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

Tarefa Prática – Exercícios de criptografia simétrica, hash, MAC, PBKDF e Criptografia Autenticada em Java

Você irá usar as bibliotecas criptográficas fornecidas pelo provedor Bouncy Castle (https://www.bouncycastle.org/latest_releases.html) e deve incluir os pacotes nas bibliotecas dos projetos. As bibliotecas estão no diretório chamado "bibliotecas" e estão compactadas no arquivo zip da tarefa. Usaremos o provedor BCFIPS que é a versão FIPS da Bouncy Castle e em alguns casos o provedor BC que é o provedor padrão da Bouncy Castle.

O capítulo 3 do livro Beginning Cryptography with Java (arquivo Beginning Cryptography with Java.chm) pode ser lido para entender alguns dos exemplos fornecidos. Este livro está no formato CHM que é o formato Microsoft Compiled HTML Help. Assim, no Windows o arquivo abre normalmente. No Linux e MacOs deve ser obtido um programa para leitura do livro: no Linux pode ser o xCHM, ChmSee ou outro equivalente; no MacOs tem o CHM Reader.

1. Abra o **projeto2CodigoLivro** e teste o seu funcionamento. Responda:

   1.1. Qual algoritmo é usado no código? Em qual modo?

   1.2. Explique o que faz o método generateKey da classe KeyGenerator https://docs.oracle.com/javase/7/docs/api/javax/crypto/KeyGenerator.html.

   1.3. Explique como são usados os métodos *init*, *update* e *doFinal* para cifrar e para decifrar os dados nesse código. Leia a documentação e entenda bem o funcionamento desses métodos.

2. Crie um programa que permite ao usuário entrar com uma string pelo teclado, o programa cifra a string e mostra a string cifrada na tela. O código deve "sortear" uma boa chave e IV. Use o modo CTR (counter) do algoritmo AES para cifrar. Use o projeto3Aes para auxiliar.

3. Nesse projeto você irá programar dois sistemas de decifragem, um usando o AES em **modo CBC** e outro usando o AES no **modo contador** (*counter mode* – **CTR**). Em ambos os casos um IV de 16 bytes é escolhido de forma aleatória. Para o modo CBC use o esquema de padding PKCS5. Para o modo CTR use NoPadding.

   Inicialmente iremos testar apenas a função de decifragem. Use o projeto3Aes para auxiliar a responder as questões. Nas questões seguintes você recebe uma chave AES, um IV e um texto cifrado (ambos codificados em hexa) e o seu objetivo é recuperar o texto plano/texto decifrado. A resposta de cada questão é o texto decifrado (frase legível).

   3.1.
   - Chave CBC: 53efb4b1157fccdb9902676329debc52
   - IV: d161fbaa4c64ecf7d2c4abd885751273
   - Texto cifrado em modo CBC:

701f7fa45d9bb922c3cb15a519ba40ede1769eb753650886d6e69ebcad9c2816002679896a65a921d25e00793078474e3dbeca9a2838031c490e5ae9d1ea143f

3.2.

- Chave CTR: a05e2679204241af07f6857d150a1fcd
- IV: 468ce1126a37b07138e78eab48344712
- Texto cifrado em modo CTR:

36466b5fddcfcb1b8a9479eb8c489e7139a3c35020b1e5ee808b39ff18b6abd812afe7dbbca40e15df391a7c07ece1c8e10a49368b86a946c8379cd8fa01a47f1956671144b0ca18a4c812cde8f7b9

4. Crie um programa que recebe duas strings pelo teclado, calcula o hash (resumo criptográfico) e o MAC de cada uma das strings escrevendo o resultado na tela. Teste e explique o funcionamento do programa com entrada de strings iguais e depois com entrada de strings diferentes.
5. Crie um programa que recebe uma string pelo teclado e cifra a string usando CRIPTOGRAFIA AUTENTICADA (AES no modo GCM). O programa também deve gerar uma boa chave usando PBKDF2.
6. (Peso: 50% da nota da tarefa) Desenvolva um programa que usa criptografia autenticada no modo GCM e PBKDF2 para implementar uma aplicação útil. Você é livre para criar seu próprio exemplo para implementar (cifrar arquivos, jogos que usam criptografia, ...) !! NÃO É PERMITIDO: ter chaves e IV fixos e escritos no próprio código. Parâmetros devem ser guardados cifrados em arquivo (não podem ser guardados em texto plano).

   6.1. 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.

   6.2. Apresentação desta questão (vídeo):

      6.2.1. os alunos devem gravar a apresentação desta questão

      6.2.2. deve ser apresentado o código sendo executado (aplicação rodando)

      6.2.3. deve ser explicado o código, chamando a atenção para os requisitos pedidos (GCM, 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 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 questão.

Referências:

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