Engenharia de Segurança
Projeto de Desenvolvimento 1

Melânia Pereira Duarte Oliveira Paulo Ricardo Pereira
{pg47520, pg47157, pg47554}@alunos.uminho.pt
Grupo 8

2 de maio de 2022

1 Introdução

Serve o presente documento como relatório do trabalho desenvolvido para a UC de Engenharia de Segurança do Mestrado em Engenharia Informática na área da Criptografia e Segurança da Informação.

Neste relatório é apresentado o enunciado do problema que foi proposto e todas as decisões tomadas para o desenvolvimento da sua solução. É ainda indicado o modo como o programa deve ser instalado e utilizado.

2 Enunciado

"O objetivo deste projeto é disponibilizar um serviço de cofre digital, garantindo que os documentos lá depositados só são acedidos pelos seus titulares."

O cofre digital tem as seguintes operações para o exterior:

- depositar documento (qualquer tipo de documento), sendo que o depositante tem que identificar quem lhe pode aceder. Como resultado é devolvido ao depositante o hash do documento depositado e a chave de decifra do mesmo, cifrada com a chave pública do depositante. Caso o depositante identifique que o documento só pode ser acedido por várias pessoas em conjunto (n pessoas de um conjunto de m pessoas, com n ¡= m), essa chave de decifra é partida pelas m pessoas (utilize o esquema de Shamir para partilha de segredo que iremos ver numa das próximas aulas teóricas) e fornecida a cada uma, cifrada com a chave pública de cada uma.

- fornecer documento, desde que seja pedido pelo(s) depositante(s) que lhe podem aceder, e fornecida a chave correta de decifra (já decifrada pela chave privada do depositante).

Internamente, e como atua como fiel depositário do que lhe é confiado, o cofre digital gera a hash do documento e cifra-o, guardando apenas o par (hash,

documento cifrado). A chave de cifra é única para cada documento e não é guardada, mas é devolvida ao(s) depositante(s), conforme descrito na operação de depositar. Para fornecer o documento, o cofre digital tem que receber a chave de decifra (ou as várias componentes da mesma, caso só seja possível aceder por vários depositantes em conjunto), e só devolve o documento original caso o consiga decifrar. No caso de receber várias componentes da mesma, deve existir um mecanismo que permita que os vários depositantes forneçam a sua componente da chave sem necessitarem de a mostrar aos restantes depositantes."

Em resumo, e segundo a interpretação do grupo, o pedido é o desenvolvimento de um serviço que permita o depósito e posterior levantamento (acesso) de ficheiros, recorrendo a um técnica de envelope digital onde o ficheiro deve ser cifrado segundo uma chave simétrica gerada e, essa chave, deve ser cifrada usando cifra de chave pública, com a chave pública do depositante. Além disto, deve ser, ainda, possível que o depositante decida que o ficheiro depositado possa ser acedido, no mínimo, por um subconjunto de um conjunto de pessoas autorizadas; neste caso, a chave de cifra do ficheiro deve ser partida (usando o esquema de Shamir para partilha de segredo) e cada parte deve ser cifrada com a chave pública de um dos participantes.

Ao grupo 8 foi ainda pedido que a realização deste projeto fosse feita em *Javascript*, ou outra linguagem à escolha, que não Java ou Python. O grupo **decidiu fazer a implementação em JavaScript**.

3 Bibliotecas usadas

Foram usadas apenas três bibliotecas de criptografia:

- crypto-js
- crypto
- shamirs-secret-sharing

Além destas, foram usadas ainda as bibliotecas *prompt-sync*, *fs* e *path* para ler *input* do teclado, abrir, ler e escrever ficheiros e aceder à informação de caminho de um ficheiro, respetivamente.

4 Decisões tomadas

Estando no papel de desenvolvedor de uma solução para um problema apresentado, é natural que decisões tenham que ser tomadas, algumas mais livres e outras mais restringidas por requisitos que estejam adjacentes ao enunciado do problema, de forma a se obter uma solução que seja coerente e precisa.

Começando pela funcionalidade de depositar um documento, deve ser referido que:

- para gerar a hash do ficheiro, optou-se pelo SHA256, por ser uma função de hash que oferece grande segurança;
- para a cifragem do ficheiro, foi decidido usar a cifra de blocos AES
- a chave de cifra do ficheiro é gerada então, usando o AES, com um tamanho de 128 bits
- para armazenar a hash e o ficheiro cifrado (assim como outras componentes que são necessárias), o grupo recorreu à escrita destes dados em ficheiro (cujo nome é `files.txt`)
- para o caso de o depositante indicar que o ficheiro só pode ser acedido por um número n de um conjunto de m pessoas, recorreu-se ao esquema de Shamir para partilha de segredo (como indicado no enunciado); para tal, foi usado o pacote *shamirs-secret-sharing* do Node.js, que parte um segredo num nº de partilhas indicado (neste caso, m) com um threshold indicado (neste caso, n)
- no caso de haver partilha da chave simétrica, cada uma das partes resultantes da partilha de Shamir é cifrada usando a chave pública da pessoa associada a essa parte; para isto, é pedido ao utilizador que insira o nome das pessoas que poderão aceder ao ficheiro, assim como as suas respetivas chaves públicas (é necessário que o depositante se inclua na lista das pessoas que fornecer)
- no caso de não haver partilha, a chave é cifrada apenas com a chave pública do depositante
- para a distribuição das partes por cada pessoa, e também para que o depositante receba a chave cifarda no caso de não haver partilha, o grupo decidiu criar um ficheiro para cada uma das pessoas indicadas; esse ficheiro terá como nome, o nome do ficheiro cifrado seguido do nome da pessoa; este ficheiro terá como conteúdo o hash do ficheiro, que servirá para o identificar aquando da obtenção do mesmo, e a parte (ou o seu todo) da chave cifrada

Seguindo com a funcionalidade de fornecer um documento, deve ser referido ainda:

- um ficheiro é identificado pela sua hash, logo, para obter determinado ficheiro, o utilizador deve inserir a respetiva hash
- é pedido ao utilizador que insira um conjuno de dados necessários À decifra do ficheiro, dos quais faz parte a sua parte da chave simétrica de cifra do ficheiro e, para que não haja a possibilidade de outros depositantes verem a sua parte, o grupo recorreu ao uso de uma função que "esconde"o texto que está a ser escrito na linha de comandos - `prompt.hide()` (função usada também para a introdução da palavra passe da chave privada

- as partes introduzidas são combinadas usando, como na funcionalidade de depósito, o pacote *shamirs-secret-sharing*

Para ficheiros PDF, foi necessário fazer um tratamento inicial, no depósito, e final (aquando do fornecimento do mesmo), visto que este tipo de documento possui uma codificação binária e não "textual".

O grupo decidiu desenvolver este programa com um conjunto sucessivo de menus de entrada, onde se pede que o utilizador introduza aquilo que é necessário e onde é especificado ainda a forma como deve ser introduzido, facilitando assim o uso do programa por parte dos utilizadores, mas também a parte de "parsing"dos dados de *input* pelo programa.

5 Uso do programa desenvolvido

Para a o uso do programa é necessário ter instalado na máquina o **npm**, um gestor de pacotes para o *Node.js* e o próprio **node**. Informações para a sua instalação podem ser encontradas aqui.

Com o *npm* e o *node* instalados na máquina, é necessário, então, instalar o conjunto de pacotes necessários à execução do programa, da seguinte forma:

```
npm install
```

Depois de instalados todos os pacotes necessários, o programa está pronto a ser utilizado com o seguinte comando

```
npm start index.js
```

Para a utilização do programa é necessário ter ficheiros PEM de chaves RSA pública e privada. Podem ser obtidos através dos seguintes comandos:

```
openssl genrsa -aes128 -out priv.pem 2048
openssl rsa -in priv.pem -out pub.pem -pubout -outform PEM
```

Ao iniciar o programa, será apresentado um menu, deve ser escolhida uma opção e, de seguida, introduzir todos os dados que forem pedidos, da maneira especificada.

Para a obtenção de um ficheiro, a sua hash e a parte (todo) da chave cifrada deve ser copiada do ficheiro fornecido no depósito e colada aquando do seu pedido na execução do programa.

Apresentam-se a seguir um conjunto de *screenshots* que exemplificam o uso do programa para o ficheiro *exemplodoc.docx*.

Figura 1: Ficheiro *exemplodoc.docx*

```
mel@MEIAVIDA AP2-PD1 % npm start index.js

> ap2-pd1@1.0.0 start
> node index.js "index.js"

Choose one option:
1 - Deposit file
2 - Get file
0 - Leave

> 1
Enter the filename:
> exemplodoc.docx
Do you want this file to be accessed by more than one person? y/n
> n
Enter your public key file:
> pub.pem


Choose one option:
1 - Deposit file
2 - Get file
0 - Leave

>
```

Figura 2: Depósito do ficheiro *exemplodoc.docx*

Figura 3: Ficheiro resultante do depósito do ficheiro *exemplodoc.docx*, contendo a sua hash e a chave cifrada

```
Choose one option:
1 - Deposit file
2 - Get file
0 - Leave

> 2
Enter the file hash:
> ce4dcad88dcb3209900ecdd420aed5765955498db880a059525fe21fb0961fcd
Enter parts separated by spaces or the entire encrypted key if it was not shared:
>
Enter the private keys separated by spaces in the same order as the parts:
>priv.pem
Enter the passphrase for the #1 private key entered:
>


Choose one option:
1 - Deposit file
2 - Get file
0 - Leave

> 
```

Figura 4: Obtenção do ficheiro *exemplodoc.docx*

Figura 5: Ficheiro decifrado obtido através do programa