Trabalho de Implementação 2
Gerador e Verificador de assinatura

Gabriel Martins de Almeida, 190013371
Pedro de Tôrres Maschio, 190018763

[1]CIC0201 - Segurança Computacional - T01

1. Introdução

RSA é um dos primeiros sistemas de criptografia de chave pública, surgiu devido à necessidade de se comunicar de forma segura em um canal inseguro, nesse sistema a chave de criptografia é pública, ao contrário da chave de descriptografia, que é secreta. Esse modelo é descrito como cifração assimétrica, devido à característica de ser trivial calcular a função em uma direção e extremamente complexa na direção contrária, mas esse sistema não é eficaz para cifração de grandes mensagens, logo é utilizado em conjunto com uma cifração simétrica para aumentar a eficiência do algoritmo.

De forma simplificada esse modelo funciona da seguinte forma: cada indivíduo terá uma chave pública e uma privada, para se comunicar com este indivíduo será necessário utilizar uma função para calcular o hash da mensagem, gerar uma chave simétrica e a utilizará para criptografar a mensagem. Posteriormente será utilizada a chave pública do destinatário para cifrar o hash obtido e a chave simétrica, o alvo da mensagem receberá a mensagem cifrada com a, o hash calculado através da mensagem e a chave simétrica cifrada pelo, então utilizara a sua chave privada para descriptografar a chave simétrica e o hash, com esses dados o indivíduo utilizará a chave simétrica para descriptografar a mensagem, calcular o hash da mensagem recebida e comparar com o hash recebido, pois qualquer alteração na mensagem irá também alterar o hash gerado.

2. Criptografia

2.1. Criptografia simétrica

A criptografia utilizada neste trabalho será o padrão Advanced Encryption Standard (AES), este algoritmo funciona da seguinte forma, primeiro será necessário dividir a mensagem em blocos de 16 bytes, onde teremos um bloco de 128 bits ($16 * 8 = 128$). Em seguida, será necessário realizar a expansão de chave aplicando a ”Rijndael’s key schedule” e a chave expandida obtida será utilizada em passos futuros. Após isso, será realizado uma soma entre os blocos da mensagem e da chave inicial gerando um novo bloco de bytes.

O algoritmo AES realiza o passo de substituição, trocando cada byte com um código de acordo com uma tabela pré-estabelecida chamada Rijndael S-box. No passo de mudança de linhas, o algoritmo muda as linhas dos blocos obtidos durante o processo de substituição, deslocando a primeira linha para a segunda, a segunda para a terceira, etc. a última linha será deslocada para a primeira, gerando um processo parecido com um deslocamento logico a direita. Em seguida, o próximo passo é misturar as colunas, este passo é realizado multiplicando cada coluna por uma matriz predefinida, gerando um novo bloco de código. Então, o último passo consiste em adicionar a chave expandida obtida nos passos inicias ao bloco que obtivemos no passo anterior.

Em seguida, o algoritmo irá repetir os passos citados anteriormente por muitas rodadas, dependendo do comprimento da chave: 128 bits = 10 rodadas, 192 bits = 12 rodadas e 256 bits = 14 rodadas. Por fim, o algoritmo realizará mais uma rodada, mas não passara pela etapa de substituição de bytes, devido a não gerar grandes mudanças no resultado e demandar grande poder de processamento.

Como a criptografia AES utiliza um modelo simétrico, ele usa a mesma chave para criptografia de dados e descriptografia. Então, a descriptografia AES começa com a chave inversa e o algoritmo inverte cada ação até decifrar a mensagem.

2.2. Criptografia assimétrica

A criptografia RSA é baseada na premissa de que o algoritmo será fácil de calcular em uma direção, mas quase impossível ao contrário, esse comportamento é obtido ao multiplicar dois grandes números primos, é trivial verificar que P * Q = N, mas é inviável a partir de N dizer quais primos foram utilizados para obter o número, isso se deve ao "problema da fatoração do inteiro" que torna inviável resolver esse problema com os recursos computacionais atuais.

O primeiro passo para criptografar uma mensagem com RSA é gerar as chaves. Para isso, precisamos de dois números primos (P e Q) que serão selecionados com um algorítimo de primalidade, os números P e Q precisam ser suficientemente grandes e relativamente distantes para garantir a eficiência do algorítimo, posteriormente será necessário calcular P * Q = N.

Uma vez que temos esses valores, devemos calcular o Totient de Carmichael descrito pela seguinte equação: $\lambda(N) = mmc(p-1,\, q-1)$, também é necessário gerar a chave pública, como é um valor público não é tão importante ser um número aleatório e é geralmente definido por E = 65.537. Sendo assim, após calcular esses dados é possível utilizar a equação: $Ciphertext = Message^E \, mod(N)$ para obter a mensagem cifrada.

Para gerar a chave privada, devemos calcular o inverso modular de E = 65.537 com o valor o obtido da equação de Totient de Carmichael para encontrar D, descrito pela seguinte equação $D = 1/E \; mod \; \lambda(N)$, após encontrar esse valor podemos descriptografar as mensagens que foram criptografas com a chave pública seguindo a seguinte fórmula $Mesagem = C^D \; mod(N)$.

3. Implementação

A implementação geral do sistema de verificação de assinaturas foi desenvolvido na linguagem de programação Python, em sua versão 3.10.4. Esta linguagem foi escolhida por prover suporte nativo à computação de grandes números inteiros (big ints), possuir muitas bibliotecas nativas para manipulação de arquivos, conversão entre tipos e clareza sintática.

3.1. Implementação do AES

O desenvolvimento do AES foi baseado no documento FIPS 197 do NIST, que pode ser encontrado aqui: `https://csrc.nist.gov/publications/detail/fips/197/final`. Assim como descrito na seção 2.1, o algoritmo AES trabalho por meio de várias rodadas, aplicando diversas operações em uma matriz de bytes de tamanho 4x4. O nome de cada operação - que foi implementada em forma de função em Python está descrito na Figura 1. O processo de decifração consistem em inverter os passos relacionados em cada operação. Abaixo está uma explicação de cada trecho da implementação

- **sub_bytes**: Neste passo, recebemos a matriz 4x4 com os dados e os substituímos pelo seu representante na S-box.
- **shift_rows**: Este passo consiste em fazer um *left shift* na matriz recebida. A primeira linha não é considerada, mas as linhas 2, 3 e 4 são deslocadas 2, 3 e 4 vezes à esquerda, respectivamente.
- **mix_cols**: O passo mais complexo do AES, não é aplicado nas linhas, mas sim nas colunas. Os bytes de cada coluna são submetidos a substituições por equações do campo de Galois. As tabelas que foram utilizadas nesse passo foram obtidas aqui: `https://en.wikipedia.org/wiki/Rijndael_MixColumns`.
- **add_key**: Neste passo, após a matriz ter passado por diversas operações, é feito um XOR bytes a byte com a chave de criptografação.

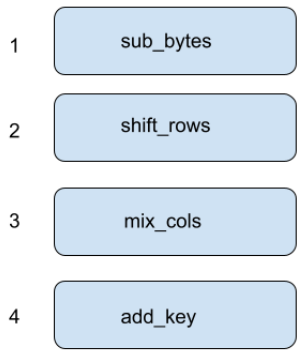

Figure 1. Passo utilizados para cifrar uma mensagem com o AES

Além desses passos, é preciso considerar duas situações: o que acontece se a mensagem não for múltiplo de 16, e o que acontece se a mensagem for maior que a chave?

Para resolver essas situações, duas estratégias são utilizadas:

- **Padding**: existem diversas técnicas para realizar o padding, que consiste na inserção de bytes na mensagem original (no início, meio ou fim). De modo a torná-la múltipla de 16. Para este trabalho, utilizamos o PKCS 7, que consiste na insersão de n bytes de valor n ao final da mensagem.
- **Expansão de chave**: consiste em aplicar diversas operações na matriz que contém a chave de modo a fazer várias matrizes, que serão utilizadas nos passos anteriores.

3.1.1. AES CTR

O AES pode ser implementado com diversos ”modos”. Para este trabalho, foi utilizado o modo Counter (CTR). Que utiliza um Initializing Vector (iv), também chamando de nonce. Durante o processo de criptografação e descriptografação, faremos a operação XOR dos blocos com o IV criptografado. Após isso, esse IV será incrementado um byte, por isso é chamado counter. Para o AES CTR, chamamos a função *encrypt* tando para criptografar tanto para descriptografar.

3.2. Implementação do RSA

O processo de implementação do RSA é consideravelmente mais simples do que o AES. Para implementá-lo, faz-se necessário a geração de dois números primos, p e q, de 1024 bits. Algoritmos mais simples, como crivo de Eratóstenes e métodos mais triviais se tornam inviáveis em termos de tempo de computação e complexidade. Para resolver este problema, utilizamos o teste de primalidade Miller-Rabin.

O teste de Miller-Rabin funciona por meio de rounds, por ser um método probabilístico, não teremos certeza que o número é de fato primo. Mas saberemos que a probabilidade de ele não ser primo é muito baixa, isso é suficiente para o algoritmo.

Conforme explicado anteriormente, cada elemento que quer se comunicar com o RSA possui uma chave pública e uma chave privada. A chave pública é composta pelo par $(p * q, d)$, em que d é o $MMC(d, phi)$, em que phi é igual a $(p - 1) * (q - 1)$. E a chave privada é composta pelo par (n, e), em que é é um número primo, que em nossa implementação é igual a 65537.

Para criptografar/descriptografar a mensagem com o RSA, tranformamos ela em um inteiro e então aplicamos a operação $mensagem^k mod\ v$, em que k é igual a d ou e, a depender de estar utilizando a chave pública ou privada, e em que v é igual a n.

Em nosso trabalho foi preciso utilizar um esquema de *padding* chamado OAEP (Optimal Asymmetric Encryption Padding), que foi padronizado na RFC 2437. O objetivo de utilizar esse esquema de *padding* em conjunto com o RSA é adicionar um elemento de aleatoriedade no processo de criptografação e descriptografação. De forma semelhante ao AES, o OAEP consiste em uma série de passos que são aplicados na mensagem. De modo geral:

- Para a criptografação, o OAEP aplica as operações e depois criptografamos com o RSA normalmente;
- Para a descriptografação, primeiro descriptografamos com o RSA e depois removemos os paddings da mensagem descriptografa.

Mais detalhes do algoritmos OAEP implementado podem ser obtidos aqui `https://www.rfc-editor.org/rfc/rfc8017#section-7.1`.

4. Conclusão

A realização deste trabalho foi bastante desafiadora no sentido de compreender bem o funcionamento de uma assinatura digital. Também foi difícil lidar com diferentes formados de entrada e da conversão entre os tipos lidos, p.e: entre bytes para int, de int para bytes novamente e também a utilização da biblioteca base64 para codificação e decodificação dos arquivos lidos. De modo geral foi interessante aprender como a comunicação segura pela internet se tornou possível a partir dos avanços nos métodos de criptografia assimétricos.