Secure Authentication

Establishing authentication in a secure communication channel

Trabalho realizado por:
Tomás Costa - 89016
João Marques - 89234

Índice

Introdução

O projeto consiste no desenho e implementação de um protocolo que permita a autenticação mútua entre dois pontos, sendo que o canal seguro já é fornecido pelo trabalho prático anterior da criação de um canal seguro de comunicações usando chaves simétricas. Pretende-se que seja possível trocar um ficheiro entre o cliente e o servidor usando o protocolo, caso o servidor considere que o utilizador tem essas permissões. O utente pode-se autenticar com senhas diretas ou com o cartão de cidadão. O servidor deverá conseguir provar a sua identidade, evitando-se ataques de MiTM ou impersonação.

Considerações

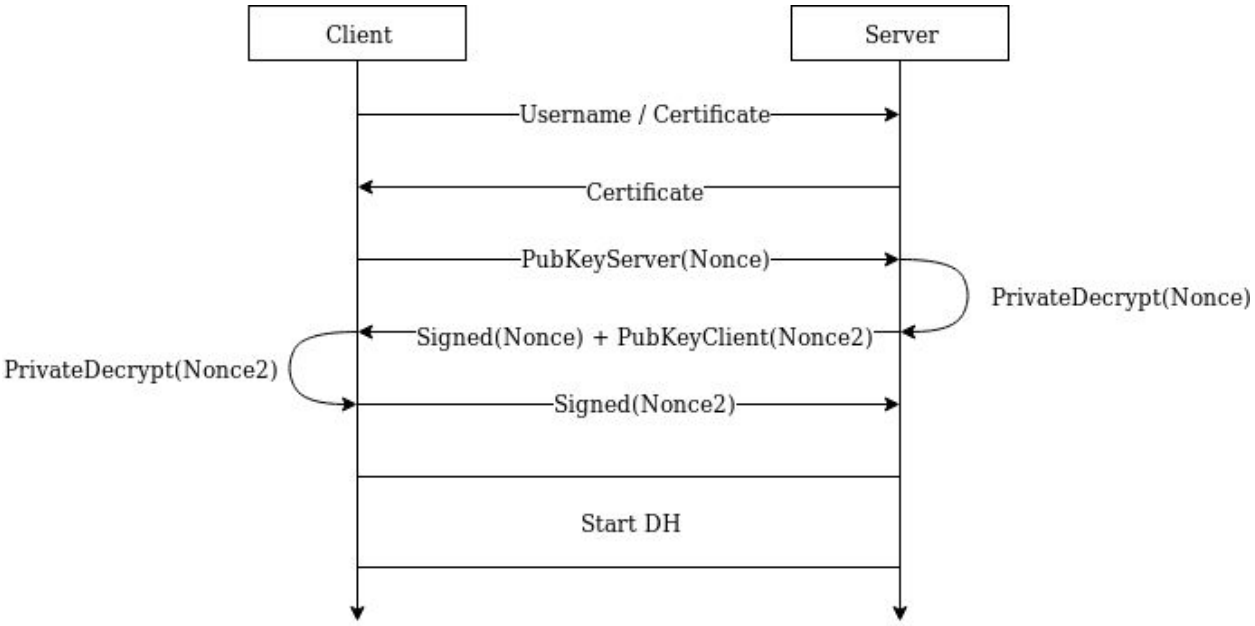
A estrutura inicial já continha o estabelecimento de um canal de comunicações seguro que foi criado para o projeto anterior. Sendo que este foi utilizado para partilhar os certificados e dados de autenticação tanto do cliente como do servidor.

O trabalho consiste em quatro grandes objetivos que, englobam tanto o desenho como a implementação, estes são:

- Protocolo para autenticação de utentes através da apresentação de senhas
- Mecanismo para controlo de acesso
- Protocolo para autenticação de utentes através do cartão de cidadão
- Protocolo para autenticação do servidor através de certificados X.509

Arquiteturas Inicias

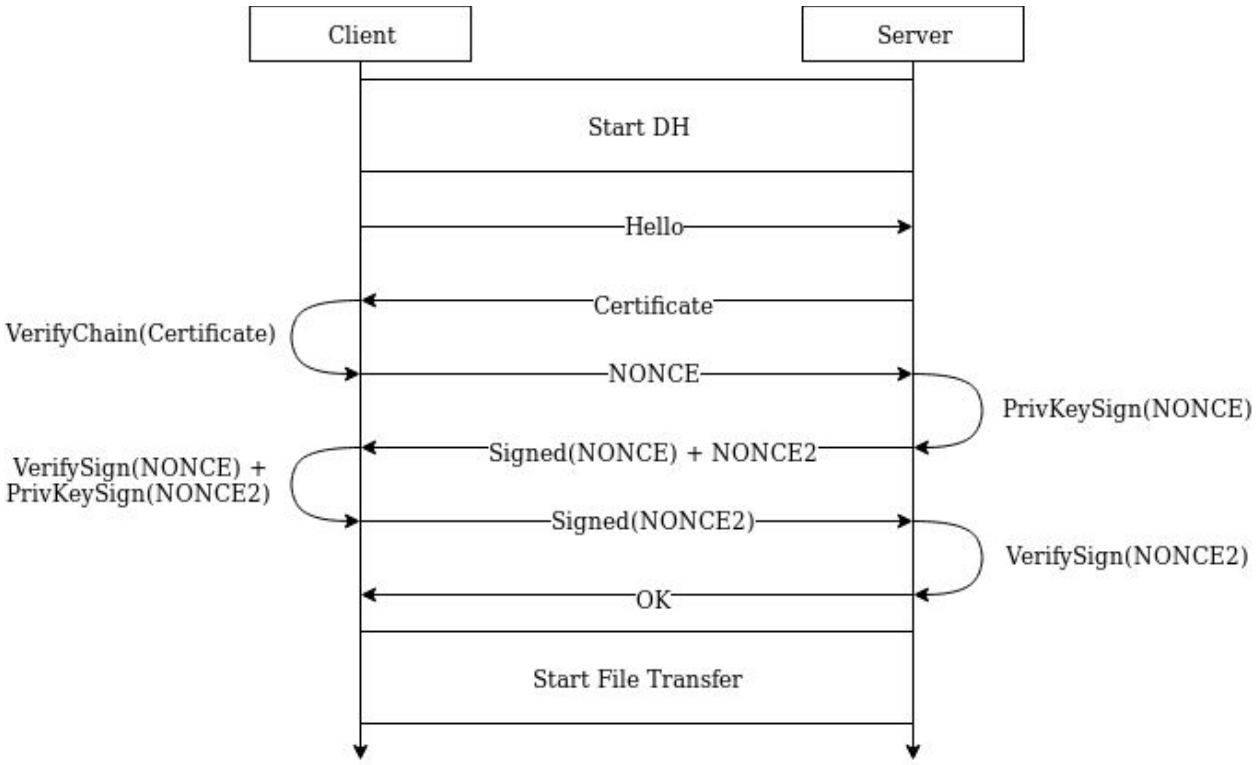
Antes de começarmos a implementar e, visto que havia a liberdade de desenhar qualquer arquitetura, pensamos em várias arquiteturas que foram posteriormente descartadas por vários motivos.
Apesar de termos definido várias arquiteturas, é mais relevante mencionar a penúltima e o porquê de não a termos utilizado.
A arquitetura era a seguinte:



Sendo que existem vários erros na arquitetura, os de maior relevância são: a partilha dos dados do username em plaintext antes de verificar o servidor; uso impróprio do DH (este pode já ser usado no início para os dados serem trocados com chaves simétricas); não há garantia de autenticidade, mas sim de confidencialidade, visto que estamos a encriptar com a pública do remetente.
A correção destes erros levou-nos a nossa arquitetura final, representada no próximo tópico.

Arquitetura Final

A arquitetura final é a seguinte:

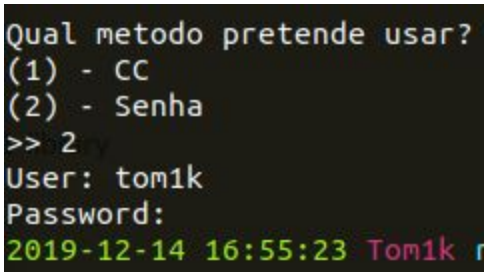


Sendo que começamos por negociar as chaves simétricas e, posteriormente, passamos à autenticação encriptada com essas mesmas chaves.
O cliente contacta o servidor com um HELLO, o servidor envia o seu certificado e o cliente envia ou o certificado do seu Cartão de Cidadão ou os dados de login no servidor, depois é feito um desafio-resposta para cada entidade e após estas verificações todas (assumindo que o cliente tem acesso de escrita), o cliente pode começar a transferir o ficheiro.

Protocolo para autenticação de utentes através de senhas

O mecanismo de autenticação por senhas, passa por o utilizador inserir dois tipos de dados: username e password.
Caso estes dados não se encontrem na base de dados de utilizadores, o processo vai ser terminado (explicado mais detalhadamente na secção de controlo de acesso). Um exemplo teste é:

- USERNAME: tom1k
- PASSWORD: 123hiperseguro



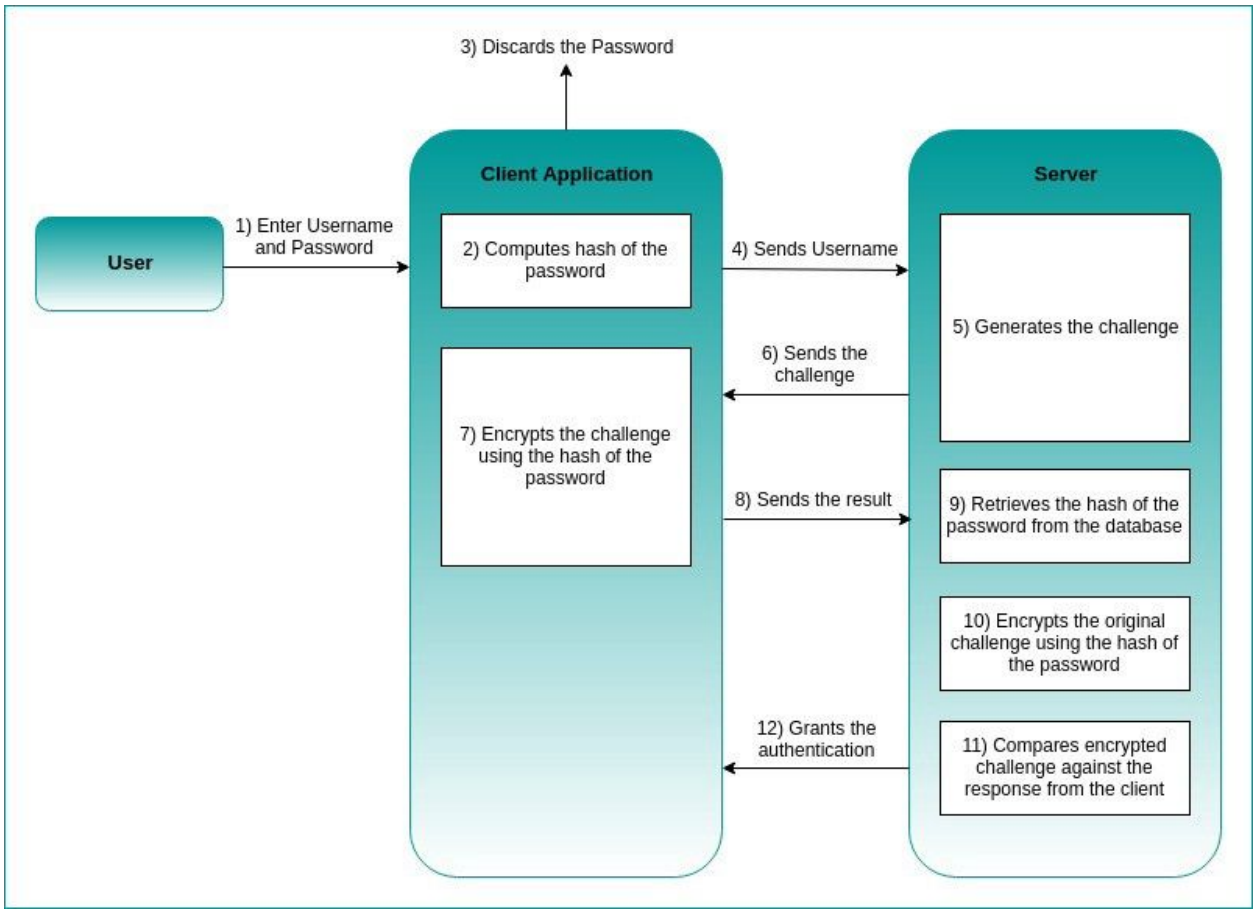
Visto que não temos par chaves associado ao username, o que fazemos e enviar um NONCE do cliente ao servidor, o servidor responde assinando com a sua privada e manda também um NONCE (NONCE2).

O cliente, por sua vez, valida a resposta ao NONCE vinda do server e responde ao NONCE2 da seguinte forma:

1. Hash da password inserida pelo utilizador
2. HMAC do hash da password em 1.
3. Assina NONCE com HMAC anterior

O servidor quando recebe a resposta, vai repetir o processo de hashing com a password que tem armazenada no sistema e, vai comparar a sua resposta ao desafio com a resposta do user ao desafio, caso sejam iguais e porque o cliente está a tentar autenticar com a password corresponde a sua conta e portanto pode ser validada a autenticação.
O seguinte processo foi implementado conforme descrito em: https://medium.com/@nipunadilhara/challenge-response-authentication-protocol-850925f50813

Segue o diagrama do desafio:



Mecanismo para controlo de acesso

Para o mecanismo de controlo de acesso, está implementado um ficheiro que serve como base de dados de utilizador, denominado de 'userdb' que segue o seguinte esquema:

```
tomas:BI843122680:123:AUTH_WRITE
tom1k:BI984226910:123hiperseguro:AUTH
joao:BI300923620:passsupersegura1234:AUTH_WRITE
```

A notação do ficheiro é a seguinte: um utilizador por linha com as suas informações delimitadas por ':' .
O seu username na posiçao 0, o seu número do BI na posiçao 1, password na posição 2 e as suas permissões na posição 3.
Existem 2 tipos de permissões: AUTH (permissão para fazer login) e AUTH_WRITE (permissão para login e escrita).
E apesar de o sistema estar construido apenas com a funcionalidade da escrita de ficheiros, esta notação permite escalabilidade e adaptabilidade futura.

Protocolo para autenticação de utentes através do cartão de cidadão

Quando o utilizador opta pelo login através do CC, após a validação do certificado do servidor e da validação do challenge, o cliente envia o certificado de autenticação do Cartão de Cidadão para o servidor, que por sua vez o valida também.

Para este processo, precisamos de carregar os certificados do CC como certificados intermédios no servidor. Também carregamos as CRLs e Delta-CRLs específicas para cada certificado, através das extensions do certificado.

Após isso, o processo é similar à autenticação com password, com a diferença que o challenge é assinado pela private key do CC, o valor que identifica o utilizador é o seu número de CC, por sua vez associado a cada user no lado do servidor. A public key do CC, obtida pelo servidor através do certificado, é depois usada para verificar a assinatura do challenge.

Protocolo para autenticação do servidor através de certificados X.509

Para autenticaçao do servidor, este possui um certificado, por sua vez assinado por uma CA Root (com certificado auto-assinado), que foi gerado por nós. Por essa razão, este certificado Root precisa de estar registado do lado do cliente como raiz confiável.

Para facilitar todo este processo, colocámos todo o código relacionado com estas verificações na classe **Certificate_Validator**. Esta classe recebe 3 (listas de) diretórios: localização(ões) dos certificados raiz, dos certificados intermédios, e onde colocar as CRLs que serão descarregadas dinamicamente.

Quando recebemos um certificado, começamos por carregar as respetivas CRLs para uma lista. Depois disso, construímos a cadeia de certificação, procurando o issuer de cada certificado na lista de CAs que temos carregada, até encontrar o certificado raiz. Com a cadeia contruída, validamos a mesma, verificando a assinatura de cada certificado com a chave do issuer, e comprovando que o certificado não se encontra em nenhuma CRL (já asseguramos que estão todos válidos pois, caso a sua data de expiração já tenha passado, não chega a ser carregado para a nossa lista).

Caso a cadeia de certificação seja validada, a função devolve True, e podemos seguir com o processo de autenticação.

Mecanismos de segurança extra

Para completar o nosso trabalho ainda mais um pouco, adicionamos um mecanismo que pretende replicar o 2FA (Two Factor Authentication), quando o modo de autenticação é por senha. Logo, temos escolha de múltiplas perguntas, restringimo-nos à pergunta "Qual o nome do seu primeiro animal de estimação?"
Portanto, adicionamos um campo a base de dados de utilizadores para a resposta a esta pergunta.

```
tomas:BI718625680:123:AUTH_WRITE:caozitos
tom1k:BI718625690:123hiperseguro:AUTH:fernando_pessoa
joao:BI167829510:passsupersegura1234:AUTH_WRITE:sasha
```

O ideal seria termos um gerador aleatório de códigos para outro dispositivo, no entanto não conseguimos aplicar este gerador visto que haviam muitas restrições, que o impossibilitavam.

A autenticação passou a ser feita assim: O utilizador insere o username, a sua pass, mas, no final insere também a resposta à pergunta:

```
Qual metodo pretende usar?
(1) - CC
(2) - Senha
>> 2
User: tom1k
Password:
Nome do primeiro animal de estimação?
>> fernando pessoa
```

Que gera o resultado:

```
Is secret question correct? ->  True
```

(visto que que a resposta do username 'tom1k' é fernando_pessoa)

Conclusão

Com a conclusão deste trabalho, estamos bastante satisfeitos com o que realizamos, pois, permitiu-nos desenvolver: o raciocínio, através da refutação das diferentes arquiteturas iniciais, criatividade da solução, visto que a liberdade era grande e o esquema final não tinha apenas uma solução, e, mais importante, permitiu-nos aprofundar conceitos lecionados nas aulas de Segurança Informática em Organizações.

Bibliografia

Fontes mais relevantes à realização deste trabalho:

```
* [Slides 5 and 6](https://joao.barraca.pt/teaching/sio/2019/)
* [Cryptography.io](https://cryptography.io)
* [StackOverflow (Several Doubts)](https://stackoverflow.com/)
```