratório) codifica a sua chave privada simétrica individual (`sk_i`), usada somente neste processo de *wrap/unwrap*, em formato *base64* e envia-a para o enclave através de um canal seguro. Esta chave é utilizada para o processo de *key wrapping* ou *unwrapping* da chave mestra.

O enclave, ao receber esta chave, armazena-a temporariamente na lista de chaves de envolvimento. Se estiver a gerar uma nova chave mestra, e se o número de chaves recebidas atingir o esperado, procede ao *key wrapping* com a função `ecall_get_wrapped_master_key`. Caso contrário, se estiver em modo de recuperação, irá tentar decifrar a chave mestra armazenada com as chaves recebidas, através da função `ecall_unwrap_master_key`.

Para além disso, o código é constituído por funções auxiliares como `base64_decode` para decodificar assinaturas/chaves em base64, `read_file` e `write_file` para I/O de ficheiros binários e `is_remote_newer` que permite verificar se a versão de um ficheiro armazenado remotamente (no *Google Cloud Storage*) é mais recente do que a versão local. Caso a versão remota seja mais recente, é iniciado automaticamente um *download* para garantir que a operação seja realizada sobre os dados mais atualizados. Caso contrário, mantém-se a utilização da versão local existente.

## 6.5 Enclave/

De seguida, será abordado o conteúdo do diretório `Enclave/`, o qual contém os ficheiros responsáveis pela definição e **implementação das funcionalidades** que serão executadas no interior do enclave.

O ficheiro `Enclave.cpp` define todas as **funções *ECALL*** referidas em cima, incluindo as que calculam todas as operações estatísticas suportadas. Adicionalmente, encontram-se *hardcoded*, no enclave, as **chaves públicas de cada entidade**, utilizadas para verificar as assinaturas das mensagens recebidas pela aplicação. Além das funções *ECALL*, o ficheiro `Enclave.cpp` define também diversas **funções auxiliares internas** que não são acessíveis diretamente a partir do exterior do enclave, mas que são fundamentais para o funcionamento seguro da aplicação.

Entre estas funções destacam-se:

- `ecc_verify` — responsável por validar assinaturas digitais usando as chaves públicas *hard-coded* das entidades autorizadas.
- `recursive_wrap` — aplica o mecanismo de *key wrapping* recursivo, cifrando a chave mestra com múltiplas chaves privadas fornecidas por entidades externas.
- `parse_csv` — faz o processamento e parsing de dados CSV decifrados, estruturando os dados para posterior análise estatística.

Estas funções são chamadas internamente por outras *ECALLs*, como `ecall_process_stats`, para garantir modularidade, reutilização de código e uma clara separação entre as tarefas de segurança, *parsing* e validação de dados.

O ficheiro `Enclave.edl` permite a **comunicação** entre a **aplicação** e o **enclave**, declarando quais são as funções que podem ser chamadas pela aplicação (*ECALLs*) e as que o enclave pode chamar na aplicação (*OCALLs*).

O ficheiro `Enclave.lds` é um **linker script** que controla quais são os símbolos que devem ser exportados pelo enclave. Nele são declarados explicitamente os símbolos essenciais ao funcionamento do *SGX*, enquanto todos os restantes símbolos são definidos como locais, evitando exposições desnecessárias.

Por fim, o ficheiro `Enclave.config.xml` define os parâmetros de configuração utilizados no processo de **assinatura do enclave**. Neste ficheiro são especificados aspetos como o tamanho máximo da pilha (`StackMaxSize`), tamanho do *heap* (`HeapMaxSize`), número de *TCS* (*Thread Control Structures* - `TCSNum`) e outras opções como a ativação ou desativação do modo de depuração.

## 6.6 Comunicação entre o Cliente, App e o Enclave

A comunicação entre os clientes (por exemplo, hospital ou laboratório) e o enclave SGX ocorre de forma indireta, mediada pela aplicação principal (`App.cpp`) que corre fora do enclave, num ambiente não confiável. Esta interação é feita através de uma ligação **SSH** segura, estabelecida pelo cliente para o servidor onde reside o enclave.

Através dessa ligação SSH, o cliente remoto interage com dois **pipes** nomeados no sistema de ficheiros Linux:

- `/tmp/sgx_pipe` — usado para enviar comandos à aplicação.
- `/tmp/sgx_response` — usado para receber as respostas da aplicação.

### Caminho da Comunicação

1. O cliente estabelece uma ligação SSH e escreve comandos no **pipe** `/tmp/sgx_pipe`, usando um formato estruturado com campos separados por `|`, contendo dados como o tipo de operação (`encrypt`, `stat`, `addkey`), o assinante, dados codificados, assinaturas e *timestamps*.
2. O conteúdo recebido no *pipe* é processado pelo ficheiro `App.cpp`, que valida os parâmetros e decide que função *ECALL* chamar no enclave, de acordo com a operação pretendida.
3. Dentro do enclave, a lógica associada à operação é executada. Por exemplo:
  - `ecall_process_encrypt` — valida assinaturas e autoriza a cifração.
  - `ecall_process_stats` — valida assinaturas duplas e executa estatísticas sobre dados cifrados.
  - `ecall_add_wrapping_key` — adiciona uma chave individual ao processo de *key wrapping*.
4. Sempre que o enclave precisa de informações do exterior (por exemplo, tempo atual), utiliza funções *OCALL*, como `ocall_get_time`, que são implementadas no lado não-confiável.
5. A resposta da operação é escrita no **pipe de resposta** (`/tmp/sgx_response`) e pode ser lida pelo cliente.

### Segurança da Comunicação

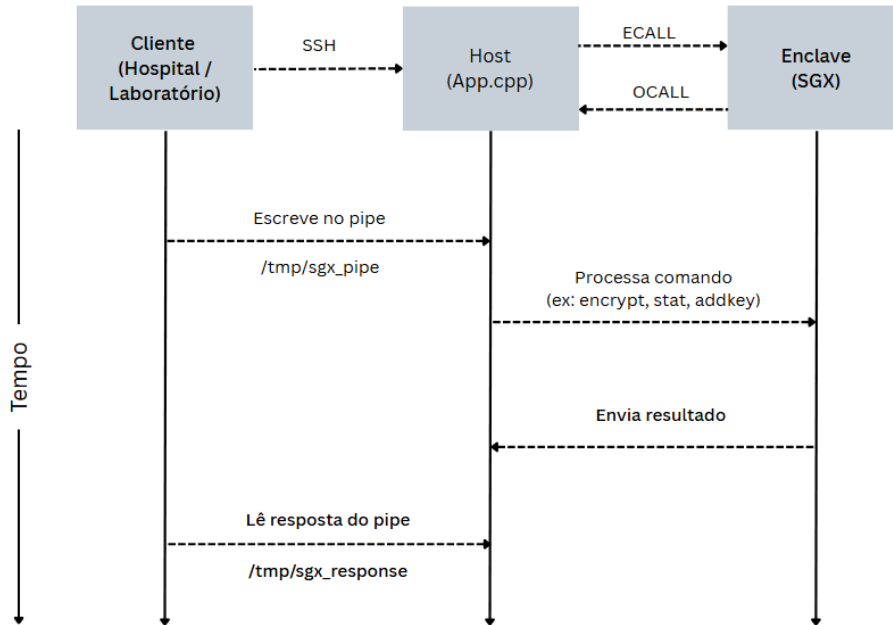
Embora o canal entre o cliente e a aplicação *SGX (pipe)* não seja seguro por si só, a segurança é garantida por:

- **Validação de Assinaturas Digitais** — Todas as mensagens sensíveis recebidas são assinadas com ECDSA e verificadas dentro do enclave com chaves públicas *hardcoded*.
- **Verificação de *Timestamps*** — Impede ataques de *replay*, rejeitando mensagens fora da janela temporal definida (5 minutos).
- **Isolamento do Enclave** — Toda a lógica crítica é executada dentro do enclave, isolada do sistema operativo e de possíveis atacantes.



Este modelo assegura que, mesmo num ambiente *host* potencialmente comprometido, os dados e operações mantêm-se protegidos pela raiz de confiança proporcionada pelo *SGX*.

A Fig. 7 ilustra o fluxo de comunicação entre o cliente (hospital ou laboratório), a aplicação não-confiável (*App.cpp*) e o enclave *SGX*.



**Figura 7:** Fluxo de comunicação entre o cliente, aplicação não-confiável (*host*) e o enclave *SGX* ao longo do tempo.

## 6.7 Ficheiros Auxiliares

Para além dos ficheiros referidos anteriormente, o projeto inclui ainda um conjunto de ficheiros auxiliares essenciais para a **compilação** correta, **assinatura** e **configuração** do enclave.

O ficheiro `Makefile` é responsável por **automatizar todo o processo de construção da aplicação e do enclave**. Ele deteta automaticamente a arquitetura (x86 ou x64) e define *flags* de compilação conforme o modo escolhido, HW (*hardware*) ou SIM (simulador). Para além disso, este ficheiro define as regras de compilação dos componentes da aplicação não confiável (*App.cpp*) e do enclave (*Enclave.cpp*), bem como a geração dos ficheiros de suporte (via *sgx\_edger8r*) e a assinatura final do enclave através da ferramenta *sgx\_sign*.

Fora do projeto, encontra-se o ficheiro `client.cpp` que implementa um cliente na **linha de comandos** que interage com a aplicação *SGX*. Inicialmente, o programa carrega variáveis de ambiente a partir de um ficheiro `.env`, o qual contém informações como **nome de utilizador SSH**, *host*, caminho da **chave SSH** e o **bucket de destino** no GCS. A comunicação do cliente com o enclave dá-se **via SSH**. O cliente carrega a sua chave privada a partir de um ficheiro binário guardado localmente (`ecc-<id>-privkey.bin`), com id correspondente ao "hospital" ou "lab". Essa chave é utilizada para gerar uma estrutura `EVP_PKEY` com o suporte da *OpenSSL*, baseada na **curva NIST P-256**. A partir dela, o cliente pode **gerar assinaturas digitais ECDSA** de mensagens que irá enviar ao enclave. Por fim, o menu do cliente *SGX* permite ao utilizador interagir com um enclave seguro através das três funcionalidades principais descritas na Sec.6.4.

Ao mesmo nível, existe o ficheiro `sign_message.cpp` que permite **assinar digitalmente uma mensagem** utilizando a chave privada de uma entidade. Ao executar o binário com dois argumentos (o caminho para o ficheiro da chave privada binária e a mensagem a assinar), o programa carrega a chave, assina a mensagem, e imprime a assinatura resultante.

Existe também o ficheiro `Enclave.private.pem` que corresponde à **chave privada** utilizada

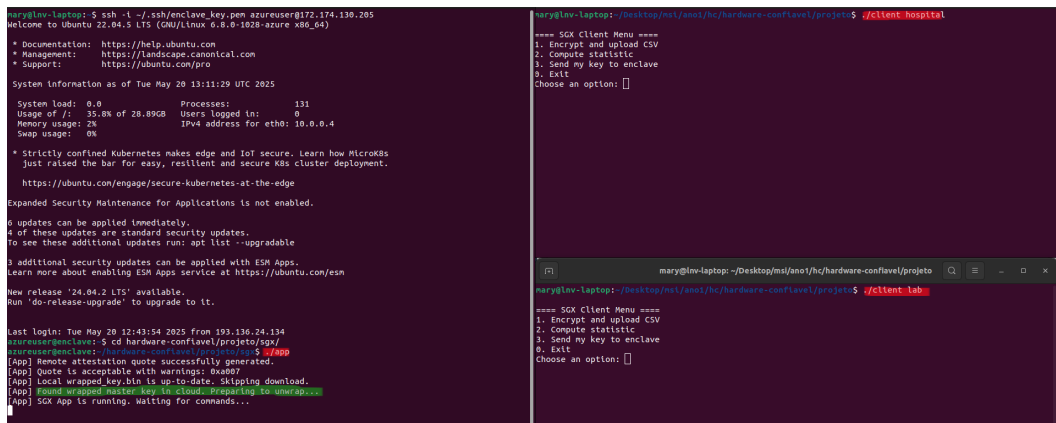
no processo de **assinatura do enclave**. Este ficheiro é essencial para garantir a integridade e autenticidade do binário final gerado (`enclave.signed.so`), permitindo que o *SGX* reconheça o enclave como um componente legítimo e confiável.

Por fim, encontra-se também o ficheiro `gen_key.py` que foi utilizado para **gerar dois pares de chaves** privadas ECC (SECP256R1) de 32 *bytes* e as chaves públicas correspondentes, utilizados pelas entidades.

## 7 Fluxo de Execução

Nesta secção, apresenta-se o fluxo de trabalho do programa implementado.

Na Fig.8, está representado o *layout* da aplicação e do menu do cliente. Inicialmente, a aplicação sugere que foi encontrada uma chave mestra (*wrapped*) na *cloud*, ou seja, está preparada para ser *unwrapped*.



```
mary@lin-laptop:~$ ssh -l -i ~/.ssh/enclave_key.pem azureuser@172.174.130.205
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1028-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System Information as of Tue May 20 13:11:29 UTC 2025

System load:  0.0          Processes:    131
Usage of /:   35.8% of 28.89GB    Users logged in:  0
Memory usage: 2%            IPv4 address for eth0: 10.0.0.4
Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.
   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

6 updates can be applied immediately.
4 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

3 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

New release '24.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

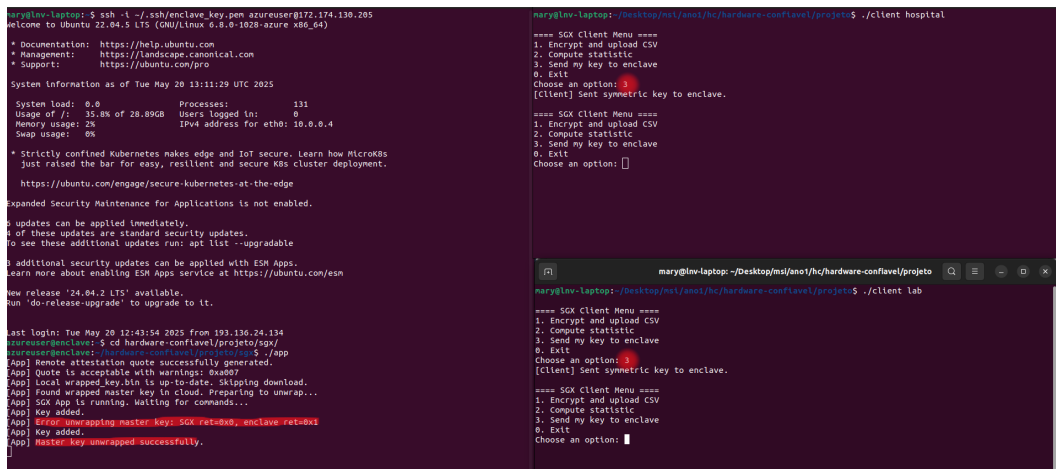
Last login: Tue May 20 12:43:54 2025 from 193.136.24.134
azureuser@linclave:~$ cd hardware-conf/lvel/projeto/sgx/
azureuser@linclave:~/hardware-conf/lvel/projeto/sgx$ ./app
[App] Remote attestation quote successfully generated.
[App] Quote is acceptable with warnings: none?
[App] Local wrapped key.bin is up-to-date. Skipping download.
[App] Found wrapped master key in cloud. Preparing to unwrap...
[App] SGX App is running. Waiting for commands...

mary@lin-laptop:~/Desktop/msi/ano1/hc/hardware-conf/lvel/projeto$ ./client-hospital

==== SGX Client Menu ====
1. Encrypt and upload CSV
2. Compute statistic
3. Send my key to enclave
0. Exit
Choose an option: 3
```

Figura 8: *Layout* do menu do cliente.

Para que a chave mestra seja devidamente obtida pelo enclave, cada cliente deve enviar a sua chave privada simétrica. Como se observa na Fig.9, o hospital envia primeiro a chave, no entanto, não é possível obter a chave mestra, pois a outra entidade (laboratório) ainda não partilhou a sua. Assim que o laboratório envia a sua chave secreta, o enclave consegue dar *unwrap* à chave mestra.



```
mary@lin-laptop:~$ ssh -l -i ~/.ssh/enclave_key.pem azureuser@172.174.130.205
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1028-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System Information as of Tue May 20 13:11:29 UTC 2025

System load:  0.0          Processes:    131
Usage of /:   35.8% of 28.89GB    Users logged in:  0
Memory usage: 2%            IPv4 address for eth0: 10.0.0.4
Swap usage:   0%

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.
   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

6 updates can be applied immediately.
4 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

3 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

New release '24.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Tue May 20 12:43:54 2025 from 193.136.24.134
azureuser@linclave:~$ cd hardware-conf/lvel/projeto/sgx/
azureuser@linclave:~/hardware-conf/lvel/projeto/sgx$ ./app
[App] Remote attestation quote successfully generated.
[App] Quote is acceptable with warnings: none?
[App] Local wrapped key.bin is up-to-date. Skipping download.
[App] Found wrapped master key in cloud. Preparing to unwrap...
[App] SGX App is running. Waiting for commands...
[App] Key added.
[App] Error unwrapping master key: SGX ret=0xb, enclave ret=0x1
[App] Key added.
[App] Master key unwrapped successfully.

mary@lin-laptop:~/Desktop/msi/ano1/hc/hardware-conf/lvel/projeto$ ./client-hospital

==== SGX Client Menu ====
1. Encrypt and upload CSV
2. Compute statistic
3. Send my key to enclave
0. Exit
Choose an option: 3

[Client] Sent symmetric key to enclave.

==== SGX Client Menu ====
1. Encrypt and upload CSV
2. Compute statistic
3. Send my key to enclave
0. Exit
Choose an option: 3

[Client] Sent symmetric key to enclave.
```

Figura 9: Envio da chave de cada cliente.

O próximo passo seria cada entidade armazenar os seus *datasets* na *cloud*. Para isso, como se observa na Fig.10, enviam-se os ficheiros `.csv` no seu formato base64 e o enclave procede ao seu *upload*.



## 8 Validação dos Requisitos

A solução desenvolvida foi avaliada em função dos requisitos funcionais e de segurança definidos previamente. Com base nessa análise, é possível afirmar que todos os requisitos foram cumpridos, incluindo melhorias e extensões que não estavam previstas na fase inicial.

No que diz respeito ao **armazenamento seguro de dados sensíveis**, o sistema garante que toda a informação é cifrada e processada exclusivamente dentro de enclaves *SGX*, assegurando que nunca é exposta ao sistema operativo ou a terceiros. Os cálculos estatísticos foram implementados para suportar múltiplas operações matemáticas, tornando a **solução mais escalável e completa**.

A **autenticação** e a **autorização das entidades** foi assegurada através de chaves públicas *hardcoded* e chaves privadas, que permitem validar que apenas participantes autorizados contribuem e acedem aos dados. O sistema, ainda, requer a assinatura de todas as entidades envolvidas antes da execução de qualquer operação, garantindo controlo total e prevenção contra acessos isolados ou não autorizados.

A **atestação remota** foi implementada com o apoio de um servidor externo, garantindo que o enclave é legítimo e executado num ambiente seguro. A comunicação entre entidades e enclave é protegida por canais seguros, reforçando a confidencialidade e autenticidade dos dados trocados.

Um **requisito relevante adicional** foi a introdução de uma solução baseada em **key wrapping**, que não constava dos requisitos iniciais. Esta funcionalidade foi incorporada para reforçar a resiliência do sistema em cenários onde podem falhar o enclave ou o *hardware*, o que melhorou significativamente a garantia de continuidade operacional com elevados níveis de segurança.

Por fim, a **escalabilidade** do sistema foi tida em consideração desde o início do projeto. A arquitetura foi desenhada para permitir facilmente a integração de novas entidades e o aumento do volume de dados sem comprometer o desempenho ou a segurança, validando assim o cumprimento deste requisito.

Em conclusão, todos os requisitos definidos foram concretizados com sucesso e a solução final inclui ainda mecanismos adicionais que reforçam a robustez, a flexibilidade e a segurança da implementação.

## 9 Conclusão

Este projeto teve como objetivo o desenvolvimento de uma solução segura e confiável para armazenamento e processamento colaborativo de dados sensíveis em ambientes distribuídos, com particular ênfase na área da saúde. Para tal, recorreu-se ao uso de *hardware* confiável, concretamente o *Intel SGX*, que permitiu assegurar a confidencialidade, integridade e autenticidade dos dados e das operações realizadas.

Ao longo do trabalho, foi implementado um sistema robusto que permite que diferentes entidades, como, neste caso, hospitais e laboratórios, armazenem e processem dados médicos sem nunca exporem diretamente os registos individuais. Através do uso de enclaves *SGX*, os dados são processados em ambiente isolado, garantindo que apenas resultados agregados, como estatísticas, são revelados.

A solução implementada suporta a cifração de ficheiros, a execução segura de cálculos estatísticos e o uso de múltiplas chaves para proteger a chave mestra através de um mecanismo de *key wrapping*. A atestação remota com o *Azure Attestation Service* garante que apenas enclaves legítimos participem no sistema, e a autenticação baseada em assinaturas digitais evita que entidades não autorizadas acessem aos dados.

Para além disso, a aplicação suporta recuperação segura após falhas, persistência de chaves na *cloud*, e comunicação segura entre os clientes e o enclave, demonstrando um forte alinhamento com os requisitos funcionais e de segurança definidos.

A implementação e validação da solução demonstraram a viabilidade do uso de enclaves *SGX* em contextos colaborativos com requisitos elevados de privacidade e confiança. Este trabalho abre caminho para a criação de sistemas distribuídos mais seguros, aplicáveis não só à saúde, mas também a outras áreas sensíveis como finanças, defesa ou investigação científica.

Como trabalho futuro, poderá explorar-se a migração para outras plataformas de execução confiável mais recentes, como o *Intel TDX (Trust Domain Extensions)*, que oferece benefícios adicionais de isolamento ao nível da máquina virtual, mantendo a compatibilidade com muitos dos princípios aplicados neste projeto.

## Referências

- [1] Fatemeh Nargesian, Erkang Zhu, Renée J. Miller, Ken Q. Pu, and Patricia C. Arocena. Data lake management. *Proceedings of the VLDB Endowment*, 12:1986–1989, 8 2019.
- [2] Daniel Ehnes. The magic of intel’s sgx. a tutorial on programming a secure enclave, 2018.
- [3] Ferdinand Brasser, Urs Müller, Alexandra Dmitrienko, Kari Kostiainen, Srdjan Capkun, and Ahmad Reza Sadeghi. Software grand exposure: Sgx cache attacks are practical. *11th USENIX Workshop on Offensive Technologies, WOOT 2017, co-located with USENIX Security 2017*, 2 2017.
- [4] Michael Schwarz, Samuel Weiser, and Daniel Gruss. Practical enclave malware with intel sgx. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11543 LNCS:177–196, 2 2019.
- [5] JP Aumasson and Luis Merino Kudelski Security. Sgx secure enclaves in practice: Security and crypto review. 2016.
- [6] Jo Van Bulck, Marina Minkin, Ofir Weisse, Daniel Genkin, Baris Kasikci, Frank Piessens, Mark Silberstein, Thomas F Wenisch, Yuval Yarom, Raoul Strackx, and Ku Leuven. Foreshadow: Extracting the keys to the intel sgx kingdom with transient out-of-order execution.
- [7] Intel Corporation. Intel sgx sdk for linux. <https://github.com/intel/linux-sgx#build-the-intelr-sgx-sdk-and-intelr-sgx-sdk-installer>, 2024.
- [8] Intel Corporation. Install the intel® sgx sdk. <https://github.com/intel/linux-sgx#install-the-intelr-sgx-sdk-1>, 2024.
- [9] Microsoft Learn. Fluxo de trabalho de validação de enclave intel® sgx. <https://learn.microsoft.com/pt-br/azure/attestation/workflow>, 2025.