Análise da eficiência de uma proposta de grafos direcionados ponderados na Criptografia RSA

Leonardo de Almeida Cavadas[1], **Igor Machado Coelho**[1,2], **Helio do Nascimento Cunha Neto**[1]

[1]Instituto de Matemática e Estatística – UERJ, RJ, Brasil

[2]Instituto de Computação – UFF, RJ, Brasil

Abstract. *Graph theory has numerous applications in computing, including information security. This paper analyzes how the application of paths in weighted directed graphs impacts key generation, focusing on reducing computational costs by avoiding the repeated generation of prime numbers with each key creation, as in the traditional RSA algorithm and in Multiprime RSA. This analysis uses a pre-constructed graph, where weights are assigned based on the prime number generation algorithm in an interval between a and b by Ezz-Eldien et al., which has time complexity $O(b)$ and space complexity $O(b-a)$. Despite the high initial cost due to graph construction, subsequent key generation becomes more efficient. However, like variants of traditional RSA, it remains vulnerable to quantum computing.*

Resumo. *A Teoria dos Grafos oferece diversas aplicações na computação, inclusive na segurança da informação. Este trabalho analisa como a aplicação de caminhos em digrafos ponderados impacta a geração de chaves, focando na redução de custos computacionais ao evitar a geração recorrente de primos a cada criação de chaves, como no algoritmo RSA tradicional e no Multiprime RSA. Esta análise utiliza um grafo previamente construído, onde os pesos são atribuídos com base no algoritmo de geração de primos em um intervalo, entre a e b, de Ezz-Eldien et al., que apresenta complexidade de tempo $O(b)$ e espacial $O(b-a)$. Apesar do custo inicial elevado devido à construção do grafo, a geração subsequente de chaves torna-se mais eficiente. No entanto, como as variantes do RSA tradicional, continua vulnerável à computação quântica.*

1. Introdução

A crescente necessidade de proteção de dados e a evolução constante das técnicas de invasão impõem desafios significativos à segurança da informação. Desta forma, a utilização de ferramentas matemáticas avançadas, como a teoria dos grafos, se mostra promissora para aprimorar os métodos criptográficos existentes. Em particular, o emprego de grafos direcionados ponderados permite a exploração de novas estratégias para a geração de chaves criptográficas, ampliando a complexidade e a robustez dos sistemas, como o RSA. Este trabalho avalia se o uso de caminhos em digrafos ponderados pode reduzir o custo de geração de chaves em variantes do RSA, considerando desempenho, complexidade, reutilização de primos e implicações de segurança.

A geração de números primos tem uma parte do custo na criação das chaves da criptografia assimétrica, podendo ser caro e repetitivo, onde o *Multiprime RSA* ainda requer a geração frequente de novos primos. Uma possibilidade a ser avaliada é representar

Cadernos do IME - Série Informática
e-ISSN: 2317-2193 (online)
DOI: 10.12957/cadinf.2026.94187

números primos como pesos de arestas em um grafo direcionado ponderado e utilizar caminhos nesse grafo para compor os parâmetros da geração de chaves. Essa estratégia pode reduzir a geração recorrente de primos, mas introduz custos adicionais relacionados à construção, armazenamento, busca de caminhos e proteção do próprio grafo.

Este trabalho investiga se a estrutura combinatória de grafos direcionados ponderados oferece vantagens práticas para a geração de chaves RSA. A análise considera tempo de execução, custo inicial, reutilização de primos, risco de comprometimento do grafo e comparação com Multiprime RSA e amostragem direta de primos.

Também é incorporado o algoritmo de geração de números primos em um intervalo, de Ezz-Eldien *et al.* [Ezz-Eldien et al. 2024], que é mais eficiente do que métodos tradicionais como Crivo de Eratóstenes ou Miller-Rabin. Isso melhora a construção inicial do grafo ao distribuir primos como pesos de forma rápida e com baixo custo de memória. Ao utilizar a estrutura combinatória dos grafos, o artigo mostra que é possível obter um número massivo de combinações de chaves pela vasta conectividade das arestas, aumentando a entropia e a segurança do sistema sem regerar os primos.

Este trabalho é organizado em quatro seções. A Seção 2 mostra trabalhos relacionados, a Seção 3 explica a criptografia de chave pública, a criptografia RSA e tópicos da Teoria de Grafos utilizados no trabalho, a Seção 4 explica o trabalho desenvolvido e a Seção 5 é a conclusão.

2. Trabalhos relacionados

Wiemer [Wiener 1990] criou o *RSA Rebalanceado*, ao melhorar o desempenho do algoritmo de decifração/assinatura deslocando o trabalho para o algoritmo de cifragem/verificação.

Collins *et al.* [Collins et al. 1998] implementaram outra melhoria da criptografia RSA, ao utilizar k números primos $(p_1, p_2, \ldots, p_k)$ ao invés de dois números primos, e chamado de *Multiprime RSA*. Esta implementação possibilita a divisão das etapas de cifração e decifração em sub-tarefas, permitindo seu desenvolvimento de maneira paralela e, assim, otimizando o desempenho.

Takagi [Takagi 1998] desenvolveu a *MultiPower RSA*, propondo um módulo apropriado $p^k q$, baseado na criptografia RSA, que resiste a dois dos algoritmos de fatoração mais rápidos, nomeadamente o crivo do campo numérico e o método da curva elíptica. Também aplicaram o algoritmo de decifração rápida módulo p^k. O processo de decifração proposta é mais rápido do que a decifração do RSA, utilizando o Teorema Chinês do resto.

Paixao e Gazzoni Filho [Paixao and Gazzoni Filho 2005] descreveram uma combinação eficiente de duas variantes da criptografia RSA (*MPrime* e *Rebalanced RSA*), analisadas por Boneh e Shacham [Boneh and Shacham 2002], chamada *RPrime RSA*. Para módulos de 2048 bits, o processo de decifração resultante é cerca de 8 vezes mais rápido que o apresentado por Quisquater e Couvreur [Quisquater and Couvreur 1982] e cerca de 27 vezes mais rápido que a criptografia RSA original.

Ezz-Eldien *et al.* [Ezz-Eldien et al. 2024] apresentaram dois novos algoritmos para gerar números primos: um que gera primos até um determinado limite e outro que gera números primos dentro de um intervalo especificado. Estes algoritmos inovadores baseiam-se em fórmulas de números compostos por ímpares, o que lhes permite obter me-

lhorias de desempenho em comparação com os algoritmos de geração de números primos existentes. Os resultados experimentais mostram que os algoritmos propostos superam os algoritmos de geração de números primos já estabelecidos, tais como Miller-Rabin, Crivo de Atkin, Crivo de Eratóstenes e Crivo de Sundaram sobre o tempo médio de execução.

Diferentemente dos trabalhos relacionados que buscam melhorias no desempenho da criptografia RSA por meio da utilização de múltiplos primos (como o Multiprime RSA), módulos específicos (como no MultiPower RSA) ou redistribuição do custo computacional (como no RSA Rebalanceado). Este trabalho examina uma abordagem que modela a geração de chaves como busca de caminhos em grafos direcionados ponderados que modela a geração de chaves como a busca de caminhos em grafos direcionados ponderados, cujas arestas são associadas a números primos, reduzindo significativamente a necessidade de geração recorrente de primos para cada chave, mantendo um elevado número de combinações possíveis.

3. Conhecimentos Básicos

Nesta seção são mostrados os conhecimentos básicos para o entendimento do trabalho: a criptografia assimétrica RSA, alguns conceitos de grafos e os principais algoritmos para a geração de números primos.

3.1. Criptografia RSA

Este algoritmo encripta a mensagem M utilizando uma função de cifração $E(M)$ com a chave pública $PU = \{e, n\}$, criando um texto cifrado C (Equação (1)). Na decifração, é utilizada uma função de decifração $D(C)$ sobre o texto cifrado com a chave privada $PR = \{d, n\}$. Assim, o texto C é decifrado mostrando a mensagem original M (Equação (2)).

$$C = E(M) = M^e \; mod \; n \quad (1) \qquad M = D(C) = C^d \; mod \; n \quad (2)$$

O valor de n é a multiplicação dos dois números primos escolhidos, os valores d e e são inversos multiplicativos de $\phi(n)$, onde $\phi(n)$ é o Totiente de Euler. Os cálculos para encontrá-los são mostrados na Criptografia RSA. Note que d não é conhecido, por fazer parte da chave privada, e e é conhecido, por fazer parte da chave pública. A criptografia RSA [Rivest et al. 1978] utiliza o valor de n, que é calculado de acordo com a Equação (3), o valor de e, de acordo com a Equação (5), e o valor de d, Equação (6). Onde p e q são números primos muito grandes.

$$n = p \cdot q \quad (3) \qquad \phi(n) = (p-1) \cdot (q-1) \quad (4)$$

$$mdc(e, \phi(n)) = 1 \quad (5) \qquad d \equiv e^{-1} \; mod \; \phi(n) \quad (6)$$

Rivest, Shamir e Adleman [Rivest et al. 1978] mostram que para calcular o valor de e é utilizado o Algoritmo de Extensão de Euler para encontrar o Máximo Divisor Comum entre $\phi(n)$ e e, isto é,$(mdc(e,\ \phi(n)))$, que é calculado pela determinação da sequência $x_0, x_1, \ldots, x_k$, onde $x_0 = \phi(n)$, $x_1 = d$ e $x_{i+1} \equiv x_{i-1}(mod\ x_i)$. Este cálculo termina quando encontra $x_k = 0$, então $mdc(e,\ \phi(n)) = x_{k-1}$, onde e é relativamente primo a $\phi(n)$. Para calcular o valor de d, inverso multiplicativo de e, é utilizada a Equação (6). Assim, as chaves pública e privada são, respetivamente, $PU = \{e, n\}$ e $PR = \{d, n\}$, usadas nas Equações (1) e (2).

3.2. Teoria de Grafos

Um *grafo*, ou *grafo não direcionado*, $G = (V, E)$ é um conjunto finito não vazio V de vértices e um conjunto E de arestas, onde cada aresta $e \in E$ será denotada pelo par de vértices $e = (v, w)$ que a forma. Quando o grafo é *direcionado*, ou *digrafo*, $D = (V, E)$, também possui um conjunto finito não vazio V dos vértices, e um conjunto E de arestas, porém cada aresta $e = (v, w)$ possui uma única direção de v para w. Diz-se também que e é divergente de v e convergente a w [Szwarcfiter 2018].

Um *passeio* é definido como uma sequência finita e alternada de vértices $(v_1, v_2, \ldots, v_k)$ e arestas $(v_1v_2, v_2v_3, \ldots, v_{k-1}v_k)$, começando e terminando com vértices, de modo que cada aresta seja incidente com os vértices que a precedem e a seguem. Nenhuma aresta aparece (é coberta ou atravessada) mais de uma vez em um passeio, sendo que um vértice pode aparecer mais de uma vez. Um passeio no qual nenhum vértice aparece mais de uma vez é chamado de *caminho* ou *caminho simples*, comumente denotado por P_n. O número de arestas em um caminho é chamado de *comprimento do caminho* [Deo 2016].

Um grafo ponderado $G = (V, E)$ (nas arestas) é um grafo juntamente com uma função ω, que associa um número real não negativo a cada aresta $e \in E$, denotando o peso da aresta e [Szwarcfiter 2018].

Dados $n \in \mathbb{N}$ e $p \in [0, 1]$, definimos $G(n, p)$ como o grafo aleatório com n vértices obtido ao adicionarmos uma aresta entre cada par de vértices $u, v \in V(G)$ de forma aleatória e independente com probabilidade p [Botler et al. 2021]. Neste trabalho, é utilizado um digrafo aleatório ponderado, onde suas arestas terão números primos como peso, para a geração das chaves de criptografia, mostrada na Seção 4.

3.3. Geração de números primos

A geração de números primos é uma etapa crucial na implementação da criptografia RSA, uma vez que a segurança do sistema baseia-se na dificuldade de fatorar o produto de dois grandes números primos. Existem diversos algoritmos eficazes para a identificação desses números primos, cada um com suas particularidades, complexidade de tempo e espaço.

O Crivo de Eratóstenes é um dos métodos mais tradicionais e amplamente utilizados para encontrar todos os números primos até um determinado limite n. Este algoritmo funciona marcando iterativamente os múltiplos de cada número primo a partir de 2. Os números que permanecem desmarcados após este processo são considerados primos. Sua complexidade de tempo é $O(n\ log(log\ n))$ e a complexidade de espaço é $O(n)$, o que o torna eficiente para gerar primos até limites razoavelmente grandes. No entanto, ele pode não ser o mais adequado para gerar primos em intervalos específicos ou para ser executado em paralelo em processadores modernos [Ezz-Eldien et al. 2024].

O Crivo de Sundaram é uma alternativa que se baseia na identificação de números primos até um determinado limite e na remoção de números que podem ser expressos como a soma de dois primos consecutivos. Os números restantes, após aplicar um conjunto de fórmulas, são primos. Este método apresenta uma complexidade de tempo de $O(n\ log\ n)$ e uma complexidade de espaço de $O(n)$, embora seja menos eficiente do que o Crivo de Eratóstenes para a geração de primos em um limite fixo [Ezz-Eldien et al. 2024].

Em contrapartida, o Crivo de Atkin representa uma abordagem mais moderna e otimizada, oferecendo uma complexidade de tempo de $O(n\ log(log\ n))$ e uma complexidade de espaço de $O(n^{\frac{1}{2}} log\ n)$. Crivo de Atkin é mais eficiente para a geração de números primos do que os Crivos de Eratosthenes e de Sundaram, porém requer mais memória [Ezz-Eldien et al. 2024].

Além da geração de números primos, a primalidade pode ser testada usando o teste de Miller-Rabin, que é um método probabilístico. Este teste verifica se um número específico é primo, utilizando uma abordagem que envolve aritmética modular. Sua complexidade de tempo é $O(k\ log\ n)$, onde k é o número de iterações, e é particularmente útil para testar números grandes e evitar falsos positivos [Ezz-Eldien et al. 2024].

Por fim, o teste de primalidade AKS é um algoritmo determinístico que determina se um número é primo. No entanto, é mais complexo e pode exigir maior utilização de memória, o que pode limitar sua aplicabilidade em algumas situações [Ezz-Eldien et al. 2024]. A complexidade de tempo do AKS é $O(log^6\ n)$ [Harahap and Khairina 2019].

Esses métodos de geração e teste de números primos são fundamentais para garantir a robustez e a segurança das chaves criptográficas empregadas na criptografia RSA. A Tabela 1 [Ezz-Eldien et al. 2024] apresenta uma comparação dos métodos de números primos acima com seus objetivos e limitações.

Ezz-Eldien *et al.* [Ezz-Eldien et al. 2024] apresentam, na Tabela 2, uma comparação entre os algoritmos mais recentes, destacando suas complexidades de tempo e espaço, a geração baseada em intervalo e seu desempenho em aplicações de criptografia. Os autores também mostram as mesmas características, na Tabela 3, de um algoritmo proposto por eles para geração de primos dentro de um intervalo $[a, b]$. Baseado nestas tabelas, os algoritmos propostos são melhores em relação aos métodos já conhecidos pelos suas complexidades.

O Algoritmo 1 recebe dois parâmetros, o início a e o fim b, gerando todos os números primos dentro do intervalo $[a,\ b]$ especificado. Se o início do intervalo a for maior que o fim do intervalo b, o algoritmo retorna uma lista vazia, indicando que não há primos em um intervalo inválido e quando $a = b$, o algoritmo verifica se o único número é primo e o retorna. Por exemplo, se $a = b = 17$, o algoritmo retorna o valor $[17]$. O algoritmo identifica corretamente os intervalos que não contêm números primos. Por exemplo, para o intervalo $[14, 16]$, ele retorna uma lista vazia, pois não há primos entre 14 e 16. O algoritmo inicia duas listas vazias, A_1 e B_1, para guardar os números compostos ímpares no intervalo especificado. Ao percorre cada valor do intervalo $[a,\ b]$ e, para cada valor é calculado os números compostos ímpares m e m_1, nas linhas 11 e 13 respectivamente. O algoritmo pára quando $m_1 > 2y$. Então, o algoritmo retorna uma lista de números primos sendo a diferença de termos entre as listas A_1 e B_1 [Ezz-Eldien et al. 2024].

O Algoritmo 1, apesar de complexo para entende, lida eficientemente com intervalos grandes, mantendo a sua complexidade de tempo linear, garantindo que pode ser escalado para entradas maiores sem degradação do desempenho. A estrutura do algoritmo assegura a robustez, validando o intervalo de entrada e tratando casos especiais de forma adequada. Mantém a confiabilidade ao gerar com precisão números primos dentro de intervalos válidos, mesmo quando o intervalo contém apenas um número ou nenhum

Tabela 1. Métodos de geração de números primos e suas limitações [Ezz-Eldien et al. 2024].

Método	Objetivo	Limitação
Crivo de Eratóstenes	Geração de primos até um limite especifico usando uma abordagem direta e eficaz.	Especificado até um determinado limite. Alto uso de memória.
Crivo de Sundaram	Geração de primos com um limite pequeno.	Especificado até um determinado limite. Não é possível identificar primos que não são ímpares.
Crivo de Atkins	É uma ferramenta poderosa para gerar primos com grande intervalo.	Mais complexo. Usa muita memória.
Miller-Rabin	Um método probabilístico para determinar se um determinado número é primo ou composto.	Método probabilístico que pode produzir números falso positivos.
AKS	Método deterministico para verificar se um número específico é primo ou composto.	Dificuldade de implementação. Mais complexo.

Tabela 2. Análise comparativa dos algoritmos do estado da arte. [Ezz-Eldien et al. 2024].

Algoritmo	Complexidade de Tempo	Complexidade de Espaço	Geração baseada no intervalo	Desempenho em aplicações de criptografia
Crivo de Eratóstenes	$O(n\,log(log\,n))$	$O(n)$	Não bem adaptado	Moderado
Crivo de Sundaram	$O(n\,log\,n)$	$O(n)$	Não bem adaptado	Moderado
Crivo de Atkins	$O(n\,log(log\,n))$	$O(n^{\frac{1}{2}}\,log\,n)$	Não bem adaptado	Moderado
Primalidade de Miller-Rabin	$O(k\,log\,n)$	$O(1)$	Pode ser adaptado	Alto (probabilístico)
Teste de primalidade AKS	$O(log^6 n)$	$O(log^3 n)$	Não bem adaptado	Alto (determinístico)

Tabela 3. Análise dos algoritmos propostos por [Ezz-Eldien et al. 2024].

Algoritmo	Complexidade de Tempo	Complexidade de Espaço	Geração baseada no intervalo	Desempenho em aplicações de criptografia
Algoritmo de geração de primos em um intervalo	$O(b)$	$O(b-a)$	Bem adaptado	Alto

primo [Ezz-Eldien et al. 2024].

Algoritmo 1: Geração de primos em um intervalo [Ezz-Eldien et al. 2024]

Entrada: Início a e fim b
Saída: Lista de números primos no intervalo $[a, b]$.

início
- $y \leftarrow \frac{a}{2}$
- $y_1 \leftarrow \frac{b}{2}$
- $x_1 \leftarrow \left\lfloor \sqrt{2 \times y} \right\rceil$
- $N \leftarrow \left\lfloor \sqrt{2 \times y_1} \right\rceil$
- $N_1 \leftarrow \frac{(2\times y)-9}{6}$
- $A_1 \leftarrow lista\ vazia$
- $B_1 \leftarrow lista\ vazia$
- $primos \leftarrow$ lista vazia
- **para** n *no intervalo* $[a,\ b]$ **faça**
  - $m \leftarrow \sum_{n=x_1}^{N} 4n^2 + 4n + 1 + \sum_{l=0}^{N_1} l(4n+2)$
  - $g \leftarrow \sum_{n_1=1}^{N} \left| \left\lfloor \left(\frac{y}{2n_1+1} - n_1 \right) \right\rceil \right|$
  - $m_1 \leftarrow \sum_{n_1=1}^{N} 4n^2 + 4n + 1 + \sum_{l_1=g}^{N_1} l_1(4n_1+2)$
  - adiciona m em A_1
  - adiciona m_1 em B_1
- **fim**
- $primos \leftarrow B_1 - A_1$
- retornar $primos$

fim

4. Análise do modelo baseado em grafos para a criptografia RSA

A abordagem analisada neste trabalho considera a geração de chaves criptográficas RSA a partir de caminhos em digrafos ponderados, como o *Multiprime RSA*, de Collins *et al.* [Collins et al. 1998]. O modelo analisado utiliza digrafos aleatórios para estruturar o processo de criação de chaves. Sob este modelo combinatório, a geração de chaves é modelada como a identificação de caminhos simples, onde os pesos das arestas são primos pré-gerados. O estudo destaca que, ao utilizar a estrutura de caminhos no grafo,

é obtido um ganho significativo em entropia devido ao número massivo de combinações de caminhos possíveis (P_n), o que reduz a dependência da geração frequente de novos números primos.

Em um digrafo ponderado, a quantidade de caminhos P_n possíveis, com $n \geq 4$, é muito grande e, entre o mesmo par de vértice $u, v \in V(G)$, existem diversos caminhos diferentes, e de tamanhos diferentes, possíveis. Porém existe a possibilidade de gerar as mesmas chaves pública e privada mais de uma vez. A razão por utilizar uma quantidade de vértices maior ou igual a 4 ($n \geq 4$) no caminho P_n, é para garantir a utilização de uma quantidade de números primos maior que a criptografia RSA tradicional. Por exemplo, se $n = 4$ a quantidade de números primos a ser utilizada será 3.

$$n = \prod_{i=1}^{k} p_i \qquad (7) \qquad\qquad \phi(n) = \prod_{i=1}^{k} (p_i - 1) \qquad (8)$$

Utilizando os valores calculados nas Equações 7 e 8 (generalizações das Equações 3 e 4) nas Equações 5 e 6, os valores de n, d e e serão utilizados na Equação 1, transformando a mensagem original em mensagem cifrada, e na Equação 2, voltando a mensagem cifrada para a mensagem original.

O modelo analisado considera a criação das chaves pública e privada com o uso de múltiplos primos na proposta de encontrar um caminho $P_{n'}$ em um grafo direcionado ponderado.

A geração de números primos, no Algoritmo 2, que serão os pesos das arestas do digrafo ponderado, será através do Algoritmo 1, proposto por Ezz-Eldien *et al.* [Ezz-Eldien et al. 2024], para gerar um número primo entre o intervalo $[a, b]$, pois é mais eficiente do que os crivos mostrados na Tabela 2, com complexidade de tempo $O(b)$ e complexidade espacial $O(b - a)$. Com este algoritmo e o grafo direcionado ponderado, não haverá necessidade de gerar os números primos a todo momento que for gerar chave de criptografia, como o ***Multiprime RSA***.

Na linha 5, é gerado o digrafo aleatório D com os parâmetros n'' e p'. Na linha 6, são gerados os números primos pelo Algoritmo 1. Na linha 7, são sorteados, de forma aleatória e podendo haver repetição, os números primos que serão os pesos de cada aresta de D. Depois do digrafo criado, é buscado um caminho $P_{n'}$ entre um par de vértices $u, w \in V(D)$, na linha 8. Da linha 10 até a linha 13, são calculados os valores da criptografia RSA para a criação das chaves PU e PR.

Há a possibilidade de divisão em sub-tarefas na cifração e na decifração. Na cifração, cada sub-tarefa pode ser tratada como $C_i = M^{e_i} \ mod \ p_i$, onde e_i é o expoente correspondente ao primo p_i e na decifração, o texto cifrado pode ser decriptado para cada primo $M_i = C^{d_i} \ mod \ p_i$. Nos dois momentos pode ser utilizado o Teorema Chinês do Resto combinando as respostas C_i, para chegar no texto cifrado, e M_i, para voltar para a mensagem original. Porém, essa abordagem geralmente é utilizada quando se tem acesso aos primos, como em armazenamento local [Collins et al. 1998].

A matriz adjacência da Tabela 4, junto com a Figura 1[1], mostra um exemplo de

[1]Os pesos de cada aresta, do digrafo ponderado da Figura 1, não foram colocados para que não dificultasse o seu entendimento. Por isso, a matriz adjacência está após o digrafo.

digrafo D gerado de forma aleatória, com 15 vértices e probabilidade $p = 0,5$ de haver arestas entre cada par de vértices, utilizado para medir os tempos de geração de chaves da Figura 2.

Algoritmo 2: Geração das chaves PU e PR

Entrada: n'' (*número de vértices desejado para o digrafo*), p' (*probabilidade de haver arestas entre cada par de vértices.*), a (limite inferior) e b (limite superior) para o Algoritmo 1

Saída: Chaves PU e PR.

início
- D //$digrafo\ a\ ser\ utilizado$
- $lista\ primos$ //$lista\ de\ primos\ que\ será\ gerada$
- $P_{n'}$ //$caminho\ em\ D\ a\ ser\ utilizado$
- $D \leftarrow D(n'', p')$
- $lista\ primos \leftarrow Algoritmo\ 1(a, b)$
- distribuir os números primos da $lista\ primos$ para cada arestas de D de forma aleatória, pode ocorrer repetição.
- encontrar um caminho $P_{n'}$, com $n' \geq 4$ (n' = quantidade de vértices)
- $n \leftarrow \prod_{i=1}^{n''-1} p_i$, $p_i \in P_{n'}$ (n da criptografia RSA)
- $\phi(n) \leftarrow \prod_{i=1}^{n''-1}(p_i - 1)$
- $e \leftarrow número$
- **while** $mdc(e, \phi(n)) \neq 1$ **do**
  - $e \leftarrow número$
- **end**
- $d \leftarrow e^{-1}\ mod\ \phi(n)$
- retorna $PU \leftarrow \{e, n\}$ e $PR \leftarrow \{d, n\}$

fim

O caminho entre dois vértices é determinado utilizando a Busca em Profundidade (*Deep-First Search - DFS*) modificado, onde um caminho simples pode ser encontrado em tempo $O(V + E)$ [Sedgewick 2002]. Como em um grafo é possível encontrar diversos caminhos entre dois vértices, há possibilidade de criar diversas chaves para a criptografia. No digrafo exemplo da Figura 1, ele possui 105 arestas e $440.343.922$ caminhos $P_{n'}$ existentes, com $n' \geq 4$, considerando caminhos onde os vértices de origem e destino são diferentes e sem repetição de vértices e arestas, onde não se passa por uma aresta ou vértice que já foi passado anteriormente. Apesar da possibilidade de geração de chaves iguais, é possível que seja pequena porque entre dois vértices existem diversos caminhos possíveis.

A complexidade espacial do Algoritmo 2 será $O(n^2)$ pela necessidade de guardar o grafo direcionado na forma de sua matriz de adjacência, onde n é seu número de vértices. Porém, diferentes caminhos podem possuir trechos em comum e para que o invasor consiga quebrar a segurança teria que descobrir os outros números primos utilizados na criação das chaves. Exemplos de caminhos que esta ocorrência no digrafo da Figura 1: $4 \rightarrow 10 \rightarrow 0 \rightarrow 5 \rightarrow 8 \rightarrow 11$ e $2 \rightarrow 1 \rightarrow 10 \rightarrow 0 \rightarrow 6 \rightarrow 12 \rightarrow 14$.

Os gráficos na Figura 2 mostram os tempos de execução do *Multiprime RSA* (Fi-

Figura 1. Exemplo de digrafo ponderado D.

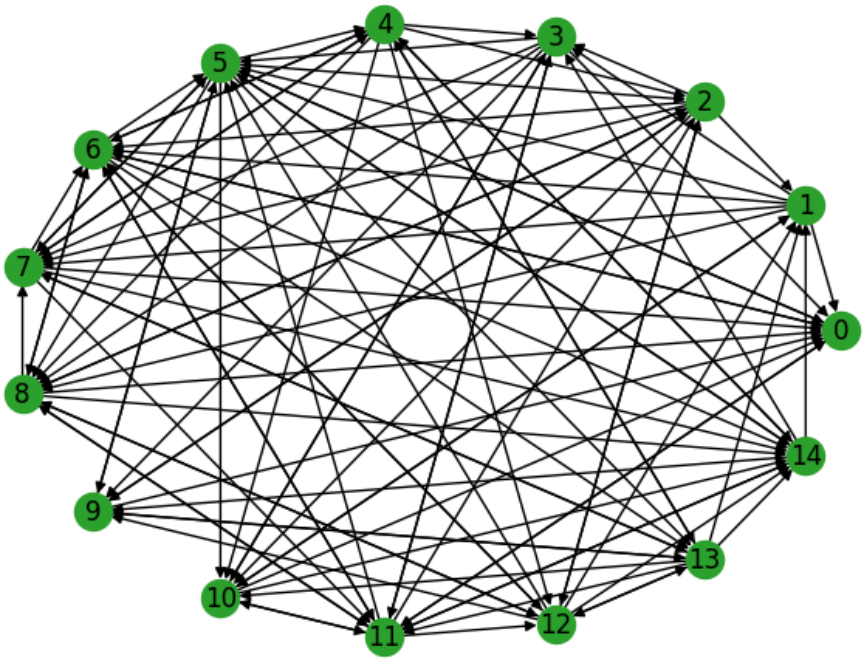
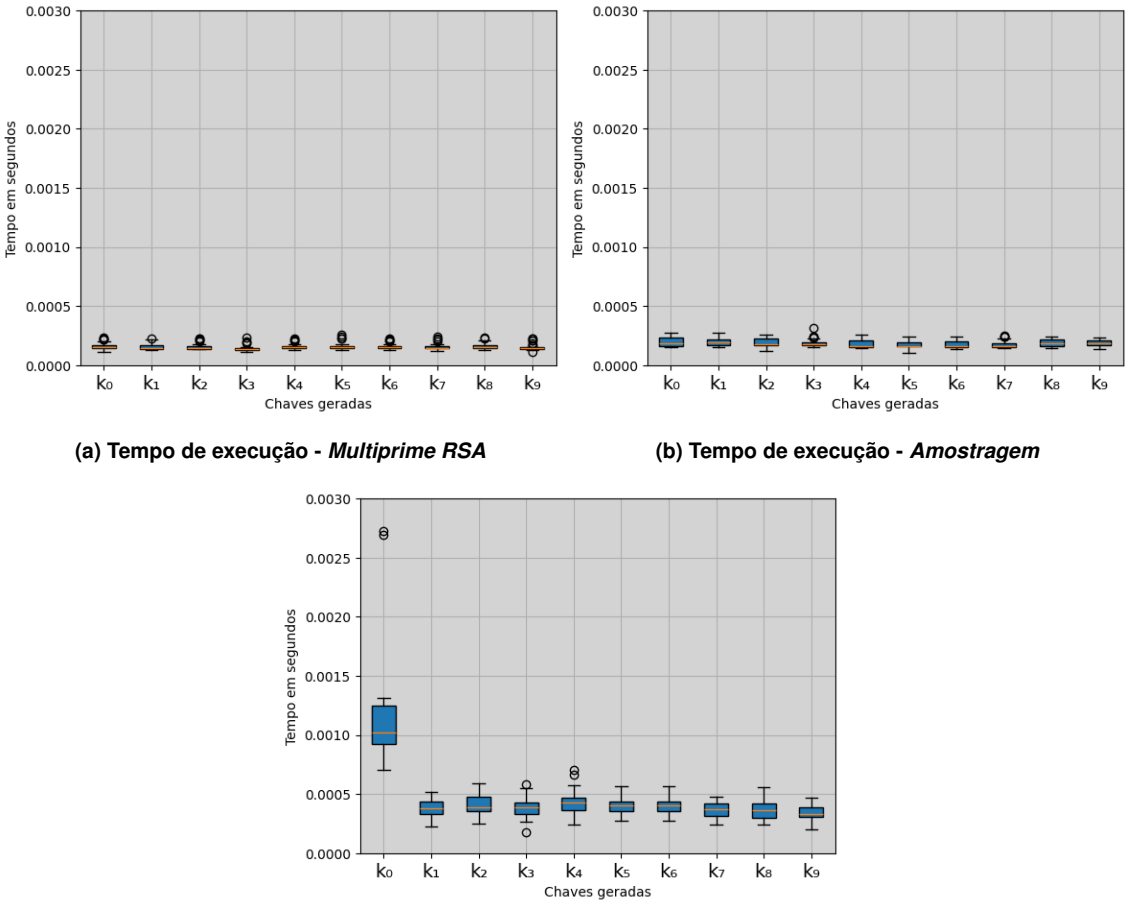


Tabela 4. Matriz adjacência do Digrafo da Figura 1.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
0	0	0	0	0	0	19	71	0	0	0	0	19	0	0	0
1	59	0	0	41	0	59	11	79	37	0	29	0	0	0	0
2	0	59	0	73	0	0	19	71	11	43	11	0	11	0	0
3	53	0	0	0	0	73	0	67	29	89	31	89	0	0	0
4	0	0	19	29	0	0	43	29	61	0	11	0	71	13	59
5	89	0	73	0	47	0	0	11	19	47	31	0	0	89	37
6	73	0	0	0	43	67	0	0	83	0	0	41	83	59	61
7	29	0	0	0	71	61	97	0	0	0	0	97	0	17	19
8	97	0	71	0	0	0	41	29	0	0	0	29	73	0	97
9	79	0	31	0	0	13	0	0	0	0	0	0	0	59	0
10	37	79	0	43	0	0	0	0	0	0	0	59	0	0	23
11	71	0	83	73	0	19	61	0	79	0	19	0	71	0	97
12	0	97	19	0	0	71	47	0	11	83	0	0	0	17	53
13	0	83	0	0	41	0	0	17	0	41	71	31	67	0	31
14	0	71	0	29	67	37	0	0	0	59	0	19	0	0	0

gura 2a), do uso de uma amostra de números primos pertencentes a uma lista de números primos (Figura 2b) e do modelo baseado em grafos (Figura 2c). O objetivo dos gráficos é comparar os tempos de execução de cada algoritmo, executando-os 30 vezes para a criação de cada uma de 10 chaves k_i, $0 \leq i \leq 9$ e medindo o tempo de execução.

Figura 2. Tempo de execução do *Multiprime RSA*, da *Proposta RSA* e de *Amostragem*.



(a) Tempo de execução - *Multiprime RSA*

(b) Tempo de execução - *Amostragem*

(c) Tempo de execução - *Modelo proposto*

De todos os algoritmos utilizados, o *Multiprime RSA* (Figura 2a) mostrou ser o que tem a execução mais rápida. O algoritmo que utiliza amostra de uma lista de números primos (Figura 2b), onde é utilizado o Algoritmo 1 para gerar a lista de números primos dentro de um intervalo e escolhidos uma amostra desta lista, possui o tempo de execução maior do que *Multiprime RSA* porém bem próximo e mais constante, pois possui menos pontos de *outliers*.

O tempo de criação da chave k_0 pelo algoritmo proposto (Figura 2c) é mais alto, em comparação ao *Multiprime RSA*, por ser necessário a geração do digrafo aleatório $G(n, p)$ e a geração dos números primos como peso de cada aresta. Os tempos seguintes, que marcam somente o encontro de um caminho entre dois vértices, são bem menores. No entanto, os tempos de criação das chaves k_i, com $1 \leq i \leq 9$, são aproximadamente de 2 a 3 décimos de milésimos de segundo mais alto em comparação com o *Multiprime RSA*. As marcações circulares indicam tempos de execução *outliers*, tempos que se diferenciam

drasticamente dos outros.

As Tabelas 5, 6 e 7 mostram os intervalos de confiança (95%) na geração de cada uma das chaves de cada algoritmo na Figura 2. Onde são mostrados que a geração das chaves estará dentro do intervalo de tempo [*Mínimo*, *Máximo*]. No caso do algoritmo *Multiprime RSA*, o tempo mínimo e máximo da geração de cada chave são muito próximos, possuindo uma constância menor por possuir bastantes *outliers*. Na abordagem baseada em grafos, como na criação da primeira chave há a geração do digrafo ponderado e dos números primos, o intervalo de tempo na geração de k_0 será maior e para as geração das outras chaves, os intervalos também são muito próximos e possui uma constância maior por possuir menos *outliers*.

Tabela 5. Intervalo de confiança de 95% dos tempos do *Multiprime RSA*.

Chave	Mínimo	Máximo	Chave	Mínimo	Máximo
k_0	$0,000150$	$0,000173$	k_5	$0,000151$	$0,000175$
k_1	$0,000148$	$0,000170$	k_6	$0,000149$	$0,000168$
k_2	$0,000148$	$0,000168$	k_7	$0,000146$	$0,000167$
k_3	$0,000133$	$0,000153$	k_8	$0,000153$	$0,000173$
k_4	$0,000149$	$0,000167$	k_9	$0,000142$	$0,000163$

Tabela 6. Intervalo de confiança de 95% dos tempos do *Algoritmo 2*.

Chave	Mínimo	Máximo	Chave	Mínimo	Máximo
k_0	$0,000974$	$0,001313$	k_5	$0,000376$	$0,000426$
k_1	$0,000354$	$0,000407$	k_6	$0,000376$	$0,000426$
k_2	$0,000370$	$0,000439$	k_7	$0,000344$	$0,000392$
k_3	$0,000359$	$0,000427$	k_8	$0,000341$	$0,000402$
k_4	$0,000383$	$0,000462$	k_9	$0,000318$	$0,000366$

Tabela 7. Intervalo de confiança de 95% dos tempos da algoritmo de amostra de uma lista de primos.

Chave	Mínimo	Máximo	Chave	Mínimo	Máximo
k_0	0.000183	0.000212	k_5	0.000164	0.000186
k_1	0.000184	0.000207	k_6	0.000162	0.000188
k_2	0.000180	0.000206	k_7	0.000164	0.000188
k_3	0.000175	0.000202	k_8	0.000176	0.000200
k_4	0.000164	0.000191	k_9	0.000177	0.000198

O algoritmo que busca uma amostra em uma lista de primos possui uma eficiência melhor do que o Algoritmo 2, apesar dos dois possuírem complexidade linear na quantidade de primos utilizada, a busca pelo caminho entre dois vértices em um digrafo ponderado é menos eficiente.

O sistema de gerenciamento de chaves descreve como as chaves criptográficas são criadas, armazenadas com segurança, distribuídas aos respectivos detentores de chaves e

usadas de acordo com as especificações do protocolo. Portanto, é a base da maioria dos sistemas criptográficos e deve ser tratado com cuidado [Mulder et al. 2023]. O digrafo ponderado seria guardado em um sistema de gerenciamento de chaves, mantendo-o em segurança evitando ataque ou/e acesso a ele e aos pesos (números primos) utilizados na geração das chaves, comprometendo o grafo ou a lista de pesos pode predizer facilmente as chaves subsequentes.

Um método de segurança possível para, caso o digrafo seja acessado de forma não autorizada, manter os números primos desconhecidos seria substituir os pesos das arestas por outros números primos após um período de tempo X (a cada 15 dias, por exemplo) e substituir o digrafo a cada período de tempo Y, onde $Y > X$, (a cada 6 meses, por exemplo). Desta forma, a chance de conhecer os números primos, das chaves que já foram criadas, seria muito menor.

Com a solução mencionada anteriormente, que envolve a geração periódica de novos pesos ou de um novo digrafo, a possibilidade de gerar chaves idênticas é quase nula. Além disso, a abordagem garante que a utilização de caminhos entre dois vértices, que já foram utilizados, seja diferente pois os pesos serão diferentes.

Todavia, a abordagem neste trabalho continua fraca contra a computação quântica, pois o algoritmo de Shor permite que um computador quântico fatore números inteiros grandes em tempo polinomial, quebrando a base da segurança do RSA e de todas as suas variantes.

5. Conclusão

Este trabalho analisa uma abordagem baseada em digrafos ponderados para geração de chaves RSA. Os resultados indicam que, embora a reutilização de primos pré-gerados reduza a necessidade de nova geração de primos a cada chave, o custo associado à construção do grafo, à atribuição dos pesos e à busca de caminhos não resulta em ganho de desempenho quando comparado ao Multiprime RSA e à amostragem direta de uma lista de primos.

Como a geração do digrafo aleatório usa um algoritmo de geração de grafos aleatórios com complexidade linear $O(n + m)$ de Batagelj e Brandes [Batagelj and Brandes 2005], a geração de números primos do Algoritmo 1 é linear e encontrar um caminho entre dois vértices também é linear, então a criação das chaves pública e privada, mostrada no Algoritmo 2, terá um maior custo no momento de associar, em cada aresta, os números primos gerados.

O dígrafo gerado com seus pesos (como o exemplo da Figura 1) não pode ser conhecido pelo atacante, pois com este conhecimento seria possível descobrir as chaves de criptografia através do ataque de força bruta ao descobrir cada caminho possível. Com o custo inicial mais elevado, pela construção do grafo e atribuição dos pesos, o tempo de geração subsequente das chaves é consideravelmente reduzido, porém ainda é maior do que o algoritmo *Multiprime RSA* e a abordagem por *Amostragem* de números primos (de acordo com a Figura 2), não havendo a necessidade do uso de grafos.

Então, a utilização de digrafos ponderados não apresenta ganho de desempenho na geração de chaves, pois a busca do caminho entre dois vértices deixa o algoritmo ineficiente, mesmo com as melhorias de desempenho e complexidade na geração de primos. O

sistema continua se apoiando na fatoração de inteiros como problema de segurança, quebrado pelo algoritmo de Shor, que pode fatorar números grandes em tempo polinomial em computadores quânticos, tornando esta proposta fraca contra computadores quânticos.

Agradecimento

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Referências

Batagelj, V. and Brandes, U. (2005). Efficient generation of large random networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 71(3):036113.

Boneh, D. and Shacham, H. (2002). Fast variants of rsa. *CryptoBytes*, 5(1):1–9.

Botler, F., Collares, M., Martins, T., Mendonça, W., Morris, R., and Mota, G. (2021). *Combinatória*. 33º Colóquio Brasileiro de Matemática. Editora IMPA, 1ª Edition.

Collins, T., Hopkins, D., Langford, S., and Sabin, M. (1998). Public key cryptographic apparatus and method. US Patent 5,848,159.

Deo, N. (2016). *Graph Theory with applications to Engineering ans Computer Science*. DOVER PUBLICATIONS, INC, 1ª Edition.

Ezz-Eldien, A., Ezz, M., Alsirhani, A., Mostafa, A. M., Alomari, A., Alserhani, F., and Alshahrani, M. M. (2024). Computational challenges and solutions: Prime number generation for enhanced data security. *PloS one*, 19(11):e0311782.

Harahap, M. K. and Khairina, N. (2019). The comparison of methods for generating prime numbers between the sieve of eratosthenes, atkins, and sundaram. *Sinkron: jurnal dan penelitian teknik informatika*, 3(2):293–298.

Mulder, V., Mermoud, A., Lenders, V., and Tellenbach, B. (2023). *Trends in DataProtection and Encryption Technologies*. Springer, Open Access.

Paixao, C. A. M. and Gazzoni Filho, D. L. (2005). An efficient variant of the rsa cryptosystem. In *Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg)*, pages 14–27. SBC.

Quisquater, J.-J. and Couvreur, C. (1982). Fast decipherment algorithm for rsa public-key cryptosystem. *Electronics letters*, 18(21):905–907.

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.

Sedgewick, R. (2002). *Algorithms in C: Part 5 - Graph Algorithms*. Always Learning. Addison-Wesley, 3ª Edition.

Szwarcfiter, J. L. (2018). *Teoria Computacional de Grafos: Os algoritmos*. Elsevier, 1ª Edition.

Takagi, T. (1998). Fast rsa-type cryptosystem modulo $p^k q$. In *Advances in Cryptology—CRYPTO'98: 18th Annual International Cryptology Conference Santa Barbara, California, USA August 23–27, 1998 Proceedings 18*, pages 318–326. Springer.

Wiener, M. J. (1990). Cryptanalysis of short rsa secret exponents. *IEEE Transactions on Information theory*, 36(3):553–558.

Apêndice

A. *Script* usado para a contagem de caminhos

```
def contar_caminhos(G):

    total_caminhos = 0

    caminhos = []

    for k in range(0,15):

        for i in range(0, 15):

            if i != k:

                caminhos.append(list(
                nx.all_simple_paths(G, 0, i)))

        for i in range(0,len(caminhos)):

            for j in range(0,len(caminhos[i])):

                if (len(caminhos[i][j]) >= 4):

                    total_caminhos += 1

        caminhos.clear()

    return total_caminhos
```