Trabalho segurança

Manuela Matos Correia de Souza - 16/0135281
Ian Nery Bandeira - 170144739

Universidade de Brasília

1 Geração de chaves

A geração das chaves RSA foi realizada conforme a especificação do algoritmo, através da função `genKey`. Essa função executa o teste de primalidade de Miller-Rabin em inteiros escolhidos aleatoriamente no intervalo $(2^{511}, 2^{512})$ até que dois primos (p e q) sejam encontrados, para garantir que n possui 1024 bits. Esses primos definem a primeira parte da chave pública: $n = pq$. A segunda parte da chave pública é o inteiro $e = 65537$, definido dessa forma por ser o maior primo de Fermat.

Com os valores de p e q, definimos a função totiente de Euler $\phi(n) = (p-1)(q-1)$ e calculamos d de tal forma que d seja o inverso multiplicativo de $e \mod \phi(n)$. Para calcular d, usamos o algoritmo de Euclides expandido, já definido para o teste de primalidade de Miller-Rabin. Como decisão de projeto, para caso d seja negativo, o d resultante é $(d + \phi(n)) \mod \phi(n)$.

A chave privada é composta dos primos p, q e do inteiro d. Salvamos a chave pública e privada em um arquivo .json.

2 Assinatura e verificação

Uma vez que as chaves foram geradas, podemos assinar um documento usando o algoritmo RSA. Para construir a assinatura, usamos a função `sign`, que recebe como argumento a mensagem que queremos assinar em formato de string. Fazemos o hash dessa mensagem utilizando a função de hash SHA3. Em seguida fazemos o padding do resultado usando a função `padded`, que implementa o OAEP. O resultado do padding é transformado em um inteiro k e submetido à encriptação RSA utilizando a chave privada (n, d): $assinatura = k^d \mod n$.

Para verificar uma assinatura de um texto, utilizamos a função `verify`, que recebe como entrada a assinatura s e a mensagem m. Essa função executa a desencriptação RSA de s utilizando a chave pública: $s^e \mod n$. Em seguida, converte o resultado de inteiro para uma string e desfaz o padding utilizando a função `unpadded`. Para que a assinatura seja válida, o resultado desse processo deve ser igual ao hash da mensagem, que obtemos a partir da mensagem de passada como argumento.

3 OAEP

Antes de gerar a assinatura do documento, submetemos a mensagem hasheada a um algoritmo de padding. Esse procedimento é realizado para inserir aleatoriedade no processo de cifração, uma vez que o algoritmo RSA sozinho é determinístico e, portanto, inseguro.

No OAEP, utilizamos duas funções hash H e G e dois ko e $k1$, onde $k0$ é o tamanho de uma string composta apenas pela repetição do caractere '0' e k1 é o tamanho de uma string de caracteres aleatórios. Na nossa implementação, fizemos $H = G = SHA3$ e $k0 = k1 = 8$. A escolha do tamanho de $k0$ e $k1$ foi aleatória, apenas com a condição de serem múltiplos de 8. Para realizar o padding, utilizamos a função `padded` e para desfazê-lo utilizamos a função `unpadded`. Essas funções funcionam conforme a especificação do algoritmo OAEP.