Segurança de Redes e Sistemas de Computadores
2016/2017, 2º Semestre

Aula Prática (LAB 2)

Exercícios de programação usando:

- **Criptografia Simétrica com cifras de blocos e modos de cifra de blocos**
- **Funções de Síntese Seguras (*Secure Hash Functions*)**
- **Autenticação e Integridade usando *MACs* (*Message Authentication Codes*)**

Para os próximos pontos, comece por obter o arquivo lab2.tgz. Abra o arquivo de modo a obter o material que lhe é fornecido.

---

1. Exercícios de Programação com Criptografia Simétrica e suas parametrizações com modos de cifra de blocos e *padding* normalizado.

Para os próximos pontos, comece por obter o arquivo lab2.tgz. Abra o arquivo de modo a obter o material que lhe é fornecido.

1.1 Criptografia Simétrica com métodos de cifra por blocos

Vamos começar pelo conteúdo da diretoria

Exercicios-modos-cifra-simetrica

O objetivo será verificar como utiliza a JCE para programar com criptografia simétrica e como pode usar os modos criptográficos simétricos, ex: ECB (SimpleECBExample), CBC (SimpleCBCExample), CTR (SimpleCTRExample) ou CFB (como faria ?)

Deve ensaiar os programas e modificar de modo a compreender o funcionamento, bem como o impacto de usar os vários modos bem como o papel do *padding*. Tente compreender as implicações nas parametrizações das funções de cifra bem como o impacto no tamanho dos blocos (*Plaintext* vs. *Ciphertext*)

Faça os exercícios que lhe vão ser sugeridos na aula prática pelo docente.

Analise agora o código em InlineIvCBCExample.java e NonceIvCBCExmaple.java.

- O que encontra de interessante nestes programas em relação à gestão de vetores de inicialização (IVs) ou *nonces* usados como IVs ?

Vamos agora observar em mais detalhe na prática a utilização da parametrização de *padding* nos algoritmos criptográficos simétricos por blocos., bem como a importância desta parametrização e sua normalização.

Para o efeito, verificar a diretoria:

ExemplosVerifPadding

Esteja atento à explicação que será feita para verificar o impacto desta parametrização quer do ponto de vista da segurança quer do ponto de vista do impacto em relação à dimensão da informação *plaintext* bem como da informação *ciphertext*.

2. Exercícios de Programação com Criptografia Simétrica e suas parametrizações com modos de cifra de blocos e *padding* normalizado.

2.1 Secure Hashing (ou uso de Funções para Síntese Segura de Mensagens)

Vamos agora utilizar o conteúdo da diretoria

Exercicios-secure-hashing

Deve seguir a seguinte sequência para poder compreender o que está envolvido:

TamperedExample.java

TamperedExample2.java

São dois exemplos que demonstram um ataque do tipo *Message Tampering* que atenta contra a INTEGRIDADE das mensagens. Esteja atento à demonstração inicial que lhe será feita na aula prática.

Verifique como a simples utilização de criptografia simétrica não consegue evitar este ataque que é um ataque à INTEGRIDADE. Verifique que pode fazer ataques deste tipo, a partir dos programas fornecidos, sem que o emissor e o receptor se apercebam do que estão em presença de um atacante.

Verificaremos de seguida como resolver o problema da INTEGRIDADE.

Comece por compreender o código em:

TamperedWithDigestExample.java

Utilize este programa para ver como os ataques de Message Tampering podem ser defendidos com métodos seguros de síntese ou digestão de mensagens (conhecidas por secure hash methods)

... Tudo parece assim permitir proteger a INTEGRIDADE.

... Ou será que não?

Estude e verifique o programa:

TamperedDigestExample.java

Exercício: Esteja atento à demonstração que lhe será feita e tire as suas conclusões do que aprendeu.

O que há a fazer para conseguir ter uma proteção definitiva face a ataques que visam quebrar a integridade e a autenticidade das mensagens ?

Questão importante: como usar corretamente a combinação de cifras simétricas (para proteger a CONFIDENCIALIDADE) e sínteses seguras (para proteger a INTEGRIDADE) ? Qual a melhor solução e como tratar na prática o TRADEOF entre segurança e desempenho ?

3. Exercícios de Programação com Criptografia Simétrica, Funções de Síntese Segura de Mensagens (*Secure Hashing*) e Códigos de Autenticação de Mensagens (*MACs, HMACs e CMACs*)

3.1 MACs (*Message Authentication Codes*), HMACs (ou *Hash-Based MACs*) e CMACs (ou Cryptographic MACs)

Vamos agora utilizar o conteúdo da última diretoria

Exercicios-MAC

Esta diretoria tem um único exercício.

Verifique como pode usar MACs. Neste caso podemos usar HMACs (ver o caso de **TamperedWithHMacExample.java**) ou CMACs (no caso de **CipherMacExample.java**) para prevenir em definitivo problemas de proteção das comunicações para resistir a ataques à integridade de mensagens com provas incluídas de autenticidade das mesmas (ou seja *Message Aauthentication Codes*).

Desafios (trabalho de casa)

Exercício 1 (para programar)
A partir dos exercícios dos desafios lançados no LAB-1, tente agora iterar esses desafios protegendo as mensagens de CONFIDENCIALIDADE E INTEGRIDADE, usando MACs para implicitamente verificar a integridade com autenticidade das mensagens.

Exercício 2 (para definir uma solução)
Quer construir um pequeno sistema de auditoria que verificasse ou atestasse a integridade do file-system dos computadores dos laboratórios do DI, de modo a prevenir eventuais intrusões que tivessem em vista modificar por exemplo ficheiros considerados críticos (ex., aplicações, ficheiros com configurações críticas, ficheiros associados a devices, etc). Como o faria ? Como proporia uma solução para o efeito ? Qual era a arquitetura do sistema que conceberia? Como a programaria ?
