Verificador de Hashs automatico

para ficheiros

|  |  |
| --- | --- |
| **Curso**: | Lic. Engenharia Informática |
| **Unidade curricular:** | Sistemas Distribuídos |
| **Ano letivo:** | 2022/2023 |
| **Nº, nome , GitHub de aluno:** | 1702072 Filipe Santos, NobleLip |
| **Docente:** | Paulo Vieira |
| **Email:** | skate777771@hotmail.com |

**Índice**

[**Descrição do Trabalho 2**](#_Toc135245047)

[**Implementação do Trabalho 3**](#_Toc135245048)

[**Funcionamento do trabalho 7**](#_Toc135245049)

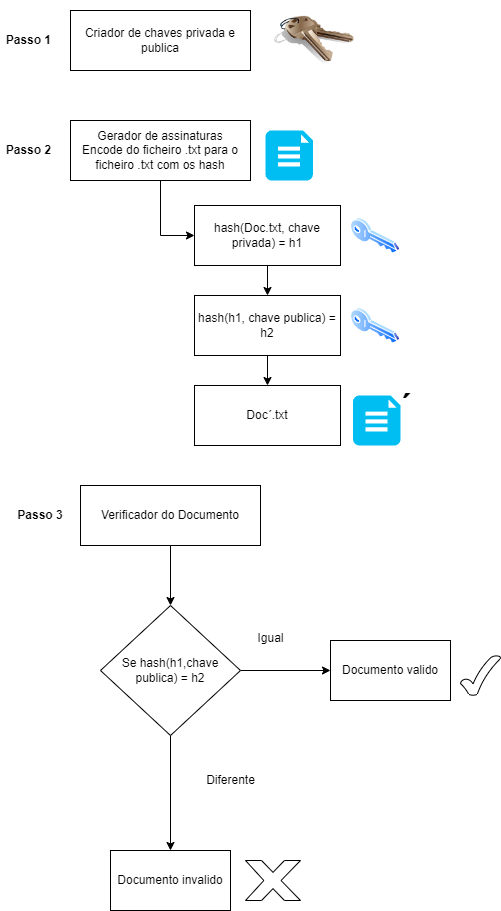
[**Conclusão 11**](#_Toc135245050)

[**Bibliografia 11**](#_Toc135245051)

# 

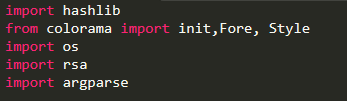
# Descrição do Trabalho

Neste trabalho foi proposta a automatização da assinatura de ficheiros de modo a verificar a veracidade dos ficheiros e vídeos publicados, usando chaves publicas e privadas conseguimos obter dois hashs dos ficheiros que vão ser cruciais para verificar se o ficheiro foi adulterado ou se continua legitimo, sendo assim, foi necessário a construção de um algoritmo de modo a parcelar o trabalho e ferramentas necessárias para a criação do projeto, sendo este o seguinte:



# Implementação do Trabalho

Apos alguma pesquisa de como deveria realizar o trabalho, encontrei 4 módulos que são cruciais para a realização do trabalho e um modulo que serve para a parte estética do trabalho.



* **HashLib** - Este módulo implementa uma interface comum para muitos algoritmos diferentes de hash seguro e resumo de mensagem. Estão incluídos os algoritmos de hash seguro FIPS SHA1, SHA224, SHA256, SHA384 e SHA512 (definido no FIPS 180-2), bem como o algoritmo MD5.
* **Os** – Responsável por disponibilizar todos os ficheiros no diretório para uma mais fácil automatização da assinatura, não sendo assim necessário inserir o nome individual dos ficheiros.
* **Rsa** - Python-RSA é uma implementação RSA puramente Python. Ele suporta criptografia e descriptografia, assinatura e verificação de assinaturas e geração de chaves de acordo com PKCS#1
* **Argparse** - facilita a criação de interfaces de linha de comando amigáveis. O programa define quais argumentos requer, também gera automaticamente mensagens de ajuda e uso.

E o modulo estético:

* **Colorama –** Usado para colorir os outputs em consola de modo a distinguir os hash resultantes das assinaturas.

Com os módulos necessários comecei a implementação do projeto.

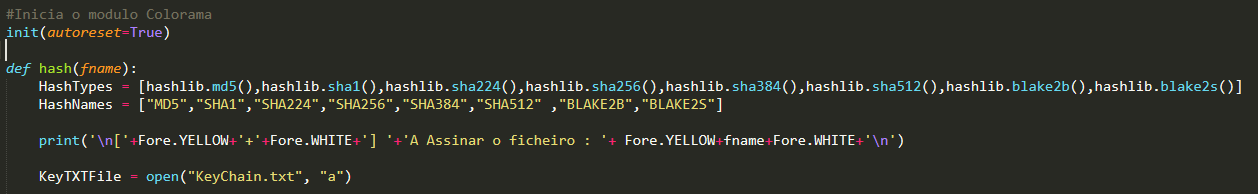
Primeiro foram definidos os parâmetros que o programa aceita:

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

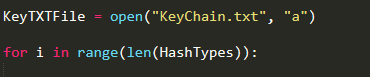
* **--f** – Nome de apenas um ficheiro de modo a não assinar todos os ficheiros desnecessariamente.
* **--a** – True/False , se for igual a True assina todos os ficheiros presentes no diretório.

De Seguida foi criada a função principal de assinatura de ficheiros:



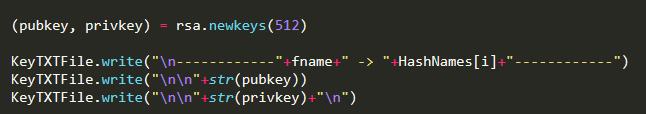
Para começar foi inicializado o colorama na primeira linha, de seguida foi definida a função e criado dois arrays responsáveis por armazenar as funções do modulo hashlib e os respetivos nomes das mesmas, de modo a usar um loop sem ter que chamar uma a uma.

De seguida é feito um print de modo a deixar a saber o utilizador que o processo de assinatura começou.

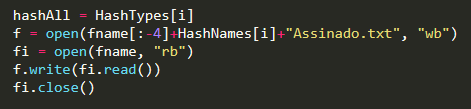


De seguida é aberta a “carteira de chaves” publicas e privadas, esta carteira apenas existe de modo a verificar a veracidade do programa, ao guardar as chaves publica e privada pode ser provado que as assinaturas são de facto reais.

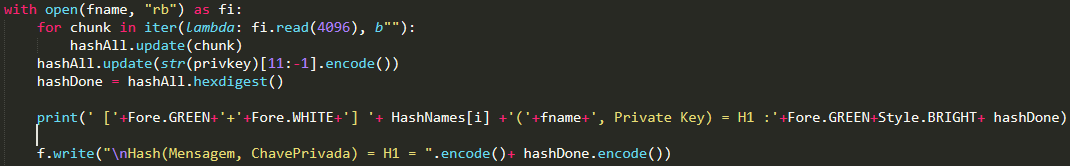
Depois de aberta a “carteira de chaves” é iniciado um loop que percorre todos os métodos de assinatura presentes nos arrays referidos anteriormente.



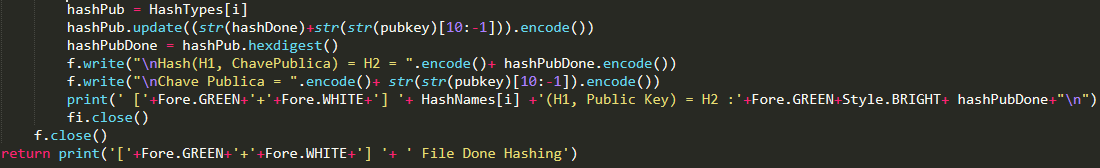
Aqui são criadas ambas as chaves publica e privada e escrita na “carteira de chaves” juntamente com o nome do ficheiro que foi assinado de modo a poder verificar a veracidade de cada assinatura.



De seguida é criado o ficheiro resultante da assinatura e escrita a informação do ficheiro inicial.

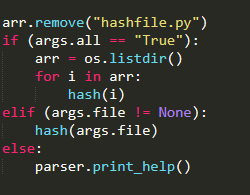


Aqui é aberto o ficheiro inicial a ser assinado, feita a leitura de 4096 bytes em 4096 bytes de modo a deixar a assinatura menos lenta, de seguida é adicionada a chave privada ao ficheiro e obtemos assim o “H1” ou ( Hash(Ficheiro, Chave Privada) = H1 ) , também é feito o print do H1 de modo a avisar o utilizador que foi sucedida a primeira assinatura e guardado no ficheiro assinado o H1.



É chamada novamente uma função do array das funções de assinatura visto que já foram feitos vários updates o que não seria possível dentro deste contexto, precisamos exclusivamente do H1 juntamos assim a chave publica e obtemos o H2 ou (Hash(H1, Chave Publica)), que é o que é feito na segunda linha, depois disto escrevemos no ficheiro assinado o H2 e também a chave publica de modo a ficar disponível para a verificação da veracidade do documento, é feito um print de modo a deixar o utilizador a saber que foi sucedida a assinatura na totalidade e é feito um return que conclui a assinatura do ficheiro na totalidade.

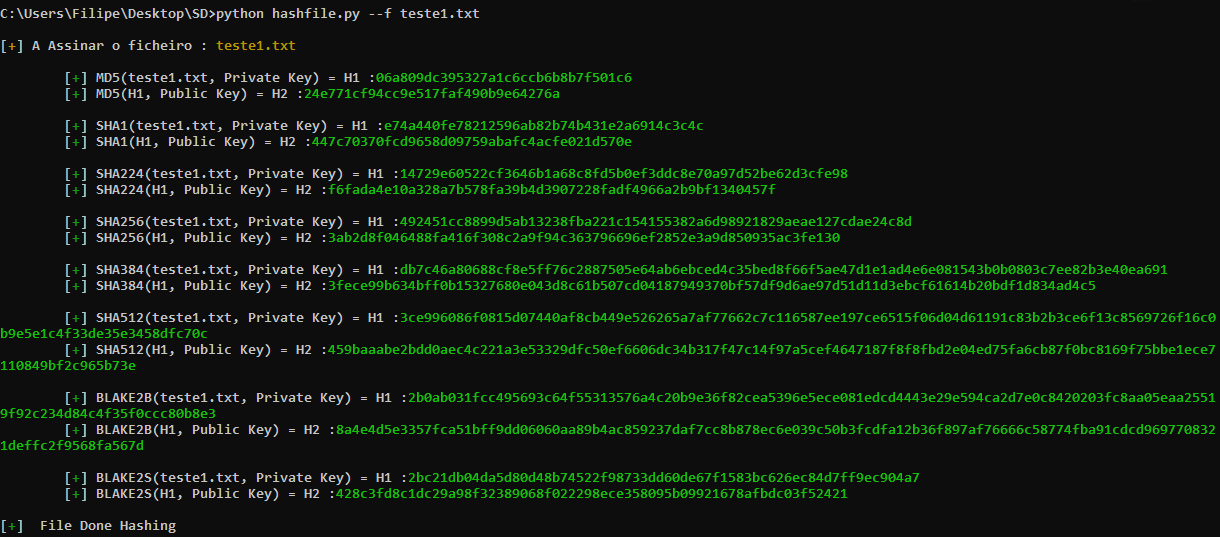
Depois de toda a função estar concluída basta chamar a merda, para isso temos duas formas de chamar, ou pedimos que assine individualmente um ficheiro, ou pedimos para adicionar todos os ficheiros presentes no diretório, caso nenhuma opção seja selecionada é feito um print de como usar o programa de modo a avisar o utilizador.



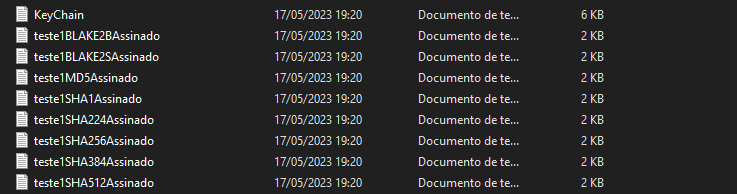
# Funcionamento do trabalho

Para o teste do programa foram criados 3 ficheiros teste, cada ficheiro teste tem um diferente paragrafo gerado pelo Lorem Ipsum e também temos um vídeo, para a assinatura de vídeos.

Sendo assim é feito agora o teste da escolha de apenas um ficheiro para a encriptação:

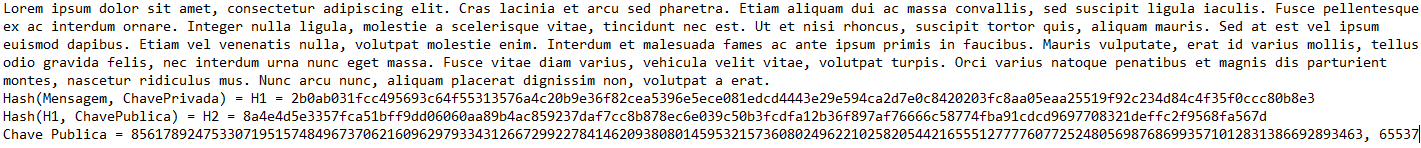


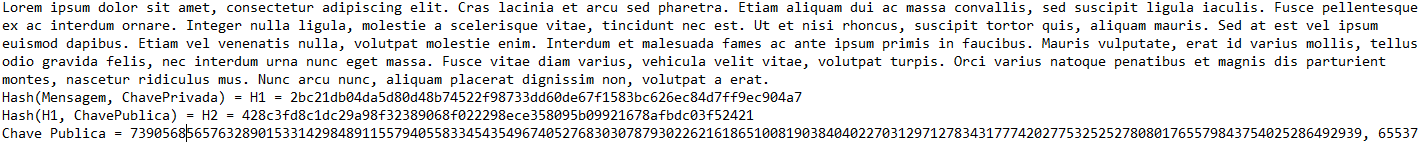
O resultado do programa são 8 ficheiros assinados com diferentes métodos de encriptação, md5, sha1, sha224, sha256, sha384, sha512, blake2b e blake2s e claro também é criado a “Carteira de chaves”:

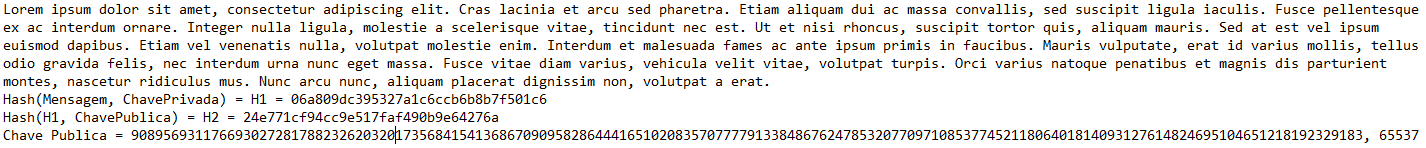


O output dos ficheiros é todo idêntico mudando apenas o método de assinatura, para exemplificar foi mostrar os 3 primeiros métodos:

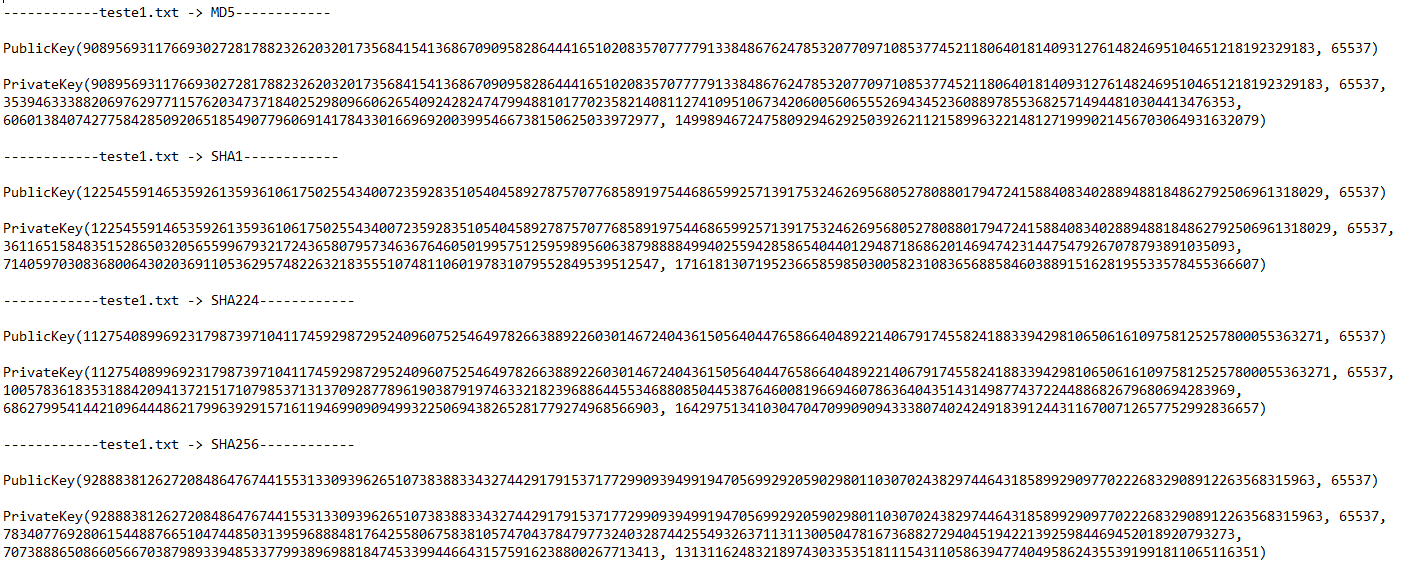
Blake2b:

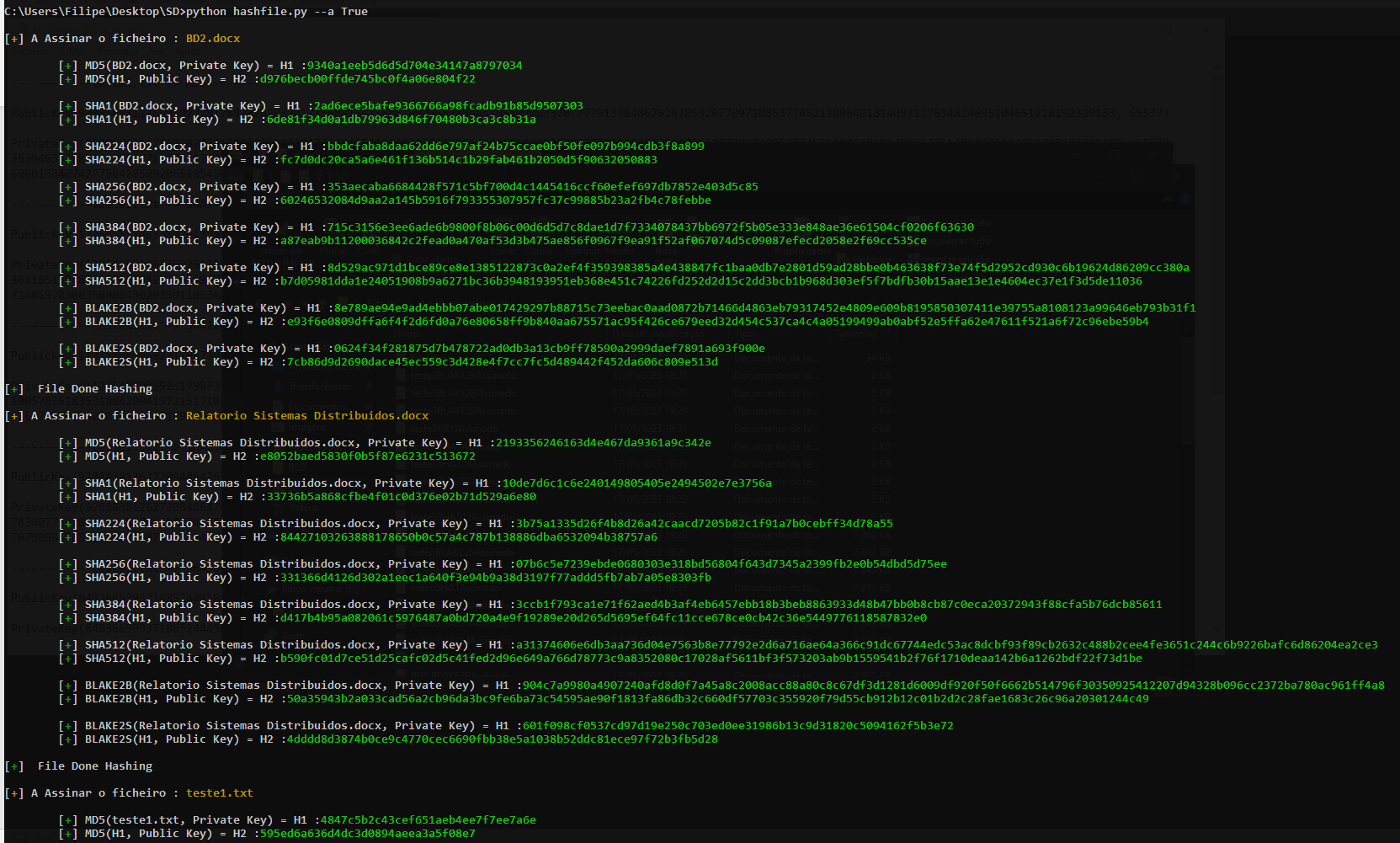


Blake2s:

Md5:

Em relação a carteira de chaves este é o seguinte output:



Agora vou exemplificar usando o programa para encriptar todos os ficheiros presentes no diretório:

Sendo isto uma pequena parte do output, e também temos os ficheiros assinados:

Uma imagem com texto, captura de ecrã, menu, design

Descrição gerada automaticamente

Sendo esta também uma parte da quantidade dos ficheiros que foram encriptados.

# Conclusão

Com este trabalho consegui aprofundar vários métodos de assinatura e também perceber o quão importante é a necessidade de assinar tudo o que é informação importante, não só conseguimos impedir a modificação e adulteração da informação como também conseguimos eliminar todas as notícias falsas da internet.

Sendo as “Fake News” um dos maiores problemas da internet e agora com a inteligência artificial os “Deep Fakes”, a assinatura de vídeos e noticias conseguia facilmente erradicar todo esse problema, e conseguíamos assim perceber quem são fontes seguras.  
Também poderíamos usar este método para impedir a adulteração da história, que como sabemos até hoje foi tão alterada.  
Em suma ,este trabalhou proporcionou-me uma melhor compreensão sobre o assunto de assinatura de ficheiros e também ferramentas que posso utilizar para praticar o mesmo futuramente.

# Bibliografia

1. Documentos fornecidos pelo docente em contexto de aula
2. <https://stuvel.eu/python-rsa-doc/usage.html>
3. <https://docs.python.org/3/library/os.html>
4. <https://docs.python.org/3/library/argparse.html>
5. <https://github.com/tartley/colorama>