PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

Safe Vault, sistema para proteção de pastas e arquivos

Thiago Lages de Alencar

PROJETO FINAL DE GRADUAÇÃO

CENTRO TÉCNICO CIENTÍFICO - CTC

DEPARTAMENTO DE INFORMÁTICA

Curso de Graduação em Engenharia da Computação

Rio de Janeiro, mês de ano

Thiago Lages de Alencar

Safe Vault, sistema para proteção de pastas e arquivos

Relatório de Projeto Final 1, apresentado para **Bacharelado em Ciência da Computação** da PUC-Rio como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Anderson Oliveira da Silva.

Rio de Janeiro

Setembro de 2018.

1. Introdução

Serviços de armazenamento como *Google Drive*[12], *Dropbox*[13] e *Onedrive*[14], permitem ao usuário o conforto de acesso remoto às suas informações, sem se preocupar com a manutenção recursos físicos. Porém existem usuários que se recusam a armazenar informações sensíveis nesses provedores, pois consideram a proteção insuficiente. Portanto observa-se que, a confidencialidade é uma grande preocupação quando se fala de armazenamento na internet[1][2].

O armazenamento na nuvem pode aumentar os riscos de comprometimento das informações de diversas maneiras, como por exemplo: vazamento de informação, furto de conta, fraudes, uso indevido, etc[2][3]. Esses riscos podem ser reduzidos se os dados forem protegidos utilizando protocolos de segurança. Tais protocolos dão ao usuário controle sobre quem pode acessar os dados, e isso garante a proteção das informações do usuário.

Diante desse cenário, o sistema visa a funcionar como uma ferramenta para o usuário armazenar de forma segura, seus arquivos em serviços de armazenamento ou no próprio dispositivo que está utilizando. A ideia é que o sistema seja apenas uma ponte para o usuário garantir a segurança de seus dados nos serviços de armazenamento.

Com o intuito de respeitar a privacidade do usuário, nenhuma fiscalização é realizada nos dados durante o armazenamento no serviço. Durante a recuperação dos dados, a ferramenta utilizará dos conhecimentos de segurança para identificar qualquer invasão na privacidade dos dados.

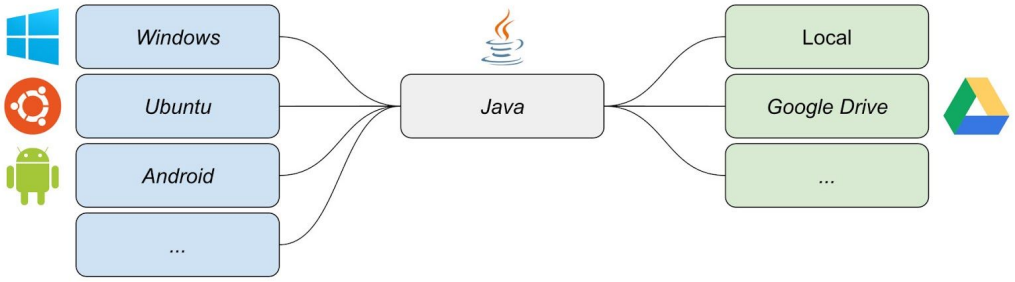
1.1 Sobre o ambiente computacional

A intenção do *Safe Vault* durante esse projeto é ser capaz de atingir *Windows*, *Ubuntu* e *Android* mas mostrando que tem potencial de atingir outros utilizando a estrutura base dele.

Para alcançar o máximo possível de plataformas foi escolhido desenvolver na linguagem de programação *Java*, versão *SE* 8 pois era a mais recente quando o

projeto se inicializou. Junto de provedores criptográficos disponíveis na *Java Cryptography Architecture* (JCA)[9].

Com finalidade de se comunicar diretamente com um serviços da nuvem, é necessário utilizar a API daquele serviço. Ao fim desse projeto será apresentado o sistema se comunicado com a API do *Google Drive*[10][11] como exemplo de comunicação com a nuvem.



O ambiente de desenvolvimento será o *Eclipse[17]*, versão *Eclipse Photon 27 June 2018*.

1.2 Sobre a adequação do trabalho como Projeto Final

Durante esse projeto foram utilizados diversos conceitos: modelagem de dados utilizando a linguagem *UML*[18], programação orientada a objetos com *Java*, designs patterns, conceitos de segurança da informação, etc.

2. Situação Atual

2.1 Descrição e avaliação de tecnologias e sistemas existentes

Grande parte dos serviços de armazenamento oferecem o armazenamento não criptografado, o que implica no usuário confiar que suas informações vão ter a confidencialidade respeitada. Não existe certeza dessa confidencialidade, pois se alguém obter a credencial de acesso do usuário, toda a informação no provedor do serviço vai estar comprometida.

Provedores e empresas de terceiros que garantem essa segurança, costumam armazenar a chave que protege as informações do usuário[6]. Isso delega o controle do usuário para os provedores e empresas, não resolvendo o problema de confidencialidade[1].

O *Safe Vault* dá ao usuário a chave capaz de proteger a informação, assim garantindo a confidencialidade das informações. Uma desvantagem é que o serviço de armazenamento não poderá fornecer recursos como: busca, modificação, inserção e remoção sobre as informações seguras[1]. Essas facilidades devem ser implementadas pela ferramenta que controla a proteção dos dados.

Para prover a segurança das informações do usuário, é utilizado algoritmos de criptografia. A informação é protegida por cifragem de forma que mesmo se uma pessoa maliciosa obtiver os dados cifrados, será muito difícil decifrar sem ter a chave de criptografia[2]. A segurança da informação é garantida pelos seguintes princípios básicos: *Integridade*, *Autenticidade*, *Confidencialidade e Disponibilidade*.

Integridade envolve proteger a informação de alterações sem permissão explícita do proprietário[2]. O estado da informação quando resgatada, deve ser igual ao estado quando gerada[5]. Algoritmos de resumo de mensagem (digest) são utilizados para verificar a integridade das informações[2].

Autenticidade consiste em identificar corretamente um usuário ou computador[2]. É uma maneira de medir o grau de confiança de que a origem da informação é a mesma que ela alega ser[7]. É necessário verificar a autenticidade

em todo processo de identificação e transferência de informação. Através da assinatura digital pode-se validar a autenticidade das informações[2].

Confidencialidade garante que apenas as pessoas às quais a informação é destinada conseguem compreendê-la. Utilizando algoritmos de criptografia, é possível mascarar a informação original, através de cifragem. A segurança depende da garantia de segredo da chave utilizada no algoritmo.

Disponibilidade visa a manter a informação disponível para uso sempre que houver necessidade dela. O serviço de armazenamento compartilha desse princípio básico.

Ferramentas como *Cryptomator*[15] e *Boxcryptor*[16] garantem a segurança dos dados na máquina local e aproveitam que serviços de armazenamentos como *Google Drive*, *Dropbox* e *OneDrive* oferecem ferramentas de sincronização para transmitir esses dados seguros para a nuvem. Essa tática envolve o usuário possuir esses serviços de sincronização para transmitir o arquivo para nuvem, ou seja, para cada serviço de armazenamento que o usuário desejar utilizar, mais um software de sincronização na máquina dele.

Safe Vault, o sistema desse projeto, tem como foco fornecer um software/aplicação capaz de interagir com os serviços de armazenamento diretamente, dessa maneira diminuindo a necessidade de outros softwares e concentrando todo o foco de armazenamento em apenas uma ferramenta.

3. Objetivos do trabalho

O trabalho visa a implementar uma ferramenta que permita garantir a segurança da informação do usuário em serviços de armazenamento, de forma a garantir os princípios de integridade, autenticidade e confidencialidade. A ferramenta é separada em três partes: *Engine*, *Plugin* e *Interface*.

A *Engine* é responsável por aplicar todos os protocolos de segurança nas informações do usuário, receber requisições das interfaces e fazer requisições aos plugins. A *Engine* deve ser única para todas as plataformas e deve interagir com as interfaces de usuário dos diferentes sistemas operacionais e com os serviços de armazenamento (através de *Plugin*). Para garantir a segurança da informação, a Engine fará uso de algoritmos de hash (digest), padrões de assinatura digital e algoritmos criptográficos simétricos e assimétricos.

O *Plugin* interage com o serviço de armazenamento para realizar tarefas requisitadas pela *Engine*. Cada *Plugin* fica responsável por se comunicar com um dos provedores de armazenamento. A *Engine* faz requisições padronizadas independente do *Plugin* para executar operações básicas no serviço de armazenamento, como: criar arquivo, deletar arquivo, ler arquivo, escrever arquivo, criar pasta, listar pasta e deletar pasta. Existe a implementação de *Plugin* padrão para interagir com o sistema de arquivos local na máquina do usuário.

A *Interface* é o meio de comunicação do usuário com a *Engine*, e vice-versa. Através dela o usuário descobre o estado das informações na nuvem e solicita ações à *Engine*. Todas as interfaces de usuário, independentemente de sistema operacional, solicitam as mesmas ações de forma padronizada para a *Engine*.

4. Atividades Realizadas

4.1 Estudos preliminares

- Conhecimento sobre a linguagem de programação *Java*.
- Conhecimento sobre IDE *Eclipse.*
- Conhecimento desatualizada da plataforma *Android.*

4.2 Estudos conceituais e de tecnologia

- Estudo sobre a API do *Google Drive.*
- Estudo sobre conceitos de segurança da informação: *Integridade*, *Autenticidade* e *Confidencialidade*.
- Estudo sobre técnicas e ferramentas para garantir a segurança da informação: *Resumo de Mensagem*, *Assinatura Digital*, *Envelope Digital*, *Criptografia Simétrica* e *Assimétrica*.

4.3 Testes e Protótipos para aprendizado e demonstração

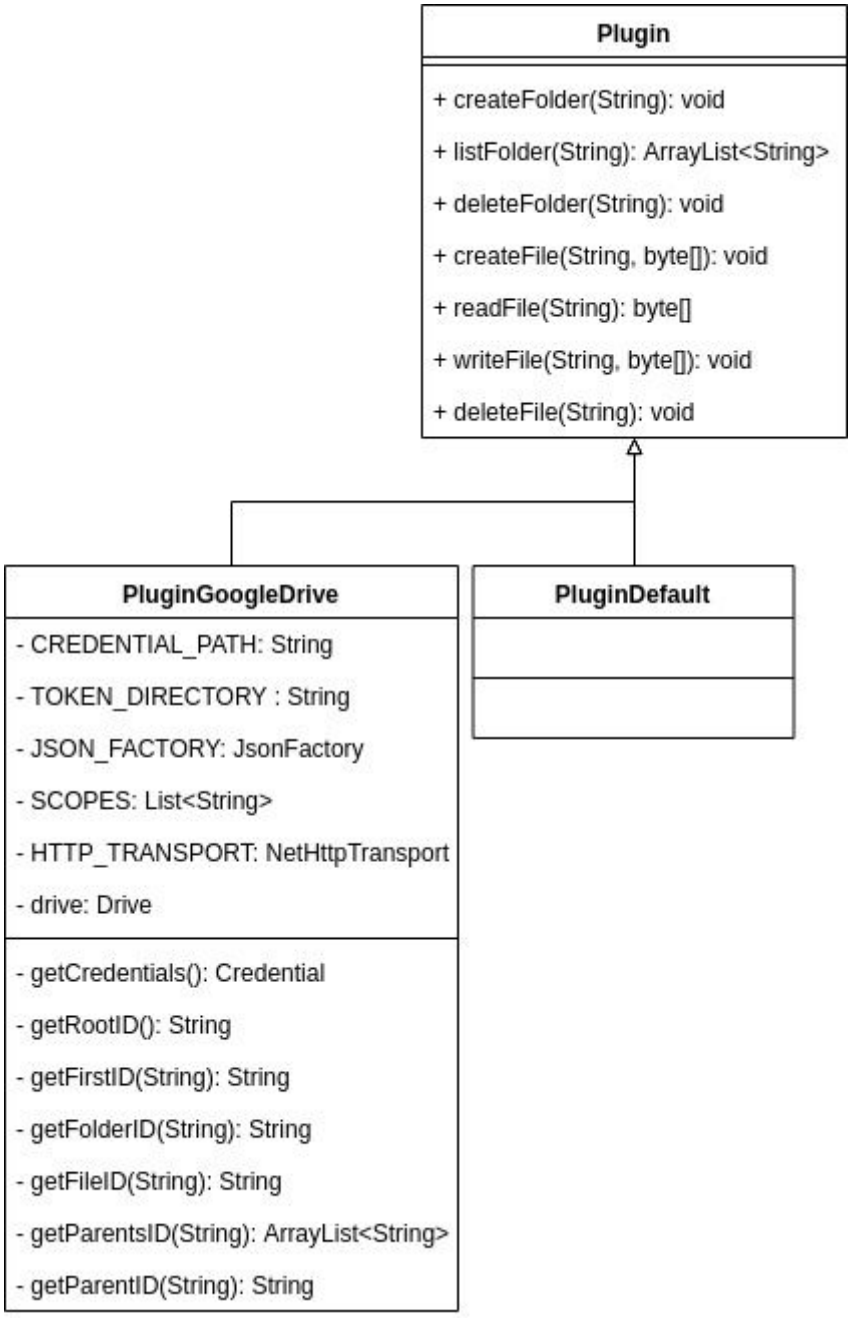
- Teste do algoritmo de *Resumo de Mensagem* em uma String.
- Testes dos algoritmos de *Criptografia Simétrico* e *Assimétrico* para cifrar e decifrar uma String.
- Criação de uma *Assinatura Digital* sobre uma string, utilizando a biblioteca *Java* de assinatura (java.security.Signature).
- Criação de uma *Assinatura Digital* sobre uma string, sem utilizar a biblioteca *Java* de assinatura.
- Construir um programa que lê um arquivo com uma lista de *Digest* de arquivos e confere se os *Digest* estão correto com os arquivos originais.
- Criar um programa para fazer download de arquivos no *Google Drive*.
- Criar um programa para fazer upload de arquivos no *Google Drive*.
- Upload de arquivo cifrado e download do mesmo, para verificar se está recuperando com sucesso.
- Criar um protótipo de Plugin para armazenar e recuperar localmente os arquivos.

- Criar um protótipo de Plugin para armazenar e recuperar do *Google Drive* os arquivos.
- Protocolo de segurança dos arquivos.

4.4 Plugin Google Drive

Toda requisição para a API do *Google Drive* envolve enviar um token de autorização que identifica o usuário, para obter esse token foi preciso requisitar uma credencial *OAuth 2.0* no *Google API Console*. Com essa credencial o usuário consegue se identificar com a conta do Google e conseguir um token temporário para utilizar a API [19]. Durante o início do projeto a aplicação requisita permissão completa ao Google Drive do usuário, deve-se ir diminuindo a quantidade de permissões conforme se mostre adequado no projeto.

Com o acesso a API foi possível executar testes básicos (upload, download, acessar informações, queries, etc) e criar dois protótipos: Plugin do Google Drive e Plugin Default


Plugin
+ createFolder(String): void
+ listFolder(String): ArrayList<String>
+ deleteFolder(String): void
+ createFile(String, byte[]): void
+ readFile(String): byte[]
+ writeFile(String, byte[]): void
+ deleteFile(String): void
PluginGoogleDrive
- CREDENTIAL_PATH: String
- TOKEN_DIRECTORY : String
- JSON_FACTORY: JsonFactory
- SCOPES: List<String>
- HTTP_TRANSPORT: NetHttpTransport
- drive: Drive
- getCredentials(): Credential
- getRootID(): String
- getFirstID(String): String
- getFolderID(String): String
- getFileID(String): String
- getParentsID(String): ArrayList<String>
- getParentID(String): String
PluginDefault

4.5 Protocolo de segurança dos arquivos

Procedimento que a *Engine* deve adotar sempre que o usuário desejar fazer upload de dados:

- Gerar chave simétrica.
- Utilizar a chave simétrica para cifrar os dados do usuário, com isso criando os dados cifrados.
- Utilizar a chave pública do usuário para cifrar a chave simétrica, com isso criando o envelope digital.
- Aplicar o algoritmo de hash nos dados originais do usuário para gerar o digest.
- Utilizar a chave privada do usuário para cifrar o digest, com isso criando a assinatura digital.
- Criar um container, formado por: assinatura digital, envelope digital e dados cifrados.

Procedimento que a *Engine* deve adotar sempre que o usuário desejar fazer download de dados (oposto do procedimento de upload):

- Separar o container em 3 partes: assinatura digital, envelope digital e dados cifrados.
- Utilizar a chave pública do usuário para decifrar a assinatura digital, com isso obtendo o digest.
- Utilizar a chave privada do usuário para decifrar o envelope digital, com isso obtendo a chave simétrica.
- Utilizar a chave simétrica para decifrar os dados cifrados, com isso obtendo os dados originais.
- Gerar o digest dos dados originais e conferir se igual ao digest obtido pela assinatura digital.

5. Revisão do Plano de Ação

Para fornecer a segurança da informação aos usuários, foi necessário estudar os conceitos de integridade, autenticidade e confidencialidade[2], e as técnicas que garantem a funcionalidade desses conceitos, como: *Resumo de Mensagem*, *Assinatura Digital*, *Criptografia Simétrica* e *Assimétrica*[2]. A *Engine* ainda será construída com base na *Java Cryptography Architecture* (JCA)[9]. Estudo da API para o *Plugin* do *Google Drive* foi concluído e um protótipo foi criado e testado.[10][11]. Para demonstrar como o sistema pode ser adaptado a várias plataformas, foi incluso a *Interface Android*.

6. Cronogramas

O processo de criação do software é incremental e engloba as seguintes atividades:

1. Estudo dos conceitos: *Integridade*, *Autenticidade*, *Confidencialidade*, *Resumo de Mensagem*, *Assinatura Digital*, *Envelope Digital*, *Criptografia Simétrica* e *Assimétrica*. (realizado)
2. Teste de algoritmos Criptográficos Simétricos e Assimétricos. Cifrar e decifrar informações. (realizado)
3. Teste de *Resumo de Mensagem* e *Assinatura Digital*. (realizado)
4. Teste de armazenamento e recuperação de informação cifrada na máquina do usuário, armazenar localmente. (realizado)
5. Estudo da *Google Drive* API. (realizado)
6. Teste armazenamento e recuperação de informação cifrada no serviço *Google Drive*. (realizado)
7. Teste criptografar e armazenar no *Google Drive*.
8. Padrões e boas práticas que vão ser utilizada no código do sistema.
9. Diagramas da *Engine*, *Plugin* e *Interface*.
10. Desenvolvimento da *Engine*.
11. Teste da interação da *Engine* com as informações.
12. Desenvolvimento do *Plugin* padrão, armazenamento local. (realizado)
13. Teste da interação da *Engine* com o *Plugin* padrão.
14. Desenvolvimento do *Plugin* do *Google Drive*. (realizado)
15. Teste da interação da *Engine* com o *Plugin* do *Google Drive*.
16. Desenvolvimento da *Interface*, *Java Graphic User Interface* (GUI).
17. Teste da *Interface Java* GUI com a *Engine*, utilizando o *Plugin* padrão.
18. Teste da *Interface* Java GUI com a *Engine*, utilizando o *Plugin* do *Google Drive*.
19. Estudo da plataforma *Android* para criação da *Interface* dele. (incluído)
20. Desenvolvimento da *Interface* para *Android*. (incluído)
21. Teste da *Interface Android* com a *Engine*, utilizando o *Plugin* padrão. (incluído)

22. Teste da *Interface Android* com a *Engine*, utilizando o *Plugin* do *Google Drive*. (incluído)

	SEMANA																	
ATIVIDADE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	✔	✔	✔	✔														
2		✔	✔	✔														x
3		✔	✔	✔														x
4				✔	✔													x
5					✔	✔	✔											
6						✔	✔											
7							x											
8								x	x	x		x		x		x	x	
9									x	x		x		x		x		
10										x	x							
11											x							x
12												✔	✔					
13													x					x
14														✔	✔			
15															x			x
16																x	x	
17																	x	x
18																		x
19							x	x										
20														x	x	x	x	
21																	x	x
22																		x

7. Referências bibliográficas

[1] YUSUF HAIDER M, SIVA SELVAN. **Confidentiality Issues in Cloud Computing and Countermeasures: A Survey**. Departamente of Computer Science and Engineering. Manipal Institute of Technology (Manipal University). Disponível em: <https://www.researchgate.net/publication/305689086_Confidentiality_Issues_in_Cloud_Computing_and_Countermeasures_A_Survey>

[2] ALFREDO KURY ABÍLIO, ANDRÉ CONCEIÇÃO DA GRAÇA, CRISTIANO PEDRO DA SILVA, MAURO SÉRGIO DOS SANTOS AMORIM. **Comunicação Segura na Internet: Métodos, Infra-estrutura de Chaves Públicas e Padrões**. Trabalho de Conclusão de Curso.

[3] PRADEEP KUMAR TIWARI. **Cloud Computing Security Issues, Challenges and Solution**. International Journal of Emerging Technology and Advanced Engineering, 2012. Disponível em: <https://www.researchgate.net/publication/271522943/download>

[4] MAHESH U. SHANKARWAR, AMBIKA V. PAWAR. **Security and Privacy in Cloud Computing: A Survey**. CSE Department, SIT, Symbiosis International University, 2015. Disponível em: <https://www.researchgate.net/publication/282925703/download>

[5] CAROLINA GWOZDZ POERSCH, GIULLYAN METZKER KUNTZE. **Modelo de Coleta e Análise de Evidências em Sistemas Computacionais**. Universidade do Sul de Santa Catarina, 2010. Disponível em: <https://www.riuni.unisul.br/bitstream/handle/12345/3229/100889_Carolina.pdf?sequence=1>

[6] Google Cloud. **How Google Uses Encryption to Protect Your Data**. Acessado em: 13 Set. 2018. Disponível em: <https://storage.googleapis.com/gfw-touched-accounts-pdfs/google-encryption-whitepaper-gsuite.pdf>

[7] SEAN PEISERT, ED TALBOT, TOM KROEGER. **Principles of Authentication**. California, USA, 2013. Disponível em: <https://www.nspw.org/papers/2013/nspw2013-peisert.pdf>

[8] Oracle. **The Java™ Tutorials**. Acessado em: 13 Set. 2018 <https://docs.oracle.com/javase/tutorial/index.html>

[9] Oracle. **Java Cryptography Architecture (JCA) Reference Guide**. Acessado em: 13 Set. 2018 <https://docs.oracle.com/javase/8/docs/technotes/guides/security/crypto/CryptoSpec.html>

[10] Google. **Google Drive APIs REST**. Acessado em: 13 Set. 2018 <https://developers.google.com/drive/api/v3/about-sdk>

[11] Google. **Google API Client Libraries**. Acessado em: 13 Set. 2018 <https://developers.google.com/api-client-library/>

[12] Google. **Google Drive**. Disponível em: <https://www.google.com/drive/>

[13] Dropbox Inc. **Dropbox**. Disponível em: <https://www.dropbox.com>

[14] Microsoft. **OneDrive**. Disponível em: <https://onedrive.live.com>

[15] Skymatic. **Cryptomator**. Disponível em: <https://cryptomator.org/>

[16] Boxcryptor. **Boxcryptor**. Disponível em: <https://www.boxcryptor.com>

[17] Eclipse Foundation. **Eclipse**. Disponível em: <https://www.eclipse.org/>

[18] UML. **UML**. Disponível em: <http://www.uml.org/>

[19] Google. **Using OAuth 2.0 to Acecss Google APIs**. Disponível em: <https://developers.google.com/identity/protocols/OAuth2>