Instituto Superior de Engenharia de Lisboa
LEIRT, LEIM, LEIC
Segurança Informática
Primeiro trabalho, Semestre de Inverno de 22/23
Entregar até 29 de outubro de 2022

1. Considere um novo modo de operação definido por:

   - Seja $x = x_1, \ldots, x_L$ a divisão nos blocos x_i do texto em claro x.
   - RV é um vector aleatório, com a dimensão do bloco, gerado por cada texto em claro x.
   - Seja $y_i = E(k)(x_i \oplus RV)$, para $i = 1, \ldots, L$, onde E é a operação de cifra, k é a chave da cifra, $\oplus$ denota o ou-exclusivo bit a bit.

   1.1. Defina o algoritmo de decifra para este modo de operação.

   1.2. Compare este modo de operação com o modo CBC quanto a: *a)* possibilidade de padrões no texto em claro serem evidentes no texto cifrado, *b)* capacidade de paralelizar a cifra.

2. O RFC 4880, "OpenPGP Message Format", especifica a cifra de mensagens (denominados objectos) como uma combinação entre esquemas assimétricos e simétricos:

   «[...] first the object is encrypted using a symmetric encryption algorithm. Each symmetric key is used only once, for a single object. A new "session key" is generated as a random number for each object (sometimes referred to as a session). Since it is used only once, the session key is bound to the message and transmitted with it. To protect the key, it is encrypted with the receiver's public key. [...]»

   Justifique a utilização desta abordagem com dois tipos de chave e explique sucintamente o processo de decifra de uma mensagem (object).

3. A engine classe `Signature` da JCA contém, entre outros, os seguintes métodos:

```
void initSign(PrivateKey privateKey)
void initVerify(PublicKey publicKey)
void update(byte[] data)
byte[] sign()
boolean verify(byte[] signature)
```

   3.1. Explique sucintamente o processamento realizado internamente no método `sign` com o objetivo de fazer a assinatura. Pode usar na explicação os métodos referidos que entenda relevantes.

   3.2. Considere que é instanciado um objeto `Signature` com a transformação "RSAwithMD5". Se em virtude de uma vulnerabilidade detectada na função de *hash* $MD5$ for computacionalmente fazível, dado x, obter $x' \neq x$ tal que $MD5(x') = MD5(x)$, quais as implicações deste ataque para as assinaturas geradas/verificadas pelas transformação referida?

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

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

   4.2. Porque motivo a proteção de integridade dos certificados X.509 não usa esquemas MAC (Message Authentication Code)?

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

5. Usando a biblioteca JCA, realize em Java uma aplicação para geração de *hashs* cripográficos de ficheiros. A aplicação recebe na linha de comandos *i)* o nome da função de *hash* e *ii)* o ficheiro para o qual se quer obter o *hash*. O valor de hash é enviado para o *standard output*.

   Teste a sua aplicação usando certificados (ficheiros `.cer`) presentes no arquivo `certificates-and-keys.zip`, em anexo a este enunciado. Compare o resultado com os valores de *hash* apresentados pelo visualizador de certificados do sistema operativo (ou outro da sua confiança).

6. Usando a biblioteca JCA, realize em Java uma aplicação para cifrar ficheiros com um esquema híbrido, ou seja, usando 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 recebe na linha de comandos a opção para cifrar (`-enc`) ou decifrar (`-dec`) e o ficheiro para cifrar/decifrar.

   No modo para cifrar, a aplicação também 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 devem ser codificados em Base64. 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 para decifrar, a aplicação recebe também 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 a codificação e descodificação em *stream* de bytes em Base64 deve usar a biblioteca Apache Commons Codec [1].

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

22 de setembro de 2022

Referências

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