UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
FACULDADE DE COMPUTAÇÃO
ATIVIDADE ORIENTADA DE ENSINO

GABRIEL BARBOSA ALCÂNTARA
GABRIEL HIROFUMI OKANO

CONCEITOS BÁSICOS DE CRIPTOGRAFIA

Campo Grande - MS
4 de dezembro de 2023

GABRIEL BARBOSA ALCÂNTARA
GABRIEL HIROFUMI OKANO

CONCEITOS BÁSICOS DE CRIPTOGRAFIA

Atividade orientada de ensino do Curso de Engenharia da Computação da Faculdade de Computação da Universidade Federal de Mato Grosso do Sul, como objetivo de enriquecer o conhecimento adquirido durante o curso de graduação

Orientador(a): Dra. Ana Karina Dourado Salina de Oliveira

Campo Grande - MS
4 de dezembro de 2023

Resumo

Esse trabalho tem como objetivo principal apresentar conceitos básicos de criptografia, entre eles: cifra de bloco, modos de operação, criptografia simétrica, criptografia assimétrica, assinatura digital, funções hash. Também são apresentados e descritos os algoritmos mais populares AES e RSA da criptografia simétrica e assimétrica, respectivamente.

Sumário

1 Introdução

A Segurança da Informação desempenha um papel crucial na era digital, onde a crescente troca e armazenamento de dados demandam medidas robustas de proteção. A criptografia emerge como ferramenta essencial para assegurar a confidencialidade e integridade das informações, sendo fundamental para preservar a confiança nas transações, comunicações, na privacidade individual e na proteção contra ameaças cibernéticas, garantindo que apenas usuários autorizados tenham acesso aos dados sensíveis. Essa prática, codifica dados, garantindo acesso autorizado e resguardando a confiança no ambiente digital.

Como pilar fundamental da Segurança da Informação, a criptografia consiste na prática de codificar e decodificar dados para proteger sua confidencialidade e autenticidade. Essa ciência, baseada em algoritmos matemáticos complexos, é crucial para preservar a privacidade e segurança das informações no cenário digital em constante evolução, onde a troca e armazenamento de dados ocorrem exponencialmente.

Com o aumento exponencial do uso de dispositivos com recursos limitados, como sensores IoT (Internet das Coisas) e dispositivos embarcados, a necessidade de garantir a segurança desses sistemas tornou-se urgente. Esses dispositivos, muitas vezes com capacidades de processamento e armazenamento restritas, demandam soluções de criptografia específicas para garantir a proteção adequada dos dados que manipulam. Tais algoritmos são adaptados para implementação nesses ambientes limitados, como sensores, cartões inteligentes, dispositivos de cuidados de saúde, entre outros, uma vez que implementar algoritmos tradicionais acaba se tornando uma tarefa desafiadora. [1]

2 Cifras de Bloco

Cifra de Bloco é uma função $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$. Essa notação significa que temos uma string formado de 0 ou 1 com tamanho k, outra string de tamanho n, e retorna uma string de tamanho n.

Para cada chave $K \in \{0,1\}$ e mensagem $M \in \{0,1\}$ temos uma função $E_K(M) = E(K,M)$. É necessário que essa função seja uma permutação, ou seja, essa função é reversível. Portanto existe apenas uma inversa E_K^{-1} tal que $E_K^{-1}(K, E(K,M)) = M$. [2][3]

Em termos mais simples, cifras de blocos dividem uma mensagem em pequenos pedaços, de forma a garantir que esses pedaços sempre tenham o mesmo tamanho, para que possam ser processados.

Abaixo exemplos de algoritmos que utilizam cifra de bloco: [4]

- **Lucifer/DES** - Lucifer foi o precursor do DES (*Data Encryption Standard*), criado nos anos 70 e utilizado até os anos 90, quando ficou obsoleto devido ao pequeno tamanho da chave, de apenas 56 bits. No final dos anos 90, o *National Institute of Standards and Technology* (NIST) promoveu uma competição para substituí-lo.

- **Rijndael** - Cifra de bloco vencedor da competição do NIST, uma versão modificada é usada no AES (*Advanced Encryption Standard*). Mais detalhes serão vistos em 4.1.

- **Serpent** - Finalista da competição, ficou em segundo lugar. Entre os finalistas, era o mais seguro, porém mais lento que o Rijndael.

- **IDEA** - Inicialmente criado para substituir o DES, o IDEA (*International Data Encryption Algorithm*) foi utilizado no *Pretty Good Privacy* (PGP). Sua adoção foi restrita devido ao uso de patentes, que expiraram em 2012.

3 Modos de Operação

Na criptografia de cifras de blocos, os dados são divididos em múltiplos blocos, e cada bloco é encriptado individualmente. Se os dados forem insuficientes para preencher um bloco, utiliza-se *padding*. Um modo de operação descreve como aplicar uma operação de cifra de bloco repetidamente, de forma a processar dados maiores que um bloco.

3.1 *Electronic Codebook* (ECB)

O *Electronic Codebook* (ECB) é o modo de operação mais simples. Os blocos são encriptados usando a mesma chave, porém se múltiplos blocos forem encriptados utilizando a mesma chave, então partes repetidas geram o mesmo resultado, formando padrões na saída, o que permite analisá-los mesmo sem o conhecimento da chave.

Esse problema é evidente na criptografia de imagens, a Figura 1 ilustra um exemplo no qual as cores são perdidas, porém o layout da figura ainda é compreensível. Isso se deve ao fato de múltiplos blocos, como o fundo branco e a parte laranja, são repetidos múltiplas vezes, gerando os mesmos blocos de saída. Apenas as bordas entre o branco e laranja aparentam estar distorcidas, que foi onde não se repetiram padrões.

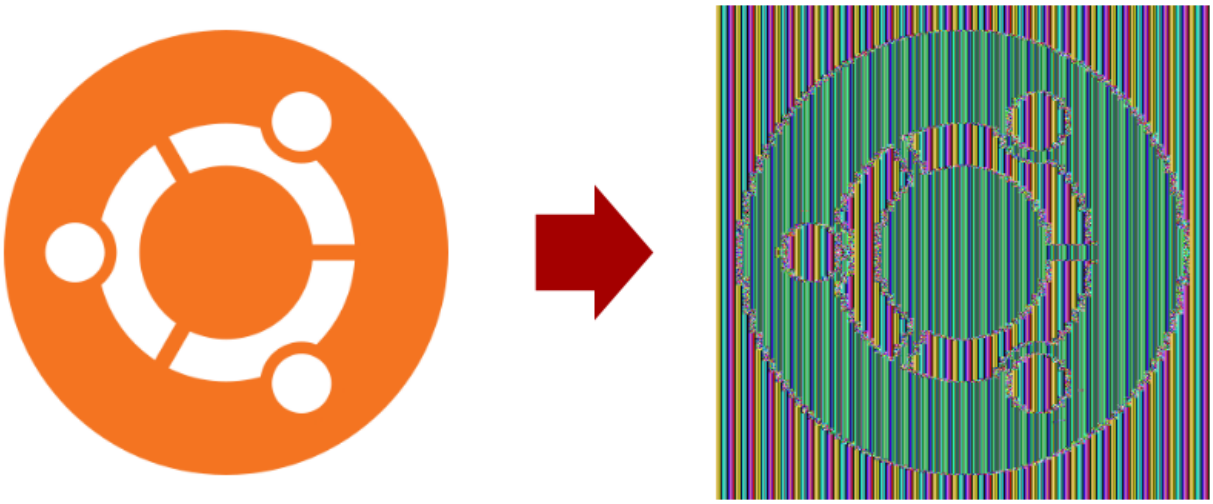

Figura 1: ECB[5]

Esse problema é chamado de falta de difusão, portanto é necessário uma forma de gerar resultados diferentes caso dois blocos sejam iguais.

3.2 *Cipher Block Chaining* (CBC)

Cipher Block Chaining é, como o próprio nome sugere, um encadeamento de blocos, conforme ilustrado na figura 2.

- Inicialmente, um valor aleatório, conhecido como *nonce* ou *Initial Vector* (IV), é gerado;
- É feito um XOR com o primeiro bloco da mensagem m_0, encriptando-o em seguida;
- Em seguida, é feito um XOR entre o texto cifrado gerado na saída $c_0 = \text{encrypt}(m_0 \oplus \text{nonce})$ e o próximo bloco m_1;
- Este processo é repetido em todos os blocos seguintes.

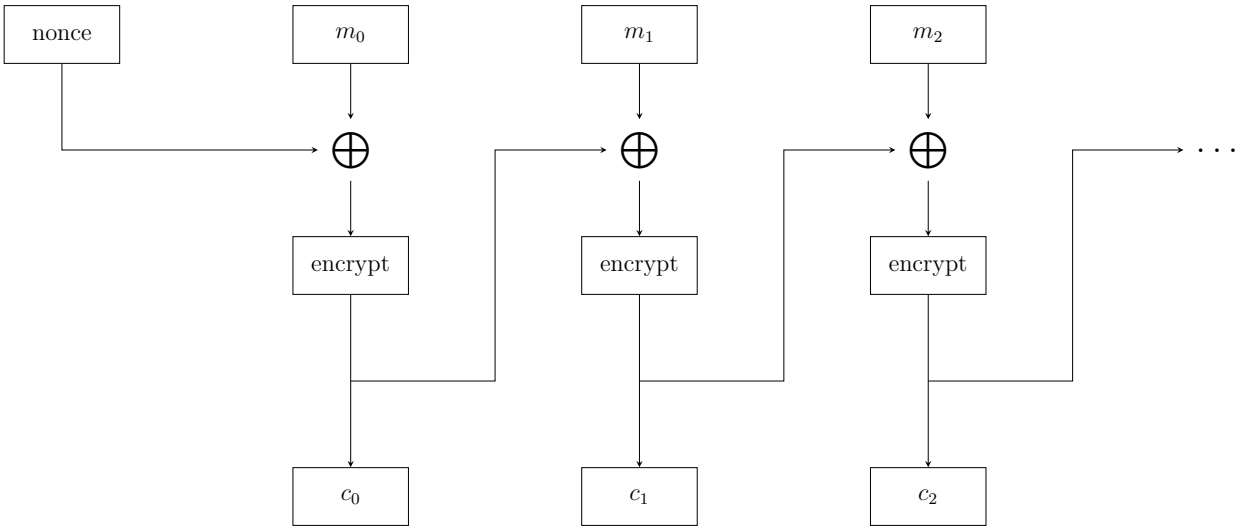

Figura 2: Diagrama de Blocos CBC

A principal desvantagem desse modo de operação é a lentidão. Observe que todo o processo é linear, para obter os blocos seguintes é necessário computar os blocos anteriores, o que também impede a paralelização.

3.3 *Counter Mode* (CTR)

Counter Mode é essencialmente transformar uma cifra de bloco em cifra de fluxo, encriptando-o em seguida.

- Um *nonce* é gerado, junto com um contador *counter* inicializado em zero;
- Esses valores são concatenados e encriptados com a função $encrypt(nonce \mid\mid counter)$;
- O resultante é feito um XOR com seu respectivo bloco, gerando o texto cifrado de saída;
- O contador é incrementado, e o processo se repete nos blocos seguintes.

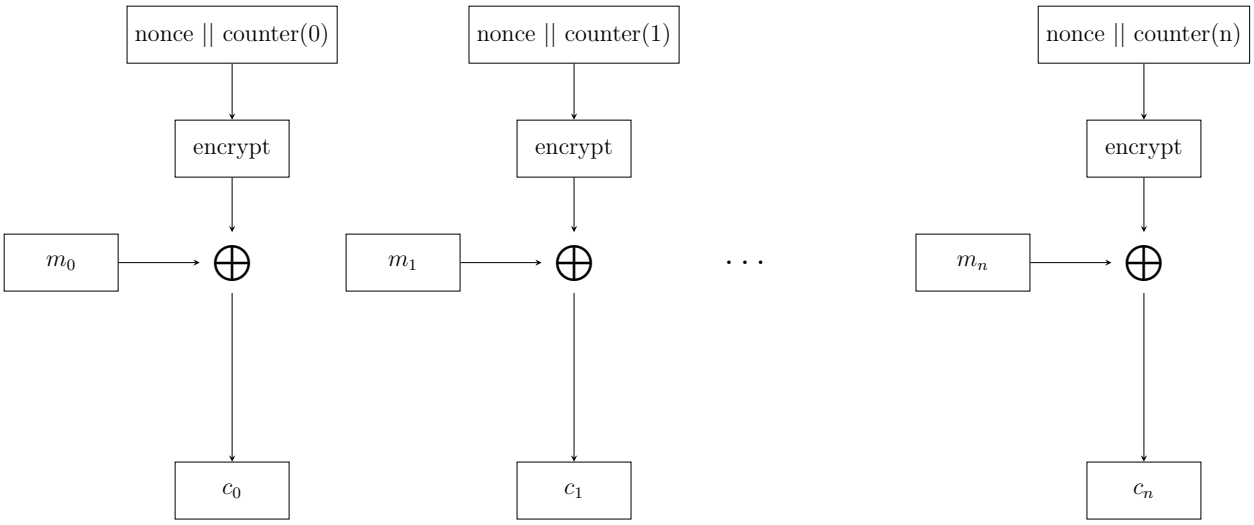

Figura 3: Diagrama de Blocos CTR

Esse modo de operação possui melhor desempenho, pois não há dependência entre as etapas, tornando o processo paralelizável. É importante notar que o tamanho de bits do contador precisa ser grande o suficiente; se usarmos um contador de 32 bits, após 2^{32} iterações, o contador reseta. Se estivermos usando um contador de 32 bits e uma cifra de bloco de 128 bits, não será seguro encriptar mais que $128 \times 2^{32} = 64$ GB de dados. [2][3][5]

4 Criptografia Simétrica

O esquema de criptografia simétrica permite a comunicação segura entre duas partes, que compartilham da mesma chave K, utilizada para cifrar e decifrar uma mensagem M. Conforme ilustrado na Figura 4, Alice envia a mensagem "Hello Word", encriptando-a com uma chave K, que será decriptada por Bob usando a mesma chave. Esse processo é idêntico no reverso; se Bob enviar uma mensagem para Alice, o processo será o mesmo. Se a mensagem for interceptada por Eve (eavesdropper), ela não conseguirá entender o conteúdo da mensagem.

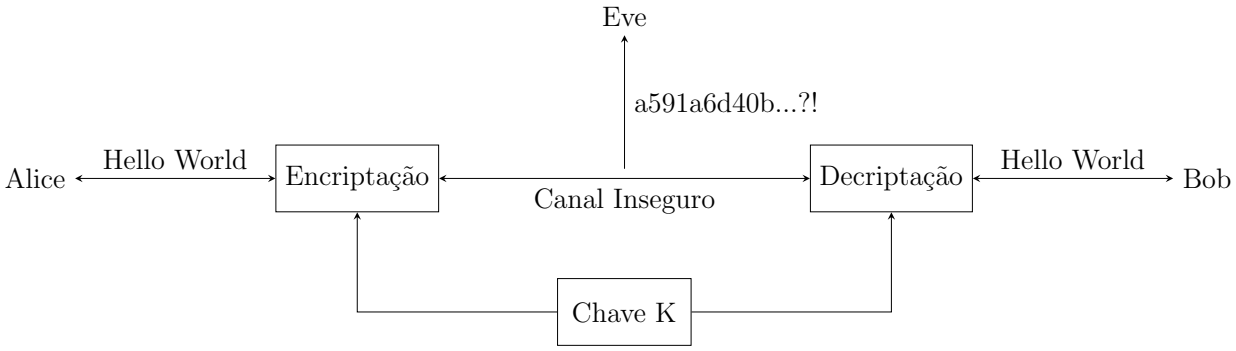

Figura 4: Alice e Bob se comunicam com a mesma chave K

A criptografia simétrica fornece o serviço de confidencialidade e tem a vantagem de ser computacionalmente mais eficiente em comparação com a criptografia assimétrica, devido à realização de operações mais rápidas.[2][3]

4.1 AES (Advanced Encryption Standard)

O algoritmo de *Rijndael* é especificado como uma cifra de bloco simétrico capaz de processar blocos de dados de 128 bits, utilizando chaves de cifra com comprimentos de 128, 192 e 256 bits. Ele também pode lidar com tamanhos de bloco e comprimentos de chave maiores, no entanto, estes não são abordados.

A entrada e saída do algoritmo consistem em sequências de 128 bits. A chave de cifra para o algoritmo AES é uma sequência de 128, 192 ou 256 bits. Outros comprimentos de entrada, saída e chave cifrada não são permitidos por este padrão.

A chave será utilizada na geração de outras chaves, que por sua vez serão empregadas nas rodadas, conforme ilustrado no diagrama de blocos da Figura 5.

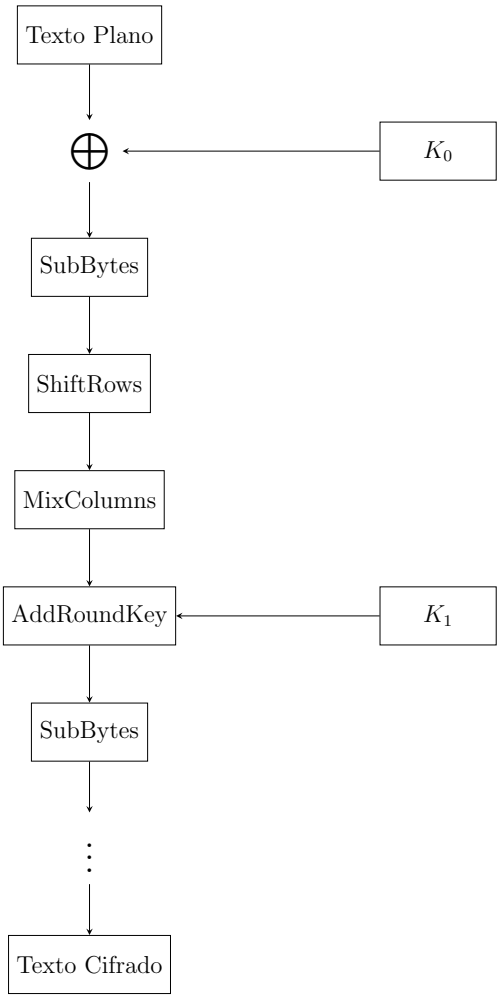

Figura 5: Diagrama de Blocos AES

Inicialmente, a mensagem será dividida em blocos de 16 bytes organizados em uma matriz

4x4 de 128 bits:

B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

Tabela 1: *CipherBlock*

Essa matriz passará pela operação XOR com a chave respectiva da rodada.

4.1.1 SubBytes

O valor resultante do $CipherBlock \bigoplus K_n$ será substituído pelo valor correspondente do Rijndael S-box lookup table.

4.1.2 ShiftRows

Cada linha da matriz terá um shift à esquerda, de acordo com o número de sua linha $0 \leq n \leq 3$. Ou seja, o valor do shift será $S' = S(n, :) << n$, resultando na matriz:

B_0	B_4	B_8	B_{12}
B_5	B_9	B_{13}	B_1
B_{10}	B_{14}	B_2	B_6
B_{15}	B_3	B_7	B_{11}

4.1.3 MixColumns

Cada coluna passa pela seguinte multiplicação de matrizes:

$$\begin{bmatrix} b'_{0,c} \\ b'_{1,c} \\ b'_{2,c} \\ b'_{3,c} \end{bmatrix} = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} b_{0,c} \\ b_{1,c} \\ b_{2,c} \\ b_{3,c} \end{bmatrix}$$

para $0 \leq c \leq 3$, último round não possui MixColumns.

4.1.4 AddRoundKey

Recebe a chave para a próxima rodada e realiza uma operação XOR com a saída do passo anterior.

Vale notar que os processadores mais recentes já possuem o AES implementado em hardware. Por exemplo, um AMD Ryzen 5 5600X, que tem um preço sugerido de 300 USD, é capaz de atingir 10,6 GB/s. [6][7]

5 Criptografia Assimétrica

Um dos desafios da criptografia simétrica é o compartilhamento da chave de forma segura. Esse problema é resolvido com os pares de chaves pública e privada, também conhecido como criptografia assimétrica. No esquema de criptografia assimétrica, é gerado um par de chaves, onde cada chave pública tem uma chave privada correspondente. A chave pública é revelada publicamente, enquanto a chave privada é mantida em segredo. Quando uma chave é usada para cifrar, a outra é usada para decifrar. Para garantir a confidencialidade, utiliza-se a chave pública do destinatário para cifrar a mensagem. Já para a assinatura, emprega-se a chave privada do remetente para cifrar a mensagem.[2][3]

Na Figura 6, Alice envia a mensagem "Hello World"para Bob, cifrando-a com a chave pública de Bob. Ao receber a mensagem, Bob utiliza sua chave privada para decifrar, garantindo assim o serviço de confidencialidade.

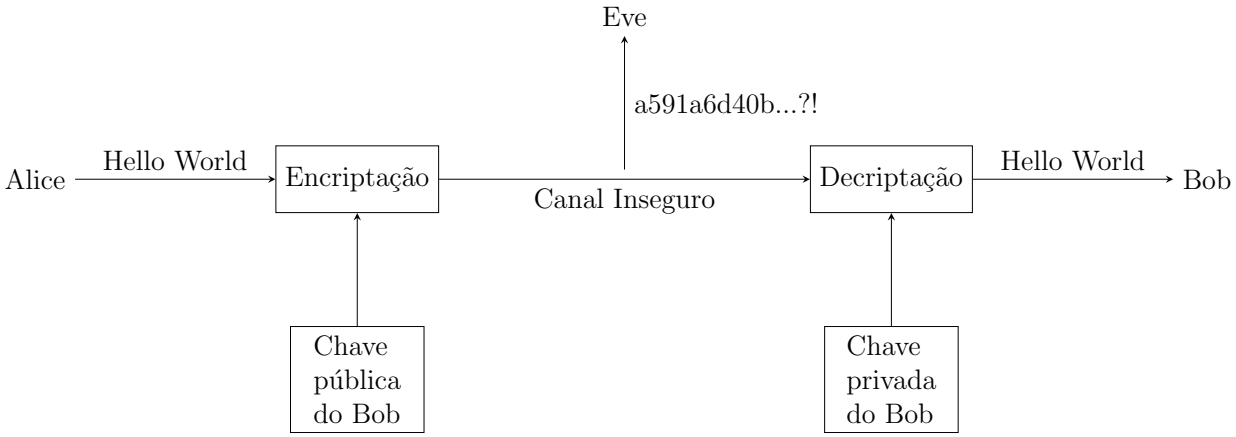

Figura 6: Alice envia mensagem para Bob

Na Figura 7, podemos observar o processo de assinatura, onde Bob deseja garantir a autenticidade da mensagem que enviou. Nesse caso, Bob cifra a mensagem com sua chave privada, e Alice decifra a mensagem utilizando a chave pública de Bob.

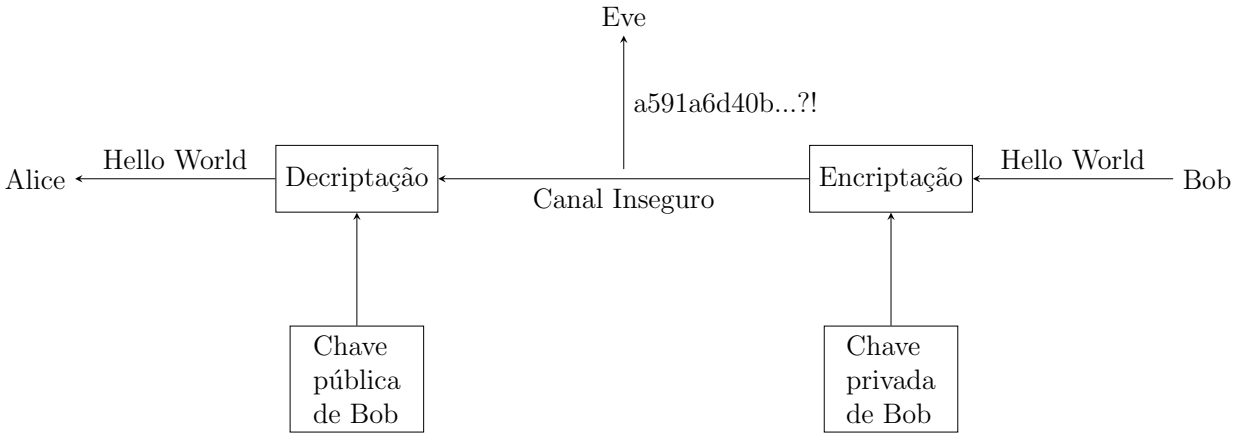

Figura 7: Bob assina a mensagem com sua chave privada.

5.1 RSA (Rivest–Shamir–Adleman)

Williams Stanley Jevons escreveu em 1874: "O leitor pode dizer que dois números multiplicados juntos produzirão o número 8.616.460.799? Acho improvável que alguém além de mim venha a saber", a resposta é 89681×96079, ambos números primos. A razão dele ter escrito isso se deve ao fato de que é extremamente difícil determinar os fatores de números grandes, quando este é o produto de dois números primos, e essa é a idéia por trás do RSA. [8]

A função φ de Euler conta quantos números inteiros k entre $1 \leq k \leq n$ possuem $\text{mdc}(k, n) = 1$, uma observação interessante é que se n for um número primo, então $\varphi(n) = n - 1$. Essa função pode ser fatorável em $\varphi(p * q) = \varphi(p) * \varphi(q)$, portanto se ambos p e q forem primos temos que $\varphi(p * q) = (p - 1) * (q - 1)$.

O teorema de Euler diz que se tivermos dois inteiros m e n de forma que $\text{mdc}(m, n) = 1$, então:

$$m^{\varphi(n)} \equiv 1 \pmod{n}$$

Se elevarmos a uma potência $k \in \mathbb{Z}$, $k > 0$:

$$m^{k*\varphi(n)} \equiv 1^k \pmod{n}$$

E multiplicarmos por m:

$$m * m^{k*\varphi(n)} \equiv m * 1^k \pmod{n}$$

Temos então que:

$$m^{k*\varphi(n)+1} \equiv m \pmod{n}$$

Vamos então chamar dois números primos e e d para encriptação e decriptação respectivamente, de forma que:

$$m^{ed} \equiv m \pmod{n}$$

Portanto temos a equivalência

$$d = \frac{k * \varphi(n) + 1}{e}$$

Isso significa que só é possível decriptar uma mensagem se souber a fatoração de n, o único jeito de quebrar essa criptografia será por força bruta, e se esse for um número suficientemente alto, só será possível quebrá-lo por meio de computadores quânticos que (ainda) não existem.

O algoritmo é dividido em três componentes, o algoritmo de geração de chaves, o algoritmo de criptografia e o algoritmo de descriptografia. Os algoritmos a seguir, ilustram esses três componentes.

Algoritmo de Geração de chaves RSA
1: Gere 2 números primos aleatorios suficientemente grandes p e q
2: Computar $N = p * q$
3: Computar $T = (p - 1) * (q - 1)$
4: Escolher dois inteiros E e D de forma que $(E * D) \bmod T = 1$
5: (N,E) são liberados como chave pública
6: (N,D) são mantidos em segredo como chave privada

Algoritmo de Criptografia RSA
Entrada: A chave pública (N, E)
Saída: Um texto cifrado C
1: Selecionar um inteiro M, onde $0 < M < N$ satisfaça $\text{mdc}(M, N) = 1$
2: Calcular $C \equiv M^E \pmod{N}$
3: Saída do texto cifrado C

Algoritmo de Descriptografia RSA
Entrada: Um texto cifrado C e a chave privada (N, D)
Saída: Um texto simples M
1: Calcular $M \equiv C^D \pmod{N}$
2: Saída do texto simples M

6 Funções Hash

As funções de hash são funções unidirecionais, existindo assim somente uma operação possível e não reversível, o cálculo do hash.

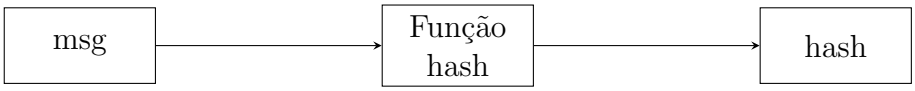

Figura 8: Diagrama de blocos de uma função hash

A mensagem é utilizada como entrada para a função de hash que então retorna o valor do hash, ou apenas o hash da mensagem.

$$h = H(msg)$$

Uma vez calculado o hash é impossível, a partir do hash, recuperar a mensagem original *msg*. Não importando o tamanho ou formato da mensagem *msg*, o hash calculado terá sempre um tamanho fixo em bits (tipicamente 128, 160, 256 ou 512 bits). Ou seja, ao utilizar uma função hash de 256 bits, uma mensagem *msg* de 320KB ou uma mensagem de 8GB, terão valores de hash com o mesmo tamanho de 256 bits.[2][3]

Toda função hash é necessário apresentar essas três propriedades:

- Resistência pré-imagem

  É necessário ser computacionalmente inviável reverter uma função hash, ou seja, dado um h e uma função $h = H(msg)$ será inviável encontrar o valor msg que gerou h. Isso protege de um ataque em que o atacante possui apenas o hash e está procurando o valor que gerou esse hash.

- Resistência de segunda pré-imagem

  Dada uma entrada x e um hash $H(x)$, será computacionalmente inviável encontrar um y, tal que $H(x) = H(y)$. Caso um atacante possua uma entrada e um hash, essa propriedade impede que a entrada original seja substituída.

- Resistência de colisões

  Dada uma função hash, deve ser computacionalmente inviável encontrar duas entradas diferentes x e y tal que $H(x) = H(y)$. Essa propriedade garante que as colisões ainda são possíveis, mas são extremamente improváveis de ocorrer. Além disso, se uma função possui resistência de colisões, logo ela também possui resistência de segunda pré-imagem.

6.1 *Secure Hash Algorithms* (SHA)

Secure Hash Algorithms são uma família de algoritmos hash, que incluem: [3]

- **SHA-1** - Função hash de 160 bits, baseado no MD5, desenvolvido pelo *National Security Agency* (NSA) em 1995. O tamanho de 160 bits é considerado curto, portanto deixou de ser utilizado após 2010.

- **SHA-2** - Família de funções hash SHA-256 e SHA-512, mais populares no momento. São utilizados em diversas aplicações e protocolos como TSL, SSL, SSH, *Bitcoin*, entre vários outros. Diferem apenas no tamanho de bloco, o SHA-256 tem bloco de 32 bit e gera uma saída de 256 bits, enquanto que o SHA-512 tem bloco de 64 bits e gera uma saída de 512 bits. Também desenvolvido pelo NSA em 2001.

- **SHA-3** - Função hash conhecida anteriormente como *Keccak*, vencedor de uma competição do NIST em 2012. Estrutura interna difere significativamente dos seus predecessores, chamado de função esponja. [9]

7 Assinatura Digital

Conforme foi explicado na seção anterior, as chaves publicas e privadas são inversas entre si, ou seja, se uma for usada para encriptação, a outra será usada para decriptação, e vice-versa. Se a chave privada do emissor for utilizada, o destinatário tem certeza de quem enviou a mensagem, e não foi alterado por um ataque *man in the middle*.

Para garantir a integridade de uma assinatura, pode-se utilizar uma função hash (Resumo) em conjunto com a assinatura. A função hash é utilizada para processar o documento, produzindo um pequeno pedaço de dados, chamado de hash. A função hash gera um resumo de tamanho fixo.

Para se garantir uma assinatura digital tem-se que verificar as seguintes propriedades:

- Um usuário não pode forjar a assinatura de outro usuário e as assinaturas digitais devem ser únicas para cada usuário;

- O remetente de uma mensagem não pode invalidar a assinatura de uma mensagem;

- O destinatário da mensagem não pode modificar a assinatura contida na mensagem;

- Um usuário não pode ser capaz de retirar a assinatura de uma mensagem e colocar em outra.

O esquema de Assinatura Digital calcula o hash da mensagem e depois criptografa o hash com a chave privada do emissor. A assinatura digital serve como uma garantia de que o documento é uma cópia verdadeira e correta do original, garantindo também a autoria da mensagem. As assinaturas digitais, assim como outras convencionais, podem ser forjadas, a diferença é que a assinatura digital pode ser matematicamente verificada. [2][3]

8 Considerações Finais

Ao explorarmos os conceitos fundamentais da criptografia simétrica e assimétrica, bem como o funcionamento do algoritmo RSA e a aplicação da assinatura digital, torna-se evidente a importância dessas técnicas para a segurança da comunicação digital.

A criptografia simétrica destaca-se pela eficiência computacional, enquanto a assimétrica, por meio de chaves pública e privada, resolve o desafio do compartilhamento seguro de chaves. O algoritmo RSA, baseado na dificuldade de fatorar números grandes, oferece uma solução robusta, sendo amplamente utilizado para a comunicação segura na era digital.

No contexto da assinatura digital, a combinação de funções hash e criptografia assimétrica proporciona uma maneira confiável de garantir a integridade e autenticidade das mensagens. A capacidade de verificar matematicamente as assinaturas digitais oferece uma camada adicional de segurança.

Por fim, é notável a evolução tecnológica, com processadores modernos já incorporando implementações de algoritmos criptográficos em hardware, como no caso do AES. Entretanto, permanece o desafio contínuo de acompanhar e desenvolver técnicas de criptografia capazes de resistir a possíveis avanços computacionais, incluindo a chegada dos computadores quânticos.

Diante disso, a compreensão desses princípios e a aplicação adequada dessas técnicas são essenciais para garantir a privacidade, integridade e autenticidade das comunicações em um mundo digital em constante transformação.

Referências

[1] *O que é criptografia de dados? Definição e explicação*. Acesso em 10 de outubro de 2023. URL: https://www.kaspersky.com.br/resource-center/definitions/encryption.

[2] Mihir Bellare e Phillip Rogaway. *Introduction to Modern Cryptography*. 2005.

[3] N. Ferguson, T. Kohno e B. Schneier. *Cryptography Engineering: Design Principles and Practical Applications*. John Wiley Consumer, 2010.

[4] Acesso em 20 de outubro de 2023. URL: https://www.tutorialspoint.com/cryptography/block_cipher.htm.

[5] Acesso em 25 de setembro de 2023. URL: https://andrea.corbellini.name/2023/03/09/authenticated-encryption/.

[6] United States National Institute of Standards e Technology (NIST). "Announcing the ADVANCED ENCRYPTION STANDARD (AES)". Em: *Federal Information Processing Standards Publication 197* (26 de novembro de 2001). URL: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf.

[7] Acesso em 15 de março de 2023. URL: https://www.vortez.net/articles_pages/amd_ryzen_5_5600x_ryzen_7_5800x_review,10.html.

[8] Rivest, Shamir e Adleman. "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems". Em: *Communications of the ACM. 21 (2): 120–126.* (Fevereiro de 1978). URL: https://web.archive.org/web/20230127011251/http://people.csail.mit.edu/rivest/Rsapaper.pdf.

[9] Guido Bertoni et al. *Sponge Functions*. Acesso em 21 de Outubro de 2023. 2007. URL: https://keccak.team/files/SpongeFunctions.pdf.