Disciplina: Segurança Computacional

Aluno: Miguel de Sousa Santos Matrícula: 140077464

Relatório Descritivo

Trabalho de Implementação

Gerador/Verificador de Assinaturas

O algoritmo RSA realiza a assinatura e a verificação digital de arquivos, garantindo as características básicas das assinaturas, que são: autenticidade, não repúdio e integridade.

A criptografia RSA consiste em gerar um par de chaves através de números primos grandes; Codificação, e Decodificação. Sendo que a segurança proporcionada por esse algoritmo está na dificuldade computacional em fatorar esses números. O problema da fatoração de inteiros possui alta complexidade computacional (a fatoração é muito lenta), se tornando inviável fatorar um número muito grande.

A criptografia de chave assimétrica é uma forma de criptografia em que a chave usada para decriptografar a mensagem não pode ser calculada usando a chave para criptografar a mensagem. Ela possui a vantagem de poder tornar a chave de cifração pública sem que haja risco de que outras pessoas possam usar isso para decifrar a mensagem, permitindo assim que qualquer um possa mandar uma mensagem ao dono da chave sem que outros possam acessar o conteúdo desta mensagem. Por outro lado, ela possui a desvantagem de ser mais lenta para cifrar decifrar a mensagem do que a criptografia de chave simétrica.

Geração das chaves:

1. São escolhidos dois números primos p e q.
2. É calculado n = p.q. e φ(n) = (p – 1)(q - 1), onde φ é a função totiente de Euler.
3. Escolhe-se um ‘e’ tal que 1 < e < φ(n) e m.d.c.(φ(n), e) = 1.
4. Depois e calculado ‘d‘ de forma que d.e ≡ 1 (mod φ(n)).
   1. A chave pública é o par de números n e ‘e’ ;
   2. A chave privada é o par de números n e d.

Codificação:

Primeiramente deve-se converter a mensagem em números. Uma forma é associar a cada caractere um numero de dois dígitos. Depois, divide-se a mensagem em blocos, que devem ter valor menor que n. Cada bloco é encriptado separadamente da seguinte forma: Seja m o bloco. O bloco encriptado será c ≡ m e (mod n).

Decodificação:

Para recuperar a mensagem ‘m’ da mensagem encriptada ‘c’ basta fazer outra potenciação modular: m ≡ c d (mod n).

A assinatura digital usa o mecanismo de geração de mensagens assinadas pela chave privada e validadas pela chave publica.

A assinatura RSA utiliza uma técnica parecida à criptografia RSA, sendo que a chaves pública e privada são as mesmas mudando apenas a forma como vão ser usadas. Como as chaves da assinatura RSA são as mesmas da criptografia RSA, elas são calculadas da mesma forma que foi explicada anteriormente. Como a finalidade da assinatura não é proteger a mensagem e sim garantir a origem, então não precisamos assinar a mensagem toda, mas apenas algo que produza o mesmo resultado. Sendo assim, neste caso iremos assinar um hash (resumo) da mensagem, pois, se o hash for bem feito, não se consegue gerar intencionalmente outra mensagem com o mesmo hash. Para assinar uma mensagem Miremos inicialmente calcular H = hash(M) e a assinatura da mensagem será elevado a D módulo N. Com isso, teremos um número que representará a assinatura da mensagem. Para autenticar a mensagem (M, A) recebida, sendo Mo conteúdo da mensagem e A a assinatura, iremos calcular H = hash(M), e H2 = (A elevado a E módulo N). Se H e H2 forem iguais, então a mensagem foi autenticada com sucesso, pois este cálculo garante que quem gerou a assinatura realmente possuía a chave privada.

A implementação deste trabalho utiliza a estrutura proposta no algoritmo RSA de criptografia assimétrica. E foi desenvolvido utilizando a linguagem de programação Python 3.9.0 e tem as principais funcionalidades do protocolo, que são criptografia, assinatura digital, transferência de chaves públicas e privadas de criptografia, além de outras funcionalidades como compressão, geração de chave de criptografia e armor (transformação dos dados para a base64 e adição de algumas informações extras na mensagem).

O programa inicia gerando a chave privada, e a chave pública que é depende da primeira. Através da função:

def gerar\_chave(tamanho\_chave):  
 random\_generator = Random.new().read  
 key = RSA.generate(tamanho\_chave, random\_generator)  
 chave\_privada, chave\_publica = key, key.publickey()  
 return chave\_publica, chave\_privada

De posse destas, a chave pública pode ser distribuída, devendo a chave privada ficar em sigilo.

O método padrão de assinar um arquivo com RSA é calcular um hash do arquivo (por exemplo, MD5 ou SHA-1). Com a chave privada o remetente assina digitalmente suas mensagens/arquivos, podendo enviar aos destinatários os documentos com as propriedades da criptografia do RSA.

Neste programa, a função assinar\_arquivo mostrada abaixo, recebe com parâmetro a mensagem, a chave privada e a função hash a ser utilizada. O retorno da função é a assinatura digital (hash) gerada.

def assinar\_arquivo(message, priv\_key, hashAlg="SHA-256"):  
 global hash  
 hash = hashAlg  
 signer = PKCS1\_v1\_5.new(priv\_key)  
 if (hash == "SHA-512"):  
 digest = SHA512.new()  
 elif (hash == "SHA-384"):  
 digest = SHA384.new()  
 elif (hash == "SHA-256"):  
 digest = SHA256.new()  
 elif (hash == "SHA-1"):  
 digest = SHA.new()  
 else:  
 digest = MD5.new()  
 digest.update(message)  
 return signer.sign(digest)

A função verificar\_assinatura, recebe a mensagem, a assinatura atual e a chave pública do remetente.

def verificar\_assinatura(message, signature, pub\_key):  
 signer = PKCS1\_v1\_5.new(pub\_key)  
 if (hash == "SHA-512"):  
 digest = SHA512.new()  
 elif (hash == "SHA-384"):  
 digest = SHA384.new()  
 elif (hash == "SHA-256"):  
 digest = SHA256.new()  
 elif (hash == "SHA-1"):  
 digest = SHA.new()  
 else:  
 digest = MD5.new()  
 digest.update(message)  
 return signer.verify(digest, signature)

Para verificar a assinatura é novamente gerado o hash do arquivo, e então tenta decifrar, com a chave pública, o hash cifrado. Como resultado da função verificar\_assinatura e realizada a comparação do hash original do arquivo com o novo hash. Caso sejam iguais, tem a certeza de que a mensagem não foi alterada durante a transmissão.

Dessa forma, as propriedades de autenticidade e integridade estão garantidas, com um overhead computacional menor do que se tivesse cifrado a mensagem completa.

Referências:

https://repositorio.ufu.br/bitstream/123456789/12511/1/Diss%20Winicius.pdf

https://www.cavalcantetreinamentos.com.br/blog/material-sala-de-aula/Seguranca%20em%20Redes/Outros/Comandos%20OpenSSL.txt

https://www.diegomacedo.com.br/assinatura-e-certificacao-digital/

https://gist.github.com/dennislwy/0194036234445776d48ad2fb594457d4