Trabalho de Implementação 2
Gerador/Verificador de Assinaturas

Fernanda Macedo de Sousa, 17/0010058
Oscar Etcheaverry Barbosa Madureira da Silva, 17/0112209

[1]Dep. Ciência da Computação – Universidade de Brasília (UnB)
CIC0201 - Segurança Computacional - Turma B - 2021.2

Resumo. *Este relatório apresenta a descrição do processo de implementação de um gerador e verificador de assinaturas RSA em arquivos, como segundo trabalho da disciplina de Segurança Computacional.*

1. Introdução

Um dos mecanismos de segurança utilizados atualmente é a assinatura digital. Segundo [Brown and Stallings 2017], uma assinatura digital refere-se a dados anexados a uma unidade de dados ou transformação criptográfica de uma unidade de dados que permita que um receptor da unidade de dados prove a origem e a integridade da unidade de dados e se proteja contra falsificação (por exemplo, pelo receptor).

No processo de geração e verificação de assinaturas RSA em arquivos é possível explorar diversos conceitos aprendidos na disciplina como cifração simétrica e assimétrica, além do uso de funções hash e o processo de assinatura digital.

1.1. Objetivos

O presente projeto tem como objetivo a implementação de um gerador e verificador de assinaturas RSA em arquivos. Essa implementação divide-se em três partes: Parte I — Geração de chaves e cifra simétrica; Parte II — Assinatura; Parte III — Verificação.

2. Implementação

O programa foi implementado utilizando a linguagem de programação Python, pois esta linguagem oferece diversas facilidades, como operações de inteiros grandes, facilidades de execução, dentre outras. A versão do Python utilizada foi a 3.8.5.

Os pontos chaves da implementação foram a implementação do algoritimo RSA (Rivest-Shamir-Adleman) e a implementação do algoritimo AES (Advanced Encryption Standard).

O projeto foi estruturado em seis arquivos Python: `aes.py`, que contém o algoritmo AES; `constant.py` com os vetores auxiliares para a implementação do algoritmo AES; `rijndael_finite_number.py` que contém a implementação do grupo de Rijndael (utilizado para o AES); `rsa.py` com o algoritmo do RSA; `util.py` com algumas funções úteis para inversão modular e exponenciação de valores modulares; e por fim, o arquivo `main.py`, que contém o fluxo de execução do programa, onde há a conexão dos demais arquivos do projeto, além do tratamento de interação com o usuário.

As subseções a seguir apresentam informações sobre a compilação e execução do projeto, além da explicação sobre a implementação dos algoritmos utilizados no gerador e verificador de assinaturas RSA.

2.1. Compilação e Execução

Para executar a simulação primeiro abra dois terminais, onde serão executados o código do servidor e o código do cliente. Para executar o código servidor é necessário o comando `python main.py -1` e para executar o código cliente é necessário o comando `python main.py -2`.

2.2. Gerador e Verificador de Assinaturas

Passos necessários para a implementação do gerador e verificador:

- O servidor deve gerar a chave pública e a chave privada, e em seguida, enviar a chave pública para o cliente, conforme ilustra a Figura 1;
- O cliente deve gerar a chave de sessão e enviar a chave cifrada para o servidor, usando o RSA e a chave pública, de acordo com a Figura 2;
- O usuário 1, faz o hash da mensagem, assina com a RSA e a chave adequada, além de enviar a mensagem cifrada com AES e a chave de sessão para o usuário 2; já o usuário 2 decifra a mensagem com AES e a chave de sessão, faz o hash da mensagem decifrada e verifica se é correspondente como o hash decifrado que foi enviado pelo usuário 1; os papéis dos usuários 1 e 2 podem ser tanto do servidor e do cliente, respectivamente, e vice-versa. Conforme apresentam as Figuras 3 e 4.

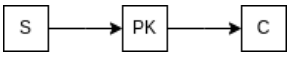

Figura 1. Servidor enviando chave pública para o cliente.

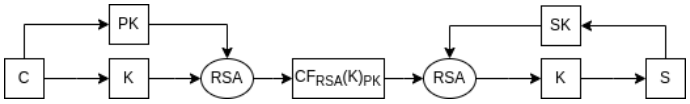

Figura 2. Cliente enviando chave do AES cifrada pelo algoritmo RSA para o servidor.

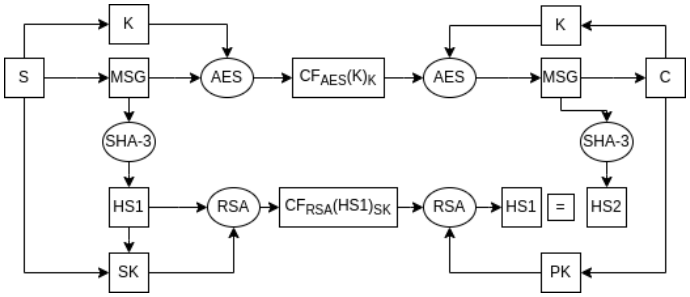

Figura 3. Servidor enviando mensagem e hash para ser recebido, decifrado, e ter o hash verificado pelo cliente.

2.3. RSA

O RSA é um algoritmo de cifração assimétrica baseado em teoria dos números. A sua segurança advém do uso de números muito grandes, o que impossibilita a quebra de

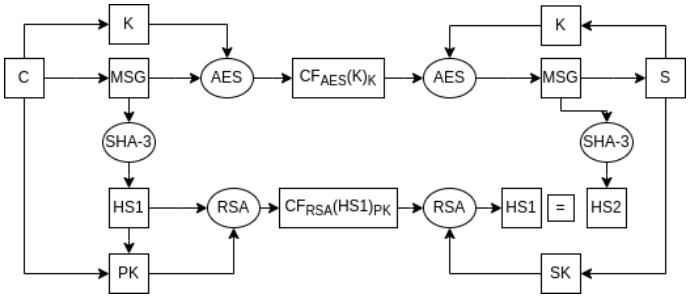

Figura 4. Clente enviando mensagem e hash para ser recebido, decifrado, e ter o hash verificado pelo servidor.

chave devido ao atual poder computacional dessa geração de computador [Geeks 2021] [Wikipedia 2022b].

O primeiro passo para realizar esse algoritmo foi entender e implementar o teste de primalidade Miller-Rabin, teste que permite verificar se um número é primo com um bom grau de confiabilidade. Esse teste é uma extensão do teorema de Euler que diz que $a^{\varphi(n)} \equiv 1 \pmod{n}$, onde $\varphi(n)$ é a função totiente de Euler (função que conta o número de coprimos a n, que estão entre 1 e n). Portanto, a função `rsa_key_generator` chama a rotina `prime_generator` para as variáveis p e q, esta rotina por sua vez utiliza o teste Miller-Rabin para gerar um possível primo aleatório.

Na função `rsa_key_generator` faz-se `n = p * q` e `phi_n = (p - 1) * (q - 1)`, em seguida é testado para cada candidato para a variável `e` um número aleatório que pode ser inversível módulo `phi_n`, se for inversível coloca-se o inverso em `d`; assim retornando `(e, n)` para ser a chave pública e `(d, n)` a chave privada. As variáveis foram nomeadas de acordo com os slides da disciplina.

Para cifrar e decifrar foram usadas as funções `rsa_encript` e `rsa_decript`, respectivamente. Estas funções são semelhantes, elas recebem a mensagem como um vetor de bytes e o transforma em um número inteiro, fazem a exponenciação deste número com o expoente da chave módulo "n" da chave.

2.4. AES modo CTR

O AES é um algoritmo de cifração simétrico de bloco. O modo CTR é um dos modos de operações de cifras de bloco do qual um valor aleatório somado ao índice do bloco da mensagem é cifrado, e esse valor cifrado é utilizado para fazer um xor com a mensagem. Assim, utilizando do AES no modo CTR, por causa das propriedades do xor, basta somente fazer o algoritmo de cifração do bloco do AES [AppliedGo] [Geeks 2022] [Wikipedia 2022a].

O processo de cifração do AES é dividido em quatro partes:

- SubBytes (função `sub_bytes`): substitui os bytes de um bloco por outro que está em uma tabela pré-calculada;
- ShiftRows (função `shift_rows`): para cada linha i de 0 até 4, rotacionar a linha i em i bytes.
- MixColumns (função `mix_columns`): multiplica (faz transformação linear de) cada coluna do bloco por uma matriz (2), no campo finito de Rijndael;

- AddRoundKey (função `round_key`): faz o xor da chave estendida com o bloco.

As regras são aplicadas da seguinte forma (na função `encrypt_block`), primeiro a chave é estendida, em seguida é aplicada AddRoundKey, e em seguida 9 rodad as das partes SubBytes, ShiftRows, MixColumns e MixColumns de forma consecutiva, e por fim, aplicar SubBytes, ShiftRows e AddRoundKey, respectivamente.

A função `ctr_encrypt_decrypt` realiza a cifração e decifração AES no modo CTR.

3. Conclusão

Em vista dos objetivos apresentados na seção 1.1, foi possível implementar as três partes envolvidas em um gerador e verificador RSA, exceto o OAEP. Portanto, apesar das dificuldades encontradas e limitações do trabalho, diversos conceitos da disciplina foram melhor consolidados, uma vez que esse trabalho proporcionou uma experiência prática com cifração simétrica, cifração assimétrica e assinatura digital.

Dos pontos necessários para a avaliação do trabalho, foram completados os seguintes itens:

- geração de chaves com teste de primalidade (Miller-Rabin)
- cifração e decifração RSA
- formatação/parsing
- AES modo CTR

Referências

[AppliedGo] AppliedGo. Aes rijndael cipher explained as a flash animation, youtube. https://youtu.be/gP4PqVGudtg.

[Brown and Stallings 2017] Brown, L. and Stallings, W. (2017). *Segurança de computadores: princípios e práticas*, volume 2. Elsevier Brasil.

[Geeks 2021] Geeks, G. F. (2021). Rsa algorithm in cryptography — geeks for geeks. `https://www.geeksforgeeks.org/rsa-algorithm-cryptography/`. [Online; acessado em 30 de abril de 2022].

[Geeks 2022] Geeks, G. F. (2022). Advanced encryption standard (aes) — geeks for geeks. `https://www.geeksforgeeks.org/advanced-encryption-standard-aes/`. [Online; acessado em 30 de abril de 2022].

[Wikipedia 2022a] Wikipedia (2022a). Advanced encryption standard — wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Advanced_Encryption_Standard`. [Online; acessado em 30 de abril de 2022].

[Wikipedia 2022b] Wikipedia (2022b). Rsa (cryptosystem) — wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/RSA_(cryptosystem)`. [Online; acessado em 30 de abril de 2022].