Um Novo Algoritmo de Criptografia Caótica Utilizando Técnicas de Ordenação a Partir da Medida Invariante

Murilo Daniel Brisolara Cruz, Everton Granemann Souza e Chiara das Dores do Nascimento

***Resumo*— A criptografia caótica é uma maneira eficiente de criptografar textos, porque utiliza recursos de embaralhamento intrínsecos do atrator. Neste trabalho, é apresentado um algoritmo gerador de números pseudo-aleatórios utilizando o mapa logístico. Sua estrutura é baseada no protocolo de Baptista, e utiliza partições da medida invariante associadas a caracteres alfanuméricos. Simulações mostram que a implementação de métodos de ordenação, como BubbleSort, QuickSort, e QuickSort/Etaoin, tornam o algoritmo de encriptação mais rápido que o protocolo de Baptista, com destaque ao QuickSort. A encriptação mostrou um nível de segurança aceitável nos testes de Wald-Wolfowitz e eficácia para mensagens longas.**

***Palavras-Chave*— Criptografia, Caos, Algoritmos de Ordenação, Mapa Logístico.**

***Abstract*— Chaotic cryptography is an efficient way to encrypt texts since it uses the attractor's intrinsic shuffling features. In this work, a pseudo-random number generator algorithm using the logistic map is presented. Its structure is based on the Baptista protocol, which uses invariant measurement partitions associated with alphanumeric characters. Simulations show that the implementation of sorting methods, such as BubbleSort, QuickSort, and QuickSort/Etaoin, make the encryption algorithm faster than the Baptista protocol, especially QuickSort. Encryption showed an acceptable level of security in Wald-Wolfowitz tests and effectiveness for long messages.**

***Keywords*— Cryptography, Chaos, Sorting Algorithms, Logistic Map.**

I. Introdução

Criptografia é o processo de codificação de texto através de um algoritmo, com o intuito de torná-lo restrito a um determinado grupo de interesse [1].

No entanto, mesmo com uma base matemática sólida, algoritmos de criptografia ainda apresentam fraquezas, comprometendo a segurança da informação. Alguns dos algoritmos mais difundidos mundialmente, como o 3DES (do inglês, *Triple Data Encryption Standard*) e o AES (do inglês, *Advanced Encryption Standard*) ainda mostram fragilidade em relação a ataques de força bruta [2]. Nesse sentido, a busca por alternativas aos protocolos de criptografia já existentes ainda permanece em aberto.

O trabalho de Pecora et al. [3] mostrou à comunidade científica a possibilidade de utilizar aplicações do caos, mediante fenômenos de auto-sincronização de oscilações caóticas, como possíveis mecanismos de criptografia [4], [5].

A partir desse esforço inicial, diversos trabalhos foram publicados no campo da criptografia caótica. Dentre eles, um dos mais populares na literatura é o protocolo desenvolvido por Baptista [8]. O algoritmo utiliza a dependência sensível às condições iniciais, característica intrínseca dos atratores caóticos, para a geração segura de números pseudo-aleatórios [9]. Em contrapartida, os resultados fornecem uma cifra com um longo tempo de encriptação e um espaço de criptografia pequeno [10], [11].

Nesse trabalho, apresentamos um algoritmo baseado no protocolo de Baptista. Os esforços foram concentrados na elaboração de um algoritmo de encriptação mais rápido através de métodos de ordenação, criando uma associação eficiente entre a mensagem e o algoritmo.

Para tanto, considerou-se o conceito de medida invariante, que representa a frequência de visitação de uma determinada órbita no espaço de fase [12], e associou-se os intervalos mais frequentes da órbita com as letras mais frequentes do texto. No protocolo original de Baptista, isso era feito de maneira fixa através de uma tabela pré-definida. Nessa situação, pode-se associar caracteres com grande frequência de aparição na frase à intervalos com baixa visitação no atrator, o que torna o algoritmo mais lento.

Duas técnicas de ordenação foram testadas: O Bubblesort e o Quicksort. Além disso, uma versão alternativa do Quicksort (Etaoin) também foi implementada utilizando uma tabela, ao invés do próprio método de ordenação, durante a computação da frequência de letras no texto. Essa tabela contém a frequência média de ocorrência das letras dentro da língua inglesa [13] e simplifica um dos procedimentos inter-operacionais do algoritmo. A segurança da criptografia foi medida através do nível de aleatoriedade da mensagem encriptada via teste estatístico não-paramétrico de Wald-Wolfowitz.

Os resultados mostram que a implementação dos métodos de ordenação tornaram os algoritmos mais rápidos que o protocolo original de Baptista, com destaque ao QuickSort, que foi mais eficiente em regiões onde a visitação dos intervalos é irregular, e mostrou uma aleatoriedade suficiente para os parâmetros de controle analisados.

O artigo é organizado da seguinte forma. Na seção II é especificado o protocolo de Baptista, os métodos de ordenação, os emissores desenvolvidos e o teste de Wald-Wolfowitz. Na seção III é comparado o tempo de encriptação dos algoritmos propostos com o algoritmo de Baptista, avaliado a aleatorie-

Murilo Daniel Brisolara Cruz, Mestrado em Engenharia Eletrônica e Computação, Universidade Católica de Pelotas, Pelotas-RS, e-mail: murilo.cruz@sou.ucpel.edu.br; Everton Granemann Souza, Universidade Católica de Pelotas, Pelotas-RS, e-mail: everton.granemann@ucpel.edu.br; Chiara das Dores do Nascimento, Universidade Católica de Pelotas, Pelotas-RS, e-mail: chiara.nascimento@ucpel.edu.br. Este trabalho foi realizado com o apoio da CAPES/PROSUC (88887.342504/2019-00).

dade da criptografia e discutido os resultados. Na seção IV é concluído o trabalho.

II. Algoritmos, Métodos de Ordenação e Segurança

O mapa logístico é um mapa real unidimensional que, quando iterado, gera séries temporais periódicas ou caóticas [15]. Sua equação é descrita por

$$x_{n+1} = r x_n (1 - x_n), \tag{1}$$

onde $r \in [0; 4]$ é um parâmetro de controle e x representa a variável de estado para o tempo n no intervalo de $[0; 1]$.

O protocolo de Baptista particiona de maneira fixa e equidistante as regiões do atrator caótico e atribui um único caractere para cada partição. Cada intervalo do atrator tem um tamanho ϵ, que é calculado como a razão entre os extremos do intervalo pelo número total de caracteres (S) contidos na tabela ASCII, sendo descrito como

$$\epsilon = \frac{[x_{min} - x_{max}]}{S}. \tag{2}$$

A visitação do atrator não é uniforme e depende do parâmetro de controle r, ou seja, a chance da visitação de uma região do espaço de fase ser maior que outras é grande [16], o que pode tornar o algoritmo de Baptista lento. Na figura 1 é ilustrado esse comportamento para dois parâmetros de controle. No quadro (a), com $r = 3,78$, temos um atrator com visitação irregular. Já em (b), com $r = 4,00$, a visitação é a mais uniforme possível dentre as regiões caóticas disponíveis.

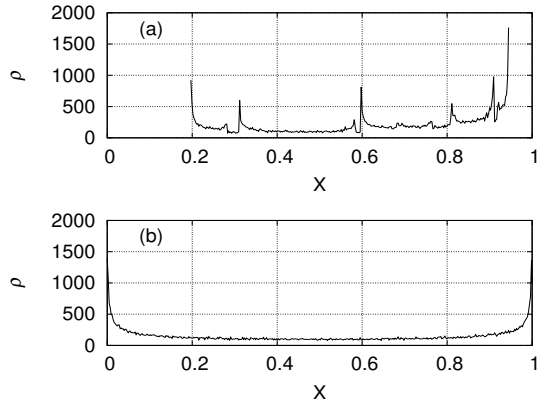

Fig. 1: Medida invariante (ρ) não normalizada para $r = 3,78$ (a) e $r = 4,00$ (b). Observe como a visitação dos intervalos de x em $r = 3,78$ possui muito mais picos do que em $r = 4,00$.

Para contornar uma possível lentidão no algoritmo causada pela visitação irregular do atrator, utilizou-se algoritmos de ordenação, como o Bubblesort e o Quicksort, para associar letras frequentes à intervalos frequentes. Dessa forma, as irregularidades da visitação do atrator podem ser utilizadas para diminuir o tempo da encriptação. A subseção a seguir descreve os métodos de ordenação utilizados.

A. Métodos de ordenação

O BubbleSort utiliza a troca de posição dos elementos (caracteres) de uma lista, de modo que todos estes estejam em ordem decrescente no fim do processo [17]. Assim, dado um vetor A_j que deseja-se ordenar em ordem decrescente, com $j = 1, \ldots, n$, onde n é o comprimento do vetor, o algoritmo compara seus elementos um a um, trocando a posição dos elementos em $j + 1$ e j, caso $A_{j+1} > A_j$.

No melhor caso, quando o vetor já está em ordem decrescente, são realizadas n operações de ordenação. Caso o vetor esteja em ordem crescente, o pior caso, são efetuadas n^2 operações. Esse procedimento é eficaz quando trabalha-se com vetores pequenos, e exponencialmente lento quando utilizado em vetores grandes.

Já o Quick Sort separa o vetor em partes, criando um pivô cada vez que o vetor é dividido [18]. O ideal é que sejam criados blocos de 7 elementos para que o algoritmo tenha uma eficiência maior [19].

Matematicamente, para um dado vetor A, este é dividido em dois subvetores ($A[i \ldots j-1]$ e $A[j+1 \ldots k]$) de forma que $A[i \ldots j-1]$ seja menor que o seu pivô $A[j]$. Os dois subvetores são então ordenados através de substituições entre o pivô e o elemento de comparação. Para uma ordem decrescente, se $A[k] > A[j]$, $A[k]$ assume a posição do pivô. Na sequência são comparados os outros elementos, sempre começando do último elemento do vetor que foi trocado.

Se os subvetores possuem o mesmo tamanho, no caso ideal, são necessárias $n \log n$ operações. No pior caso, quando todos os elementos do subvetor são maiores ou menores que o pivô, são criados subvetores com tamanhos 0 e $n-1$, 0 e $n-2$ e assim sucessivamente até o último subvetor.

B. Emissores

Conforme mencionado, foram utilizados dois algoritmos de ordenação para os emissores, o Bubblesort e o Quicksort. Ainda foi criada uma variante do Quicksort, denominada Etaoin, nomeada assim em analogia aos cinco primeiros caracteres mais frequentes na língua inglesa. Os algoritmos constituem-se de dois procedimentos principais:

- **Subrotinas textuais:** Faz o pré-processamento dos dados, como: leitura da mensagem a ser enviada, contagem dos seus caracteres, determinação da frequência de visitação no espaço de fases (medida invariante) para cada parâmetro de controle e associação dos caracteres em formato ASCII com os intervalos do atrator;
- **Iteração do mapa logístico:** É responsável por executar a encriptação dos dados baseado na tabela criada no procedimento anterior.

A figura 2 mostra o esquema de encriptação utilizado no algoritmo do emissor. No primeiro estágio (esquerda do diagrama) são contabilizados os caracteres e cria-se um vetor com a frequência das letras. Simultaneamente, o atrator é particionado em 256 intervalos de tamanho ϵ, conforme equação 2, o mapa logístico é iterado para se obter a frequência de visitação de cada intervalo (direita do diagrama). Nessa parte, cada emissor vai realizar a ordenação dos vetores utilizando um dos métodos apresentados na subseção II-A, exceto o

emissor Etaoin, o qual utiliza uma tabela predefinida, com a frequência média de ocorrência das letras do alfabeto na língua inglesa. Assim, apenas sua medida invariante é ordenada através do Quicksort. Por fim, os vetores de caracteres e de intervalos do atrator são associados, vinculando os caracteres mais frequentes aos intervalos mais frequentes.

No segundo estágio, o mapa logístico é iterado n vezes até encontrar o intervalo referente a letra desejada, conforme a tabela de frequência ordenada. Após criptografar a primeira letra, a última condição é utilizado como condição inicial ($x_n = x_{n+1}$) para a encriptação do caractere seguinte da mensagem, de modo que a encriptação do caractere posterior dependa do caractere anterior. Devido a dependência sensível as condições iniciais, esse procedimento aumenta a dificuldade de um ataque por força bruta.

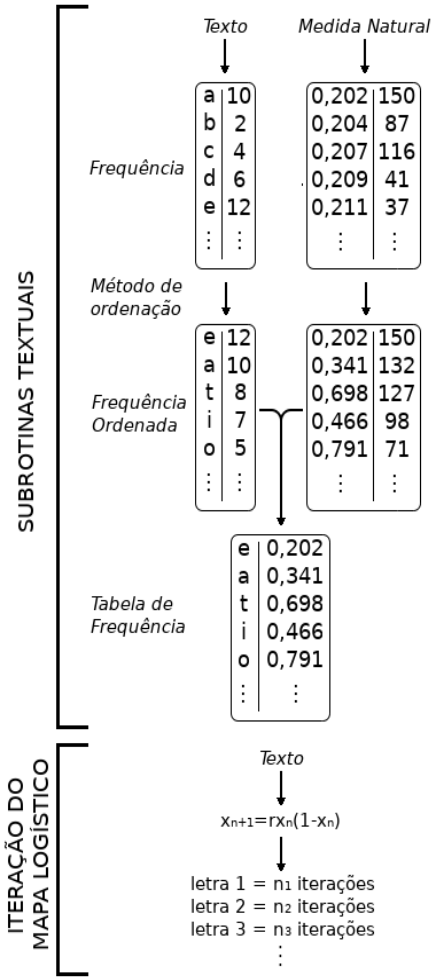

Fig. 2: Diagrama de encriptação do emissor.

C. Testes de Segurança

O teste de Wald-Wolfowitz [20], ou *Runs test*, é um teste não paramétrico, para checar a hipótese de que uma sequência origina-se de um processo randômico. Para tanto, testa-se a hipótese nula (H_0) de que uma determinada sequência é randômica. Caso H_0 seja rejeitada, a hipótese alternativa (H_1) é aceita, e a sequência é dita não-randômica.

O teste de Wald-Wolfowitz considera uma sequência de dados dicotomizada, onde é atribuído o valor 0 para valores abaixo da mediana e 1 para valores acima da mediana. Quando há uma sequência de valores iguais ou valores isolados alternados (seja 0 ou 1), na série dicotomizada, formando uma oscilação, define-se um *run*. Em seguida, é contabilizado o número de *runs* que ocorrem na sequência dicotomizada e o teste estatístico Z é calculado, conforme a equação 3

$$Z = \frac{R - \bar{R}}{s_R}, \tag{3}$$

onde R é o número de *runs*, $\bar{R}$ é o valor esperado de *runs* e s_R é o desvio padrão do número de *runs*. O valor esperado do número de *runs* ($\bar{R}$) e o desvio padrão do número de *runs* (s_R) são calculados conforme as equações 4 e 5

$$\bar{R} = \frac{2n_1n_2}{n_1 + n_2} + 1 \tag{4}$$

$$s_R^2 = \frac{2n_1n_2(2n_1n_2 - n_1 - n_2)}{(n_1 + n_2)^2(n_1 + n_2 - 1)}, \tag{5}$$

onde n_1 é o número de 1 e n_2 o número de 0 na série dicotomizada.

Quando o número de *runs* é grande ($n_1 > 10$ e $n_2 > 10$), como no caso das séries criptografadas, o valor de Z é comparado com o valor crítico ($Z_{1-\alpha/2}$) da distribuição normal [21], levando em conta o nível de significância α, que se refere ao nível de confiança do teste. Dessa forma, se o teste estatístico Z for menor que o valor crítico ($Z_{1-\alpha/2} > Z$), a hipótese nula não é rejeitada e a série é randômica.

III. Resultados e Discussão

Para mensurar a eficácia dos algoritmos de ordenação em relação ao protocolo de Baptista, foi utilizado trechos do livro [12] e gerado um arquivo de texto com 11197 caracteres. Todos os algoritmos foram executados em um processador Intel Quad Core i7-4510U, de 4ª geração com 3.1 GHz.

A tabela I exibe o tempo consumido pelos algoritmos de ordenação para cada um dos procedimentos discutidos na seção anterior. Os valores são obtidos utilizando um parâmetro de controle $r = 3,78$ a partir de uma condição $x_0 = 0,232323$, os mesmos utilizados por Baptista.

O tempo das subrotinas textuais do algoritmo proposto é maior em comparação ao algoritmo de Baptista, considerando que ambos, Bubblesort e Quicksort, necessitam de mais tempo para ordenar e criar dinamicamente as tabelas de frequência ordenada. Em contrapartida, os emissores Bubblesort e Quicksort são 41,26% e 53,68%, respectivamente, mais rápidos que o emissor Baptista. O algoritmo Etaoin foi 27,24% mais rápido em comparação ao emissor Baptista, mas 19,28% e 36,34% mais lento que os emissores Bubblesort e Quicksort.

Para comparar a velocidade dos algoritmos para as outras regiões caóticas, foi criado um mapa de razão de tempo com intervalos de tamanho re, dado por

$$re = \frac{r_{final} - r_{inicial}}{R}, \tag{6}$$

TABELA I: Tempo consumido pelo algoritmo para cada técnica.

	Baptista	Bubblesort	Quicksort	Etaoin
Subrotinas Textuais	0,008752s	0,014027s	0,013367s	0,011788s
Iteração do Mapa Logístico	0,078415s	0,037169s	0,027007s	0,051634s
Total	0,087167s	0,051196s	0,040374s	0,0634224s

onde $r_{inicial}$ é o início do espaço de parâmetros, r_{final} o fim do espaço de parâmetros e R é a quantidade de intervalos utilizados. Para as figuras que seguem foi considerado $R = 100$ dentro de um intervalo de $r = [3,56; 4,00]$.

As figuras 3, 4 e 5 mostram a razão dos tempos de criptografia entre: Bubblesort/Baptista, Quicksort/Baptista e Etaoin/Baptista, respectivamente. Para as simulações, foi considerado uma malha 100×100 com pontos equidistantes nos intervalos $r \in [3,56; 4,0]$ e $x_0 \in [0; 1]$. A escala de cores indica a velocidade relativa do algoritmo, um valor menor que 1 (branco ou verde na escala de cores) corresponde a uma região do espaço de parâmetros onde os algoritmos propostos são mais rápidos que o de Baptista. Para os emissores Quicksort e Etaoin há mais regiões verdes (menores que 1) do que laranja e vermelho (maiores que 1), oferecendo portanto, uma otimização para grande parte do espaço de parâmetros.

Igualmente, a mesma operação foi feita para o emissor Bubblesort, o qual visualmente parece mais rápido no mapa de razão em relação aos emissores Quicksort e Etaoin, uma vez que as franjas laranjas estão presentes em aproximadamente 1/3 do mapa Quicksort/Baptista e metade do mapa Etaoin/Baptista. No entanto, ao observar todo o espaço de criptografia ($r = [3,56; 4,00]$), a média dos intervalos aponta que o emissor Quicksort é mais rápido, conforme podemos ver na Tabela II, a qual será discutida na sequência.

Ainda, o emissor Bubblesort apresenta uma grande região do espaço de parâmetro ocupado por laranja e vermelho na região de $r = 3,56$ a $r = 3,77$, indicando regiões mais lentas que o emissor Baptista.

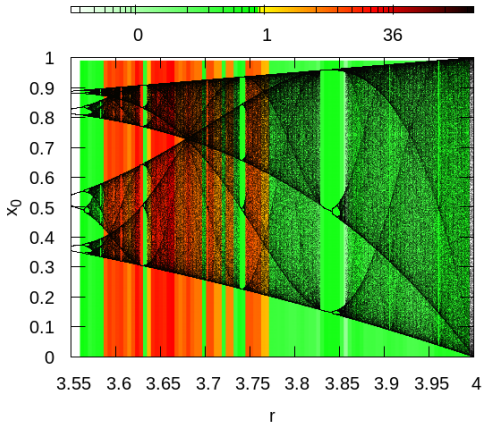

Fig. 3: Mapa da razão do tempo de encriptação. As regiões entre branco e verde identificam onde o emissor Bubblesort é mais rápido que o protocolo de Baptista.

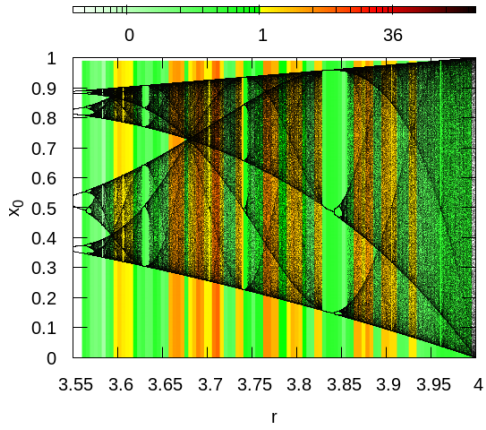

Fig. 4: Mapa da razão do tempo de encriptação. As regiões entre branco e verde identificam onde o emissor Quicksort é mais rápido que o protocolo de Baptista.

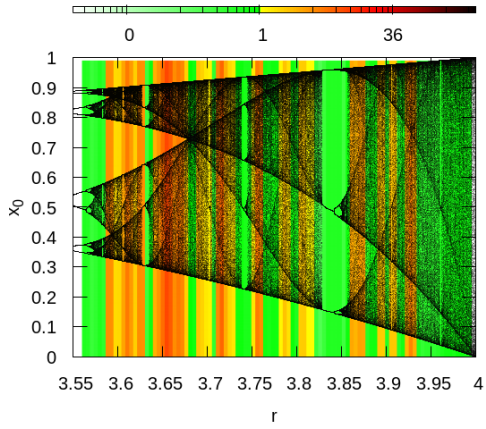

Fig. 5: Mapa da razão do tempo de encriptação. As regiões entre branco e verde identificam onde o emissor Etaoin é mais rápido que o protocolo de Baptista.

Os mapas mostrados anteriormente são dependentes do tipo e do tamanho das frases encriptadas. Dessa forma, é importante avaliar esses parâmetros afim de obter uma ideia geral da eficiência de todos os protocolos estudados. A estratégia adotada é aumentar o tamanho da frase gradativamente, pois à medida que a frase encriptada aumenta de tamanho, a frequência de ocorrência das letras na frase se aproxima mais da frequência média de ocorrência dessas letras no alfabeto. Assim, frases longas possuem uma maior similaridade média entre si do que frases curtas.

A média do tempo de execução dos algoritmos sobre os parâmetros de controle do mapa (100 valores no intervalo $r = [3,56; 4,0]$) nos fornece o tempo médio de encriptação. Na tabela II é exibido esse tempo médio, para cada algoritmo, para uma frase com 11197 caracteres.

Para se ter uma estimativa da dependência dos resultados em relação ao tamanho da frase, pode-se também avaliar os valores médios do tempo de criptografia, para o intervalo de $r = [3,56; 4,0]$, variando o número de caracteres da frase. Esse resultado é exibido na Figura 6, que mostra os resultados

TABELA II: Tempo médio consumido pelos algoritmos.

	Baptista	Bubble Sort	Quick Sort	Etaoin
Tempo Médio	1,112497s	1,096456s	0,570206s	0,957603s

da Tabela II para 9 tamanhos de frases.

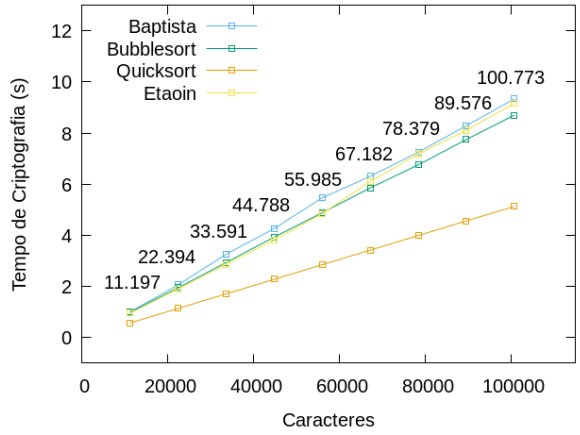

Fig. 6: Velocidade de encriptação média para os algoritmos analisados. No testes, foram utilizados 9 tamanhos de texto, encriptados a partir da condição inicial $x_0 = 0,232323$.

Através da média de velocidade exibida no gráfico, percebe-se a melhora no tempo de encriptação do algoritmo que utiliza o Quicksort. Para uma mensagem com 11197 caracteres, a redução no tempo foi de $1,00$ segundo para $0,57$ segundos, aproximadamente 43,32% de vantagem quando comparamos os tempos com o emissor de Baptista.

Para avaliar a segurança da mensagem encriptada, o teste de aleatoriedade de Wald-Wolfowitz foi aplicado para 5 regiões do atrator caótico com dinâmicas distintas: $r = 3,62$ (próxima do *onset* do caos), $r = 3,78$ (região caótica com medida invariante não-uniforme), $r = 3,82$ (antes de uma janela periódica, $r = 3,93$ (em uma região caótica ligeiramente uniforme) e $r = 4,00$ (região caótica com medida invariante quase uniforme). Para dicotomizar as séries criptografadas, considerou-se a mediana como valor de referência.

Para os testes, optou-se por uma significância $\alpha = 0,05$, ou seja, 95% de nível de confiança. Assim, quando $p < \alpha \rightarrow h = 1$ e a hipótese nula, que as séries temporais são randômicas, é rejeitada. Em contrapartida, as séries são consideradas randômicas quando a hipótese nula não é rejeitada, ou seja, quando $p > \alpha \rightarrow h = 0$. A Tabela III indica os valores obtidos nos testes de Wald-Wolfowitz para os valores de r escolhidos. A aleatoriedade foi confirmada para todas as séries temporais criptografadas pelos emissores de Baptista e o Quicksort. Já os emissores Bubblesort e Etaoin mostraram alguns pontos inseguros, como em $r = 3,78$, para ambos, e em $r = 3,62$ apenas para o Bubblesort.

IV. Conclusões

Neste trabalho, é proposto um novo algoritmo de criptografia caótica baseado nos métodos de classificação Bubblesort, Quicksort e Etaoin. Todos os algoritmos provaram ser mais rápidos em relação ao tempo de encriptação quando comparados ao emissor Baptista. Dentre todos, o algoritmo baseado em Quicksort apresentou o melhor desempenho, considerando a velocidade de criptografia em relação ao algoritmo de Baptista e o tamanho da mensagem. Os algoritmos também se provaram seguros segundo os critérios do teste de Wald-Wolfowitz.

TABELA III: Teste de Wald-Wolfowitz para alguns valores aleatórios de r. Para cada um dos quatro algoritmos abaixo, é indicado o valor de h obtido no teste, com o seu respectivo valor de p, entre parênteses.

r	Baptista	Bubblesort	Quicksort	Etaoin
3,62	0(0.42336)	1(0.044086)	0(0.43019)	0(0.14212)
3,78	0(0.19849)	1(0.024951)	0(0.081509)	1(0.0031776)
3,82	0(1.00000)	0(0.12308)	0(0.89490)	0(0.26729)
3,93	0(0.53890)	0(0.83538)	0(0.81128)	0(0.47893)
4,00	0(0.64302)	0(0.55768)	0(0.069431)	0(0.34842)

Referências

[1] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, "Handbook of Appl. Cryptography", *CRC Press. New York*, 1996.

[2] F. C. Silva e J. J. Sousa, "New Comparative Study Between DES, 3DES and AES within Nine Factors", *Journal of Computing*, v. 2, n. 3, pp. 152–157, 2010.

[3] L. M. Pecora and T. L. Carroll, "Synchronization in Chaotic Systems", *Physical Review Letters*, vol. 64, no. 8, pp. 821–824, 1990.

[4] W. Kinzel, A. Englert, e I. Kanter, "On chaos synchronization and secure communication", *Phil.Trans.R.Soc.A*, 2010.

[5] P. Stavroulakis, "Chaos Applications in Telecommunications", *Press: New York*, 2005.

[6] C. P. Silva, "A survey of chaos and its applications,"1996 IEEE MTT-S International Microwave Symposium Digest, San Francisco, CA, USA, 1996, pp. 1871-1874 vol.3.

[7] B. Jovic, "Synchronization Techniques for Chaotic Communication Systems", *Springer*, Auckland: New Zeland, 2011.

[8] M. S. Baptista, "Cryptography with chaos", *Physics Letters A*, 240(1-2), 50-54, 1998.

[9] M. A. Murillo-Escobar et al., "Implementation of an improved chaotic encryption algorithm for real-time embedded systems by using a 32-bit microcontroller", *Microprocessors and Microsystems*, 2016.

[10] D. Arroyo, G. Alvarez e V. Fernandez, "On the inadequacy of the logistic map for cryptographic applications", *Instituto de Física Aplicada*, Madrid, Espanha, 2008.

[11] L. Kocarev, "Chaos-based cryptography: A brief overview. Circuits and Systems Magazine", *IEEE*. 1. 6 - 21, 2002.

[12] K. T. Alligood, T. Sauer e J. Yorke, "Chaos: An Introduction to Dynamical Systems", *Springer-Verlag New York*, 1996.

[13] P. Norvig, "English Letter Frequency Counts: Mayzner Revisited or ETAOIN SRHLDCU", Disposto em http://www. norvig. com/mayzner. html.[acesso em 2019-11-12], 2013.

[14] M. Bellare and P. Rogaway, "Introduction to Modern Cryptography",*Ucsd Cse*, 207, 207, USA, 2005.

[15] R. M. "May, Simple mathematical models with very complicated dynamics", *Nature*, 261(5560), 459-67, 1976.

[16] E. Ott, "Chaos in Dynamical Systems", *Cambridge University Press*, University of Maryland - College Park, Maryland, Estados Unidos, 1993.

[17] T. Cormen, C. E. Leiserson, R. L. Rivest e C. Stein, "Introduction to Algorithms", *The MIT Press*, Cambridge, Massachussets, Inglaterra, 2009.

[18] P. Biggar e D. Gregg, "Sorting in the Presence of Branch Prediction and Caches", *TCD-CS*, Universidade de Dublin, Irlanda, Agosto de 2005.

[19] "Numerical Recipes in Fortran 77: The Art of Scientific Computing", *Cambridge University Press*, 1986-1992.

[20] A. Wald and J. Wolfowitz, "On a Test Whether Two Samples are from the Same Population". *Ann. Math. Stat.* 11. 147-168, 1940.

[21] J. J. Filliben, "Runs Test for Detecting Non-randomness", *NIST/SEMATECH*, January 11, 2018. Accessed on: September 21, 2020. [Online]. Available: https://www.itl.nist.gov/div898/handbook/eda/section3/eda35d.htm.