Advanced Encyption Standard (AES) e Rivest–Shamir–Adleman cryptosystem (RSA) - Projeto 2 da Disciplina CIC0201 (Segurança Computacional)

Maria Eduarda Santos - 190092556[1], Kléber Rodrigues da Costa Júnior - 200053680[1],

[1]Departamento de Ciência da Computação (CIC) – Universidade de Brasília (UnB)
Brasília, DF – Brasil

{190092556, 200053680}@aluno.unb.br

1. Introdução

Advanced Encryption Standar (AES) e Rivest-Shamir-Adleman (RSA) são algoritmos cruciais no campo da criptografia, cada um com suas próprias características, propriedades matemáticas e áreas de aplicação. A escolha entre eles depende do contexto específico e dos requisitos de segurança de um sistema ou aplicação. No presente descritivo, serão detalhados os processos de implementação de ambos métodos criptográficos, além das particularidades solicitadas no desenvolvimento de cada atividade.

2. Cifra AES

A cifra AES (Advanced Encryption Standard) é um algoritmo de criptografia simétrica amplamente utilizado para proteger informações confidenciais, selecionada pelo Instituto Nacional de Padrões e Tecnologia (NIST) dos Estados Unidos como padrão de criptografia para uso governamental e confidencial[Miller et al. 2009]. Amplamente aplicado em várias áreas da segurança da informação, tais como nos protocolos de segurança TLS/SSL de comunicação entre cliente e servidor, no armazenamento de dados em disco rígido, na segurança de dados em dispositivos móveis e no contexto governamental e militar, a cifra AES é construída a partir de vários processos que asseguram a segurança da informação, sendo eles:

Tamanho da chave e blocos: a princípio, a mensagem é formatada para que seja possível quebrá-la em blocos de 128 *bits* de mensagem (o grau de segurança da cifra depende do tamanho em que a mensagem é inicialmente quebrada, podendo este tamanho ser dado por 192 *bits* e também 256 *bits*), caracterizando estas partes como *block cipher*;

Subchaves: em seguida, faz-se necessários realizar a geração de subchaves a partir da chave originalmente utilizada, realizando, assim, a expansão da chave para aplicação nos blocos de 128 *bits*. Existirão, então, n subchaves, sendo n o número de etapas no processo de cifração, além de uma chave adicional;

Etapas de Cifração: A partir do número de *bits* definido para a quebra da mensagem em *block ciphers*, realiza-se uma quantidade k de rotinas pré-definidas por **SubBytes**, **ShiftRowsMixColumns** e **AddRoundKey** em cada bloco criado a partir da mensagem original, fazendo uso de cada subchave gerada inicialmente para cada bloco de mensagem

SubBytes: a partir do *block cipher* gerado, cada elemento do bloco (sendo 16 deles, ao todo) é substituído por um correspondente, de acordo com uma tabela pré-determinada de substituição (S-BOX), gerando assim um bloco com diferentes valores;

ShiftRows: a partir do novo *block cipher*, as linhas do bloco de informação

sofrem a aplicação de movimentos circulares dos dados ali contidos, fazendo uma operação de *shift* ao longo do bloco

MixColumns: esta etapa carateriza a criação de um novo *block cipher* a partir da multiplicação das colunas do bloco gerado pela etapa anterior com uma matriz fixa de referência. No projeto, esta matriz é dada a partir do *grupo de Galois de Rijndael*, para evitar que ataques conhecidos na área da criptografia sejam aplicados e a mensagem seja descoberta

AddRoundKey: por fim, o *block cipher* gerado a partir da etapa **MixColumns** passa pelo processo de realizar a operação **XOR** em cada um dos elementos do bloco com a subchave gerada para a rodada correspondente do processo, criando assim um bloco de mensagem criptografado

Durante a cifração, o processo AES repete as etapas mencionadas acima por 10 vezes, sendo a etapa **MixColumns** não aplicada na última rodada, e gera, assim, uma cadeia de blocos de mensagem que compõem a mensagem criptografada. O processo de decifração é semelhante ao de criptografar a mensagem, mas as etapas são executadas na ordem inversa e as subchaves são aplicadas, também, em ordem reversa.

2.1. Implementação

A implementação da cifra AES é dada pela linguagem *Python* de programação, e é distribuída ao longo dos arquivos do programa. A mensagem a ser cifrada é colocada em um arquivo externo, e então, o processo de cifração ou decifração é iniciado, dependendo da escolha do usuário.

Para realizar a cifração, uma variável auxiliar `counter` foi instanciada, com o propósito de realizar a aplicação da chave no seu correspondente *block cipher*. A chave inicialmente utilizada é dada pela geração de um número primo de 1024 *bits* (gerado a partir do método `get_prime_n_bits`), e trocada por seu *hash* correspondente (implementado pelo método `hash_128_bits`). Ambos `counter` e `chave` são armazenados em um arquivo, a ser utilizado no processo de decifração, e então, a mensagem original é preenchida com a quantidade de *bytes* necessários para se aplicar a quebra da informação em blocos de 128 *bits*.

Cada bloco gerado é transformado em um número inteiro, e então, a partir da cifração da variável auxiliar counter (agindo como subchave, como citado anteriormente), sendo esta cifração dada pelo método AES, a operação **XOR** é aplicada entre o *block cipher* e a variável auxiliar cifrada, gerando uma mensagem que será concatenada à mensagem final. Por fim, a variável auxiliar é incrementada em uma unidade (a fim de resgatar a próxima subchave do próximo bloco de mensagem), e ao final do *loop* de iteração que caracteriza as rodadas de cifração, a mensagem criptografada é armazenada em um arquivo de saída, para análise.

O método `aes.encrypt()` aplicado à variável auxiliar `counter` implementa a cifra AES no arquivo `aes.py`, o qual implementa a inicialização da classe `AES` com a criação de subchaves para aplicação ao longo das rodadas de cifração; o método de criptografia composto de 9 rodadas de cifração a partir da matriz gerada pela mensagem dada como parâmetro, e uma última rodada que não contempla a criação de um novo *block cipher* gerado a partir da multiplicação da matriz do *grupo de Galois de Rijndael*.

3. Cifra RSA

A cifra RSA (Rivest-Shamir-Adleman) é um dos algoritmos de criptografia assimétrica [Diffie and Hellman 2022] mais amplamente utilizados, inventado

por Ron Rivest, Adi Shamir e Leonard Adleman em 1977 e se baseia em operações matemáticas envolvendo números primos[Rivest et al. 1978]. O algoritmo de cifração é dado por:

Geração de chaves: escolhe-se dois números primos grandes distintos, p e q, calcula-se o produto $n = p \times q$, que será o módulo para operações criptográficas, calcula-se o valor da função *totiente* de Euler de n (denotado por $\phi(n)$, onde $\phi(n) = (p-1) \times (q-1)$), escolhe-se um número inteiro e primo relativo a $\phi(n)$, que será o expoente público, e calcula-se o inverso multiplicativo de $e \mod \phi(n)$, denotado por d, que será o expoente privado

Cifração: o remetente codifica a mensagem em um valor numérico $M < n$, eleva M à potência e, e obtém a mensagem criptografada C, onde $C = M^e \mod n$, e então, a mensagem C é enviada ao destinatário

Decifração: o destinatário recebe a mensagem C e aplica a operação inversa à de cifração, elevando C à potência de d, obtendo, então, o valor original de M, sendo $M = C^d \mod n$

A segurança do algoritmo RSA é baseada na dificuldade de fatorar números inteiros grandes em seus fatores primos, uma vez sendo computacionalmente inviável (em tempo razoável) determinar d conhecendo apenas e e n. A chave pública consiste no par (e, n), que pode ser amplamente divulgado, enquanto a chave privada consiste no valor d, que deve ser mantido em segredo.

A cifra RSA é amplamente aplicada em diferentes áreas, tais como confidencialidade de informação transmitida em rede pública, assinaturas digitais, gerenciamento de chaves utilizadas em outros algoritmos de criptografia assimétrica e protocolos de segurança. Podendo demonstrar lentidão na computação de fatores primos grandes que preservam a segurança da informação, a cifra RSA é frequentemente utilizada em conjunto com outros algoritmos criptográficos para combinar vantagens.

O projeto implementa a cifra RSA fazendo uso do teste de primalidade de Miller-Rabin ([Miller 1975][Rabin 1980]) para números primos de 1024 *bits*, além de incrementar à mensagem original a técnica OAEP de cifração [Bellare and Rogaway 1995], a fim de aumentar a segurança da criptografia assimétrica e fornecer proteção contra ataques de texto cifrado escolhido. Ainda, o projeto aplica o método RSA de criptografia em assinatura digital, e faz a verificação de autenticidade do mesmo em um conjunto de operações a serem descritas na seção 3.1[Goldreich 2009].

3.1. Implementação

A implementação da cifra foi feita na linguagem de programação *Python*, e é dada em etapas: geração de chaves públicas e privadas de 1024 *bits*, cifração e decifração usando a cifra RSA e a codificação OAEP, assinatura e verificação de mensagens criptografadas que utilizam a cifra RSA. O processo de geração de chaves públicas e privadas se dá a partir da seleção de dois números primos (p, q), os quais são selecionados a partir da geração aleatória de números no intervalo $[1 \ll 1024 - 1, 1 \ll 1024]$, sendo $\ll$ a operação de multiplicação por 2 a partir de um número k. Após selecionados, os números passam pelo teste de primalidade de Miller-Rabin, e caso o teste seja positivo, (p, q) são escolhidos como primos que compõem a construção das chaves públicas e privadas.

Em seguida, a mensagem passa pelo método `rsa.OAEPencrypt()`, o qual se encarrega de formatar e codifi-

car a mensagem a ser cifrada no formato OAEP de cifração. Este tipo de cifração é dado pela geração de um *hash*, uma semente para auxiliar a mascarar a mensagem (a qual foi composta de diferentes partes de dados, incluindo o acréscimo de 0' na composição da mensagem para que esta tenha o tamanho necessário para aplicação do método OAEP), e a concatenação destas informações para gerar a mensagem criptografada. Após, a decifração da mensagem é feita, e então, a demonstração da assinatura e verificação de assinaturas é dada a partir da geração de um *hash*, utilizado na cifração da mensagem fazendo uso do método OAEP, e então, a mensagem final é rescrita no formato BASE64, como solicitado na especificação do projeto. Por fim, o processo reverso é feito, a fim de demonstrar o processo de verificação de assinatura, e caso algum problema tenha acontecido no processo de verificação de assinatura, a mensagem original não será revelada, e um `erro de verificação` será lançado.

4. Conclusão

Por fim, é possível concluir que ambas cifras desenvolvidas no projeto garantem segurança e integridade das informações por elas protegidas. Enquanto o AES é amplamente utilizado para criptografia de dados em geral, o RSA é frequentemente utilizado para operações específicas, como troca de chaves e assinaturas digitais. O AES é conhecido por sua alta velocidade e eficiência na criptografia e descriptografia de grandes volumes de dados, enquanto o RSA é conhecido por sua aplicação em cenários onde é necessário compartilhar informações de forma segura. A implementação do projeto pode ser acessada na plataforma GitHub, e ainda está sujeita a futuras modificações e implementações de métodos que asseguram um processo de geração de mensagens cifradas poderoso e indecifrável.

Referências

Bellare, M. and Rogaway, P. (1995). Optimal asymmetric encryption. In *Advances in Cryptology—EUROCRYPT'94: Workshop on the Theory and Application of Cryptographic Techniques Perugia, Italy, May 9–12, 1994 Proceedings 13*, pages 92–111. Springer.

Diffie, W. and Hellman, M. E. (2022). New Directions in Cryptography. In Slayton, R., editor, *Democratizing Cryptography*, pages 365–390. ACM, New York, NY, USA, 1 edition.

Goldreich, O. (2009). *Foundations of cryptography. 2: Basic applications*. Cambridge Univ. Press, Cambridge.

Miller, F. P., Vandome, A. F., and McBrewster, J. (2009). *Advanced encryption standard*. Alpha Press.

Miller, G. L. (1975). Riemann's hypothesis and tests for primality. In *Proceedings of the Seventh Annual ACM Symposium on Theory of Computing*, STOC '75, page 234–239, New York, NY, USA. Association for Computing Machinery.

Rabin, M. O. (1980). Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12(1):128–138.

Rivest, R. L., Shamir, A., and Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126.