


# Relatório

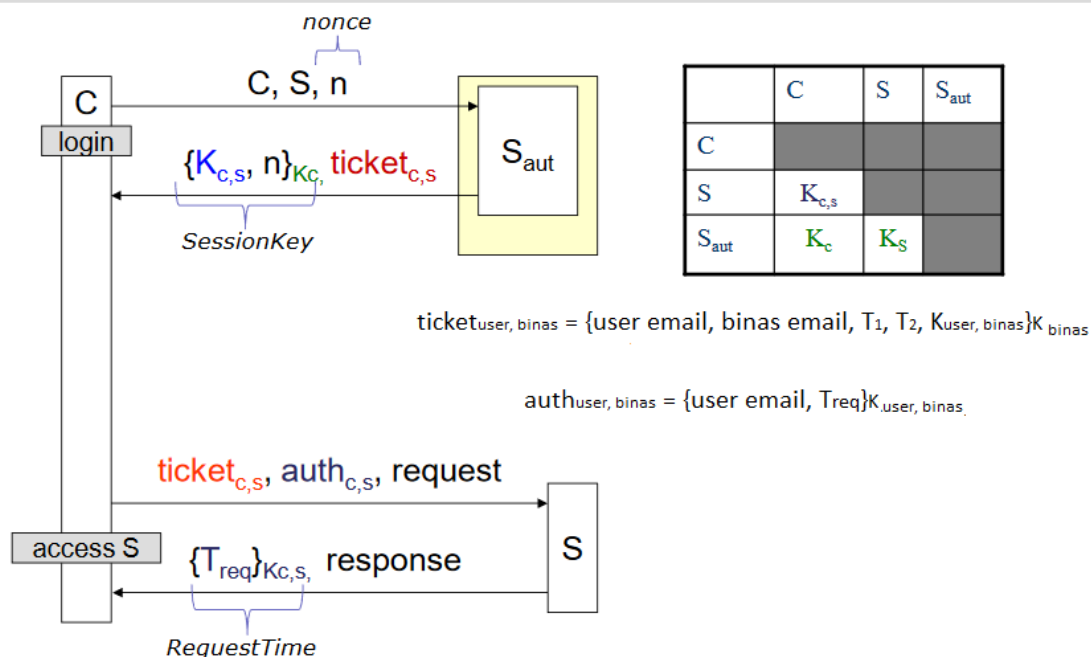
## Segurança com Kerberos

Fotografia			
Nome	Filipa Marques	David Coimbra	Lúcia Lisboa
Número	57842	84708	84738

**GitHub URL:** <https://github.com/tecnico-distsys/T07-SD18Proj>

# Aplicação do protocolo Kerberos

## Kerby: simplified Kerberos



- O cliente identifica-se no servidor de autenticação Kerberos, indicando o servidor com que pretende comunicar e um número de identificação único.
- O servidor de autenticação responde enviando uma chave de sessão, conhecida pelo cliente e servidor, encriptada com a chave do cliente. Envia também um ticket, encriptado com a chave do servidor. Através do número de identificação único, é detetada a presença de *replay attacks*. É através do *ticket* que o servidor conhece a chave de sessão.
- O cliente deve descriptar a chave de sessão com a sua própria chave. Só terá sucesso se a chave tiver sido gerada através de uma password válida.
- Ao fazer um pedido ao servidor, o cliente cria um autenticador que envia junto do ticket recebido. O autenticador inclui o identificador do cliente e um *timestamp* da altura da sua criação, e está encriptado com a chave de sessão. O *timestamp* serve como identificador único.
- É enviado também um MAC gerado a partir do pedido e da chave de sessão, de modo a verificar a integridade da mensagem.
- O servidor deve descriptar o *ticket* com a sua chave para obter a chave de sessão que é necessária para descriptar o autenticador.
- Se o ticket e autenticador forem válidos e não for detetada uma intrusão, o servidor envia, junto da resposta, o mesmo *timestamp* que recebeu no autenticador, encriptado com a chave de sessão. É gerado também um MAC através do pedido e comparado com o MAC recebido, para garantir a integridade da mensagem.
- O acesso ao servidor é controlado comparando o identificador presente no pedido com os presentes no *ticket* e no autenticador.
- O cliente compara a *timestamp* que criou com a que recebeu na resposta do servidor. Se forem diferentes, é detetado um *replay attack*.

## Handlers

**KerberosClientHandler** – Ao enviar: após a autenticação do utilizador, junta-se ao *header* da mensagem SOAP um elemento para o *ticket* e outro para um autenticador. Este último contém um *timestamp* correspondente ao instante da sua criação.

Ao receber: Compara o *timestamp* previamente criado com o recebido da resposta do servidor. Caso sejam diferentes, ocorreu um *replay attack*.

**KerberosRequestTimeHandler** – O timestamp, incluído no autenticador recebido do cliente, é encriptado com a chave de sessão e adicionado à mensagem de resposta.

**KerberosServerHandler** - Ao receber a resposta do cliente, são recolhidos o *ticket* e o autenticador da mensagem SOAP. Após os desencriptar com as respetivas chaves, verifica-se se o ticket é válido para a sessão atual e se está consistente com o autenticador.

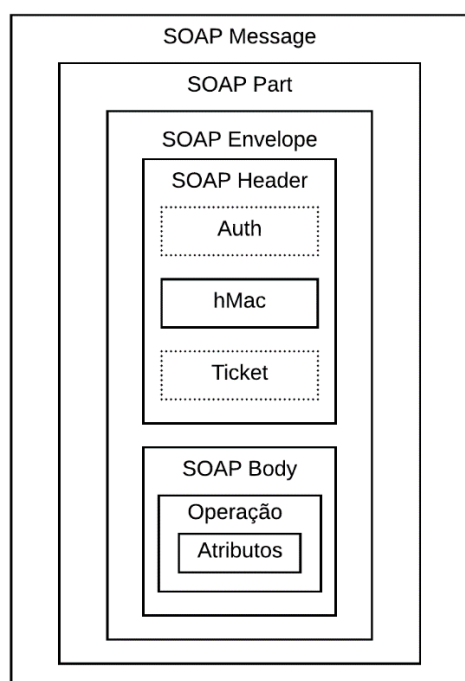
**MacClientHandler** - Aplica-se uma função de resumo à concatenação da mensagem com a chave da sessão, criando um HMAC. Este HMAC é incorporado no *header* da Soap Message.

**MacServerHandler** - É recolhido do SOAP *header* o HMAC criado pelo cliente. Após calcular o HMAC da mensagem recebida, este é comparado com o HMAC recolhido, rejeitando o pedido se não forem iguais.

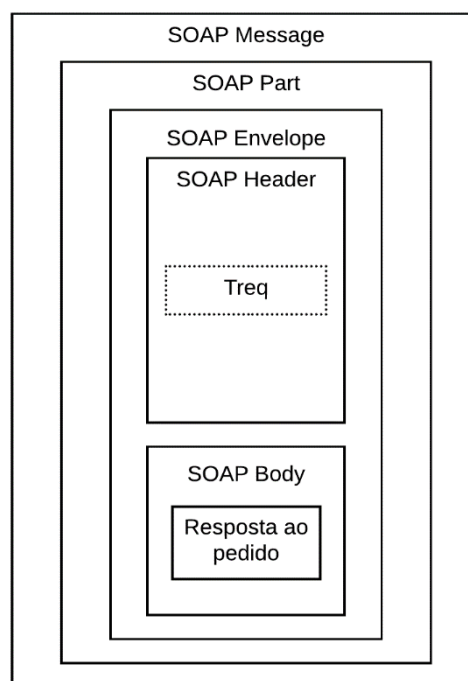
**BinasAuthorizationHandler** – Verifica se o utilizador que faz um pedido corresponde àquele que está autenticado, comparando os identificadores presentes no pedido, no *ticket* e no autenticador.

**MaliciousHandlerMACTest** – Simula um ataque ao sistema. Adiciona texto à mensagem, danificando a sua integridade.

## Descrição das mensagens SOAP



Formato da mensagem entre o cliente(binás-cli) e o servidor(binás). O Auth e o Ticket estão **encriptados**, respetivamente, com a chave de sessão e a chave do servidor.



Formato da mensagem entre o servidor(binás) e o cliente(binás-cli). O Treq(Time Request) está **encriptado** com a chave de sessão.