**A48 – Terceiro Entrega do Projeto de Sistemas Distribuídos**

 83420 – Alexandra Figueiredo

 83443 – Denis Voicu

 83520 – Mariana Loureiro

**GitHub:** <https://github.com/tecnico-distsys/A48-SD18Proj>

**Descrição da solução**

A abordagem usada tem base no algoritmo *Kerberus*. Em seguimento da entrega anterior, é assumido que existem três réplicas *Station*, assim como uma instância *Binas* a correr, e também os dois clientes, *binas-ws-cli* e *station-ws-cli*.

O cliente começa por se autenticar com o servidor *Kerby* ao requisitar um *ticket*, com a sua identificação (*email*), a identificação do servidor a contactar (*email* do *Binas*), um valor *nounce* (um valor único e aleatório que servirá depois para efeitos de validação, e a duração da sessão. Recebe duas estruturas de dados, uma *SessionKey* e um *Ticket*, em resposta. Tanto o cliente como o *Kerby* têm conhecimento da *password* do utilizador a que corresponde o *email* enviado, pelo que a chave do cliente (*Kc*) é conhecida por ambos, e pode ser usada para cifrar e decifrar a mensagem enviada.

O cliente valida o *nounce* devolvido na *SessionKey*, e, agora conhecendo a chave (também no *SessionKey*) que irá partilhar com o *Binas*, envia-lhe o *Ticket*, que não consegue abrir. O *Binas* decifra o *Ticket* com a sua chave *Ks*, que o *Kerby* conhece por ter acesso à *password* usada pelo *Binas* também. Passa a conhecer a chave partilhada *Kcs*, e verifica que o endereço do servidor no *Ticket* corresponde à sua identidade. Para ter a certeza da frescura da mensagem, os dois *timestamps* no *Ticket* devem ser inclusivos do momento atual.

Em adição ao *Ticket* (e ao pedido – *request* – propriamente dito), o cliente envia também uma estrutura de dados *Auth* ao *Binas*, que contém uma instância temporal e o seu *email*. O *Binas* valida que os *emails* (no *Ticket*, *Auth* e o *email* no *request* do cliente) são idênticos. Sendo tudo corretamente validado, envia então de volta para o cliente – novamente cifrado com a *Kcs*, e acompanhado da resposta ao pedido– o *timestamp* recebido no *Auth*, para que, do seu lado, o cliente valide a resposta por comparação com aquele que recorda enviar.

***Handlers***

Todo o processo acima é realizado pelos *handlers* do lado do cliente e do *Binas*. No processo de envio – dos dois lados – são primeiro feitas as ações referentes ao algoritmo *Kerberos*, e de seguida é aplicado um *MAC* à mensagem, cifrado com a chave partilhada *Kcs*. Neste segundo *handler*, a *string* obtida para garantir a incorruptibilidade da mensagem é simplesmente o *request*, a partir do qual tanto o cliente como o *Binas* devem gerar *MAC*s idênticos.

**Troca de mensagens**

De acordo com o *Quorum* *Consensus*, o cliente faz a operação para todos os servidores, e recebendo resposta de uma maioria deles, assume sucesso. Considerando qualquer tipo de exceção erro na resposta, e sabendo que o número de *stations* simultâneas é 3, se duas chamadas remotas devolverem corretamente, a operação foi bem-sucedida.

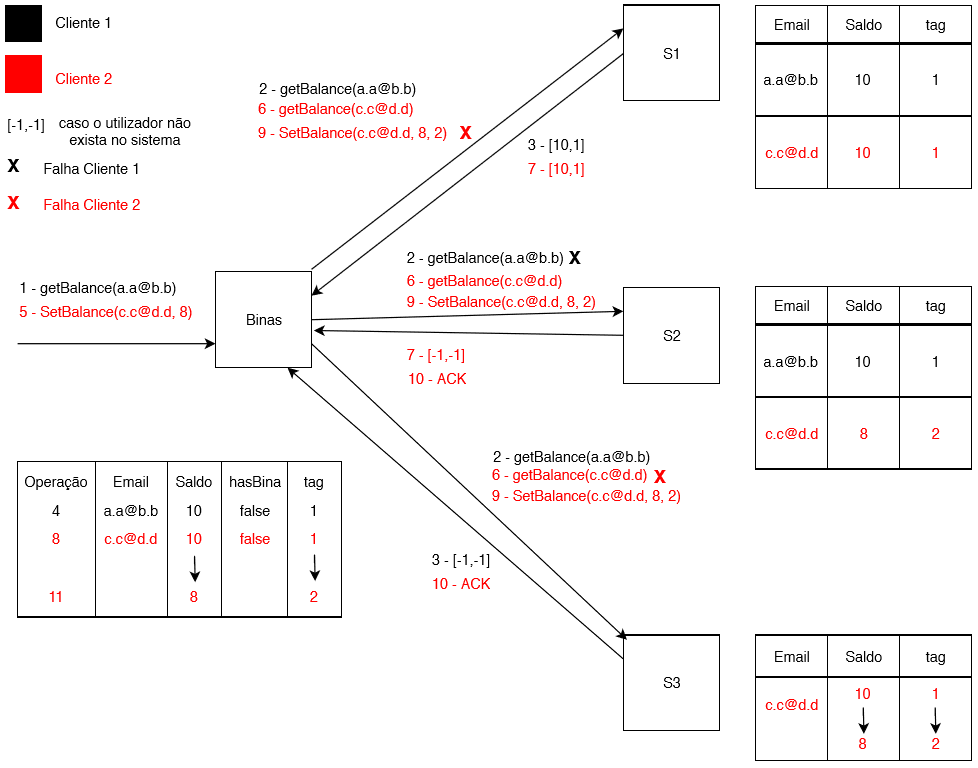
No caso do *getBalance*, a chamada enviará o email do utilizador e devolverá uma estrutura com dois valores – o saldo e a *tag*. Neste caso, é selecionado o valor cuja *tag* seja mais recente (maior *sequence* *number*). No caso do *setBalance*, é enviado o *email*, o novo saldo, e a *tag*, e o valor de retorno é *void*.

**Descrição da figura**

Assume-se que:

* *Station* S1 contém informação de 2 clientes: [a.a@b.b, 10, 1] e [c.c@d.d, 10, 1].
* *Station* S2 contém informação de 1 cliente: [a.a@b.b, 10, 1]
* *Station* S3 contém informação de 1 cliente: [c.c@d.d, 10, 1]

No início, o servidor *Binas* sofreu uma falha que resultou na perda da informação dos clientes. De seguida, é recebido um conjunto de instruções que estão numeradas cronologicamente. O cliente 1 envia uma instrução que inclui o *getBalance*(1). Como o *Binas* não tem informação sobre nenhum cliente, o binas vai obter essa informação através do envio de *getBalance*(2) às estações(S1,S2,S3). É obtida a resposta de apenas duas estações, uma vez que uma das chamadas perde-se. O *Binas* ao receber a resposta das 2 estações, compara as *tags* das repostas, escolhe a maior delas e regista essas informações(*email*, saldo, *tag*) no *Binas*(4). Caso o cliente não exista na *station* é retornado [-1,-1] como resposta.

O cliente 2 faz outra instrução que inclui o *setBalance*(5). Como o binas não tem as informações relativamente a esse cliente, é feito o *getBalance*(6) as estações. Pelo menos duas das estações respondem, e registam-se no *Binas* as informações relativas à resposta com a maior *tag*. De seguida, é enviado o *setBalance*(10) a todas as estações com os valores atualizados, ou seja, com o novo saldo e com a *tag* incrementada, e posteriormente, os valores são atualizados no *Binas*(11). Para o caso em que o servidor não tenha o utilizador com o referido *email*, então é criado esse utilizador na estação. Se já existir as informações deste, são atualizadas.