UFSM

CIÊNCIA e NATURA

ISSN 2179-460X

OPEN ACCESS

Ci. e Nat., Santa Maria, v. 47, e87825, 2025• https://doi.org/10.5902/2179460X87825

Submissão: 29/05/24 • Aprovação: 19/02/25 • Publicação: 11/04/25

Matemática

Análise da complexidade dos algoritmos de Criptografia Homomórfica

Analysis of the complexity of Homomorphic Encryption algorithms

Aline De Lurdes Zuliani Lunkes [I], **Fábio Borges de Oliveira** [I]

[I]Laboratório Nacional de Computação Científica, RJ, Brasil

RESUMO

A Criptografia Parcialmente Homomórfica é importante para a preservação da privacidade no processamento de qualquer informação sensível. Este trabalho apresenta esquemas de Criptografia Parcialmente Homomórfica aplicados a inteiros, os quais suportam apenas uma operação homomórfica, seja aditiva ou multiplicativa. Realizamos a análise das complexidades de tempo computacional associadas às operações de cifragem, decifragem, cálculo do homomorfismo e possíveis ataques a cada esquema. Além disso, comparamos os esquemas estudados, detalhando seus algoritmos e destacando suas principais características de eficiência e segurança.

Palavras-chave: Criptografia Parcialmente Homomórfica; Complexidade computational; Privacidade; Segurança

ABSTRACT

Partially Homomorphic Cryptography plays a crucial role in preserving privacy during the processing of sensitive information. This work presents partially homomorphic encryption schemes applied to integers, which support only one homomorphic operation, either additive or multiplicative. We analyze the computational time complexities associated with encryption, decryption, homomorphic computation, and potential attacks on each scheme. Additionally, we compare the studied schemes, detailing their algorithms and highlighting their key efficiency and security characteristics.

Keywords: Partially Homomorphic Cryptography; Computational complexity; Privacy; Security

1 INTRODUÇÃO

Um esquema de criptografia depende do compartilhamento de uma chave entre os pares envolvidos na troca de uma mensagem. O conceito de homomorfismo foi utilizado inicialmente em criptografia por Rivest, R. L., Adleman, L. & Dertouzos, M. L. (1978), como uma possível solução para a computação sem a necessidade de decifrar os dados. A Criptografia Homomórfica, em inglês Homomorphic Encryption - HE, é o esquema de criptografia que permite a terceiros, nuvens computacionais, executarem determinadas funções computáveis nos dados, preservando seus recursos e o formato dos dados criptografados.

Um esquema de HE é constituído por quatro operações: geração das chaves, cifrar $E(m)$, decifrar $D(E(m))$ e homomorfismo. A geração das chaves retorna ou um par de chaves secretas e públicas para a versão assimétrica de Criptografia Homomórfica ou uma única chave para a versão simétrica. O algoritmo de homomorfismo é uma operação específica na criptografia homomórfica. Ele recebe como entrada os textos cifrados das mensagens m_1 e m_2, representados como $E(m_1)$ e $E(m_2)$, e gera como saída o texto cifrado de uma função f sobre m_1 e m_2, denotado por $E(f(m_1,m_2))$. Assim, a operação f é realizada diretamente sobre as mensagens cifradas, dispensando a necessidade de prévia decifragem. O ponto crucial na HE é a preservação do formato das mensagens cifradas após um processo de homomórfico, garantindo que as mensagens possam ser decifradas corretamente. Além disso, conforme apontado por Halevi (2017), é admissível um crescimento logarítmico do tamanho do texto cifrado em função do número de operações homomórficas realizadas.

Com base nisso, temos que os esquemas de Criptografia Parcialmente Homomórfica - PHE, em inglês *Partially Homomorphic Encryption*, podem ser usados apenas para aplicações cujos algoritmos incluem apenas operações ou aditivas ou multiplicativas. Por exemplo, o esquema de criptografia de Paillier é homomórfico aditivo Paillier (1999), já o esquema de RSA é apenas homomórfico multiplicativo Rivest, R. L., Shamir, A. & Adleman, L. (1978). Algumas das aplicações de PHE são em Votação Eletrônica, Assinatura Digitral, Certificação de Website e Armazenamento de dados em nuvem. O que propomos neste trabalho é a análise das complexidades de tempo das operações de: cifrar, decifrar, homomorfismo e de

ataque dos métodos de Criptografia Parcialmente Homomórfica sobre números inteiros que constam em Acar, A., Aksu, H., Uluagac, A. S., & Conti, M. (2018). Não iremos analisar as complexidades da geração das chaves, pois estas são geradas uma única vez para cada esquema de PHE.

Além disso, os métodos: Goldwasser-Micali Goldwasser, S. & Micali, S. (1982), e KTX Kawachi, A., Tanaka, K., & Xagawa, K. (2007), que constam em Acar, A., Aksu, H., Uluagac, A. S., & Conti, M. (2018) não se enquadram em nossa proposta, pois o homomorfismo desses métodos é baseado a uma operação **XOR**, que não se enquadra na soma de números inteiros. Este trabalho está organizado da seguinte maneira: na seção 2, abordamos uma visão geral de HE. Na seção 3, abordamos com mais profundidade os algoritmos de PHE, discutindo cada um dos métodos de PHE propostos no trabalho Acar, A., Aksu, H., Uluagac, A. S., & Conti, M. (2018). Na seção 4, apresentamos a análise das complexidades dos algoritmos revisados na seção 3. Na última seção discutiremos os resultados obtidos neste trabalho.

2 CRIPTOGRAFIA HOMOMÓRFICA

Nesta seção, exploramos de forma mais detalhada a HE e suas principais propriedades. Além disso, fornecemos uma explicação aprofundada do funcionamento de cada esquema de PHE abordado neste trabalho. A Tabela 1 oferece uma visão geral das propriedades homomórficas de cada algoritmo de Criptografia de Chave Pública (PKC) analisado na seção 3, destacando suas principais características e diferenças.

Tabela 1 – Criptografia de Chave Pública (PKC)

Esquema	Fatoração de Inteiros	Logaritmo Discreto
RSA	X	
Goldwasser-Micali	X	
El-Gamal		X
Benaloh	X	
Naccache-Stern	X	
Okamoto-Uchiyama	X	
Paillier	X	
Damgård-Jurik	X	

Fonte: autores (2024)

Um esquema HE é caracterizado principalmente por quatro operações. Além disso, analisamos qual o melhor ataque para cada esquema. As operações são descritas a seguir:

Geração de Chaves: onde ocorre a geração de chaves secretas e públicas.

Cifrar: denotado por $E(m)$, onde m é a mensagem a ser cifrada.

Decifrar: denotado por $D(E(m))$, onde m é a mensagem a ser decifrada.

Homomorfismo: denotado por $E(m)\square E(m)$, onde m é a mensagem e $\square$ é a operação realizada, que pode ser aditiva ou multiplicativa, dependendo do esquema. Na Tabela 2, detalhamos as operações de homomorfismo executadas em cada esquema discutido neste trabalho.

Por exemplo,

$$\begin{aligned} E(m_1) + E(m_2) &= E(m_1 + m_2), \\ E(m_1) \times E(m_2) &= E(m_1 \times m_2). \end{aligned} \tag{1}$$

A Tabela 2 apresenta as propriedades homomórficas associadas a cada esquema descrito na seção 3.

Tabela 2 – Operações Homomórficas dos esquemas de PHE

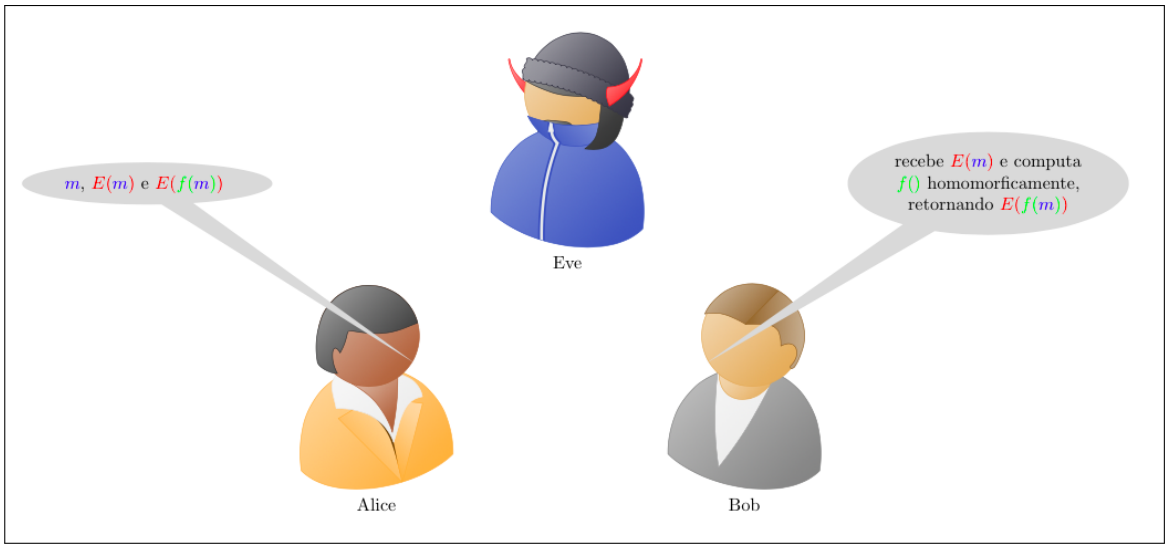
Algoritmos	Adição	Multiplicação
RSA		X
El-Gamal		X
Benaloh	X	
Naccache-Stern	X	
Okamoto-Uchiyama	X	
Paillier	X	
Damgård-Jurik	X	

Fonte: autores (2024)

Na Figura 1, apresentamos um exemplo ilustrativo do processo de homomorfismo. Nesse cenário, Alice deseja enviar uma mensagem para Bob por meio de um canal não seguro, enquanto Eve tenta interceptar a comunicação entre eles. Alice possui a mensagem tanto em sua forma original (pura) quanto cifrada. Bob, por sua vez, utiliza o homomorfismo para realizar operações diretamente sobre o texto

cifrado de Alice, garantindo que a mensagem permaneça protegida, mesmo em um ambiente vulnerável à interceptação. Além disso, ilustramos a resolução de um problema físico utilizando chaves públicas e privadas. Neste processo, o servidor (Bob) realiza o processamento das informações de forma homomórfica, preservando a confidencialidade dos dados. Após o cálculo, o servidor retorna a solução do problema ao cliente (Alice), assegurando a segurança durante todo o procedimento. Assim, a comunicação entre Alice e Bob é protegida por criptografia, garantindo a confidencialidade, autenticidade e integridade das informações, mesmo diante de possíveis ataques de interceptação.

Figura 1 – Cenário onde Alice e Bob trocam mensagem por um canal não seguro



Fonte: autores (2025)

Ataque

O esquema de PHE é adequado para aplicações específicas, sendo utilizado em algoritmos que realizam exclusivamente operações aditivas ou multiplicativas.

3 ALGORITMOS DE PHE

Nesta seção, apresentamos detalhadamente alguns algoritmos de Criptografia Parcialmente Homomórfica (PHE), com base nos conceitos apresentados em Acar, A., Aksu, H., Uluagac, A. S., & Conti, M. (2018). Para cada algoritmo, descrevemos as etapas fundamentais: geração de chaves, cifragem, decifragem, operação homomórfica e possíveis ataques.

3.1 Rivest, Shamir, Adleman (RSA)

Este método é baseado no problema de fatoração por inteiros de produtos de dois números primos grandes Rivest, R. L., Shamir, A. & Adleman, L. (1978). O RSA é o sistema criptográfico assimétrico de chave pública mais conhecido. Como este algoritmo é relativamente lento, é indicado que seus usuários, ao invés de usarem diretamente o RSA para criptografar seus dados, usem-o apenas para transmitir chaves criptográficas cifradas que serão empregadas em algoritmos simétricos. Esses algoritmos simétricos permitem que os processos de cifrar e decifrar ocorram de maneira mais rápida. Abaixo descreveremos o método de RSA.

Geração das Chaves: sejam p e q números primos gerados aleatoriamente, consideramos $n = pq$ e $\phi(n) = (p-1)(q-1)$. Seja e tal que $\text{mdc}(e, \phi(n)) = 1$. Pelo algoritmo de Euclides temos que $d = e^{-1} \mod \phi(n)$. Assim, obtemos as respectivas chaves deste esquema:

A chave pública: (e, n).

A chave secreta: (d, n).

Cifrar: considere m a mensagem a ser cifrada e M, o conjunto de mensagens possíveis de serem cifradas, com $0 \leq m < n$. Utilizaremos o par de chave pública (e, n) para cifrar m

$$E(m) = m^e \mod n, \ \forall \ m \in M. \tag{2}$$

Decifrar: a mensagem m pode ser recuperada usando o par de chave secreta (d, n) da seguinte maneira

$$D(E(m)) = E(m)^d \mod n = m. \tag{3}$$

Homomorfismo: sejam m_1 e m_2 mensagens então,

$$\begin{aligned} E(m_1) \cdot E(m_2) &= (m_1^e \mod n) \cdot (m_2^e \mod n) \\ &= (m_1 \cdot m_2)^e \mod n \\ &= E(m_1 \cdot m_2). \end{aligned} \tag{4}$$

Logo, este método não permite a adição homomórfica em mensagens cifradas, ou seja, é apenas homomórfico sobre a multiplicação.

Ataque: O melhor algoritmo para resolver o problema da fatoração de inteiros é *General Number Field Sieve* (GNFS). Para efetuar a decomposição de n em fatores primos, consultar Lara, P., & Borges, F. (2008); Pandey (2014). Além disso, o tempo de execução deste algoritmo aplicado a um inteiro de n *b*its é

$$O(exp((c+o(1))\sqrt[3]{(\ln n)(\ln \ln n)^2}), \tag{5}$$

onde c é uma constante e n é o número inteiro ímpar que queremos fatorar.

3.2 *Elgamal*

Este esquema é um algoritmo de criptografia assimétrica de chave pública utilizado para o gerenciamento de chaves. Conforme Elgamal (1985) a segurança do algoritmo é derivada da dificuldade de se extrair um logaritmo discreto em corpos finitos. Este algoritmo permite confirmar a autenticidade de uma mensagem enviada, mesmo que tenha sido enviada em um canal não seguro.

Geração das Chaves: seja g um gerador do grupo cíclico $\mathbb{G}$ com módulo p, e $n = pq$, onde p e q são números primos. Em um grupo cíclico, é possível gerar todos os elementos desse grupo usando o seu gerador. Considere $h = g^y \mod p$ onde $y \in \mathbb{Z}_p$ é qualquer, e seja x um elemento aleatório do conjunto $\{1, 2, \ldots, p-1\}$. Assim, obtemos as respectivas chaves deste esquema:

A chave pública: (p, g, h).

A chave secreta: y.

Cifrar: a mensagem m é criptografada usando g (gerador do grupo) e x, e a saída é um par de mensagens cifradas, assim

$$E(m) = (g^x, mh^x) \mod p = (g^x, mg^{xy}) = (E(m)_1, E(m)_2) \mod p. \tag{6}$$

Decifrar: seja $s = E(m)_1^{-y}$ calculado onde y é a chave secreta, assim

$$D(E(m)) = E(m)_2 \cdot s \mod p = mg^{xy} \times g^{-xy} \mod p = m. \tag{7}$$

Homomorfismo: sejam m_1 e m_2 mensagens a serem cifradas, então

$$\begin{aligned} E(m_1) \cdot E(m_2) &= (g^{x_1}, m_1 h^{x_1}) \mod n \cdot (g^{x_2}, m_2 h^{x_2}) \mod n \\ &= (g^{x_1+x_2}, (m_1 \cdot m_2) h^{x_1+x_2}) \mod n \\ &= E(m_1 \cdot m_2). \end{aligned} \tag{8}$$

Portanto, este método permite somente operações multiplicativas.

Ataque: como no RSA, utilizamos o algoritmo GNFS.

3.3 *Benaloh*

Este esquema é uma extensão do criptossistema de Goldwasser, S. & Micali, S. (1982), efetuando um número ilimitado de adições modulares, ou seja, cifra a mensagem como um bloco, em vez de bit a bit, veja Clarkson (1994); Fousse, L., Lafourcade, P., & Alnuaimi, M (2011).

Geração das Chaves: sejam um bloco de tamanho r e os números primos grandes p e q, escolhidos de modo que r divida $p-1$ e $q-1$. Além disso, r deve ser relativamente primo a $(p-1)/r$ e também a $(q-1)/r$, ou seja, $\mathrm{mdc}(r,(p-1)/r) = 1$ e $\mathrm{mdc}(r,(q-1)/r) = 1$. Segundo os autores Fousse, L., Lafourcade, P., & Alnuaimi, M (2011) para decifrar corretamente, devemos considerar $p_i = p_1 p_2 \dots p_k$ fatorado em números primos distintos. Além disso, $y \in \mathbb{Z}_n^*$ é escolhido tal que $y^{\phi/p_i} \not\equiv 1 \mod n$, onde $\mathbb{Z}_n^*$ é o subgrupo multiplicativo de números inteiros módulo n que inclui todos os números menores que r e relativamente primos a r. Então, $n = pq$ e $\phi = (p-1)(q-1)$ são calculados. Assim, obtemos as respectivas chaves deste esquema:

A chave pública: (y, n, r).

A chave secreta: (p, q).

Cifrar: seja $m \in \mathbb{Z}_r$, onde $\mathbb{Z}_r = \{0, 1, \dots, r-1\}$ e escolhendo $u \in \mathbb{Z}_n^*$ um número aleatório temos

$$E(m) = y^m u^r \mod n. \tag{9}$$

Decifrar: este processo ocorre por um processo exaustivo para $i \in \mathbb{Z}_r$ tal que

$$D(E(m)) = (y^{-i}(E(m)))^{\phi/r} = 1 \mod n, \tag{10}$$

onde a mensagem m é retornada com o valor de $m = i$.

Homomorfismo: sejam m_1 e m_2 as mensagens a operar

$$\begin{aligned} E(m_1) + E(m_2) &= (y^{m_1} u_1^r \mod n) + (y^{m_2} u_2^r \mod n) \\ &= y^{m_1+m_2}(u_1 \cdot u_2)^r \mod n \\ &= E(m_1 + m_2). \end{aligned} \tag{11}$$

Logo, esse método é apenas homomórfico aditivo.

Ataque: O Pollard ρ é o algoritmo mais eficiente para o cálculo da decifração, entretanto, a segurança do algoritmo ainda é baseada na fatoração de inteiros, portanto, utiliza-se o GNFS. O melhor algoritmo de ataque para resolver este esquema é Polllard ρ, Teske (1998). Supondo que o represente a ordem do grupo, ou seja, influencia diretamente na complexidade computacional do ataque, que é expressa pela seguinte fórmula:

$$\sqrt{\frac{\pi o}{2}}. \tag{12}$$

3.4 *Naccache-Stern*

Este é um esquema de generalização do sistema de criptografia de *Benaloh* para aumentar sua eficiência computacional Naccache, D. & Stern, J. (1998). Sua segurança se baseia no problema de maior resíduos e funciona no grupo $(\mathbb{Z}/n\mathbb{Z})^*$ onde n é um produto de dois números primos grandes.

Geração das Chaves: considere uma coleção contendo k números primos ímpares pequenos e distintos, dada por $\{p_1, \ldots, p_k\}$. Em seguida, divida esse conjunto ao meio e defina

$$u = \prod_{j=1}^{k/2} p_j, \quad v = \prod_{k/2+1}^{k} p_j.$$

Defina $\sigma = uv = \prod_{j=1}^{k} p_j$. Escolha números primos grandes a e b de modo que $p = 2au + 1$ e $q = 2bv + 1$ sejam ambos primos, e defina $n = pq$. Escolha um inteiro aleatório $g \mod n$ tal que g tenha ordem $\phi(n)/4$. Para cada índice $j \leq k$ escolhemos um número aleatório g_j, que não seja uma potência do p_j-ésimo primo do conjunto inicial, ou seja, $g^{\phi(n)/p_j} \not\equiv 1 \mod n$. Assim, com uma "probabilidade imperativa", isto é, uma probabilidade muito alta, $g = \prod_{j=1}^{k} g_j^{\sigma/p_j}$ terá uma ordem maior ou igual $\frac{\phi(n)}{4}$. Quando $k = 1$, o esquema descrito se reduz ao esquema de criptografia de *Benaloh*, que pode ser visto como um caso particular da construção mais geral apresentada.

A chave pública: (σ, n, g).

A chave secreta: (p, q).

Caso k for um número ímpar, a divisão do conjunto de k números primos ao meio resulta em duas partes que não possuem o mesmo número de elementos. Para corrigir isso, as definições de u e v podem ser ajustadas para lidar com o número ímpar de primos, ou seja,

$$u = \prod_{j=1}^{\lfloor k/2 \rfloor} p_j, \quad v = \prod_{j=\lfloor k/2 \rfloor + 1}^{k} p_j. \tag{13}$$

onde $\lfloor k/2 \rfloor$ representa o piso de $k/2$, garantindo que u contenha $\lfloor k/2 \rfloor$ primos, enquanto v contém os primos restantes do índice $\lfloor k/2 \rfloor + 1$ até k. Assim, todas as k entradas são consideradas adequadamente, mesmo para valores ímpares de k.

Cifrar: seja $m \in \mathbb{Z}/\sigma\mathbb{Z}$. Escolha um inteiro aleatório x tal que $x \in \mathbb{Z}/n\mathbb{Z},$ então

$$E(m) = x^{\sigma} g^{m} \mod n. \tag{14}$$

Decifrar: considere $m \mod p_j$ para cada j, e $E(m_j) = E(m)^{\phi(n)/p_j} \mod n = g^{m_j \phi(n)/p_j} \mod n$, em seguida, aplicamos o Teorema do Resto Chinês para calcular m_j e p_j, obtemos assim,

$$D(E(m)) = (E(m_j))^{\phi(n)/p_j} \mod n = g^{m_j \phi(n)/p_j} \mod n. \tag{15}$$

onde $m_j = m \mod p_i$. Como p_j é escolhido para ser pequeno, o m_j pode ser recuperado por pesquisa exaustiva, ou seja, comparando $E(m_j)$ para $g^{z\phi(n)/p_j}$ para $z \in \{1, \ldots, p_j - 1\}$. Obtendo m_j para cada j, segue que m pode ser recuperado aplicando diretamente o Teorema do Resto Chinês.

Homomorfismo: sejam m_1 e m_2 mensagens, então

$$\begin{aligned} E(m_1) + E(m_2) &= (x^\sigma g^{m_1} \mod n) + (x^\sigma g^{m_2} \mod n) \\ &= g^{m_1+m_2} x^\sigma \mod n \\ &= E(m_1 + m_2). \end{aligned} \tag{16}$$

Ataque: como nos métodos anteriores, para resolver o problema da fatoração de inteiros utilizamos GNFS.

3.5 *Okamoto-Uchiyama*

Em Okamoto, T. & Uchiyama, S. (1998) aparece a proposta de um novo esquema de PHE para melhorar o desempenho computacional, alterando o conjunto onde as criptografias dos esquemas anteriores de HE funcionam e preservam o domínio dos esquemas citados anteriores. Este esquema trabalha no grupo multiplicativo de números inteiros módulo N, $(\mathbb{Z}/N\mathbb{Z})^*$, com N definido como $N = p^2q$, onde p e q são números primos grandes.

Geração das Chaves: escolha um número inteiro aleatório $g \in \{2, \ldots, n-1\}$ de modo que $g_p = g^{p-1} \mod p^2$ é p. Calcule $h = g^N \mod N$, onde h é um parâmetro para melhorar o desempenho de cifrar, desde que h possa ser calculado facilmente para g e N.

A chave pública: (N, g, h).

A chave secreta: (p, q).

Cifrar: utilizando a chave pública, considere $m < p$ e escolha um número inteiro aleatório $r \in \{1, \ldots, N-1\}$ tal que

$$E(m) = g^m h^r \mod N. \tag{17}$$

Decifrar: considere o logaritmo discreto (Borges, F., Portugal, F. & Lara, S. (2007)) com base g^{p-1}, o grupo $(\mathbb{Z}/N\mathbb{Z})^* \simeq (\mathbb{Z}/p^2\mathbb{Z})^* \times (\mathbb{Z}/q\mathbb{Z})^*$ e L é um isomorfismo entre dois grupos (grupo cíclico H ao grupo aditivo $\mathbb{Z}/p\mathbb{Z}$), ou seja, $L(u) = (u-1)/n$ para cada u do subgrupo $\mathbb{Z}^*_{p_2}$.

$$D(E(m)) = L(E(m)_p) \times (L(g_p))^{-1} \mod p = m. \tag{18}$$

Homomorfismo: sejam m_1 e m_2 mensagens, então

$$\begin{aligned} E(m_1) + E(m_2) &= (g^{m_1}h^{r_1} \mod N) + (g^{m_2}h^{r_2} \mod N) \\ &= g^{m_1+m_2}h^{r_1+r_2} \mod N \\ &= E(m_1+m_2). \end{aligned} \tag{19}$$

Ataque: assim como nos casos anteriores, o algoritmo para resolver o problema da fatoração de inteiros é o GNFS.

3.6 *Paillier*

Este esquema de criptografia probabilístico é baseado na classe quadrática de resíduos, veja Paillier (1999), verificando se existe um número inteiro x tal que $x^n \equiv a \mod n^2$ para um dado inteiro a.

Geração das Chaves: sejam os números primos grandes p e q de modo que $\mathrm{mdc}(pq,(p-1)(q-1)) = 1$. Sejam $n = pq$, $\lambda = \mathrm{mmc}(p-1, q-1)$ e $\mu = \lambda^{-1} \mod n$. Escolhemos aleatoriamente $g \in \mathbb{Z}^*_{n^2}$ e verificamos se $\mathrm{mdc}(n, L(g^\lambda \mod n^2)) = 1$, onde a função L é definida por $L(u) = (u-1)/n$ para cada u do subgrupo $\mathbb{Z}^*_{n^2}$.

A chave pública: (n, g).

A chave secreta: (p, q).

Cifrar: sejam m uma mensagem e r um número escolhido aleatoriamente, então

$$E(m) = g^m r^n \mod n^2. \tag{20}$$

Decifrar: utilizando a mensagem cifrada $E(m)$ e a função L obtemos

$$D(E(m)) = [L((E(m))^{\lambda} \times (L(g^{\lambda})^{-1}) \mod n^2] \mod n = m. \tag{21}$$

Homomorfismo: sejam $m_1, m_2 \in \mathbb{Z}_{n^2}^*$, obtemos

$$\begin{aligned} E(m_1) + E(m_2) &= (g^{m_1} r_1^n \mod n^2) + (g^{m_2} r_2^n \mod n^2) \\ &= g^{m_1+m_2}(r_1 + r_2)^n \mod n^2 \\ &= E(m_1 + m_2). \end{aligned} \tag{22}$$

Como podemos ver, o esquema de Paillier não permite a multiplicação homomórfica em mensagens cifradas, ou seja, é apenas homomórfico sobre a adição.

Ataque: Como nos métodos anteriores, o algoritmo mais indicado para resolver o problema da fatoração de inteiros é o GNFS.

3.7 *Damgård-Jurik*

Este esquema é uma generalização do esquema de Paillier Damgård, I., & Jurik, M. (2001) sem perda da propriedade homomórfica, porém utilizando potências mais altas de n. Uma aplicação deste método é em votação eletrônica[1].

Geração das Chaves: escolha aleatoriamente dois números primos grandes p e q e distintos. Seja $n = pq$ (como no esquema de RSA) e $\lambda = \text{mmc}(p-1, q-1)$ (como no esquema de Paillier). Escolha um elemento $g \in \mathbb{Z}_{n^{s+1}}^*$ (o grupo quociente G/H, onde G é cíclico $\mod n^s$ enquanto H é isomorfo a $\mathbb{Z}_n^*$) tal que $g = (1+n)^j x \mod n^{s+1}$ para j conhecido primo relativo de n e $x \in H$.

Usando o Teorema do Resto Chinês, escolha $d \mod n \in \mathbb{Z}_n^*$ tal que $d \equiv 1 \mod n^s$ e $d \equiv 0 \mod \lambda$. Por exemplo, d poderia ser $\lambda \mod n^{s+1}$ para algum $s \geq 1 \in \mathbb{N}^+$, caso s seja igual a 1 obtemos o esquema original de Paillier.

A chave pública: (n, g).

A chave secreta: d.

[1] `http://security.hsr.ch/msevote/damgardjurik`

Cifrar: seja $m \in \mathbb{Z}_{n^s}$. Selecione um número randômico $r \in \mathbb{Z}^*_{n^{s+1}}$, tal que

$$E(m) = g^m r^{n^s} \mod n^{s+1}. \tag{23}$$

Decifrar: utilizando a mensagem cifrada $E(m)$ e a chave secreta d obtemos,

$$\begin{aligned}(E(m))^d &= (g^m r^{n^s})^d \mod n^{s+1} \\ &= ((1+n)^{jm} x^m r^{n^s})^d \mod n^{s+1} \\ &= (1+n)^{jmd} \mod n^s.\end{aligned} \tag{24}$$

Após este processo, substituímos $E(m)^d$ por $(g^d \mod n^{s+1}) \mod n^s$ e efetuamos novamente os cálculos utilizando um algoritmo para resolver o expoente, consultar Damgård, I., & Jurik, M. (2001), assim $g^d = (1+n)^{jd} \mod n^s$. Portanto, obtemos a expressão desejada,

$$D(E(m)) = (jmd) \cdot (jd)^{-1} \mod n^s = m \mod n^s. \tag{25}$$

Homomorfismo:

$$\begin{aligned}E(m_1) + E(m_2) &= (g^{m_1} r_1^{n^s} \mod n^{s+1}) + (g^{m_2} r_2^{n^s} \mod n^{s+1}) \\ &= g^{m_1+m_2}(r_1+r_2)^{n^s} \mod n^{s+1} \\ &= E(m_1+m_2).\end{aligned} \tag{26}$$

Ataque: Como este esquema é uma generalização do esquema de *Paillier*, o melhor algoritmo para a fatoração é o GNFS.

4 ANÁLISE DAS COMPLEXIDADES DE PHE

Nesta seção, apresentaremos o cálculo das complexidades de tempo de cada algoritmo do seguinte modo: cifrar, decifrar, homomorfismo e ataque. Assim, a complexidade de um algoritmo é uma estimativa do tempo necessário à execução, que esse algoritmo expressa em função das operações aritméticas fundamentais, e do tamanho dos dados de entrada. Abaixo trataremos do cálculo das complexidades dos

algoritmos de criptografia de chave pública de PHE, proposto na seção 3. Mostraremos qual o melhor ataque para cada esquema de PHE.

4.1 Performance de PHE

Nesta subseção, descreveremos como efetuamos os cálculos das complexidades dos algoritmos contidos na seção 3. Analisaremos, a complexidade do processo de cifrar mensagens e em seguida os demais processos.

As complexidades de cifrar são: $O(\log(e))$ no esquema de *RSA*, pois executa uma exponenciação modular e consideramos e um valor bem pequeno, $O(\log(n))$ no esquema de *Elgamal*, pois necessita de duas exponenciações modulares $\mod n$ e uma multiplicação por escalar $(m \times h)$, $O(\log(n))$ para os esquemas de *Benaloh* e *Naccache-Stern*, portanto, necessitam de duas exponenciações modulares $\mod n$. Além disso, se o esquema computa o produto de exponenciações modulares, então o esquema pode ser otimizado Borges, F., Lara, P., & Portugal, R. (2017).

Notamos que as complexidades de cifrar dos métodos citados acima, são $O(\log(n))$, exceto o algoritmo de *RSA* que é e, onde $e = 65537$, consultar Pereira, D., Aranha, M., & Borges, F. (2019).

O cálculo das demais complexidades de cifrar são: $(O(\log(N) + \log(p))$ para o esquema de *Okamoto-Uchiyama*, pois executa duas exponenciações modulares $\mod N$, uma inversão e uma exponenciação modular $\mod p$, $O(\log(n^2) + \log(n))$ no esquema de *Paillier*, $O(\log(n^s))$ no esquema de *Damgård-Jurik*, se considerarmos $s = 1$ obtemos uma versão do esquema de *Paillier*, assim, a complexidade é $\log(n^2)$ para cifrar, igual a de *Paillier*.

Trataremos da complexidade do processo de decifrar de cada esquema. A complexidade de decifrar é $O(\log(n))$ para os seguintes esquemas: *RSA* considerando $e < n$ que executará uma exponenciação modular, *Elgamal* onde n é a ordem do grupo, ou seja, uma exponenciação modular e uma inversão e *Naccache-Stern* de uma exponenciação modular e uma divisão usual sobre inteiros, bem como a aplicação do teorema do Resto Chinês, ou seja, $j \times \log(n)$, consideraremos $j = 4$, para decifrarmos corretamente. Para o esquema de *Benaloh* a complexidade é $O(2^l \times \log(n))$, onde $l = \log i$ é o número de bits de i, executará uma inversão e uma exponenciação modular, consideraremos $l \leq 2$, caso contrário, teremos problemas para decifrar.

Decifrando o esquema de *Okamoto-Uchiyama* temos que a complexidade é $O(\log(N) + \log(p))$, executando duas exponenciações modulares divididas por duas exponenciações modulares $\mod N$ e mais uma $\mod p$. A divisão é usual sobre inteiros e torna-se mais rápida. Além disso, o esquema de *Paillier* tem como complexidade para decifrar de duas exponenciações modulares divididas por duas exponenciações modulares $\mod n^2$, e mais uma $\mod n$. A divisão é usual sobre inteiros, ou seja, $O(\log(n^2) + \log(n))$. No esquema de *Damgård-Jurik* para decifrar a complexidade é de $O(\log(n^{s+1}))$, ou seja, uma exponenciação modular. Se $s \geq 1$, então $s = 1$ recai no esquema de *Paillier*, consideraremos $s = 2$, assim para decifrar obtemos a complexidade $\log(n^3)$.

No homomorfismo utilizamos operações ou aditivas ou multiplicativas em cada esquema. Assim, executamos uma multiplicação modular (MM), nos seguintes esquemas: *RSA*, *Benaloh*, *Naccache-Stern*, *Okamoto-Uchiyama*, *Paillier* e *Damgård-Jurik*. Além disso, no esquema de *Elgamal* executamos duas multiplicações modulares, uma em cada coordenada da mensagem cifrada.

Como o esquema dos autores Naccache-Stern é probabilístico a chance de decifrar a mensagem corretamente é de $1/\ln(k)$, sendo assim, k deve ser pequeno, pois assim, a probabilidade é maior de recuperar a mensagem cifrada. Por exemplo, escolhendo $K = 4$ obtemos sucesso, os resultados estão contidos na Tabela 3, onde $n = 64$ bits. Se escolhemos $K = 16$ e $n = 128$ bits, o tempo de execução de decifrar é $2.055ms$, para cifrar é $0.036ms$ e homomorfismo de $0.088ms$. Por outro lado com $k = 36$, o tempo de cifrar é $0.168ms$ e do homomorfismo $0.289ms$. Portanto, quanto maior o valor de k, maior será o tempo de execução e também terá mais dificuldades para decifrar o esquema. Além disso, devemos escolher valores que sejam múltiplos de dois, pois g tem a ordem de $g \geq \frac{\phi(n)}{4}$. Na Tabela 3, apresentamos um resumo geral dos esquemas com as suas respectivas complexidades.

Na Tabela 4, apresentamos as complexidades aproximadas, representando o custo de processamento de cada etapa dos esquemas de PHE. As informações fornecidas facilitam a comparação entre os diferentes algoritmos de PHE.

Tabela 3 – Complexidade dos esquemas de PHE

Algoritmos	Cifrar	**Decifrar**	**Homomorfismo**	**Ataque**
RSA	$O(\log(e))$	$O(\log(n))$	1 MM em n	(5)
Elgamal	$O(\log(p))$	$O(\log(p))$	2 MM em p	(5)
Benaloh	$O(\log(n))$	$O(2^l \times \log(n))$	1 MM em n	(12)
Naccache-Stern	$O(\log(n))$	$O(j \times \log(n))$	1 MM em n	(5)
Okamoto-Uchiyama	$O(\log(N) + \log(p))$	$O(\log(N) + \log(p))$	1 MM em N	(5)
Paillier	$O(\log(n^2) + \log(n))$	$O(\log(n^2) + \log(n))$	1 MM em n^2	(5)
Damgård-Jurik	$O(\log(n^s))$	$O(\log(n^{s+1}))$	1 MM em n^s	(5)

Fonte: autores (2024)

Tabela 4 – Aproximação do custo de cada processo dos esquemas de PHE

Algoritmos	**Cifrar**	**Decifrar**	**Homomorfismo**
RSA	$\log(e)$	$\log(n)$	1 MM em n
Elgamal	$\log(n)$	$\log(n)$	2 MM em n
Benaloh	$\log(n)$	$2^l \times \log(n)$	1 MM em n
Naccache-Stern	$\log(n)$	$4 \times \log(n)$	1 MM em n
Okamoto-Uchiyama	$\log(\sqrt{n^3})$	$\log(\sqrt{n^3})$	1 MM em $\sqrt{n^3}$
Paillier	$\log(n^2)$	$\log(n^2)$	1 MM em n^2
Damgård-Jurik	$\log(n^2)$	$\log(n^3)$	1 MM em n^2

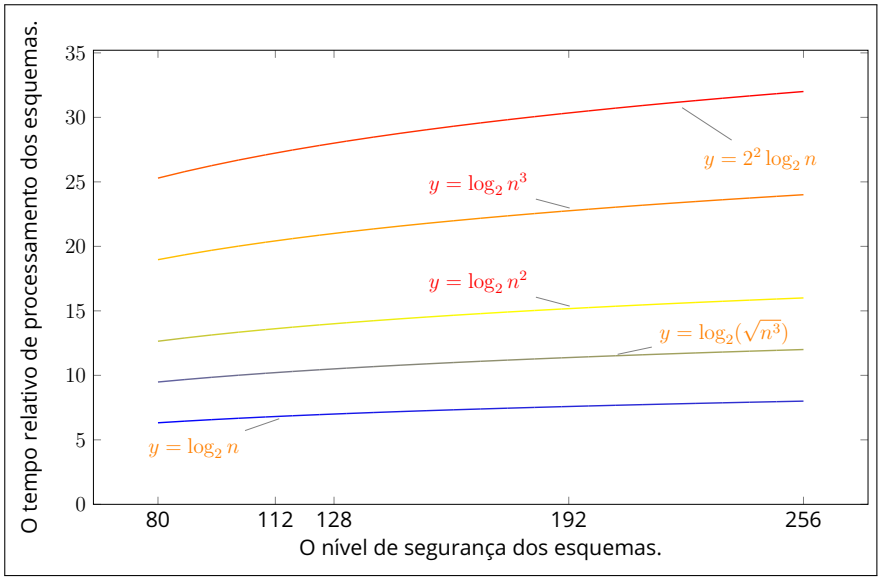
Fonte: autores (2024)

Os melhores esquemas de cifragem são o *Elgamal* e *Benaloh*, pois suas complexidades são $O(\log(n))$. Como $n = pq$ é o produto de dois números primos e $N = p^2q$, ou seja, $N > n$, temos a seguinte aproximação: $\log(N) + \log(p) \approx \log\sqrt{n^3}$. Portanto, o esquema mais eficiente para decifrar é o *Okamoto-Uchiyama*. Analisando os esquemas aditivos de PHE, o mais eficiente é o Okamoto-Uchiyama, pois possui a mesma complexidade tanto para cifrar quanto para decifrar. Embora sua complexidade seja $O(\log(N) + \log(p))$, esse esquema se destaca pela sua eficiência. O método de *Paillier* retorna a mensagem decifrada $\mod n^2$, enquanto o esquema de *Damgård-Jurik* utiliza $\mod n^s$ para cifragem e n^{s+1} para decifração, ou seja, usa valores de n de alta ordem, tornando-se mais lento que o*Okamoto-Uchiyama*.

No que diz respeito aos algoritmos multiplicativos de PHE, o mais eficiente é o *RSA*,pois executa uma exponenciação modular tanto para cifrar quanto para decifrar, além de uma multiplicação modular no homomorfismo. A complexidade para cifrar é $\log(e)$, onde $e < n$, enquanto para decifrar é $\log(n)$. Já o esquema de *Elgamal* exige o dobro de exponenciações modulares, tornando-o mais custoso.

Na Figura 2, obtemos o gráfico dos processos de cifrar, decifrar e homomorfismo de cada esquema proposto neste trabalho, com base nos resultados da Tabela 4, consideramos $n = 2048$ bits para execução, exceto no algoritmo de *Naccache-Stern*, que é de $n = 64$ bits.

Figura 2 – Complexidade dos Esquemas de PHE



Fonte: autores (2024)

Os melhores esquemas de criptografia para operações homomórficas em PHE podem ser classificados com base em suas características específicas. Para operações multiplicativas, o esquema *RSA* é amplamente utilizado, enquanto para operações aditivas, os esquemas *Benaloh* e *Okamoto-Uchiyama* se destacam pela sua eficácia e segurança. Além disso, o esquema *Naccache-Stern* é notável devido à sua complexidade de tempo $O(\log(n))$. Considerando que $n = p \cdot q$, onde p e q são números primos, e $N = p^2 \cdot q$, com $N > n$, a aproximação $\log(N) + \log(p) \approx \log(\sqrt{n^3})$ é válida.

No que diz respeito à decodificação, o algoritmo mais eficiente é o esquema *Benaloh*. Já para o cálculo do homomorfismo, o esquema *Damgård-Jurik* se destaca como o mais eficaz, apesar de envolver uma multiplicação modular $\mod n^s$. Caso $s = 1$, o esquema recai no *Paillier*, pois $n^s = n^2$, o que leva à suposição de que $s > 1$ para garantir que o esquema de *Damgård-Jurik* mantenha sua eficiência.

Analisando os algoritmos aditivos de PHE o melhor é o esquema de *Damgård-Jurik* utiliza para cifrar $\mod n^s$ e para decifrar n^{s+1}, ou seja, n de alta ordem, este esquema é mais eficiente. O método de *Paillier* retorna a mensagem decifrada

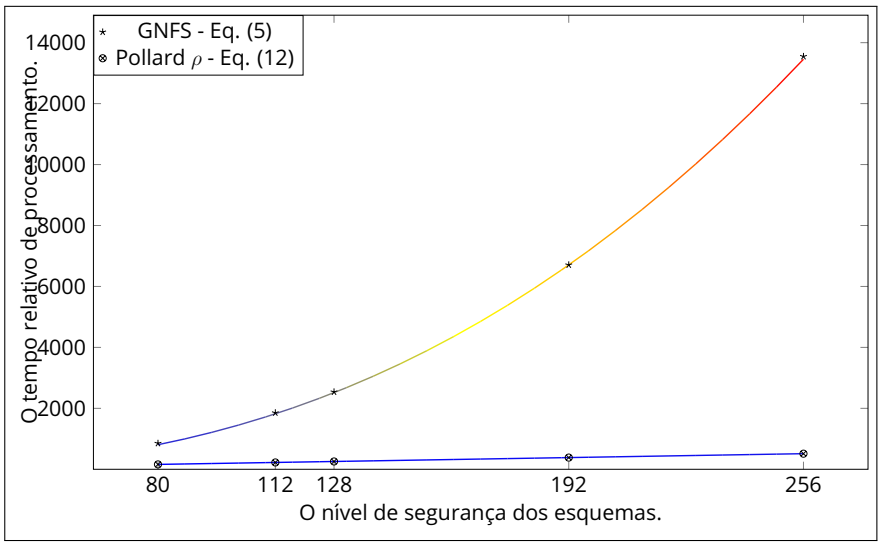
$\mod n^2$. Sendo assim mais lento que o algoritmo de *Damgård-Jurik* e *Okamoto-Uchiyama*. Além disso, *Okamoto-Uchiyama* tem complexidade de $O(\log(N) + \log(p))$ e possui a mesma complexidade para cifrar e decifrar.

O melhor algoritmo multiplicativo de PHE é o *RSA*, pois executa uma exponenciação modular para cifrar e outra para decifrar, bem como uma multiplicação modular no homomorfismo e a complexidade para cifrar é $\log(e)$, onde $e < n$ e para decifrar é $\log(n)$. No esquema de *Elgamal* temos o dobro de exponenciações modulares.

4.2 Ataque

O algoritmo mais eficiente baseado em fatoração de números inteiros é o *General Number Field Sieve*. Os esquemas propostos neste trabalho, e baseados neste ataque são: *RSA*, *Elgamal*, *Naccache-Stern*, *Okamoto-Uchiyama*, *Paillier* e *Damgård-Jurik*. Apresentamos a complexidade de tempo deste ataque na equação (5). O melhor algoritmo de ataque para o esquema de *Benaloh* é *Pollard* ρ. A complexidade deste ataque pode ser obtida na equação (12). Na Figura 3, representamos os gráficos das complexidades dos ataques de GNFS e *Pollard* ρ.

Figura 3 – Complexidade dos Ataques: GNFS e Pollard ρ



Fonte: autores (2024)

5 CONCLUSÕES

Este trabalho revisa os principais algoritmos criptográficos assimétricos aplicados a esquemas de Criptografia Homomórfica (PHE). Além disso, apresenta uma análise detalhada das complexidades teóricas e das implementações práticas de cada etapa dos esquemas criptográficos, incluindo as operações de cifragem, decifragem, homomorfismo e ataques. A análise abrange os algoritmos de PHE: *RSA*, *Elgamal*, *Benaloh*, *Naccache-Stern*, *Okamoto-Uchiyama*, *Paillier* e *Damgård-Jurik*. Os resultados indicam que o *RSA* é o esquema mais eficiente para operações multiplicativas, enquanto o *Okamoto-Uchiyama* se destaca como o melhor para operações aditivas.

AGRADECIMENTOS

Agradeço a agência de fomento CNPQ e a Petrobras pelos auxílios financeiros.

REFERÊNCIAS

Acar, A., Aksu, H., Uluagac, A. S., & Conti, M. (2018). A survey on homomorphic encryption schemes: Theory and implementation. *ACM Comput. Surv.*, 51(4), 79:1–79:35.

Borges, F., Lara, P., & Portugal, R. (2017). Parallel algorithms for modular multi-exponentiation. *Applied Mathematics and Computation*, 292, 406–416.

Borges, F., Portugal, F. & Lara, S. (2007). Autenticação e o problema do logaritmo discreto. *Revista Hífen*, 31(59/60), 106–111. http://revistaseletronicas.pucrs.br/fo/ojs/index.php/hifen/article/view/3884.

Clarkson, J. B. (1994). Dense probabilistic encryption. In *Proceedings of the Workshop on Selected Areas of Cryptography*. (pp. 120-128). Clarkson University.

Damgård, I., & Jurik, M. (2001). A generalisation, a simplication and some applications of paillier's probabilistic public-key system. In Kim, K., editor, *Public Key Cryptography*. (pp. 119-136). Springer Berlin Heidelberg.

Elgamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. In Goos, G. Hartmanis, J., editor, *Proceedings of CRYPTO 84 on Advances in Cryptology*. (pp. 469 - 472). Springer-Verlag.

Fousse, L., Lafourcade, P., & Alnuaimi, M (2011). Benaloh's dense probabilistic encryption revisited. In Vaudenay, S., & Petit, C., editor, *Progress in Cryptology – AFRICACRYPT 2011*. (pp. 348-362). Springer Berlin Heidelberg.

Goldwasser, S. & Micali, S. (1982). Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing*. (pp. 365–377). Association for Computing Machinery.

Halevi, S. (2017). Homomorphic encryption. In Lindell, Y., editor, *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*. (pp. 219–276). Springer International Publishing. https://doi.org/10.1007/978-3-319-57048-8_5.

Kawachi, A., Tanaka, K., & Xagawa, K. (2007). Multi-bit cryptosystems based on lattice problems. In Okamoto, T., & Wang, X., editor, *Public Key Cryptography – PKC 2007*. (pp. 315-329). Springer Berlin Heidelberg.

Lara, P., & Borges, F. (2008). O algoritmo de fatoração gnfs. In *Congrasso Nacional de Matemática Aplicada e Computacional*. Coordenação de Sistemas e Redes. http: //www.sbmac.org.br/eventos/cnmac/xxxi_cnmac/PDF/316.pdf.

Naccache, D. & Stern, J. (1998). A new public key cryptosystem based on higher residues. In Liu, P., Basin, D. A., & Feng, D., editor, *Proceedings of the 5th ACM Conference on Computer and Communications Security*. (pp. 59–66). Association for Computing Machinery.

Okamoto, T. & Uchiyama, S. (1998). A new public-key cryptosystem as secure as factoring. In Nyberg, K., editor, *Advances in Cryptology — EUROCRYPT 98*. (pp. 308-318). Springer Berlin Heidelberg.

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In Stern, J., editor, *Advances in Cryptology — EUROCRYPT '99*. (pp.223–238). Springer Berlin Heidelberg.

Pandey, G. (2014). A guide to general number field sieve for integer factorization. *Investigations in Mathematical Sciences*, 4, 83–98.

Pereira, D., Aranha, M., & Borges, F. (2019). HTTPS Keys in the Mediterranean. In *2019 II Workshop on Metrology for Industry 4.0 and IoT (MetroInd4.0IoT)*. (pp. 449–454). IEEE.

Rivest, R. L., Adleman, L. & Dertouzos, M. L. (1978). On data banks and privacy homomorphisms. In DeMillo, R. A., editor, *Foundations of Secure Computation*. (pp. 169-180). Academia Press.

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

Teske, E. (1998). Speeding up pollard's rho method for computing discrete logarithms. In Buhler, J. P., editor, *Algorithmic Number Theory*. (pp. 541-554). Springer Berlin Heidelberg.

Como citar este artigo

Lunkes, A. D. L. Z., & Oliveira, F. B. (2025). Análise da complexidade dos algoritmos de Criptografia Homomórfica. *Ciência e Natura*, Santa Maria, v. 47, e87825. DOI: https://doi.org/10.5902/2179460X87825.