Instituto Superior de Engenharia de Lisboa
LEIRT, LEIM, LEIC
Segurança Informática
Primeiro trabalho, Semestre de Verão de 21/22
Entregar até 18 de abril de 2022

1. Considere o esquema CI cujo objetivo é realizar cifra e autenticidade de mensagens, onde $||$ representa a concatenação de bits e $X_{1..L}$ representa os primeiros L bits de X.

$$CI(m) = T(k_1)(m)||E_s(T(k_1)(m)_{1..L})(m)$$

   T é o gerador de marcas de um esquema de *message authentication code* (MAC). E_s é o algoritmo de cifra de um esquema simétrico de cifra, cuja dimensão da chave é L bits. Porque motivo este esquema não cumpre os objectivos?

2. Porque motivo um esquema simétrico que utilize o modo de operação `CBC` precisa de ter usar um algoritmo de *padding*, enquanto que no modo `Counter` não é necessário considerar *padding*?

3. Descreva um ataque a um esquema assimétrico de confidencialidade no qual o processo de cifra é determinístico, ou seja, mensagens iguais resultam em criptogramas iguais.

4. Na biblioteca JCA, porque razão a classe `MAC` não precisa de ter um método `verify`, semelhante ao existente na classe `Signature`?

5. Considere os certificados digitais X.509 e as infraestruturas de chave pública:

   5.1. Em que situações é que a chave necessária para validar a assinatura de um certificado não está presente nesse certificado?

   5.2. Quais as consequências se uma aplicação consumidora de certificados ignorar a extensão *basic constraints*?

   5.3. Qual a diferença entre ficheiros `.cer` e ficheiros `.pfx`?

6. Considere o laboratório "Pseudo Random Number Generation Lab" [1] do projeto SEED Labs. Realize as tarefas 2.1 ("Generate Encryption Key in a Wrong Way") e 2.2 ("Guessing the Key"):

   6.1. Descreva a consequência da utilização ou não da função `time()` na chamada à função `srand()` na Listagem 1.

   6.2. Explique sucintamente o programa desenvolvido para descobrir a chave usada para decifrar o criptograma.

7. Usando a biblioteca JCA, realize em Java uma aplicação para cifra autenticada de ficheiros com um esquema híbrido, ou seja, usando sistemas de cifra simétrica e assimétrica. O conteúdo do ficheiro é cifrado com uma chave simétrica, a qual é cifrada com a chave pública do destinatário do ficheiro. A aplicação tem dois modos de operação, cifra e decifra, recebendo na linha de comandos a opção para cifrar (`-enc`) ou decifrar (`-dec`) e o ficheiro para cifrar/decifrar:

   - No modo de cifra, a aplicação recebe o certificado com a chave pública do destinatário e produz dois ficheiros, um com o conteúdo original cifrado e outro com a chave simétrica cifrada. Ambos os ficheiros (dados originais cifrados e chave cifrada) devem ser codificados em Base64 [2]. Valorizam-se soluções que validem o certificado antes de ser usada a chave pública e que não imponham limites à dimensão do ficheiro a cifrar/decifrar.
   - No modo de decifra, a aplicação recebe i) ficheiro com conteúdo original cifrado; ii) ficheiro com chave simétrica cifrada; iii) *keystore* com a chave privada do destinatário; e produz um novo ficheiro com o conteúdo original decifrado.

   Para cifra simétrica autenticada use o algoritmo AES no modo GCM com dimensão de *tag* de 128 bits. O esquema assimétrico usa o algoritmo RSA. Para a codificação e descodificação em *stream* de bytes em Base64 pode usar a biblioteca Apache Commons Codec [3].

   Considere os ficheiros `.cer` e `.pfx` em anexo ao enunciado onde estão chaves públicas e privadas necessárias para testar a aplicação.

18 de março de 2022 (v1.1 editado a 21 de março)

Referências

[1] https://seedsecuritylabs.org/Labs_20.04/Crypto/Crypto_Random_Number/

[2] https://datatracker.ietf.org/doc/html/rfc4648

[3] https://commons.apache.org/proper/commons-codec/