CIC0201 - Segurança Computacional - 2022/2

Prof. João Gondim

Trabalho de Implementação 2

Gerador/Verificador de Assinaturas

Neste trabalho, deve-se implementar um gerador e verificador de assinaturas RSA em arquivos. Assim, deve-se implementar um programa com as seguintes funcionalidades:

- Parte I: Cifração e decifração AES, chave 128 bits
  a. Geração de chave de 128 bits
  b. Cifração e decifração
  c. Extra: cifração autenticada modo XTR – Contador de Galois
- Parte II: Geração de chaves e cifra RSA
  a. Geração de chaves (p e q primos com no mínimo de 1024 bits) testando primalidade com Miller-Rabin.
  b. OAEP
  c. Cifração/decifração assimétrica RSA usando OAEP.
- Parte III: Assinatura RSA
  a. Assinatura da mensagem (cifração do hash da mensagem)
  b. Formatação do resultado (caracteres especiais e informações para verificação em BASE64)
- Parte IV: Verificação:
  a. Parsing do documento assinado e decifração da mensagem (de acordo com a formatação usada, no caso BASE64)
  b. Decifração da assinatura (decifração do hash)
  c. Verificação (cálculo e comparação do hash do arquivo)

Casos de uso:

1. Cifração de uma mensagem com AES:
   M, k -> AES_k(M)
2. Cifra híbrida:
   Enviando mensagem M para usuário A, (KA_p, KA_s) = chves assimétricas de A
   C = (AES_k(M) , RSA_KA_p(k))
3. Cifra híbrida (autenticação mútua):
   Usuário B enviando mensagem M para usuário A, (KA_p, KA_s) = chves assimétricas de A; (KB_p, KB_s) = chves assimétricas de B
   C = (AES_k(M) , RSA_KB_s (RSA_KA_p(k)), KB_p)
4. Graçao de Assinatura de A
   Usuário B enviando mensagem M para usuário A, (KA_p, KA_s) = chves assimétricas de A; H(.)=função de hash
   Sign = (AES_k(M) , RSA_KA_s(H(AES_k(M))), KA_p)
5. Verificação da assinatura
   RSA_KA_s (RSA_KA_s(H(AES_k(M)))) = H(AES_k(M)) ?

Observações:

1. Permite-se a utilização de bibliotecas públicas para aritmética modular e função de hash SHA-3
2. Não é permitida a utilização de bibliotecas públicas, como OpenSSL, para primitivas de criptográficas de cifração e decifração simétrica e assimétrica, bem como de geração de chaves.
3. A pontuação máxima será conferida os trabalhos que realmente implementarem as seguintes primitivas nos casos de uso especificados acima:
    a. geração de chaves com teste de primalidade (Miller-Rabin)
    b. cifração e decifração RSA
    c. OAEP
    d. formatação/parsing
4. A avaliação será mediante apresentação do trabalho, com a verificação das funcionalidades e inspeção do código.
5. Implementação preferencialmente individual, podendo ser em dupla. Linguagens preferenciais C, C++, Java e Python.

O que deve ser entregue: o código fonte e seu executável, descritivo (4 pg max), do OAEP, da assinatura RSA e sua verificação e do programa.

Data de Entrega: 03/07/2023, por email até 10h.

Apresentações: 03 e 55/07/2023