Projeto 3: Autenticação

Trabalho Elaborado por:

- Pedro Escaleira, 88821
- Rafael Simões, 88984

1. Índice

2. Autenticação de utentes através dum mecanismo desafio resposta

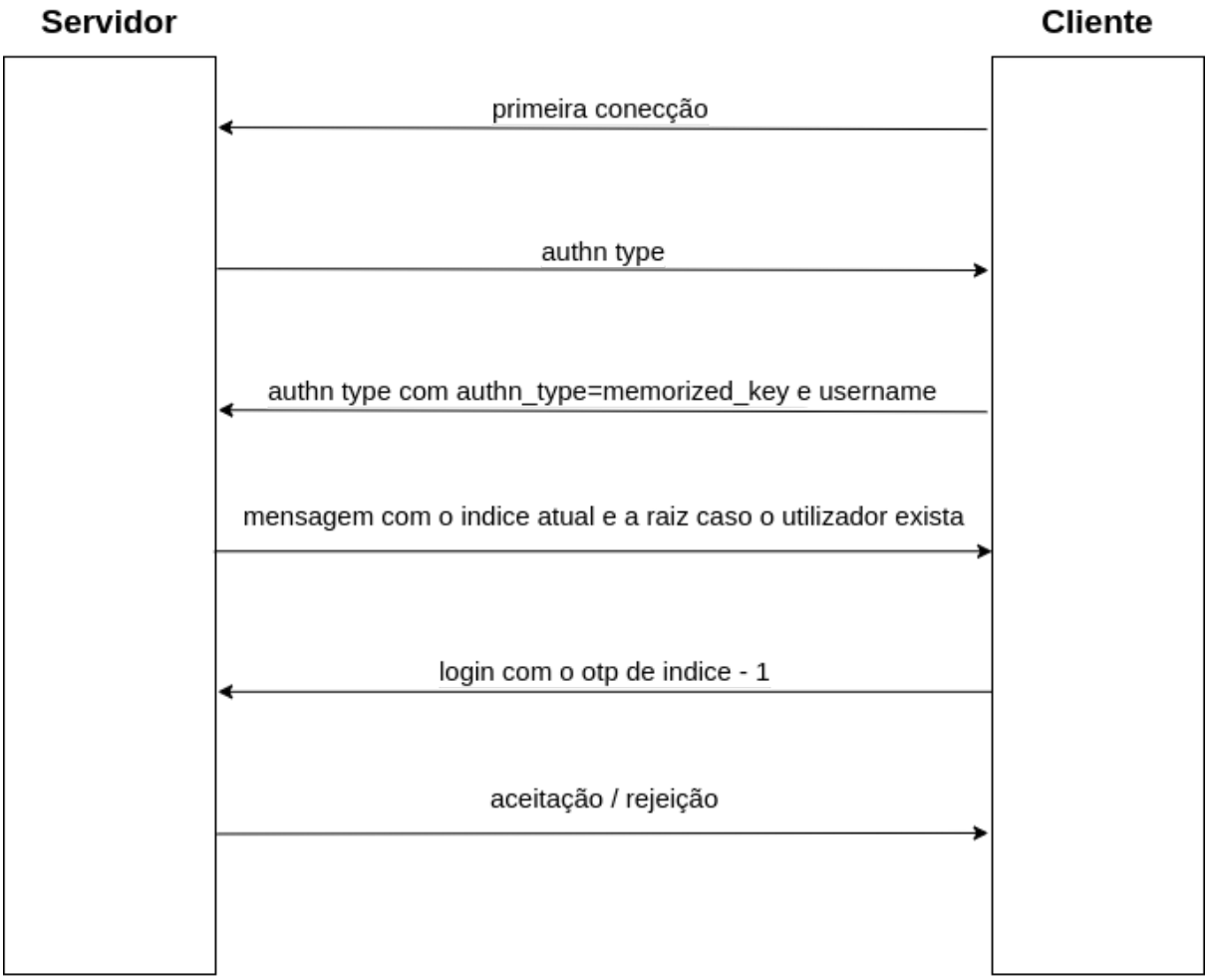
2.1. Explicação do protocolo

Para a criação deste protocolo, decidimos usar o sistema S/Key, já que a sua elegância e simplicidade nos despertou grande curiosidade.

Desta forma, o protocolo que desenvolvemos para sustentar este sistema envolve as seguintes trocas de mensagens entre o cliente e o servidor (também descritas no esquema acima):

1) Inicialmente, o cliente envia uma mensagem ao servidor que indica o início da comunicação entre ambos.
2) Como resposta, o servidor retribui com uma mensagem a pedir o tipo de autenticação que o cliente terá de realizar (necessário pelo facto de haver também autenticação usando o cartão de cidadão, como mais à frente será explicado).
3) Desta forma, o utilizador envia para o servidor o nome do utilizador que se está a autenticar.
4) Quando o servidor recebe essa mensagem, verifica se possui as credenciais deste utilizador, sendo que em caso afirmativo, envia uma mensagem ao cliente com a raíz e o índice atual do otp que se encontra armazenado na sua base de dados.
5) O cliente ao receber estes dados, calcula o valor da otp de índice - 1, sendo que envia este resultado para o servidor
6) O servidor ao receber o valor de otp de índice - 1, confronta-o com o valor de otp do índice atual, sendo que caso estes valores coincidam, envia uma mensagem de sucesso de autenticação ao cliente.

No esquema abaixo é possível obter uma simplificação em escama daquilo que foi acima explicado:


Servidor
Cliente
primeira conecção
authn type
authn type com authn_type=memorized_key e username
mensagem com o indice atual e a raiz caso o utilizador exista
login com o otp de indice - 1
aceitação / rejeição

2.2. Explicação do mecanismo de autenticação

Como já referido, o mecanismo usado para autenticação foi o S/Key. Desta forma, foram realizados os seguintes passos para que fosse possível

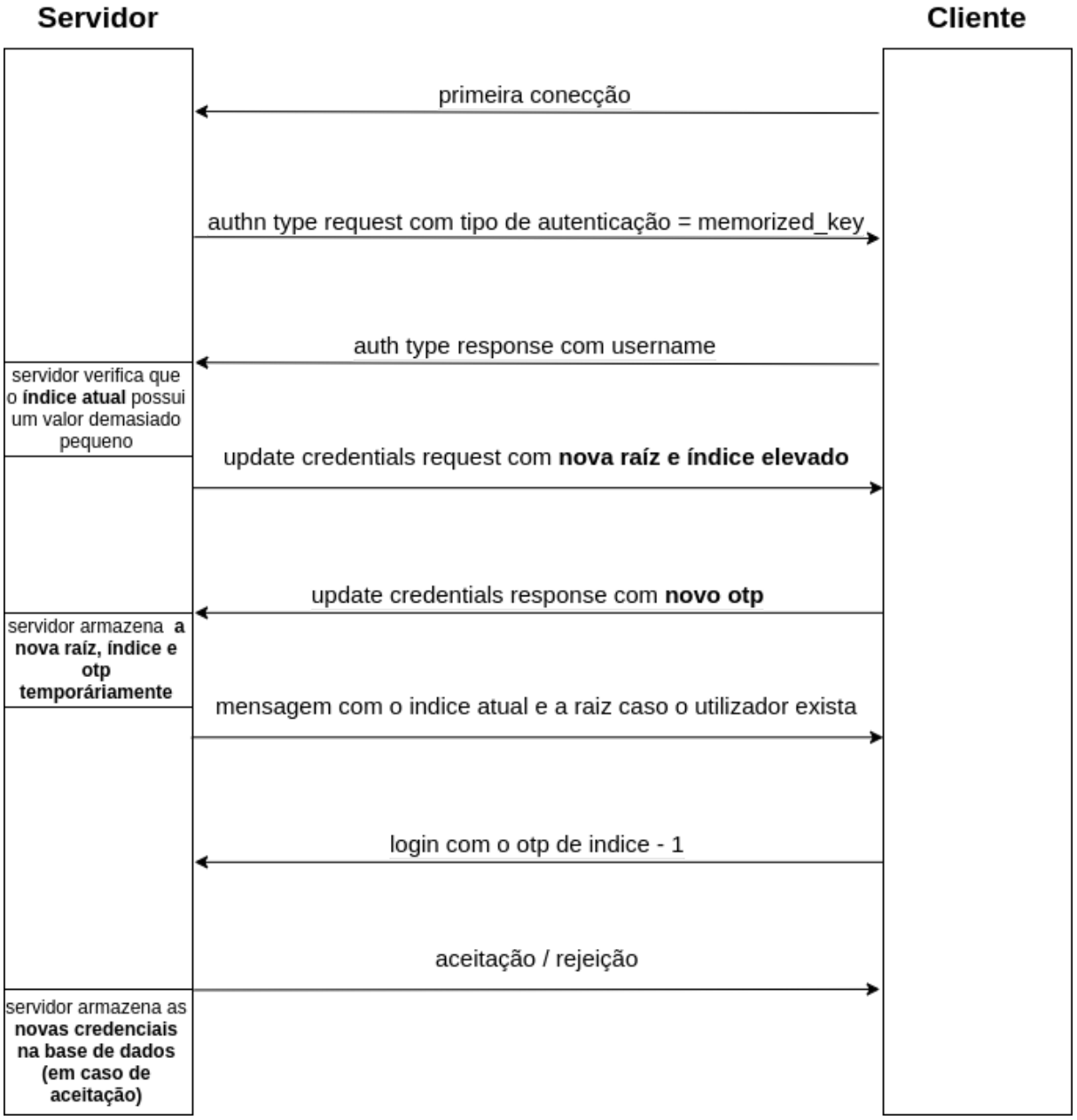
1) Para um utilizador poder ser autenticado, este tem de possuir credenciais na base de dados do servidor, isto é, o valor da raíz, do índice atual e do otp do índice atual a si associados têm de estar armazenados no lado do servidor. Desta feita, o script de python generate_credentials_new_user.py foi criado para possibilitar registar utilizadores, do lado do servidor, usando os atributos referidos e uma *password* secreta do utilizador. Este armazena cada um dos três atributos recebidos em três ficheiros binários diferentes, cujo nome está associado ao nome do utilizador do respectivo cliente. Importante referir que para gerar o valor de otp, é aplicada n vezes uma função de hash à raiz, sendo que a primeira dessas sucessivas aplicações usa a *password* memorizada do cliente.
2) Para o utilizador se autenticar posteriormente, necessita apenas de receber do servidor a raíz a ele associado e o índice que representa o número de funções de hash que foram aplicadas sobre o valor do otp corrente que o servidor possui. Desta forma, o cliente calcula o valor de otp de índice - 1. Este valor é impossível de ser replicado, já que para se calcularem as sucessivas hashes, como já foi explicado, é usada a palavra passe secreta do utilizador. No final, o cliente envia este valor ao servidor.
3) O servidor ao receber o valor de otp de índice - 1 calcula mais uma hash sobre este valor, de forma a confrontar o resultado com o otp que ele possuir armazenado. Caso os valores sejam iguais, o cliente é autenticado com sucesso e o servidor atualiza a sua base de dados, armazenando desta vez o novo valor de otp e o novo índice, de forma a que só o referido cliente se possa autenticar usando estas credenciais.

Contudo, este método descrito por si só não é robusto o suficiente, já que apresenta uma falha: quando o valor de índice chega a 0, o cliente não se pode mais autenticar. De forma a resolver esta situação crítica, construímos um “subprotocolo” para complementar o já existente:

1) Antes de enviar um desafio, o servidor verificar se as credenciais do utilizador com o *username* indicado pelo cliente, sendo que, caso chegue à conclusão que o índice atual é demasiado reduzido, envia uma mensagem ao cliente para este fazer atualização das credenciais, mesmo antes deste fazer o login. É importante que esta mensagem seja enviada antes de ser feito qualquer login, porque caso contrário, pode haver por exemplo uma falha na comunicação após o login, contribuindo isso para a redução do índice atual, levando a que este seja cada vez mais reduzido, caso este tipo de erros se verifique continuamente, originando o problema que tentamos solucionar com este novo protocolo, o que não é, de todo, o pretendido.
2) O cliente recebe a mensagem com pedido para atualizar as credenciais, isto é, recebe uma nova raíz e um índice suficientemente elevado (os dois valores enviados, obviamente, pelo servidor), sendo que gera um novo valor para o otp a partir destes e o envia para o servidor.

3) O servidor armazena estas novas credenciais temporariamente, até verificar a autenticidade do cliente. Desta forma, todos os procedimentos descritos anteriormente, desde a mensagem inicial do servidor com a raiz e índice atuais do cliente até à conclusão do login são realizados. No final, caso o login tenha sido realizado com sucesso, o servidor armazena as novas credenciais do utilizador, tendo sido por isso resolvido o problema descrito com sucesso.

Abaixo, podemos ver um esquema simplificado dos dois protocolos em funcionamento:



2.3. Exemplo de execução

2.3.1. Cliente

```
user@pc:~$ python3 client.py some_file -r
...
2019-12-15 14:02:52 vm root[3395] INFO First connection with the server
User name: escaleira
2019-12-15 14:02:57 vm root[3395] INFO Loging in
Password:
                                        ....
```

2.3.2. Servidor

```
user@pc:~$ python3 server.py
...
2019-12-15 13:50:57 vm root[3321] INFO Sending authentication type
2019-12-15 13:51:02 vm root[3321] INFO Sending challenge
2019-12-15 13:51:05 vm root[3321] INFO Loging in
2019-12-15 13:51:05 vm root[3321] INFO User logged in with success! Credentials updated.
                                        ....
```

3. Mecanismo de controlo de acesso

3.1. Explicação do protocolo

Após autenticar o cliente, o servidor procede à verificação dos acessos que esse utilizador possui. Para isso, acede a um ficheiro em *json*, onde se encontram os vários acessos de cada utilizador. Desta forma, o servidor verifica se o acesso à transferência de ficheiros por parte do utilizador está definido como sendo *True* ou *False*, pelo que, caso seja o primeiro, significa que o utilizador tem acesso para transferir ficheiros para o servidor. Sendo assim, nesse caso, o servidor envia uma mensagem de sucesso ao utilizador no final do login. Caso contrário, uma mensagem de rejeição será enviada.

É de notar também que no caso da autenticação usando-se o cartão de cidadão, que a seguir irá ser demonstrada, cada utilizador é identificado nesta base de dados através do seu número de identificação civil, já que este é único e facilmente obtido através do certificado de autenticação do cartão de cidadão.

Para facilitar a manipulação dos acessos dos utilizadores, foi criado o script *access_manager.py*, que permite de uma forma facilitada adicionar, remover e alterar acessos e visionar os já existentes.

3.2. Exemplo de execução

3.2.1. Servidor

```
user@pc:~$ python3 server.py
...
2019-12-15 13:50:57 vm root[3321] Verifying access
2019-12-15 13:51:02 vm root[3321] User has access to transfer files
                                        ....
```

4. Protocolo para autenticação através do cartão de cidadão

4.1. Explicação do protocolo

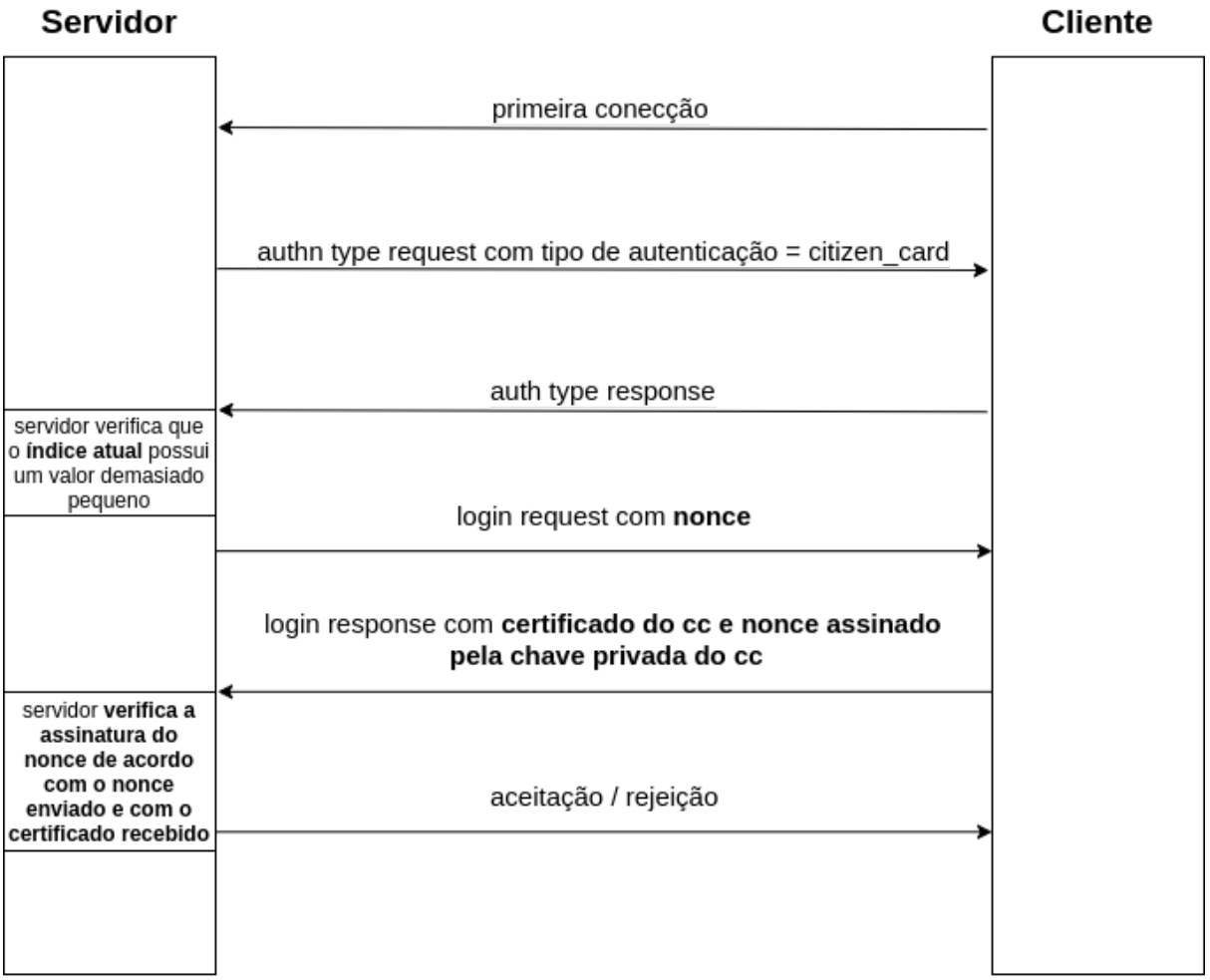
Para a criação deste protocolo, baseamo-nos na autenticação por desafio resposta com utilização de *smart cards*. Desta forma, o protocolo pode ser descrito da seguinte forma:

1) Tal como no primeiro protocolo descrito, inicialmente o cliente envia uma mensagem ao servidor para iniciar a conexão e de seguida este envia o tipo de autenticação a ser feito. A diferença é que neste caso, o tipo de autenticação é usando o cartão de cidadão.
2) O cliente, ao receber o tipo de autenticação, responde ao servidor com uma mensagem de aceitação.
3) O servidor criar um nonce, ou seja, um número random, de 64 bits, que envia para o cliente.
4) O cliente, ao receber o nonce, assina este usando o cartão de cidadão, sendo que para o fazer tem de introduzir o pin secreto deste. De seguida, quer o nonce assinado, quer o certificado de autenticação do cartão de cidadão são enviados para o servidor.
5) O servidor, ao receber estas duas entidades, realiza as seguintes tarefas, pela ordem aqui apresentada:
    a) Constrói a cadeia de certificados que termina no certificado recebido. Para isso, verifica qual o *issuer* e o *subject* de cada certificado que possui armazenado no seu disco e do certificado recebido. Ao construir a cadeia, já é feita a validação dos *common names* dos certificados, já que o nome do *issuer* dum certificado tem de ser igual ao nome do *subject* do certificado que fica acima dele na cadeia.
    b) Verifica se a cadeia de certificados foi concluída com sucesso, isto é, se foi possível chegar até um certificado autoassinado. Caso contrário, a cadeia de certificados é inválida, sendo que responde ao cliente com uma mensagem de insucesso e a ligação entre os dois termina.
    c) Com a cadeia de certificados formada, o servidor verifica se cada um dos certificados é válido. Para isso, verifica, por ordem, as seguintes propriedades dos certificados: o propósito de cada um, a validade, o estado de revocação e, por fim, as assinaturas de cada certificado, isto é, se cada certificado foi de facto assinado pelo seu *issuer*. Caso uma destas verificações

resulte na rejeição do certificado, o servidor dá por terminada a ligação com o cliente, já que o login é inválido, e nenhuma das verificações seguintes é feita. Desta forma, as verificações menos dispendiosas computacionalmente e que costumam ser objeto de erro foram as primeiras a serem feitas.

d) Com os certificados verificados, o servidor passa a verificar a assinatura do nonce. Para isso, usa o certificado que recebeu e o nonce que criara e enviara para o cliente. Em caso de sucesso, o cliente está autenticado com sucesso.
e) Apesar do utilizador está autenticado com sucesso, isso não significa, tal como vimos no ponto 3 deste relatório, que o cliente tenha acesso para transferir ficheiros para o servidor. Desta feita, o servidor obtém o número de identificação civil do utilizador e verifica se este se encontra na base de dados de acessos e, caso isso se verifique, se o utilizador identificado por esse id tem o acesso necessário para transferir ficheiros. Se possuir, o servidor envia uma mensagem de sucesso ao cliente a indicar que pode prosseguir, caso contrário envia uma mensagem a indicar que não possuir permissões e a ligação termina.

Para uma percepção facilitada e simplificada do que foi explicado, o seguinte diagrama demonstra este protocolo:


Servidor
Cliente
primeira conecção
authn type request com tipo de autenticação = citizen_card
auth type response
servidor verifica que o índice atual possui um valor demasiado pequeno
login request com nonce
login response com certificado do cc e nonce assinado pela chave privada do cc
servidor verifica a assinatura do nonce de acordo com o nonce enviado e com o certificado recebido
aceitação / rejeição

4.2. Exemplo de execução

4.2.1. Cliente

```
user@pc:~$ python3 client.py some_file -r
...
2019-12-15 14:02:52 vm root[3395] INFO First connection with the server
2019-12-15 14:02:57 vm root[3395] INFO Loging in
                                        ....
```

Como se pode observar, neste caso não é pedido o nome do utilizador e a sua palavra passe, já que é tudo tratado pelo cartão de cidadão (apesar de aqui não ser possível observar, na realidade é aberta uma janela para o utilizador inserir o PIN do cartão de cidadão).

4.2.2. Servidor

```
user@pc:~$ python3 server.py --authentication=citizen_card
...
2019-12-15 13:50:57 vm root[3321] INFO Sending authentication type
2019-12-15 13:51:02 vm root[3321] INFO Sending challenge
2019-12-15 13:51:05 vm root[3321] INFO Loging in
2019-12-15 13:51:05 vm root[3321] INFO User logged in with success! Credentials updated.
                                        ....
```

5. Protocolo de autenticação do servidor

5.1. Explicação do protocolo

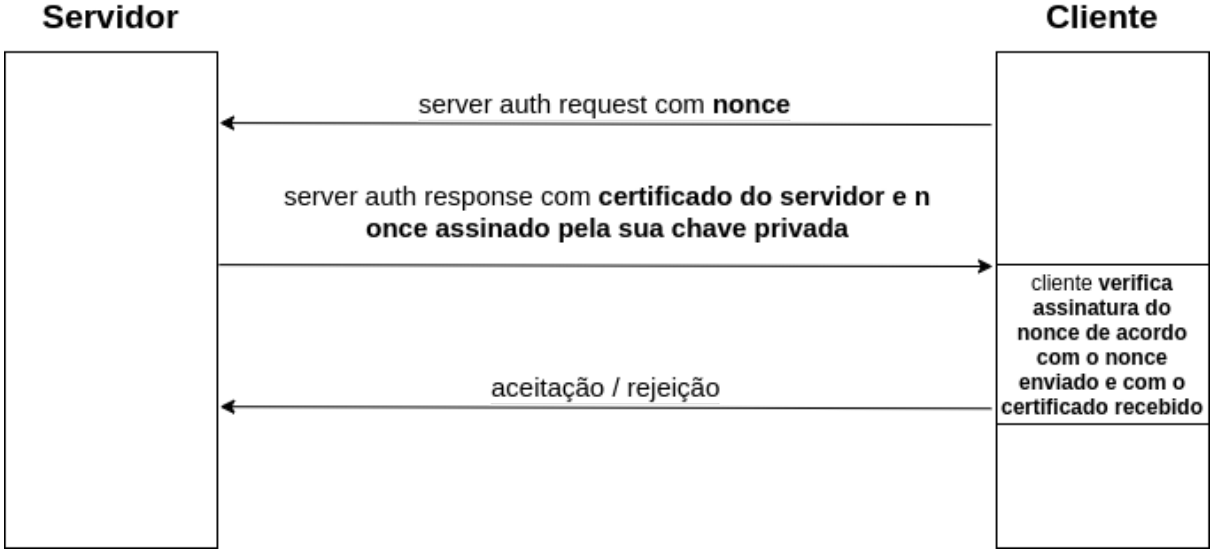
Foi-nos também proposto neste trabalho criarmos um protocolo que permitisse que o servidor fosse autenticado pelo cliente, de forma a que este último tivesse plena confiança de que de facto estava a transferir ficheiros para a entidade correta. O protocolo funciona da seguinte maneira:

1) Inicialmente, o cliente envia mensagem ao servidor para iniciar a conexão, a pedir que o servidor se autentique, mesmo antes do cliente efetuar qualquer login. Nesta mensagem, para que o servidor se possa logar, é enviado também um valor aleatório de 64 bits, um nonce, para que ele se possa logar através de desafio resposta.
2) Quando o servidor recebe esta mensagem, assina o nonce com a sua chave privada e, concluído esse passo, envia o resultado de volta para o cliente em conjunto com o seu certificado de autenticação.
3) O cliente recebe o certificado e a assinatura do nonce e faz as mesmas verificações do certificado e da assinatura do nonce já descritas no protocolo apresentado atrás. Caso a verificação seja um sucesso, o cliente aceita o servidor como sendo de confiança e continua a comunicação com este.

Para a criação deste protocolo, foram essenciais dois processos:

- Criação de dois certificados através da ferramenta XCA, onde criamos um certificado CA, ou seja, autoassinado e outro certificado assinado por este.
- Criação dum domínio para devolver o CRL do certificado CA criado.

Mais uma vez, apresentamos um esquema simplificado do protocolo discutido:



5.2. Exemplo de execução

5.2.1. Cliente

```
user@pc:~$ python3 client.py some_file -r
...
2019-12-15 14:02:52 vm root[3395] INFO First connection with the server
2019-12-15 14:02:57 vm root[3395] Verifying server
                                        ....
```

5.2.2. Servidor

```
user@pc:~$ python3 server.py
...
2019-12-15 13:50:57 vm root[3321] Authenticating server
                                        ....
```

6. Conclusão

Dada a conclusão dos protocolos descritos, foi necessário agrupá-los os 4 no trabalho já desenvolvido para o projeto 2. Desta forma, foi essencial descobrir qual a ordem em que cada um ia ser usado na comunicação, para que esta fosse o mais segura possível. Sendo assim, chegamos à conclusão de que o primeiro protocolo a ser usado deveria ser o de autenticação do servidor, já que caso contrário, um atacante poderia enviar a raíz e o índice que seriam utilizados no próximo login feito entre o cliente e o servidor e, desta forma, o cliente retornar as credenciais suficiente para o atacante fazer login no servidor.

Para além disso, o protocolo de controlo de acessos foi implementado de forma a se o último a ser executado, porque seria desnecessário verificar se um utilizador tem acesso à transferência de ficheiros antes deste fazer login. O login poderia falhar e esta verificação teria feita em vão.

Desta forma, o seguinte esquema põe em evidência a ordem em que foi aplicado cada um dos protocolos: