UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Tarefa Prática – Criptografia Simétrica, MAC e PBKDF2

Você irá usar as bibliotecas criptográficas fornecidas pelo provedor Bouncy Castle. As bibliotecas estão no diretório chamado "fips". Usaremos o provedor BCFIPS que é a versão FIPS da Bouncy Castle.

O arquivo disponibilizado no moodle (na área da definição da tarefa) possui os exemplos que você **DEVE EXECUTAR** e **ENTENDER** para poder desenvolver esta tarefa.

Desenvolva uma aplicação para que seja possível Alice conversar com Bob usando mensagens criptografadas: Alice envia uma mensagem para Bob e depois Bob envia uma mensagem para Alice.

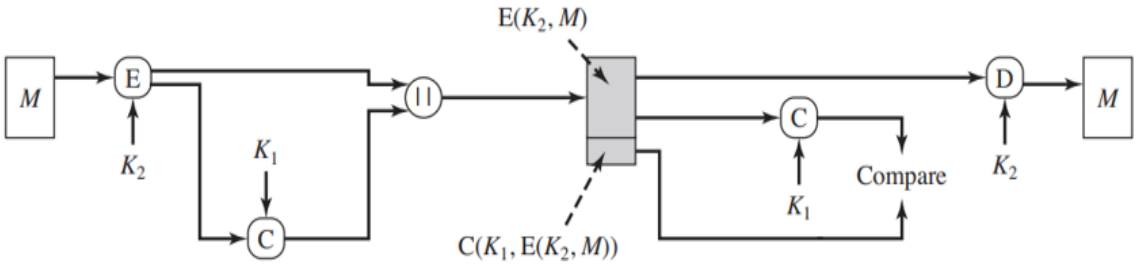
Requisitos da aplicação:

a) Alice e Bob podem ser usuários (objetos de uma classe Usuário, por exemplo) que são instanciados no início da aplicação. Não é necessário usar Sockets (o uso de Sockets é opcional, caso você queira aprender).

b) Alice e Bob devem trocar mensagens cifradas: Alice envia para Bob e depois Bob envia para Alice. O emissor da mensagem precisa construir a mensagem cifrada realizando manualmente o processo *Encrypt-then-Mac*. A chamada do método usado pelo emissor pode ser da seguinte maneira:

```
enviaMensagem (Usuario target, String blocoMsgEncryptThenMac, String keyWrapped)
```

c) A mensagem cifrada (`blocoMsgEncryptThenMac`) é construída manualmente com Encrypt-then-Mac, representado na figura. Para fazer a cifragem (Encrypt) deve ser usado o AES no modo CTR com chaves de 128 bits. Para fazer o MAC deve ser usado o HMAC (HMacSHA256). Deve ser construído todo o processo representado na figura durante a emissão e a recepção.



(c) Autenticação e confidencialidade da mensagem; autenticação ligada ao texto cifrado

d) O emissor envia uma mensagem cifrada para o receptor. O receptor, ao receber a mensagem cifrada, irá realizar o processo de verificação descrito na figura demonstrando que a mensagem está íntegra. Depois disso, escreverá na tela a msg decifrada.

e) O receptor não poderá usar chave e IV armazenados em variáveis na memória no momento da decifragem/verificação de MAC. Você deve agir como se o emissor e receptor estivessem localizados em máquinas diferentes.

f) Para gerar as chaves/IVs deve ser usado o PBKDF2. A chave usada para cifrar deve ser "wrapped" (encapsulada/cifrada) usando o processo descrito no exemplo KWExample.java dos exemplos (projeto DavidProj). *Key wrapping* significa cifrar uma chave usando outra chave.

g) Pode ser usada uma "senha mestre" (senha de admin) para derivar uma chave a ser usada pelo programa usando o PBKDF2. Essa chave derivada pode ser derivada quantas vezes forem necessárias para derivar novas chaves, caso sejam úteis.

h) Decisões próprias sobre formatos e parâmetros devem ser feitas.

NÃO É PERMITIDO: ter chaves e IV fixos e escritos no próprio código.

Se forem guardados, os parâmetros devem ser guardados em arquivo cifrado.

A. O código fonte deve ser postado no moodle, juntamente com um tutorial de execução da aplicação. Deve ser possível executar a aplicação com os arquivos anexados dentro do código.

B. Apresentação desta questão (vídeo):

i) os alunos devem gravar a apresentação desta questão

ii) deve ser apresentado o código sendo executado (aplicação rodando)

iii) deve ser explicado o código, chamando a atenção para os requisitos pedidos (Encrypt-then-Mac, key wrapping, PBKDF2, **ausência de** chaves e IV fixos, parâmetros corretos). Ambos os autores devem apresentar uma parte da gravação. O link do vídeo deve ser disponibilizado na entrega da tarefa, juntamente com o código fonte. Pode ser feita a gravação com o celular ou outro software de sua escolha. Não precisa ser nada muito elaborado, desde que seja possível escutar bem a voz, visualizar bem a execução e a apresentação do código. Procure usar fontes grandes para apresentar o código. Lembre-se de defender bem a sua ideia e de cumprir os requisitos exigidos.

Avisos:
**** Se tiver cópias de código, todos os envolvidos receberão nota zero nesta tarefa.**

Referências:

1. Arquivos que estão dentro do diretório FIPS.
2. The Bouncy Castle FIPS Java API in100 Examples - https://www.bouncycastle.org/fips-java/BCFipsIn100.pdf
3. Livro e códigos exemplo do livro Beginning Cryptography with Java: diretório Example (compactado junto no zip de exemplos). Disponível em http://www.wrox.com/WileyCDA/WroxTitle/Beginning-Cryptography-with-Java.productCd-0764596330.html
4. Bouncy Castle últimas versões - https://www.bouncycastle.org/latest_releases.html
5. Documentação classes BouncyCastle - https://www.bouncycastle.org/docs/docs1.5on/index.html
6. Lista das especificações dos algoritmos criptográficos da BouncyCastle - http://www.bouncycastle.org/specifications.html
7. Tutorial - http://docs.oracle.com/javase/7/docs/technotes/guides/security/crypto/CryptoSpec.html
8. Documentação Java Cryptography Architecture Java 8 – https://docs.oracle.com/javase/8/docs/technotes/guides/security/StandardNames.html
9. Password Storage Cheat Sheet - https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Password_Storage_Cheat_Sheet.md