Segurança de Sistemas Computacionais
2016/2017, 2º Semestre

Aula Prática nº 5 (Lab 5)
Programação em Java/JCA – JCE (Java Cryptographic Extension)

Criptografia Assimétrica

Exemplos / Demonstrações e Exercícios

Verifique o código que lhe é fornecido para a aula prática (lab6).

Neste parte vamos verificar experimentalmente diversos aspectos práticos de desempenho, uso e programação com métodos criptográficos assimétricos em JAVA (com base no suporte JCA/JCE).

1. Avaliação comparativa de impacto do uso de diferentes algoritmos criptográficos.

O primeiro aspeto a ver será obtermos rapidamente uma avaliação experimental comparativa do desempenho de funções de síntese, algoritmos criptográficos simétricos e algoritmos criptográficos assimétricos.

Para este efeito propomos o seguinte exercício:

1- Utiliza a ferramenta openssl (está pré-instalada em ambientes MAC OS ou LINUX podendo ser obtida também para sistemas Windows :
   Ver em: https://www.openssl.org
   Ou procurar também, por exemplo, em:
   http://gnuwin32.sourceforge.net/packages/openssl.htm.

   Será rapidamente explicada em que consiste esta ferramenta (que para além de ser uma implementação de referencia da suite criptográfica e protocolos da pilha TLS ou SSL, é na verdade uma biblioteca de propósito geral que permite utilizar todos os algoritmos criptográficos usados na normalização TLS ou SSL. Mais tarde, voltar-se-á a usar esta ferramenta, quando se discutir em mais detalhe o protocolo TLS ou SSL (que será também discutido nas aulas teóricas).

2. Por agora vamos usar a ferramenta openssl para testar rapidamente o desempenho de diferentes métodos criptográficos, usando funcionalidade já nativamente suportada nessa ferramenta para este tipo de benchmarking.

Executando:

Openssl speed

Poderá fazer um teste rápido da eficiência de todas as implementações criptográficas da sua versão openssl. Notar que para maior rapidez e foco pode verificar comparativamente e seletivamente alguns algoritmos específicos. Propomos as seguintes observações comparativas (que depois podem ser estendidas em avaliação experimental autónoma).

```
openssl speed md5 sha1 sha256 sha512
openssl speed des cast bf aes
openssl speed aes rc4
openssl speed sha512 aes rsa
openssl speed rsa dsa
openssl speed ecdsa
openssl speed ecdh

geração de chaves RSA de vários tamanhos:

openssl genrsa NNNN
com NNNN= 512, 1024, 2048, 4096, 8192, 16384, 32768 bits

Comparar com
openssl gendh 1024

Comparar agora com:
openssl dhparam 1024 –out xxx
openssl gendh –rand xxx
```

Nota: documentação da ferramenta openssl: man openssl

Reflita e discuta nos resultados obtidos.

2. Utilização de criptografia assimétrica em JAVA (JCA/JCE)

Verifique no arquivo da aula prática o código na seguinte diretoria:

AsymCryptography

Inicialmente vamos observar o comportamento do código fornecido na diretoria **SimplesBaseRSA,** para compreender a programação com algoritmos criptográficos assimétricos (neste caso o RSA).

2.1 Verifique inicialmente o código BaseRSAExample.

O que fez este código ? Tente interpretar e compreender o que está a ser feito e o funcionamento do programa fornecido. Responda às seguintes questões:

a) Qual o tamanho das chaves utilizadas ? Note que as chaves estão a ser usadas com base numa inicialização estática dos valores dos expoentes e módulo a usar.
b) Recorde-se de como funciona o processo de geração de chaves em RSA e as operações para cifrar e decifrar. Qual o valor das chaves e qual o valor N (operação mod) subjacente à cifra e decifra RSA usadas ?
c) Faz sentido o tamanho do ciphertext produzido a partir do plaintext que está a ser cifrado ? Porquê ?
d) Modifique o código e verifique que, como esperado, o que se pode cifrar com a chave pública decifra-se com a chave privada e o que se pode cifrar com a chave privada decifra-se com a pública. Verifique que funciona, de acordo com o esperado.
e) O que acontece se o bloco plaintext for maior em dimensão que o valor N subjacente à operação mod subjacente e às chaves usadas? Isso faz sentido ? Porquê ?
f) Repare se no código fornecido se usa ou não padding. Que tipo de problemas de segurança podem existir no facto de não se utilizar padding?

2.2 Verifique agora o código fornecido na diretoria RandomKeyRSA, nomeadamente: RandomKeyRSAExample ou RandomKeyRSA.

a) O que verifica nestes casos comparativamente ao anterior ?
b) Observe experimentalmente no seu computador a diferença de desempenho relativo ao tempo de geração das chaves e ao tempo de computação (cifra/decifra) quando aumenta o tamanho das chaves geradas e utilizadas.

2.3 Verifique agora o código fornecido em PKCS1PaddedRSA

a) Repare neste código a utilização de *padding* normalizado (com o padrão PKCS#1). Tente investigar para saber em que consiste este padrão. Que tipo de problemas de segurança identifica no facto de não se utilizar padding (e tente compreender a importância de utilizar padding normalizado, ex., PKCS1 – recorde-se da aula teórica). Note que no caso de usar PKCS1 significa pré-processar uma das seguintes formas de padding:

V1: PKCS5 (M) = 0x00 || 0x01 || F || 0x00 || M
Sendo F formado da forma 0xFF 0xFF 0xFF .... etc com tamanho no mínimo de 8 bytes. Desta forma, o tamanho da mensagem não poderá ser superior do que o comprimento da chave reduzido de 11 bytes.

V2: PKCS5 (M) = 0x00 || 0x01 || R || 0x00 || M
Neste caso, R é calculado de forma pseudo-aleatória, com pelo menos 8 bytes de tamanho.

Descubra se a implementação utiliza a forma V1 ou V2. Porquê ?

2.4 Verifique agora o código OAEPPaddingRSA

Recorde-se da discussão sobre padding OAEP na aula teórica. Verifique a diferença de utilização do método de padding utilizado, comparativamente a usar o anterior (PKCS1). Notar que o padding neste caso é calculado na forma:

1. M1 = Mask ((H(P) || PS || 0x01 || M), S)
2. M2 = Mask (S, M1)
3. Mp = 0x00 || M2 || M1

Racional do processamento de padding OAEP (ver no código):

- H(): Função de Hashing subjacente à computação de padding
- Mask(x, y): Hashing obtido a partir de uma semente (y) para um gerador pseudo-aleatório interno, utilizado para mascarar o argumento x, produzing uma síntese de tamanho hLen.
- PS: Pad string inicializada om 0x00
- P: inicializado com uma string com caracteres 0x00
- S: Seed

Notar que o tamanho máximo do plaintext usando este esquema de padding será: MaxLen = kLen – 2 hLen -2

a) Independentemente do anterior, verifique que o tamanho do ciphertext obtido é o expectável. Porquê ?
b) Pela observação do programa e observando várias corridas do programa e os resultados obtidos, tente identificar os moldes da inicialização do cálculo de padding (tendo em conta os dados anteriores). Isto faz sentido mesmo quando se está em cada corrida a gerar um novo par de chaves ? Porquê ?

Exercício opcional como trabalho de casa:

Escreva um programa emissor A (cliente) que envia por um socket TCP um ficheiro a um destinatário B (que funciona como servidor TCP). O emissor e o destinatário conhecem as chaves públicas RSA um do outro e cada um deles tem guardada em segurança a sua chave privada (correspondente à chave pública conhecida).

O emissor vai enviar o ficheiro usando criptografia simétrica (AES, chave de 256 bits), transferindo o nome do ficheiro na seguinte forma:

Na primeira mensagem envia $\{Ks1\}_{KpubB}\{filename, Nonce1\}_{Ks1}$

O servidor responde da seguinte forma:

$\{KS2\}KpubA\ \{START, Nonce1+1\}_{ks2}$

De seguida A envia os blocos do ficheiro para transferir todo o ficheiro na forma: $\{i, Bi\}_{Ks3}$

Sendo Ks3 uma chave AES de 256 bits, gerada a partir das primeiras chaves-seed Ks1 e Ks2.

Questões:

a) Note que no protocolo a distribuição de KS1 e KS2 se faz com base em envelopes de chave pública
b) Podem A e B estar certos que estão a comunicar com os interlocutores corretos ? Se sim porquê ? Se não o que falta fazer ?