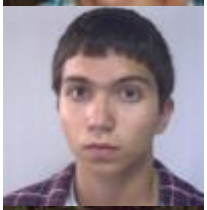
Sistemas Distribuídos 2015-2016

Grupo: A13

https://github.com/tecnico-distsys/A_13-project

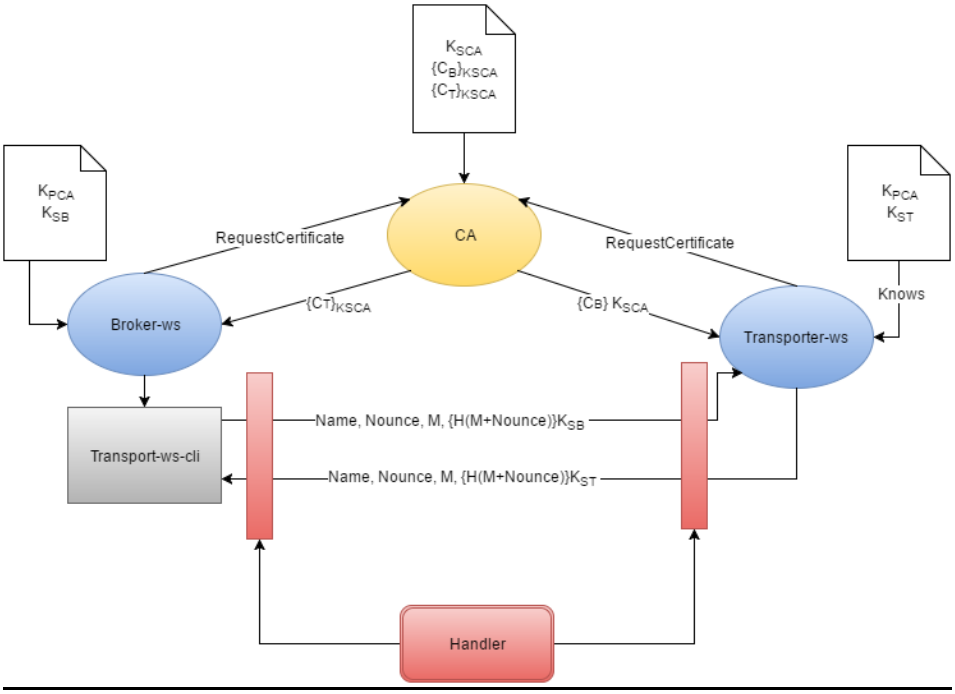
João Coelho
77983



Micael Batista
78941

Telma Correia
78572

Segurança



A figura acima descreve a solução de segurança implementada no nosso projeto.
Leia-se:
K_{PCA}: Chave publica da CA; **K_{SCA}**: Chave secreta da CA;
K_{SB}: Chave secreta do Broker; **K_{ST}**: Chave secreta do *Transporter*;
$\{C_B\}K_{SCA}$: Certificado do Broker assinado pela CA com a chave secreta;
$\{C_T\}K_{SCA}$: Certificado do *Transporter* assinado pela CA com a chave secreta;
{H(M+Nounce)}K_{SB}: *Digest* da mensagem + *nounce*, assinados com a chave secreta do *Broker*;
{H(M+Nounce)}K_{ST}: *Digest* da mensagem + *nounce*, assinados com a chave secreta do *Transporter*;

Tal como representado na figura, o Broker e o *Transporter* conhecem a chave pública da CA e as suas chaves secretas. A CA tem a sua chave secreta e ainda, certificados do Broker e *Transporter* assinados com a sua chave secreta. Esta distribuição foi feita manualmente, tal como sugerido pelo corpo docente.

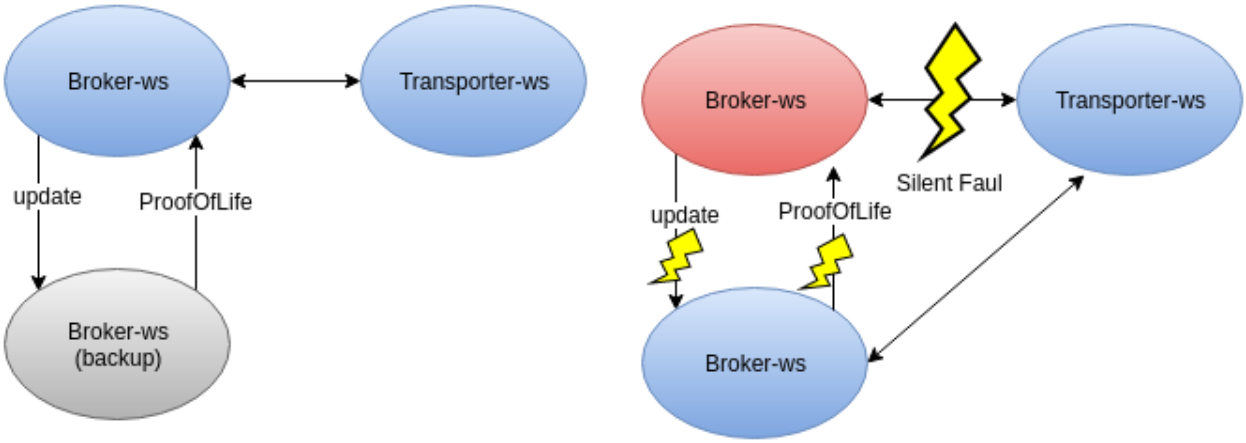
Quando o Broker envia uma mensagem, o *handler* faz um *Digest* com a mensagem e com o *nounce*. O *nounce* **garante que as mensagens não são reutilizadas**. Para gerar este nounce optámos por gerar números aleatórios pois garante que não é descoberto facilmente. O *digest* vai assinado com a chave secreta do *Broker,* (assinatura digital) para garantir a **autenticação do emissor**, a **integridade do conteúdo da mensagem** e o **não-repúdio**.

Quando o *Transporter* recebe a mensagem, o *handler* vai verificar se a mensagem não foi alterada comparando o *digest* recebido com um novo. Para desencriptar a mensagem recebida pede à CA a chave publica do emissor.

A comunicação entre o *Transporter* e o *Broker* é análoga à explicada acima.

No projeto temos apenas uma biblioteca de *Handlers* que é usada pelo *Broker* (*transporter-client*) e pelo *Transporter*.

Replicação



A figura acima representa a ligação entre os *Brokers* (primário e secundário) antes de acontecer uma falta (à esquerda).

Caso aconteça uma falta (à direita) as ligações assinaladas deixam de existir.

Considera-se implícita a existência de *Broker-Client* para comunicar com cada um dos *Brokers*.

De forma a assegurar a tolerância a faltas em caso de falha do *Broker*, foi criado um *Broker* secundário que corre paralelamente ao *Broker* primário. Ao iniciar o serviço, apenas o servidor principal se encontra registado no UDDI. De forma a manter a consistência de dados no caso de uma falha, ao executar qualquer função que altere os dados do *Broker* (*Request*, *View* & *Clear*) existe uma outra função *update* (da nova versão do *wsdl*) que vai do *Broker* principal para o secundário, que passa toda a informação necessária para manter o estado consistente.

O *Broker* secundário está sempre a postos na eventualidade do *Broker* principal falhar. Para tal, existe uma função *proofOfLife* que está constantemente a ser efetuada do servidor secundário para o primário. No caso de não receber uma resposta a tempo do lado do servidor primário, então este deve-se imediatamente registar-se no UDDI de forma a substituir o servidor primário que entrou em falta. No lado do cliente, o *front-end* deve reconhecer que houve uma falha no servidor primário e voltar a procurar um novo *Broker* no UDDI.