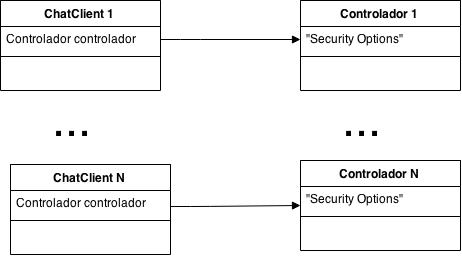
|  |
| --- |
| Universidade de Coimbra |
| Relatório STI |
| Trabalho Prático III |
|  |
| **Erbi Silva e João Batista** |
| **02/05/2015** |

|  |
| --- |
|  |

**Classe “Controlador”**

****

O sistema programado lança diversos clientes (“ChatClient”). Cada cliente terá um controlador que será responsável por controlar todas as acções que esse cliente precisa tais como, encriptação das mensagens, , manter a integridade das mensagens, entre outras, basicamente controla toda a segurança do cliente.

Classe “Mensagem”

A classe mensagem foi criada para transportar a mensagem encriptada, a chave, o ID de quem a envia e a “hash” para controlar se a mensagem foi ou não alterada pelo caminho.

Confidencialidade

A confidencialidade é garantida com encriptação e desencriptação das mensagens. Após o cliente escrever a mensagem, o controlador irá ser responsável por encriptar e enviar a mensagem do cliente. Para isso, cada cliente tem uma chave que é renovada de X em X tempo para garantir mais segurança ao utilizador. Assim sendo, são utilizados os seguintes algoritmos:

|  |  |
| --- | --- |
| **Gerar a chave:** | AES |
| **Encriptar:** | AES/CBC/PKCS5Padding |
| **Desencriptar:** | AES/CBC/PKCS5Padding |

Existe uma classe Mensagem que é responsável por transportar a mensagem encriptada bem como a chave de quem envia a mensagem.

A chave é criada com o seguinte algoritmo:

|  |
| --- |
|  |

Algoritmo Encriptação

O algoritmo de encriptação que utiliza a chave acima gerada é o seguinte:

|  |
| --- |
|  |

Algoritmo Desencriptação

O algoritmo de desencriptação que utiliza a chave acima gerada é o seguinte:

|  |
| --- |
|  |

Autenticidade/Integridade

Para garantir estes mecanismos de segurança, a mensagem guarda uma hash que é criada logo após o cliente envia a mensagem. Esta hash é gerada de acordo com o corpo da mensagem de quem a envia para garantir toda esta segurança. O algoritmo que cria a hash é:

|  |
| --- |
|  |

Manutenção da chave

Para garantir a manutenção da chave, esta é renovada de X em X tempo graças a uma thread que o servidor lança. Essa irá apagar todas as chaves anteriores, gerar novas, guardar as mesmas e é responsável por enviar uma mensagem a cada cliente indicando-lhe a sua nova chave.

|  |
| --- |
|  |

Non-repudiation

Para garantir “non-repudiation” cada cliente tem um par de chaves. O cliente encripta a sua mensagem com a sua chave privada e, o receptor receberá a chave pública para desencriptar a mesma.

Assinatura para encriptar:

|  |
| --- |
|  |

Com este pedaço de código o utilizador instancia uma “Signature” e cria uma fábrica de chaves. Isto para que se possa ir buscar a chave privada e, criar uma assinatura com a chave privada sobre a mensagem deste utilizador.

Assinatura para desencriptar:

|  |
| --- |
|  |

Aqui podemos ver que o código faz praticamente o mesmo mas, vai buscar a chave pública para verificar se combina com a assinatura.