



universidade  
de aveiro

## LEI: SEGURANÇA INFORMÁTICA E NAS ORGANIZAÇÕES

---

### P3: Autenticação

---

*Discentes:*

João Santos - 73761

Henrique Manso - 65308

*Docentes:*

Prof. João Barraca

Prof. Vítor Cunha

January 8, 2020

# Contents

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b>Métodos criptográficos utilizados</b>	<b>2</b>
2.1	Cifras assimétricas . . . . .	2
2.1.1	RSA . . . . .	2
2.2	Cifras simétricas . . . . .	3
2.2.1	AES128 . . . . .	3
2.2.2	3DES . . . . .	4
2.2.3	CHACHA20 . . . . .	4
2.3	Modos de cifra . . . . .	4
2.3.1	CBC . . . . .	4
2.3.2	CTR . . . . .	5
2.4	Controlo de integridade . . . . .	6
2.4.1	SHA-256 e SHA-512 . . . . .	6
<b>3</b>	<b>Autenticação</b>	<b>7</b>
3.1	Autenticação de cliente . . . . .	7
3.1.1	Cartão de cidadão, One-Time-Password e Desafio-Resposta . .	7
3.2	Autenticação do Servidor . . . . .	8
3.2.1	Certificados x509 . . . . .	8
<b>4</b>	<b>Workflow</b>	<b>8</b>
4.1	Desenho Ilustrativo . . . . .	9
4.1.1	Autenticação via otp-cc . . . . .	9
4.1.2	Capturas de ecrã . . . . .	10
4.2	Entidades e ficheiros necessários á operação . . . . .	14
4.3	Handshake . . . . .	15
4.4	Cifra Híbrida para transmissão do ficheiro . . . . .	16
4.5	Resultados . . . . .	18
<b>5</b>	<b>Conclusão</b>	<b>18</b>
<b>6</b>	<b>Referencias</b>	<b>19</b>

# 1 Introdução

Projeto realizado no âmbito da disciplina de segurança informática e nas organizações. O presente relatório visa explicar quais as decisões e raciocínios tomados ao longo da elaboração de todas as fases do projeto.

Este trabalho visa explorar os conceitos relacionados com o estabelecimento de uma sessão segura entre dois interlocutores com implementação de mecanismos de autenticação em ambas as partes de modo a controlar os acessos. Explora conceitos relacionados com a troca de chaves, cifras simétricas, cifras assimétricas, controlo de integridade e autenticação mutua.

O objetivo principal será a implementação de um protocolo que permita a comunicação segura cliente-servidor em que ambos são autenticados. Para isso é necessário garantir um conjunto de condições de confidencialidade e integridade das mensagens trocadas. No final deverá ser possível a troca segura de um ficheiro entre o cliente e o servidor com garantia que esse canal não permite a terceiros visualizar o seu conteúdo e que ambos os intervenientes na comunicação estão autenticados de modo a obter permissão para realizar a transferência de ficheiros.

No primeiro capítulo são abordados brevemente os métodos criptográficos utilizadas durante toda a elaboração do projeto. De seguida, é explicado mais detalhadamente os protocolos para autenticação mutua. Por fim podemos ver como se processa o workflow entre os intervenientes.

## 2 Métodos criptográficos utilizados

### 2.1 Cifras assimétricas

#### 2.1.1 RSA

Sistema criptográfico assimétrico por blocos, RSA é um dos algoritmos mais usados para transmissão segura de dados. A simetria da chave é baseada na dificuldade de fatorização de grandes números e no cálculo de logaritmos discretos.

Neste tipo de cifra é criado um par de chaves que estão diretamente associadas. A chave privada é secreta e não deve ser partilhada com ninguém enquanto que a chave publica é partilhada. As operações disponíveis de cifra e decifra são inversas, ou seja, um bloco cifrado com a chave publica apenas poderá ser decifrado com a sua chave privada associada e assim garantir confidencialidade para o único portador da chave privada. A operação inversa também é possível, assim, se cifrar um bloco com a chave privada é possível decifrar com a sua chave publica permitindo assim saber quem cifrou.

Por essa razão o RSA é um algoritmo tão popular, pois é um dos mais seguros, permite criptografia e ao mesmo tempo permite que um utilizador assine digitalmente um ficheiro.

Contudo este algoritmo é lento, por isso não deve ser utilizado para cifrar grandes conjuntos de dados. A cifra híbrida é uma excelente alternativa, que conjuga operações de cifra simétrica e assimétrica para obter melhores performances.

The image shows the letters 'RSA' in a large, bold, red, sans-serif font. The letters are slightly shadowed, giving them a 3D appearance as if they are floating above a white surface.

## 2.2 Cifras simétricas

As cifras simétricas permitem encriptar o conteúdo de ficheiros ou mensagens em que quem envia a mensagem e quem recebe usem a mesma chave secreta. Contudo as cifras simétricas permitem confidencialidade na troca de mensagens mas não permitem autenticidade de quem as enviou.

### 2.2.1 AES128

Cifra simétrica por blocos, o AES tem um tamanho de bloco fixo em 128 bits e uma chave com tamanho de 128, 192 ou 256 bits.

Esta cifra é gerada recorrendo a operações sobre uma matriz bidimensional com 4 etapas distintas:

- AddRoundKey - Combinações com a própria chave.
- SubBytes - Subtração e substituição de bytes.
- ShiftRows - Troca de posições de linha da matriz.
- MixColumns - Troca de posições de coluna da matriz.

AES é criptograficamente forte, bastante rápida e é a escolha mais popular para chaves simétricas.

### 2.2.2 3DES

Cifra simétrica por blocos, o 3DES é derivado de outra cifra simétrica, o DES. O DES com uma chave de 56 bits deixou de ser adequado nos dias de hoje devido ao tamanho da sua chave tornar possível a sua descoberta por pesquisa exaustiva.

O 3DES vem dificultar a pesquisa da chave dessa forma aplicando o algoritmo de DES 3 vezes a cada bloco. Assim o 3DES passa a ser uma aposta legítima na escolha de um algoritmo de cifra simétrica com chaves de tamanho superior.

### 2.2.3 CHACHA20

Cifra simétrica contínua. O CHACHA20 é uma derivação do SALSA20 que usa "round functions" que aumentam a difusão e aumentam a performance. Construída em funções "pseudorandom" baseadas em operações ARX, XOR e rotação esta cifra usa uma chave de 256 bits com nonce de 64 bits.

Esta cifra tem a vantagem de o utilizador pode procurar por qualquer posição no fluxo das chaves enquanto esta se encontra em execução.

## 2.3 Modos de cifra

Os diferentes modos de cifra vêm resolver o problema de cifras em blocos que eram de comprimento fixo. A utilização de diferentes modos de cifra permite que a cifragem por blocos seja mais difícil de quebrar, promovendo assim uma maior confidencialidade para mensagens com tamanho arbitrário.

Estes modos estão munidos de um vetor de inicialização que pode ser considerado como um bloco "falso", esse IV, em alguns modos não deve ser reutilizado para o o processo ser completamente aleatório. IV's reutilizados podem levar a padrões que tornam mais fácil decifrar esses mesmos blocos e destruir por completo a segurança da mensagem.

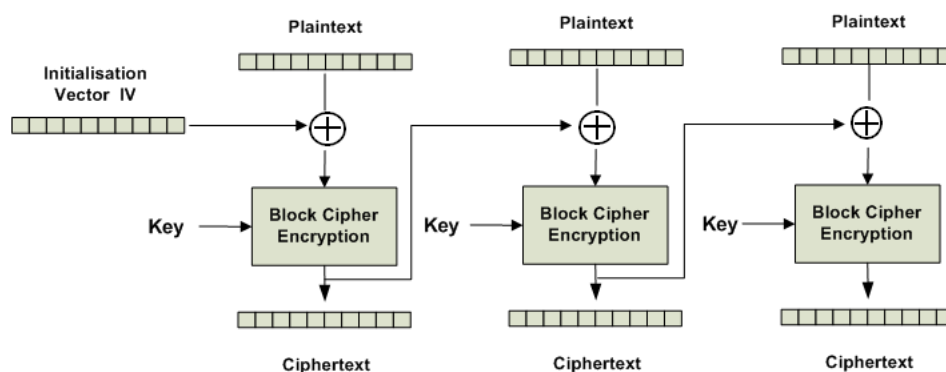
### 2.3.1 CBC

Modo de operação para cifra por blocos criptograficamente forte. O CBC aplica a cada bloco de texto uma função XOR em conjunto com o bloco cifrado posteriormente.

Desta forma todos os blocos são dependentes dos blocos anteriores. O primeiro bloco corresponde ao bloco "falso", ou seja, o vetor de inicialização. Esse vetor deve ser gerado aleatoriamente na altura de cifrar o texto e não deverá ser reutilizado de modo a não comprometer a confidencialidade da mensagem.

No processo de decifra é necessário que esse vetor de inicialização seja passado para quem fará a decifra de modo a conseguir decifrar corretamente a mensagem inicial.

Necessita de padding para alinhar o texto cifrado com o final do bloco.

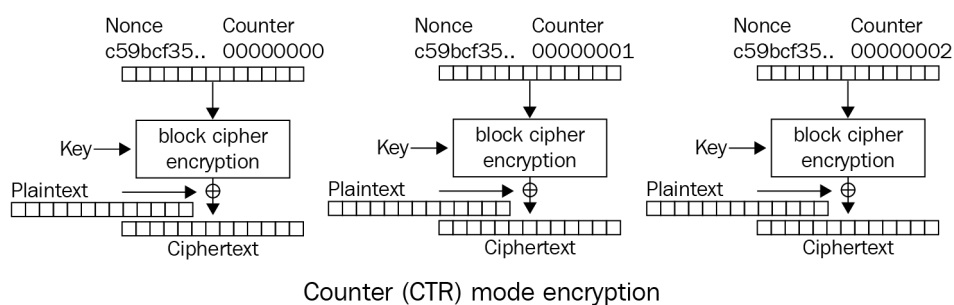


### 2.3.2 CTR

Modo de operação para cifra por blocos criptograficamente forte que transforma uma cifra por blocos numa cifra continua. Não é recomendada para cifras com blocos inferiores a 128 bits e não necessita de padding.

Vetor de inicialização funciona com realimentação e sofre operações XOR sobre o texto inicial. A cada bloco cifrado é incrementado o vetor de inicialização gerando confusão na entrada da cifra.

#### Stream Mode



## 2.4 Controlo de integridade

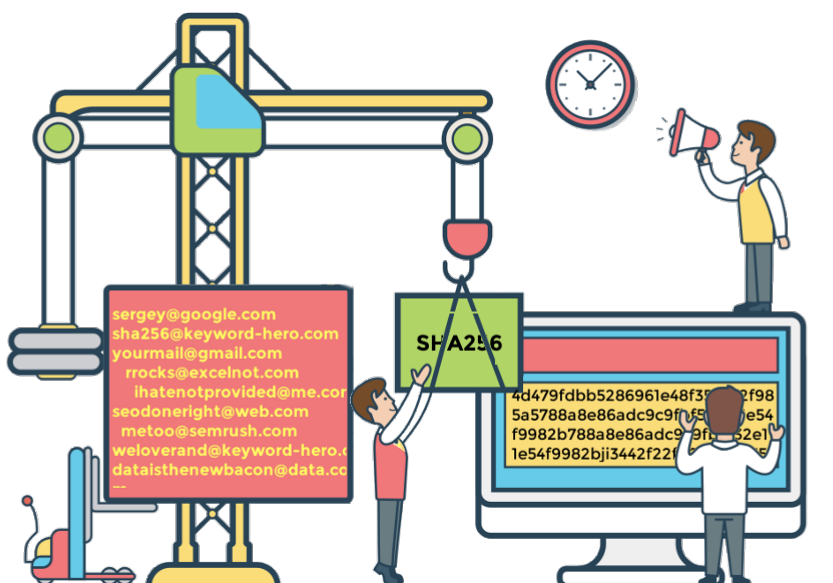
O controlo de integridade verifica se ocorreu alguma adulteração do seu conteúdo entre a origem e o destino. Desta forma é adicionada á mensagem uma informação adicional que identifique essa mesma mensagem. No momento em que a mensagem é recebida é necessário efetuar essa mesma operação de modo a observar que não houve alteração entre as mesmas.

O algoritmo deve ser forte contra colisões e uma pequena alteração no texto deve gerar uma alteração drastica no resultado.

### 2.4.1 SHA-256 e SHA-512

Função de síntese baseada em MD4, funciona como uma "impressão digital" dos textos.

Estas sínteses são geradas com chaves grandes, o que permite grande resistência à descoberta de texto e colisão, ou seja, dois textos nunca vão gerar a mesma síntese. O SHA-256/512 baseia-se em difusão e confusão em funções de compressão e são, neste momento, os algoritmos de síntese mais seguros. Contudo os mecanismos de integridade apenas são implementados corretamente quando associados a algum tipo de autenticação por parte de quem envia a mensagem, algo que neste projeto não será avaliado.



## 3 Autenticação

Nesta secção são explicados os processos utilizados no desenho e planeamento dos protocolos de autenticação do cliente e servidor.

### 3.1 Autenticação de cliente

#### 3.1.1 Cartão de cidadão, One-Time-Password e Desafio-Resposta

Para a autenticação do cliente utilizamos uma solução "overkill" complementada sempre com a implementação do mecanismo de one-time-password.

Assim, o cliente poderá autenticar-se com uma combinação de cc/otp ou apenas otp. O processo começa sempre pela geração da one-time-password, em que a key utilizada é pré-distribuída para o cliente e servidor. Esta key foi gerada com um `os.urandom(128)`. Depois de obtida a key do ficheiro pré-distribuído a otp é gerada de acordo com a hora atual e com validade de 30 segundos.

O cliente agora pode escolher de que modo se pretende autenticar. A primeira mensagem que é enviada para o servidor contém qual o tipo de autenticação que quer e um token da geração da otp. Quando o servidor recebe a mensagem do cliente verifica que esta contém esse mesmo token da otp e verifica que esta é válida para aquele instante temporal. No caso desse token não ser válido, ou seja, não foi criado pela key pré-distribuída ou não está de acordo com o instante temporal pré-definido é negado o acesso e fecha a conexão.

Se o token é verificado com sucesso então o servidor envia uma mensagem com um challenge para o cliente. Este challenge é gerado com recurso à biblioteca "secrets" que produz um nonce unico (URL-SAFE). Depois de recebido o challenge o cliente assina esse challenge com o cartão de cidadão. Para assinar o challenge é necessário assinar com a chave privada de assinatura digital presente no cartão, para isso é necessário a introdução do pin de autenticação. A mensagem enviada para o servidor deverá conter o challenge recebido para ser comparado com o que foi enviado, a assinatura do challenge, e o certificado de assinatura digital (KEY\_USAGE=digital\_signature) presente no cartão para ser possível ao servidor extrair a chave publica de modo a efetuar a verificação do mesmo.

Quando o servidor recebe a resposta ao challenge realiza um conjunto de verificações de modo a dar por terminado o processo de autenticação do cliente. Em primeiro lugar é feita a verificação do certificado enviado pelo cliente, ou seja, é comparado o certificado recebido com os certificados contidos no servidor. Assim, é verificada a validade do certificado e é percorrido (verificação da cadeia) até à raiz até encontrar uma CA válida (não contida na crl). De seguida podemos então verificar a assinatura



produzida pelo cliente comparando o nonce enviado. Caso todo o processo seja concluído com sucesso é enviada uma mensagem de sucesso para o cliente e podem agora ser negociados os algoritmos de cifra e efetuar a transferência do ficheiro. Se algum dos processos de verificação falhar, como a verificação do challenge ou a cadeia de certificação então o acesso é negado ao cliente e é fechada a conexão.

O cliente pode também optar por apenas se autenticar com recurso á otp, assim apenas será enviado o nome de utilizador e o token para ser verificado do lado do servidor, no caso de ser verificado com sucesso é possível passar para a fase de negociação de cifras e consequente transferência de ficheiros. Se a verificação não for bem sucedida é negado o acesso.

**No decorrer da validação da cadeia de certificação do cartão de cidadão notamos que quando a percorremos até ao ROOTca está contida numa crt, mesmo depois de atualizados os certificados em:**

**<https://www.ecce.gov.pt/certificados/>**

**Desta forma a validação é efetuada se o certificado anterior ao ROOT é CA e não está contido na crt.**

## 3.2 Autenticação do Servidor

### 3.2.1 Certificados x509

O servidor é autenticado utilizando certificados x509. A geração desse mesmo certificado é feita pelo servidor, é auto-assinado e encontra-se pré-distribuído para o cliente num ficheiro .pem.

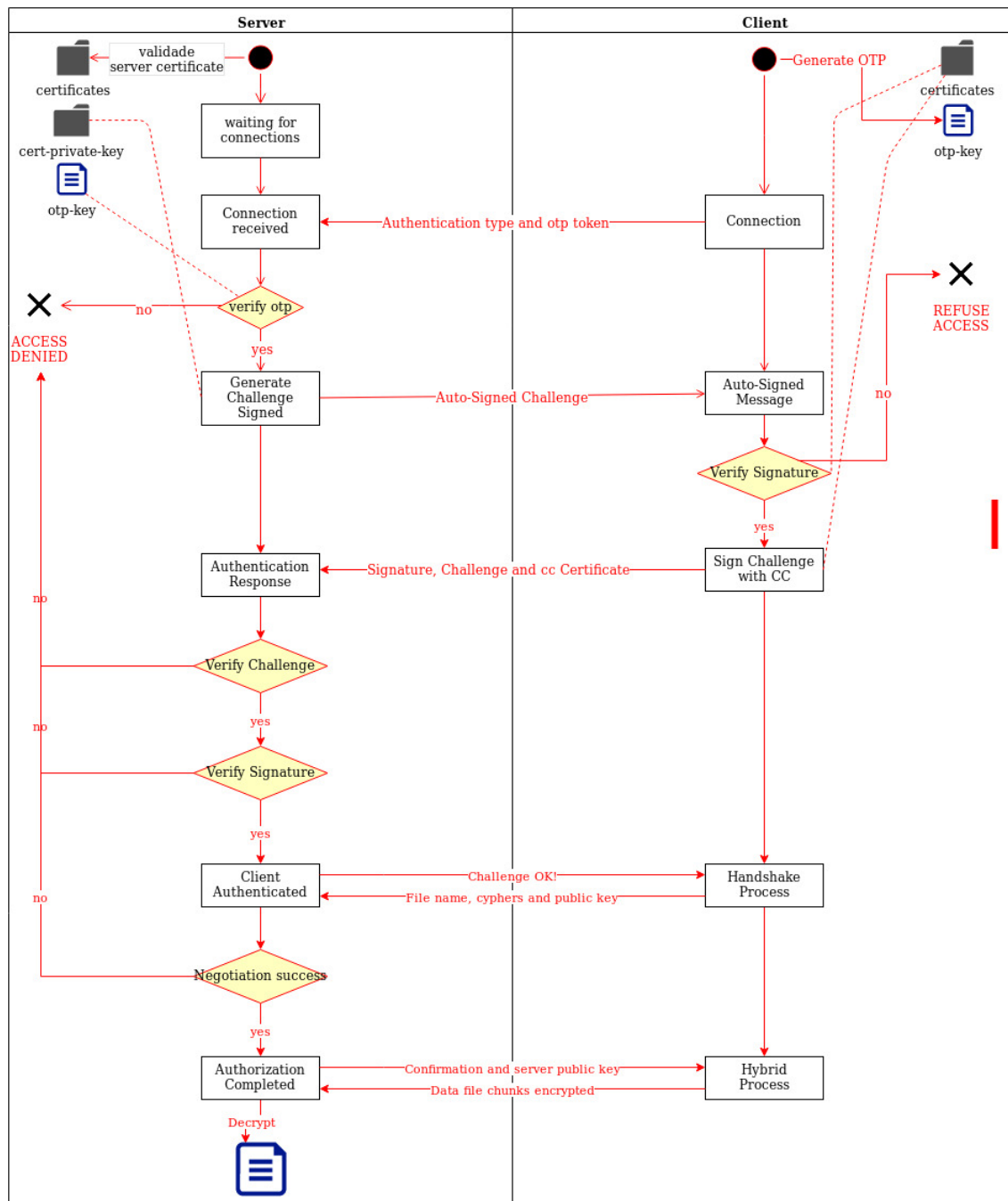
O processo de autenticação do servidor inicia-se quando este envia a mensagem de challenge para o cliente. A mensagem com o challenge request está pronta para enviar e é assinada pelo servidor com a chave privada do certificado. Quando a mensagem de challenge request chega ao cliente este extrai a assinatura e verifica com a chave publica do certificado pré-distribuído. Se a assinatura for verificada com sucesso o servidor está autenticado perante o cliente e então a comunicação pode continuar, caso a assinatura não coincida com o certificado então é terminada a comunicação e o servidor não foi autenticado.

## 4 Workflow

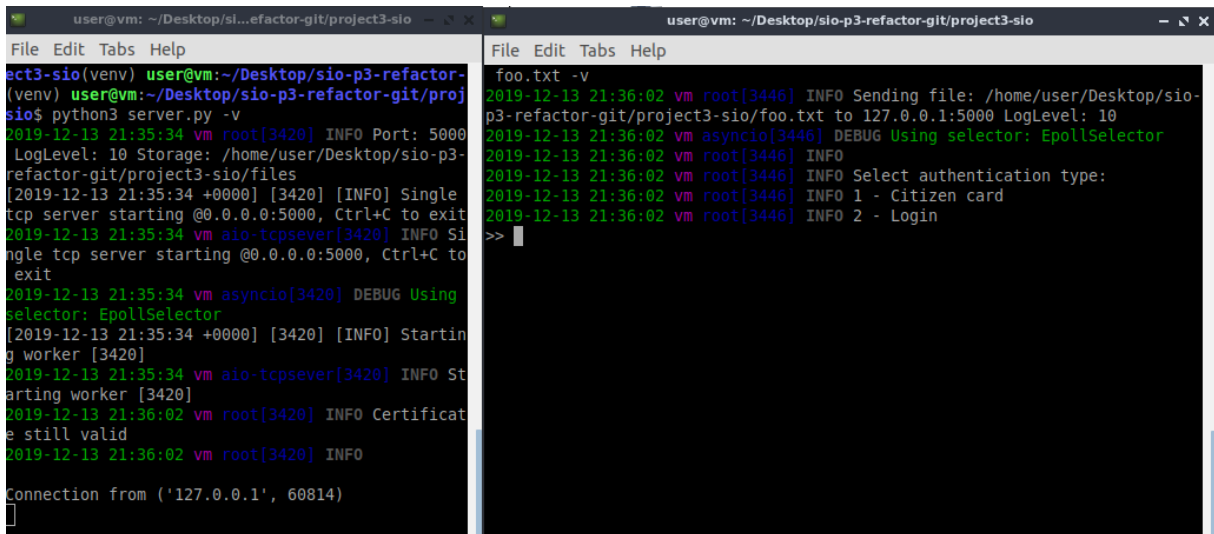
Nesta secção são expostos os pontos chave no workflow de ambas as entidades. Os diversos métodos invocados encontram-se devidamente comentados para melhor interpretação.

## 4.1 Desenho Ilustrativo

### 4.1.1 Autenticação via otp-cc



### 4.1.2 Capturas de ecrã



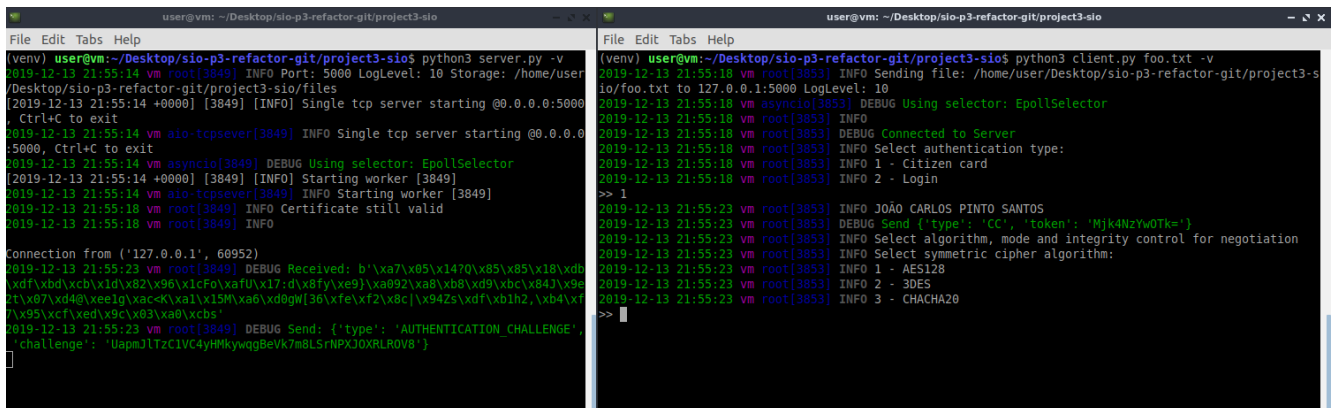
```

user@vm: ~/Desktop/sio-p3-refactor-git/project3-sio
File Edit Tabs Help
(venv) user@vm:~/Desktop/sio-p3-refactor-git/project3-sio$ python3 server.py -v
2019-12-13 21:35:34 vm root[3420] INFO Port: 5000 LogLevel: 10 Storage: /home/user/Desktop/sio-p3-refactor-git/project3-sio/files
[2019-12-13 21:35:34 +0000] [3420] [INFO] Single tcp server starting @0.0.0.0:5000, Ctrl+C to exit
2019-12-13 21:35:34 vm aio-tcpserver[3420] INFO Single tcp server starting @0.0.0.0:5000, Ctrl+C to exit
2019-12-13 21:35:34 vm asyncio[3420] DEBUG Using selector: EpollSelector
[2019-12-13 21:35:34 +0000] [3420] [INFO] Starting worker [3420]
2019-12-13 21:35:34 vm aio-tcpserver[3420] INFO Starting worker [3420]
2019-12-13 21:36:02 vm root[3420] INFO Certificate still valid
2019-12-13 21:36:02 vm root[3420] INFO
Connection from ('127.0.0.1', 60814)
>>

user@vm: ~/Desktop/sio-p3-refactor-git/project3-sio
File Edit Tabs Help
foo.txt -v
2019-12-13 21:36:02 vm root[3446] INFO Sending file: /home/user/Desktop/sio-p3-refactor-git/project3-sio/foo.txt to 127.0.0.1:5000 LogLevel: 10
2019-12-13 21:36:02 vm asyncio[3446] DEBUG Using selector: EpollSelector
2019-12-13 21:36:02 vm root[3446] INFO
2019-12-13 21:36:02 vm root[3446] INFO Select authentication type:
2019-12-13 21:36:02 vm root[3446] INFO 1 - Citizen card
2019-12-13 21:36:02 vm root[3446] INFO 2 - Login
>>

```

Início da comunicação. Cliente escolhe como se irá autenticar, gera otp e comunica ao servidor.



```

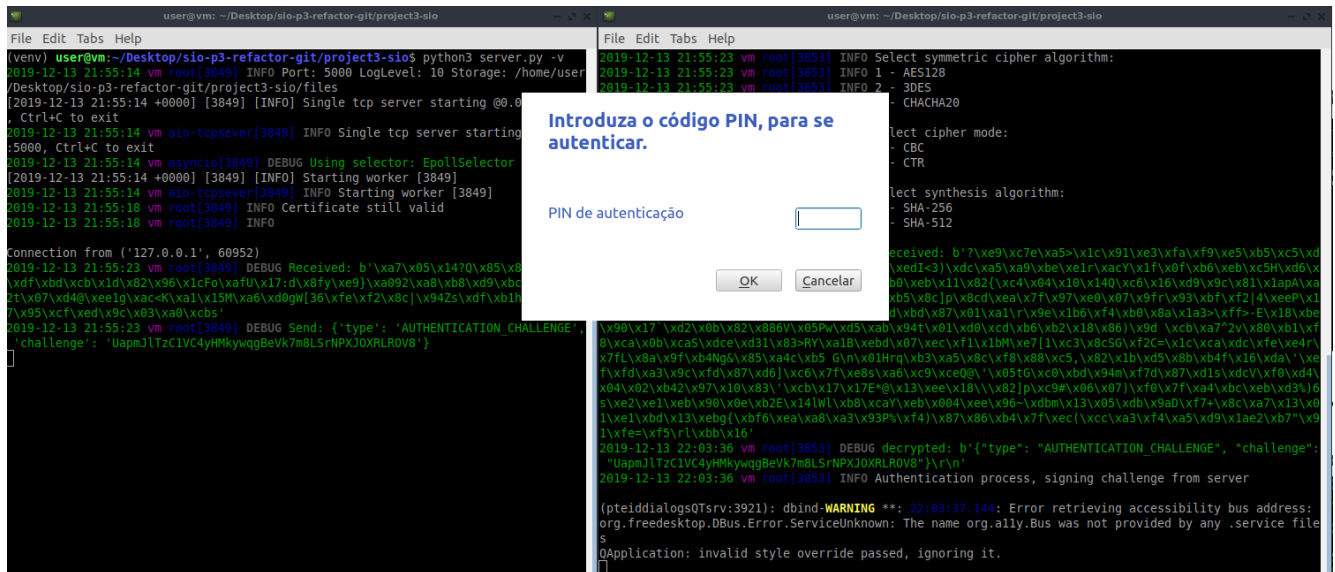
user@vm: ~/Desktop/sio-p3-refactor-git/project3-sio
File Edit Tabs Help
(venv) user@vm:~/Desktop/sio-p3-refactor-git/project3-sio$ python3 server.py -v
2019-12-13 21:55:14 vm root[3849] INFO Port: 5000 LogLevel: 10 Storage: /home/user/Desktop/sio-p3-refactor-git/project3-sio/files
[2019-12-13 21:55:14 +0000] [3849] [INFO] Single tcp server starting @0.0.0.0:5000, Ctrl+C to exit
2019-12-13 21:55:14 vm aio-tcpserver[3849] INFO Single tcp server starting @0.0.0.0:5000, Ctrl+C to exit
2019-12-13 21:55:14 vm asyncio[3849] DEBUG Using selector: EpollSelector
[2019-12-13 21:55:14 +0000] [3849] [INFO] Starting worker [3849]
2019-12-13 21:55:14 vm aio-tcpserver[3849] INFO Starting worker [3849]
2019-12-13 21:55:18 vm root[3849] INFO Certificate still valid
2019-12-13 21:55:18 vm root[3849] INFO
Connection from ('127.0.0.1', 60952)
2019-12-13 21:55:23 vm root[3849] DEBUG Received: b'\xa7\x05\x1470\x85\x85\x18\xdb\xdf\xbd\xcb\x1d\x82\x96\x1cf0\xafU\x17:d\x8fy\x99)\xa092\xa8\xb8\xd9\xbc\x84j\x9e2f\x07\x0d40\xee1g\xacK\xal\x15M\xa0\xd0gW[36\xfe\xfd\x8c]\x94Zs\xdf\x0b1h2,\xb4\x7f\x95\xcf\xed\x9c\x03\xa0\xcb5'
2019-12-13 21:55:23 vm root[3849] DEBUG Send: {'type': 'AUTHENTICATION_CHALLENGE', 'challenge': 'UapmJLTzC1VC4yHMkywqgBeV7m8LSrNPXJOXRLROV8 '}

user@vm: ~/Desktop/sio-p3-refactor-git/project3-sio
File Edit Tabs Help
(venv) user@vm:~/Desktop/sio-p3-refactor-git/project3-sio$ python3 client.py foo.txt -v
2019-12-13 21:55:18 vm root[3853] INFO Sending file: /home/user/Desktop/sio-p3-refactor-git/project3-sio/foo.txt to 127.0.0.1:5000 LogLevel: 10
2019-12-13 21:55:18 vm asyncio[3853] DEBUG Using selector: EpollSelector
2019-12-13 21:55:18 vm root[3853] INFO
2019-12-13 21:55:18 vm root[3853] DEBUG Connected to Server
2019-12-13 21:55:18 vm root[3853] INFO Select authentication type:
2019-12-13 21:55:18 vm root[3853] INFO 1 - Citizen card
2019-12-13 21:55:18 vm root[3853] INFO 2 - Login
>> 1
2019-12-13 21:55:23 vm root[3853] INFO JOÃO CARLOS PINTO SANTOS
2019-12-13 21:55:23 vm root[3853] DEBUG Send ('type': 'CC', 'token': 'Mjk4NzYwOTk=')
2019-12-13 21:55:23 vm root[3853] INFO Select algorithm, mode and integrity control for negotiation
2019-12-13 21:55:23 vm root[3853] INFO Select symmetric cipher algorithm:
2019-12-13 21:55:23 vm root[3853] INFO 1 - AES128
2019-12-13 21:55:23 vm root[3853] INFO 2 - 3DES
2019-12-13 21:55:23 vm root[3853] INFO 3 - CHACHA20
>>

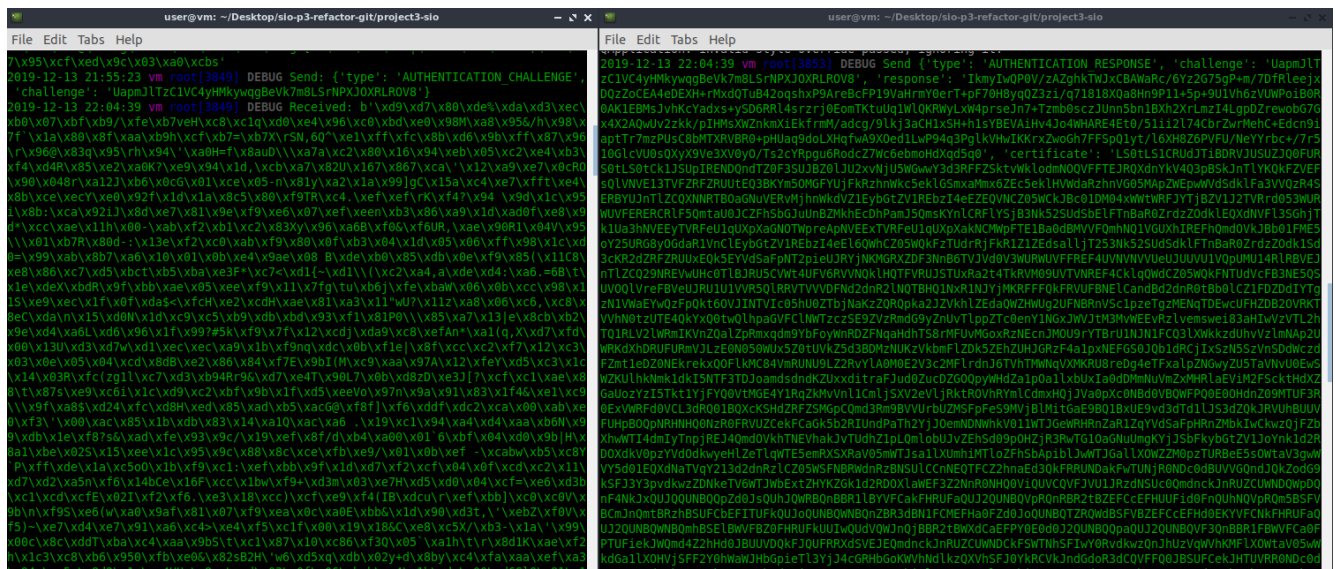
```

Servidor envia mensagem com challenge auto-assinada para o cliente para este ver-

ificar a autenticidade do servidor junto do certificado previamente distribuído.



O cliente depois de verificar a autenticidade do servidor é necessário extrair a chave de assinatura digital do cc.



Depois de extraída a chave podemos assinar o challenge. Para o servidor já autenticado é enviado o challenge recebido, a assinatura e o certificado do cartão de cidadão.

```

user@vm: ~/Desktop/sio-p3-refactor-git/project3-sio
2019-12-13 22:04:39 vm root[3849] DEBUG Verifying challenge
2019-12-13 22:04:39 vm root[3849] ERROR No CRL available, probably one of the last
... CC versions
2019-12-13 22:04:39 vm root[3849] ERROR Certificate ECRaizEstado revoked at 2018-0
8-21 21:21:20
2019-12-13 22:04:39 vm root[3849] INFO Checking if last entity is CA
2019-12-13 22:04:39 vm root[3849] INFO Entity is CA
2019-12-13 22:04:39 vm root[3849] INFO Validation chain complete
2019-12-13 22:04:39 vm root[3849] INFO Client passed challenge waiting for file
2019-12-13 22:04:39 vm root[3849] DEBUG Send { 'type': 'CHALLENGE OK' }
2019-12-13 22:04:40 vm root[3849] DEBUG Received: b'\xa2\x08\x0d\t\xcf\x08[\x0d
\x94\xda\xbf\x01\xfa\x3f\x9f\x3f\x0f\x03\xca\xee\x8b\x9f\tz\x06=\x95\x8b\x87\x
3f\xeb\x95\x84\x1f5\x190\x9b\x93p|\x00\xa6\xff\xcc\x13h\x00\x92\x1f5\n7\xfa\xefo\x
7\xebw\x9e\x0\xed\x9e\x9f8\x2\x14h|\xa8\x8d\x0c\xbb\xdb\x12\x05\xcb\x8a\x8e5\x
f1v\x1b>\xc95\x06\xef0\x0dR2\xcd\xdc{\i\x93c\x91\xfa\x04\x95\x8a\x87\xfb\x19/\
x88I>\x08\x88n\xab\xce\x96f\x9a\x07\t\x05\x08e5\x08\x16\x9a\x1f5\x9d\x06-745L\
\x05\x00M\x1a\x02\x0a7\xdb\x7f\x03\xccr\x91\x03z~\x03\x91T\x1f3\x7f\x05\x04f
\x06yM\x1b\x06\x09V\x18\xdd\x0e18\x00\x0f6\xa7\x07\x03\x7f\x08n\x07\xca\x1d\x06\x
a2(\x94T\x99\x9d\x0c\xac\x0fj|\x90\xdaM\x17W\xde\x07\xfb\xfb/I\x0aem\x03\x9d5c\x1
e7\x9a\x1f1\xfb\x09\x9a\x04\x0aag\x03\x03\x04\x0b|\x07\x01\x0a\x11\x0e\x0a\xfd\x07
\x085P0\x04\x09\x08\xda1\x0c67#\xf3\x07z\x089t(\x1d\x03\x080;\x03\x08=\x0d\x03\x0
0|\x0e\x18\xfa\t|\x9d\x0a10U\x1f0\xfb\x0a1\x04\x08\xee\x04\x0e\x05\x03\x03\x0e2\x03\x
0a\x0e\x0c\x068\x0a\x0c\x15c\x03\x0a\x0b0|\x0c0\x1d\x0f\x0b\x09\x0a1;\x0a\x91\x95\x
0c-\x0b0c\x07\x1aR\x0c\x05f\x99\x0a2\x0e0(\x01\xa0h1\kcaK*\x07\x02\x01\x0e9\x0a12
\x0a\x050c\x00b\x0c0f71\x080\x03\x07\x0d*\x03\x099\x060a1\x0e\x05f0\x04\x1f\x08f
\x072\x00\x080\x077 *\x0c\x0e0\x1a\x0e0\x17n\x05\x0a0\x090\x0e-\x1c5a\x12\x0c00\x01k
\x05\x00\x9a\x1f5\x05\x0f(\x07\x0b\x04\x08 \x0c0j\x0b\x06\x0899\x01\x0b\x08\x0c\x0
\x07\x0a\x0e\x08f\x0c\x0e\x0875\x03\x0c3t\x095\x19\x0f6*\x14p13\x02\x10\x0b42|\x1f0\x
\x91p1p\x02\x02\x0d\x01\x0a0n\x07\x096\x1a\x01\x0a7\x0a3y0P|\x07\x000\x05E\x0c\x0a40U
\x0b\x0dZ\x084\x04\x086E\x0c\x0c\x0e\x0c;\x05=\x08\x0f\x0c\x06\x02_dZ\x0d\x0b\x09f\x0a
\x09d)\x0a\x04\x95n\x08b0(\u0ec\x03t\x0af\x07(\x02\x07f7t\x08\x0c\x0d0#\x06\x03\x0f(\x
\x05 \x0e \x17j)\x08\x1a\x00\x0a2\x11'\d0\x83j|\x0f\x095\x0e\x0c\x0b\x0d7f(\x030v\x90d\x
\x0fdj=U\x02\x08b\x0c)\x03*\x0d\x01b\x055\x0a\x0b0h1\x07;\x1cR|\x07\x07f\x08\x08\x09f\x05
\x0bb\x1e\x0f0\x0a6\x0a0\x080q\x0a3\x0e0g|\x12\x0c\x0f9\x0d5\x08e\x0f2\x05-\x11\x09fP\x0f\x1
\x07\x07\x0e0\x0a0\x0d0c\x0e0\x070758 \x0d0b\x080\x0c07f7f1\x050p\x0c7 \x0c0b0\x0b0c

```

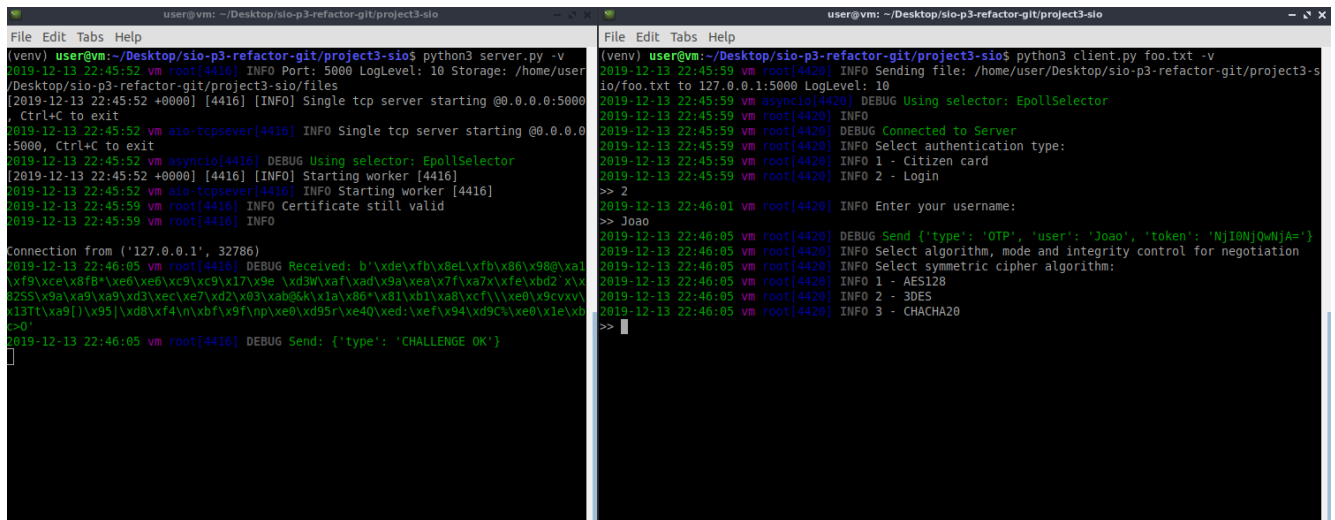
```

user@vm: ~/Desktop/sio-p3-refactor-git/project3-sio
2019-12-13 22:04:40 vm root[3853] DEBUG Received: b'\xfac\x03\n5\xfc\x94A\x94\x8fc=\x030\x010\x0e-\x
\xff6\x02\x04 \x0f\3\x1e\x01\xda\x0e0\A\x1c7,\xf0\x13\x08 \x0a5\x09\x8f1V\x01\x0c|\x9a\x9106\x03\x99
\x07>dr\x16\x02C\x0d1\x0e\x0aW)\x9c\x8f\x05\x9b-U\x06\x03\x03\x1c\x0b\x03\x06\x08\x0a\x0f1V\x01\x0b\x
1a\x0b\x04\x0c\x0d\x0d\xff0f\x09\x0b\x08a\x07\x0a5\x02\x0a8\x07\x0a\x0b\x0f6n:0W*\x03\x03\x0f\x01\x07[H
\xdd\nc\x02\x0e0f\x04\x04T\x15\x04\x0af\x09\x07f\x0cf1\x0c8-n\x17)\x04\x04\x09\x0a\x0a\x0b5v\x96(\y\x0c7\x1
c:\x0d\x8935\x19\x09)\x04\x0e\x1d\x0a1\x0c\x0b\x18\x06+\x10\x0d0\x0c1A\x0d\x02\x099M\x0d9m\x1d\x0a*22<0\x07\x1
be\x10\x07\x0a8e\x15\x0c0\x0f\x0c\x0a0\x0a1c;c\x09\x0f3\x02\x036A\x0e\x08c\x08e0y\x0e4\x0c8\x0ef\x940_g\x0bd\x0
1\x0d9\x0a0d0\x15q\x09\x03|\x11\x02\x0c9<\x0b|\xf3\x0f\x0d73P\x0c0\xaa\x05\x0af2\x19\x0a0\x99\x19\x08M\x08b\x0
8\x10M\x0c'\x17\x0b0\x04\x0c0\x02-L\x04\x082\x0a\x0e0\x91-\x0c0\x0b00\x0a2\x0c7'\x0fcv\x091\x0a0\x19\x0db|\x
cb\xff\x03|\xf80\x1f0>2\x0e)\x022\x0b00\x01\x08b\x0e0\x07f\x0c7)'
2019-12-13 22:04:40 vm root[3853] DEBUG Decrypted: b'{"type": "CHALLENGE OK"}\r\n'
2019-12-13 22:04:40 vm root[3853] DEBUG Send { 'type': 'OPEN', 'file_name': '/home/user/Desktop/sio-p3-
refactor-git/project3-sio/foo.txt', 'symmetric_cipher': 1, 'cipher_mode': 1, 'synthesis_algorithm': 1,
'client_public_key': '-----BEGIN PUBLIC KEY-----\nMIIBIjANBgkqhkiG9w0BAQEAFAACCAQEAfUWj4I
XCUJHMSJH4p0tAgT0b0c7ZB5/AR9eqe5Aq0c1WbM0cc0jv+699M0W09f32jYEUk1590f/gTUvde0e/BaLmL/ufRLf2P347
mu27L/R6382da21/6n08uYz4etny8LZ29d7zL90cFnr55CL59m120W5r1bAFcsjVRHXIXZC0z2d1VFRHJBLteedjy0jv90X
WRAjLKAjN57862z4grAX0m142z9u66d13c0+Im3A08n69VjBSB818eKjY03pCt6WNEng+E017Nm75qBL0705j8CV0peBj
+kf560b2u1yL/E112cPv7M0uB3b1LVK2Njvav\nmwIDAQAB\n-----END PUBLIC KEY-----\n"}'
2019-12-13 22:04:40 vm root[3853] DEBUG Received: b'\xf1fx10\x0d\x07ff\rR\x9d6/L1\x089\x0a\x95\xdd\x14\x
0e\x0a\x0e8L\x96\x0cc\x0cfH\x15'\x0b\x086:\x066\x10\x0bb\x0dc0|\x08\x16\x0f0\x0c\x0a0\x0e2\x06\x07\x0e8r(\x10
3\x0d0\x06\x02\x03\x07\x0c0\x92\x90HM\x07f\n\x0e4r\x0a0\x18\x07WapT|\xaa0\x08\x08\x0e23\x080\x0af\x
f0\x0c\x93\x0d \x05/\x19\x0a7\x13\x101\x0b0 \x05k\x087'\x0b0Aev\x0e0:\x01\x05\x06\x080*\x08\x07\x0e9T\x0d35/\x
0a0\x0c\x0f2\x0d\x0c\x0d\x05\x0ef\x04\x0c2 \x01\x0c9Z\x0e5\x02\x09f\x03\x0c7\x07\x0e0\x0c\x02*Eq0xfar:\x01\x0
2\x0c6\x05M\x0a0\x0a0\x14/\x05\x1b\x0e2t\x091\x1b\x0e0b\x0a0\x1b2\x08c\x0f5\x0b9\x0af\x0d0\x11\x0e5\x0f0\x0c0c7V\x
06\x0c0d1\x0e7<\x0a0*\x0c0\x0b0\x08'\x02\x066)\t0\x0e0\x91\x0e2(\x0a1\x02H00\x06M\x07f-\x1d\x0d\x0a7\x0b2\x0b3
A\x01a>\x08b\x1f'\x17\x0c03\x0d0L\x0d4\x0ff\x0a0\x13\x0c\x0c3\x0c1e\x0c3\x0e4'\x0ff\x089\x0b5\x0f1Ac0\x085R\x02
\x07\x0e9\x0d0\x02\x0c0\x03\x0d0L\x0d4\x0ff\x0a0\x13\x0c\x0c3\x0c1e\x0c3\x0e4'\x0ff\x089\x0b5\x0f1Ac0\x085R\x02
\x07\x0e9\x0d0\x02\x0c0\x03\x0d0L\x0d4\x0ff\x0a0\x13\x0c\x0c3\x0c1e\x0c3\x0e4'\x0ff\x089\x0b5\x0f1Ac0\x085R\x02
\x07\x0e9\x0d0\x02\x0c0\x03\x0d0L\x0d4\x0ff\x0a0\x13\x0c\x0c3\x0c1e\x0c3\x0e4'\x0ff\x089\x0b5\x0f1Ac0\x085R\x02
\x07\x0e9\x0d0\x02\x0c0\x03\x0d0L\x0d4\x0ff\x0a0\x13\x0c\x0c3\x0c1e\x0c3\x0e4'\x0ff\x089\x0b5\x0f1Ac0\x085R\x02

```

O servidor vai agora validar se o challenge enviado é igual ao recebido e se a assinatura é válida. Neste caso concreto da imagem corresponde a um cartão que foi renovado á 1 mês e por isso quando procuramos remotamente se o certificado se encontra numa crl ela nem existe no repositório. Outro pormenor é que quando percorremos a cadeia até à raiz verificamos que o certificado raiz se encontra revogado. Neste caso verificamos se o certificado anterior da cadeia é CA e validamos a cadeia.

O servidor envia então a mensagem de ok para o cliente, estão agora mutuamente autenticados e podem começar a transferência do ficheiro. Para isso o cliente envia para o servidor qual a forma como vai cifrar o ficheiro bem como a sua chave publica.



The image shows two terminal windows side-by-side, illustrating the SIO authentication process. The left window shows the server running a Python script, and the right window shows the client running a Python script.

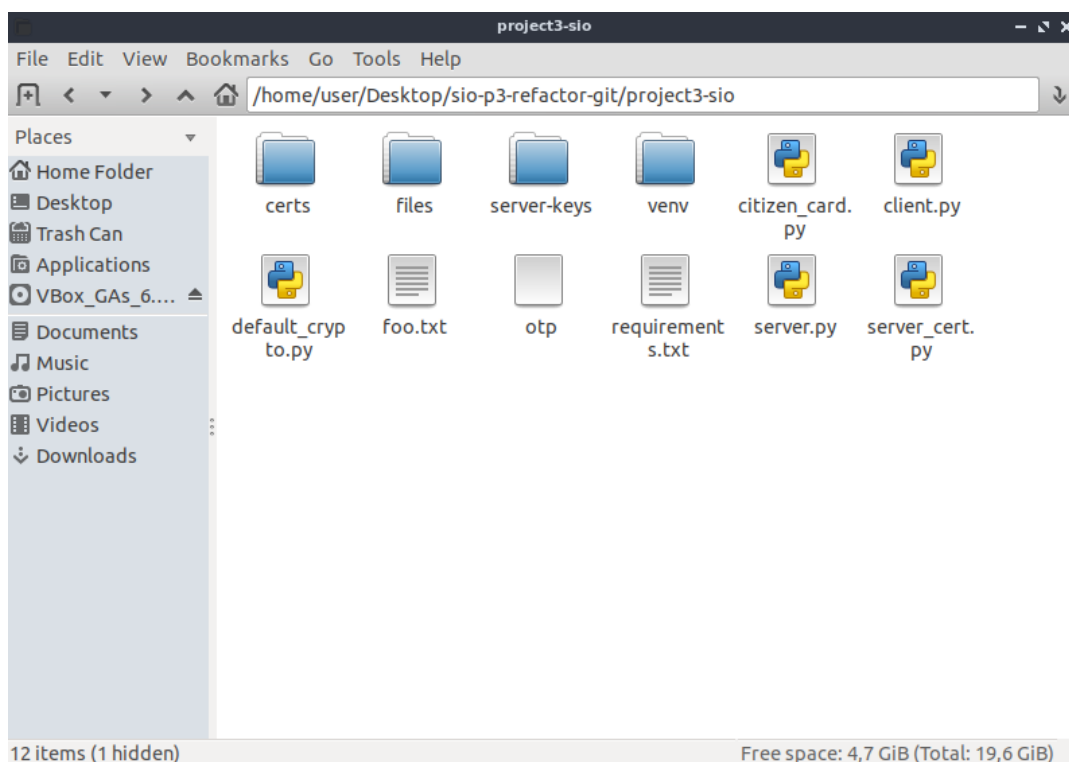
```
(venv) user@vm: ~/Desktop/sio-p3-refactor-git/project3-sio$ python3 server.py -v
2019-12-13 22:45:52 vm root[4416] INFO Port: 5000 LogLevel: 10 Storage: /home/user/Desktop/sio-p3-refactor-git/project3-sio/files
[2019-12-13 22:45:52 +0000] [4416] [INFO] Single tcp server starting @0.0.0.0:5000
Ctrl+C to exit
2019-12-13 22:45:52 vm aio:tcpserver[4416] INFO Single tcp server starting @0.0.0.0:5000, Ctrl+C to exit
2019-12-13 22:45:52 vm asyncio[4416] DEBUG Using selector: EpollSelector
[2019-12-13 22:45:52 +0000] [4416] [INFO] Starting worker [4416]
2019-12-13 22:45:52 vm aio:tcpserver[4416] INFO Starting worker [4416]
2019-12-13 22:45:59 vm root[4416] INFO Certificate still valid
2019-12-13 22:45:59 vm root[4416] INFO
Connection from ('127.0.0.1', 32786)
2019-12-13 22:46:05 vm root[4416] DEBUG Received: b'\xde\xfb\x8eL\xfb\x86\x980\xad\xfb\x9f\xce\x8f8*\xe0\xe6\xc9\xc9\x17\x9e \xd3W\xaf\xad\x9a\xea\x7f\x7a7x\xfe\xbd2 \x\x0255\x9a\xa9\xa9\x03\xec\x7d2\x03\xab06K\x1a\x86*\x81\x01\x08\xcf\\\xe0\x9cvx\x13Tt\xa9|)\x95|\xd8\xf4\n\xbf\x9f\np\xe0\xd95r\xe4Q\xed:\xef\x94\xd9C\xe0\x1e\x0'
2019-12-13 22:46:05 vm root[4416] DEBUG Send: {'type': 'CHALLENGE OK'}
```

```
(venv) user@vm: ~/Desktop/sio-p3-refactor-git/project3-sio$ python3 client.py foo.txt -v
2019-12-13 22:45:59 vm root[4420] INFO Sending file: /home/user/Desktop/sio-p3-refactor-git/project3-sio/foo.txt to 127.0.0.1:5000 LogLevel: 10
2019-12-13 22:45:59 vm asyncio[4420] DEBUG Using selector: EpollSelector
2019-12-13 22:45:59 vm root[4420] INFO
2019-12-13 22:45:59 vm root[4420] DEBUG Connected to Server
2019-12-13 22:45:59 vm root[4420] INFO Select authentication type:
2019-12-13 22:45:59 vm root[4420] INFO 1 - Citizen card
2019-12-13 22:45:59 vm root[4420] INFO 2 - Login
>> 2
2019-12-13 22:46:01 vm root[4420] INFO Enter your username:
>> Joao
2019-12-13 22:46:05 vm root[4420] DEBUG Send {'type': 'OTP', 'user': 'Joao', 'token': 'NjI0NjQwMjA='}
2019-12-13 22:46:05 vm root[4420] INFO Select algorithm, mode and integrity control for negotiation
2019-12-13 22:46:05 vm root[4420] INFO Select symmetric cipher algorithm:
2019-12-13 22:46:05 vm root[4420] INFO 1 - AES128
2019-12-13 22:46:05 vm root[4420] INFO 2 - 3DES
2019-12-13 22:46:05 vm root[4420] INFO 3 - CHACHA20
>>
```

No caso do cliente optar por se autenticar com o login apenas tem de utilizar um nome de utilizador e gerar o otp que será verificado do lado do servidor. Apenas os clientes que tenham a key pré distribuída poderão ter acesso para transferir ficheiros porque caso o servidor verifique que a otp não coincide o acesso a si é negado.

## 4.2 Entidades e ficheiros necessários á operação

Este projeto envolve duas entidades principais, client e server que pressupõem um funcionamento de um servidor para N clientes. O sistema é assíncrono, assim não terá em atenção tempo e espaço o que permite uma aproximação mais direta cliente/servidor.



O servidor é executado e simplesmente espera pela conexão de clientes ao mesmo. Quando recebe uma conexão pode efetuar trabalho de acordo com os pedidos efetuados. O objetivo do servidor é receber ficheiros sobre um canal seguro de vários clientes autenticando-se mutuamente, negociando com eles de que forma serão trocados os ficheiros.

No directório `"/certs"` encontram-se os certificados do cartão de cidadão e o certificado auto-assinado do servidor. Estes certificados são pré-distribuídos de forma a ser possível ao servidor certificar o cliente de acordo com os certificados do cartão de cidadão e para o cliente poder certificar o servidor de acordo com o certificado pré distribuído.

No directório `"/server-keys"` encontra-se a chave do certificado do servidor que apenas

é utilizado por ele para assinar mensagens e consequentemente se autenticar perante o cliente.

O ficheiro "otp" corresponde à key que é utilizada por ambas as entidades para gerar a one-time-password.

Os restantes ficheiros dizem respeito à implementação interna do cliente e servidor. Encontram-se comentados de modo a facilitar a percepção dos mecanismos utilizados.

- **citizen\_card.py** - PyKCS11, utilizado para manipulação do cartão de cidadão, obtenção dos certificados e chaves necessárias para as operações, realizar assinaturas, serializações de objetos, verificações de assinaturas, validação de cadeias de certificação e certificados revogados.
- **server\_cert.py** - Geração de certificados auto assinados para o servidor. Pressupõe criação de par de chaves RSA, geração do certificado e respetiva assinatura com a própria chave. Validação do certificado e carregamento de chaves para a operação.
- **default\_crypto.py** - Todos os métodos de criptografia utilizados, geração de chaves, carregamento de chaves, assinaturas e verificação das mesmas. Cifra/decifra de mensagens e geração e verificação de one-time-passwords.

Para verificação de certificados presentes na *crl* é necessário ligação à internet visto que estes são invocados remotamente.

### 4.3 Handshake

O servidor irá criar um par de chaves RSA para cada conexão efetuada, assim a chave pública que será transmitida a cada um deles será sempre diferente e a chave privada associada apenas irá decifrar a sua mensagem e nunca outra.

Neste ponto não existe flexibilidade de negociação de cifra assimétrica, ambos os end-points apenas aceitam RSA e como tal podemos fazer agora a negociação dos restantes processos de criptografia.

Este é o ponto onde o cliente pode negociar com o servidor como será cifrado o ficheiro, ou seja, qual a cifra simétrica que irá utilizar, o modo de cifra e qual a



função de síntese para garantir integridade da mensagem enviada.

```
2019-12-13 21:55:18 vm root[3853] INFO
2019-12-13 21:55:18 vm root[3853] DEBUG Connected to Server
2019-12-13 21:55:18 vm root[3853] INFO Select authentication type:
2019-12-13 21:55:18 vm root[3853] INFO 1 - Citizen card
2019-12-13 21:55:18 vm root[3853] INFO 2 - Login
>> 1
2019-12-13 21:55:23 vm root[3853] INFO JOÃO CARLOS PINTO SANTOS
2019-12-13 21:55:23 vm root[3853] DEBUG Send {'type': 'CC', 'token': 'Mjk4NzYwOTk='}
2019-12-13 21:55:23 vm root[3853] INFO Select algorithm, mode and integrity control for negotiation
2019-12-13 21:55:23 vm root[3853] INFO Select symmetric cipher algorithm:
2019-12-13 21:55:23 vm root[3853] INFO 1 - AES128
2019-12-13 21:55:23 vm root[3853] INFO 2 - 3DES
2019-12-13 21:55:23 vm root[3853] INFO 3 - CHACHA20
>> 1
2019-12-13 22:03:35 vm root[3853] INFO Select cipher mode:
2019-12-13 22:03:35 vm root[3853] INFO 1 - CBC
2019-12-13 22:03:35 vm root[3853] INFO 2 - CTR
>> 1
2019-12-13 22:03:36 vm root[3853] INFO Select synthesis algorithm:
2019-12-13 22:03:36 vm root[3853] INFO 1 - SHA-256
2019-12-13 22:03:36 vm root[3853] INFO 2 - SHA-512
>> 1
```

#### 4.4 Cifra Híbrida para transmissão do ficheiro

A cifras híbridas surgem como uma solução intermédia que junta o que de melhor oferecem as cifras simétricas e assimétricas. A cifra híbrida usa cifras simétricas para cifra a "granel", por estas serem mais eficientes, e cifras assimétricas para distribuição de chaves, nomeadamente para transmitir a cifra a "granel" ao interlocutor.

A implementação do próximo estado pressupõe o conhecimento da chave pública do servidor, e por isso é possível aplicar a cifra híbrida de acordo com o handshake da fase anterior.

Resta apenas proceder á cifra dos "chunks" do ficheiro que é para enviar para o servidor. Esses "chunks" são cifrados individualmente, o que dificulta a possibilidade de interceptação dos mesmos por um atacante. No caso de ser interceptado e for descoberto como decifrar, o atacante apenas conseguirá realizar a operação para

esse "chunk".

```

user@vm: ~/Desktop/sio-p3-refactor-git/project3-sio
File Edit Tabs Help
6dDaXQ4qTrHC90L8co3xfA3PUsz0Bxy46PT5XTgsVc/50Pgj/HVbTdpL+0vvBPLmxGvbWHCw1mrIIAbM1UMereLL9b3
XLyqeEN57+iIGBnRjbgX3mnVf+eeN1rMugeCaXllcgVkoEgshSE95vw+dT/hF932QUZjDEK7pyA00fyNfc+FVpSwgFRE
LBrTx4454gIC66vrdP8scVgcjugI4tifYaBIAiICu8k0siSv5rMr0ZIf730tnjaiKF7rr7/WQBoEszhnEM0BJw15hTcd
8MVFMw5+vMXhg/sjYdvuRpvTvs0pmnarcbrz1JASVKFXv3wq1GvfvgFX7N5x2sAHmnJc4uCAEFg9jaoFL5fGQn8mgYwQ
LIzdA55LhwxSj9SXLmJgqISv+on8dq2554nBXQIIRTQIR3Aui07WI+unZ9duW6ztHtvzhCjQWqFKAhmdt7+fv0Xr+1ek
FeAqGGnwrL86wAhSDK0qqgkTPsAEAQESyIuiwyBSRuTPTJWu9vf57X8/l3/69q'}
2019-12-13 22:58:50 vm root[4571] DEBUG Send {'type': 'DATA', 'data': '045d58oqqrBmcwqqj1ck
kQmioJdSjuxRsFAZmvyfF7Vsudlfu4uYwgmQ4cxeH0+5vEemwk5YwzNvSnD4qioMf1+TwDQXFMNy7KQV1jUzGdhT55Xw
RZMy4IosL5nnDW2IDUjC7c/+cpzT73+wfrnH7xjF2Ns1znvTGXPPXg7fcl0AF/99Nsjq5fv03x0lFkGIaJHbtzEBYSi
JioTggJcsPLVCCQkQv0uHdAWKYFi3NI00EtV6ZpajGRYytpZmNjAtP2nWxI/z68wXUq+0V3Pxs5c97v/3Y4nKe/+8VMY
A0KjTc8twqCAsnjMbT/7wRHp+PBRNNro3AZee0h1fV8MmM2fCpgkTxQFBHEATohoF3v/iBrx70++Hjdz1e0z05ft1tV
10kdKl0ml33e6s2bWejhvSv/1wFuagNDvLXENnr0+0MXtNU0yfGLj3U7ybFRUEUg0y8/TCMLgV0uMXBmV1Y2yFTUWRC
A0S5AcwFFA/PnfXn2z08MH/bu7n9z3/xt0iIDzHRAGBSXetv0s1iHABVZVZmdg77xfUF0TDBHBeBBGh7rhs2it2dEt+
GPlRf/9aPpLGVn7+rudDnw84xdIkgHJFBEY4GKtec7a2E9bB1m072QWABPa9jmFw3SMk3Dghu0+/UX4ZvYi7n0Qozwz01
RWPvVLO+tc+/GLaw3dc9z4A2Ji9gF154dn0Zeugpn71Q8Ss+cv0TNm9B4oIiZaOJCqQVWpKC2VRllFNyICXAJrFITAA
CfclgFclGccigRuHeiI5ZDtQBAAVHu8RVGRkU9uT93N+vFuQZ890ekM6d+HFxQW3bts7UbHgmVrrQvtvfqh/aHDgDeu3p
UI3DChy06mwZVkiDQ7CVREciTuuvhiiIEKSacrcdctschpsmt0JwoBuffjchRaWN28MzxuB20dmWlHQuieKzpmXi69mLc
Pftj9x/6dljneN0HbILOtF+XsPnfPvf9xo3PfwCwP0oisvAr9d0VLVVP80JPTZzJx9rGtiFJNnSi6PgU9VYVkwDMNou
ZataW+JtrtcUVfKqggFdEhHMenM0w95FSstq3CqmgBj5YbkDrTbkx/hYsotC1U5WSjesgFF2zahJj8PHg5c6BGMO+PDo
PGTe42xJ5WNDeRbq/5NZuzNyt06J7foTW9NSWMDGzx2X0Ehiz+o3WZ745wL733x3W8fffPLqPSs3KDK6hrIkkSTLw45i
RZQt7VtU8pubE5JP6Vn18RTLrvj0SEP33bN2pFDB358110vsfdfeJgW9h2M11MT'}
2019-12-13 22:58:50 vm root[4571] DEBUG Send {'type': 'DATA', 'data': '/c083+fm7S8aZpomDwo0Y
k6XEw6nAw6nBLVvofp8qKr2orLaC1mw4HTKcDkU0BQZDlLGRZUHgsBQX0t3zIQEHggAAAFc0hHwznvvyMt/altarMee
tLyFnd00wwdGVm5KCGuRwWVB7n7i9Tn//fZ7cWlFSgoKsH+4LIYpsllSWKy1Pw0mDEB0eGdwAF88t0cv7fIkpZdj3TDQ
HRkGEorqpr9fGvJYgCwNTUDG7fvQmxU+Fu/LF606WrfugAAHq9JREFUub+vKuh15uRfYiLD1ZDA0AQFujC4T09h+botf
t20YYzBq2pgdsBG2ASGuWA0ACTGRaOopAyc6iaXp/BdTh170lgVE0HEEIBHXKcQk222EU6LQtbpNj9yKkIlgVuGNA81
VDLSqH0nwcuCm+HNfFdkKAKJz0xZAH9e3F9+bsW6Cqqrntty7/YyzNqfvZn+u3stNPGXjIERjYtxfPysld4PVpAT06J
3whysrUup8tWb2RTRg5tF1Gb93m7Wz44P6cc5y48PPR929J7dnYwkZBMagHCGtmNfePe00ja4dMTDW/Bat+p0/aGfu7
N6TgwC365b/fvz19U++9kHVSv/f+S2NX8fz4H/eE2qqPcP2F5chKDOCA0Zz0+gWJw+mA4LDgcrugaxp8PhW6pG62k81t
ReMYiC3c5XFARue+ytI4ZpIrxTMA0u6RDSs3N296QEDGbfzpq7/eLbHnZZFu/q8fLgKYdVGNBwp1zDLXVoBkmTN0EJ
AhY8MdaMMYgChawZL2g4NALfSA7fz8+n/UbBXP80pVu/36SGMsIXa2b2L5JYrLKgAgRhLF20rLk1FRWQNRZMj0K0RZR
Wxt9q2WfWZlWkwhqS4KAsA/m/mXHbDZ20aZujExSARNx+cA5pmwLI4DhfPYbCzIlnT+1QU2nJFCKGADjkB5Bcw6fpA
0coy9jd4guqBUDjwMBAF+5NiMDgIBecAjtpgzma6j33j7WbZ33x4/zKddvTrCvveSqWMUDV9G8mXh1X9eTzJgSNHtr/p
kH9en/XzHgFzVn0x9TZS1acu3LjNt8V9zwdyxig68aQsVfc/uzL544PGjmk300jhgz8rj3+9j/XbWbDB/fnKWnpqadPv
12ivLIXDcMiIeY2XBu140xTfuzDg0SE0S4XYw+sK01/FlcHhUjsJyA+B2MWZJYM10PgVBgNenYvuTdkje9+n389d+
H8XnDG6r8sdEkj2TG8+vG3sszMrP998Q0CdGxuAHeT080MMciyDFmS4HK7wS0L'}
2019-12-13 22:58:50 vm root[4571] DEBUG Send {'type': 'DATA', 'data': 'uq5DVTWoPh+sRgsmuzjy0
L49cec1lwT0nfyWDTI5rhavWMe6JyXwXZL79t33/FvyG59+G6mqml/FZw/+PjSa7Er+1/1WNWpN3hEDQGLtZ8s5h64b0
GHXA6uu8dZ+7qxFW64YswMz30LgnGNrWmbyzF8XfXPZ+RP//uTrH7KXhrqj0bMkx0eJkWEhsLjFnd1k1pFehB3Y5soEB
oeDAjqEEArokBNARFgICopKYRgGEpzKEW+m0RhDpCKhm0vBhLBA9HE7DlysT9Ixeuuzb4be98LbvylsR6CILgAoKKqu
u7HgV6fGvjGZ99i+k+hM2YvXDaJZ9ekxF7duuQCwMdfzxy//NvbPxt2Wq7uHDjx7pqvD7Xu1/OxNezF874fs7CGX17d
Ens16t7blv+/acPH8xXrt+85bYnp/SqvVN22BTnqFAJ5wwPwJh+LvTv4kSIW6AvyrGy+AJQdQsZ+QZw7fR16aZqbMk8f
GMrj9fnfvrNTyAwISNld2Z43x5d52kkj78JI/rffdnjr8iyJMHpckF20A67SmGw65I5RBGKwFdl2E26uzGo0k6nrrnB
j6gTy9K0SHH3ZlzhvPfv67LveaB5+Mqq2rqa6cd75pn5A565vL9rDD4fFr9XD0/sJi9Ne27q556/YOyF66466dfzs20
npz370uuJzQ7f7B3nJ15Flq3e/YXa4c9CERQtoFBXRIMwoJDERRaQUUxjBrQDJcR9hfzGs33nA0G0A42Wfsz5g9/85p3
899P21PNg9wVhwYAD69+iCxLgoKKIKopKbNu9B3kFxsquq8Ddz7z0n7zr+pw/Vm+I/+S72SG//r5646aUXTzA7WRBA
W70694ZyFExcCoySsorkZKxZfL5RS1vrMYDL73D/3XzVTnzL62KP2fcqLy2eg9LV6xd/Q/XPxp4cBHDhLwKQ79KJ26dF

```

O processo de transferência de ficheiro cliente-servidor desenvolve-se da seguinte maneira:

1. Cifra do "chunk" com a cifra simétrica e mecanismo de integridade acordado.
2. Cifra com a chave publica(servidor) do vetor de inicialização, salt e password gerada automaticamente que foram utilizados para cifrar o "chunk" .
3. Envio para o servidor dos conjuntos anteriores.

O processo de recepção do ficheiro por parte do servidor é feito de maneira inversa:

1. Decifra com a sua chave privada o vetor de inicialização, salt e password que foram utilizados para cifrar o "chunk".
2. Decifra o "chunk" com cifra simétrica e mecanismo de integridade acordado utilizando para isso os dados provenientes da cifra assimétrica.
3. Faz a concatenação do "chunks" implicando a totalidade do ficheiro decifrada e disponível para visualização.

## 4.5 Resultados

Os resultados obtidos foram bastante satisfatórios, foi possível abordar todos os pontos que constituíam a elaboração deste projeto.

A autenticação de ambas as partes foi efetuada com sucesso e permite que sejam controlados os acessos por parte de clientes ao servidor. É bastante importante que ambas as entidades se autenticuem uma perante a outra de modo a não permitir ataques durante a transferência de ficheiros.

As maiores dificuldades durante todo o processo foi, em primeiro lugar a alteração de alguns processos que fizemos de maneira errada no projeto anterior e posteriormente a implementação de autenticação com o cartão de cidadão. Contudo, foi possível contornar essas dificuldades e obter resultados que vão de acordo com o esperado.

## 5 Conclusão

A concretização deste trabalho permitiu-nos evoluir bastante tecnicamente e explorar bastante mais "a fundo" os processos de autenticação, utilização de cifras simétricas, cifras assimétricas e da conjugação das mesmas.

Estabelecer um canal de comunicação seguro e autenticado entre duas entidades foi uma tarefa desafiadora mas bastante vantajosa para o nosso futuro como programadores. Bastante vantajoso foi também os guiões realizados nas aulas práticas que são facilmente adaptáveis para o contexto em que estamos a trabalhar.

## 6 Referencias

### References

- [1] <https://cryptography.io/en/latest/>
- [2] <https://joao.barraca.pt/teaching/sio/2019/>
- [3] <https://keyword-hero.com/using-sha256-hash-email-addresses>
- [4] <https://www.comparitech.com/blog/information-security/3des-encryption/>
- [5] <https://www.dei.isep.ipp.pt/asc/doc/assinatura-digital.html>
- [6] Livro de Segurança em Redes Informáticas - André Zúquete
- [7] <https://pkcs11wrap.sourceforge.io/api/samples.html>
- [8] <https://docs.python.org/3/library/secrets.html>