Proceeding Series of the Brazilian Society of Computational and Applied Mathematics, v. 11, n. 1, 2025.

Trabalho apresentado no XLIII CNMAC, Centro de Convenções do Armação Resort - Porto de Galinhas - PE, 2024

Proceeding Series of the Brazilian Society of Computational and Applied Mathematics

Um Bloco de Permutação para Construções Esponja Baseado na Transformada do Cosseno sobre Corpos Finitos de Característica Dois

Laís Maria Rodrigues de Araújo[1], Juliano B. Lima[2]
DES/UFPE, Recife, PE
José R. de Oliveira Neto[3]
DEMEC/UFPE, Recife, PE

Resumo. Neste trabalho, é proposto um novo bloco de permutação de comprimento $(2+n)\times 32$, em que n é um número natural positivo, para utilização em construções esponja. Este bloco de permutação emprega um polinômio de permutação juntamente com a transformada do cosseno do tipo 1, ambos definidos sobre o corpo finito $\mathbb{F}_{2^8}$. Quando utilizado em algoritmos *hash* que empregam construções esponja, o bloco proposto torna possível reduzir o número de rodadas de aplicação do bloco de permutação para dois em todos os algoritmos *hash* testados.

Palavras-chave. Construção Esponja, Corpos Finitos de Característica 2, *Hash*, Permutação, Transformada do Cosseno Sobre Corpos Finitos

1 Introdução

Primitivas criptográficas constituem os blocos fundamentais na construção de sistemas criptográficos [10]. A partir dessas primitivas, surgem diversos sistemas de segurança e comunicação segura, tais como algoritmos de cifragem e decifragem [8, 15, 16], e funções *hash* [2, 4, 9, 13, 27]. Nesse contexto, uma primitiva que tem ganho destaque é a construção esponja, revelando-se como uma estrutura versátil capaz de se adaptar a diversos requisitos de segurança [7]. Nos últimos anos, construções esponja vem sendo implementadas em diferentes funções *hash* [2, 4, 9, 13, 27].

A construção esponja é, especificamente, um modelo iterativo simples que emprega um bloco de permutação $P(\mathbf{I})$, em que $\mathbf{I}$ é um vetor binário de b bits. Essa construção aceita uma entrada de comprimento variável e produz uma saída de comprimento arbitrário, sendo definida por quatro parâmetros fundamentais: o tamanho do estado interno b, a taxa de bits r (de entrada ou saída da construção por ciclo), a capacidade c e o tamanho do *hash* de saída n, obedecendo $b = r + c$ [3, 18]. De uma forma geral, os blocos de permutação empregados em construções esponja propostas na literatura utilizam um conjunto de operações binárias tais como soma por contantes, operações ou-exclusivo, S-boxes e embaralhamento de bits [2, 4, 9, 13, 27]. A segurança do bloco vem do fato de que a permutação é utilizada recursivamente. Por exemplo, o ASCON-*hash*, o SHA3 e o S-Quark utilizam 12, 24 e 1024 rodadas de aplicação do bloco de permutação por ciclo, respectivamente [4, 9, 27].

Desde a década de 1970, as transformadas definidas sobre corpos finitos têm sido investigadas e empregadas em cenários práticos [21, 23]. Mais recentemente, essas transformadas têm desempenhado um papel importante em cenários relacionados a criptografia e segurança de sinais digitais [8,

[1]lais.maria@ufpe.br
[2]juliano.lima@ufpe.br
[3]joserodrigues.oliveiraneto@ufpe.br

14–18, 20]. Isso se deve basicamente ao fato de pequenas modificações no sinal de entrada gerarem mudanças significativas no sinal (transformado) de saída, o qual resulta de operações sobre as mencionadas estruturas algébricas [8]. Entre essas transformadas, encontram-se as transformadas definidas sobre corpos de característica dois, ou seja, corpos da forma $\mathbb{F}_{2^r}$, $r \in \mathbb{N}^*$ [14, 15]. Ferramentas matemáticas definidas sobre corpos de características dois têm o benefício de permitir um mapeamento direto de palavras binárias de r bits em elementos do corpo $\mathbb{F}_{2^r}$ [15].

Embora as transformadas em questão sejam adequadas para prover difusão e confusão em sistemas de segurança, é importante que também sejam adicionadas ao sistema não-linearidades, a fim de dificultar ataques [8, 10, 15]. Neste contexto, funções de permutação que trabalham com pequenos conjuntos de bits, como S-boxes e polinômios de permutação, têm sido estudados e utilizados como blocos de sistemas criptográficos [1, 19, 28].

Neste trabalho, é proposto um novo bloco de permutação visando sua utilização em construções esponja. Esse bloco de permutação emprega um polinômio de permutação [19] juntamente com a transformada do cosseno do tipo 1 sobre corpos finitos de característica 2 [14]. Adotando o bloco proposto, é possível reduzir o número de rodadas de aplicação para 2, quando utilizado em algoritmo de *hash* [2, 4, 9, 13, 27].

O restante deste trabalho é dividido da seguinte forma: na Seção 2, são apresentados os conceitos matemáticos necessários para o entendimento do trabalho; na Seção 3, o novo bloco de permutação proposto é definido; na Seção 4, os resultados dos testes relacionados a implementação do novo bloco de permutação e seu uso em funções *hash* são apresentados e confrontados com os resultados encontrados para os algoritmos orignais. Por fim, na Seção 5 são apresentadas as conclusões do trabalho.

2 Preliminares

2.1 Construções Esponja

Na Figura 1, é mostrado o esquema geral de uma construção esponja, destacando suas duas fases distintas: absorção e compressão. Primeiramente, antes de iniciar a fase de absorção, todos os bits do estado são inicializados com valor zero, e a mensagem de entrada é dividida em blocos, TS_i, de r bits. Seguindo para a fase de absorção, ocorre uma operação XOR nos blocos de mensagem de entrada de r bits, intercalada com a aplicação da bloco de permutação $P(.)$. Após o processamento de todos os blocos de mensagens, a construção esponja inicia a fase de compressão. Nesta fase, os primeiros r bits do estado são retornados como blocos de saída, H_j, intercalados com aplicações de $P(.)$, até que chegue ao comprimento desejado [6].

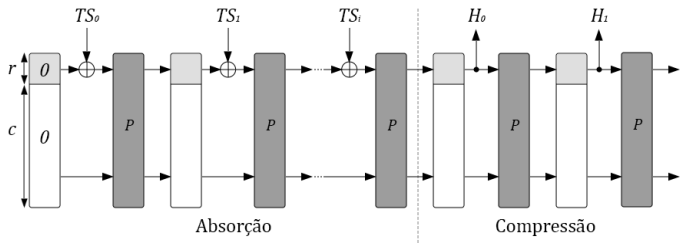

Figura 1: Diagrama de blocos de uma Construção Esponja. Fonte: Adaptado de [18].

2.2 Polinômios de Permutação

A partir dos resultados apresentados em [1], os autores de [19] apresentam casos específicos de classes de polinômios de permutação que são involuções sobre um corpo finito $\mathbb{F}_{q^2}$, em que q é a potência de um primo. Dentre essas classes, encontra-se a descrita a seguir:

Corolário 2.1. *Seja m um inteiro positivo, $q = 2^{2m}$ e k um inteiro tal que $1 \leq k \leq q/2$. Seja $f(x) = x^{q^2-2}h(x^{q-1})$, em que $h(x) = \gamma(1 + \beta x^k + \beta^{-1}x^{-k})$, $\gamma \in \mathbb{F}_{q^2}^*$ e $\beta \in \mu_{q+1}$. Então $f(x)$ é uma involução em $\mathbb{F}_{q^2}$ se e somente se $\beta^2\gamma^{(q-1)k} = 1$ [19].*

Na descrição acima, γ é um elemento primitivo do corpo finito $\mathbb{F}_{q^2}^*$ e μ_{q+1} é um subgrupo de $\mathbb{F}_{q^2}^*$ tal que $\mu_{q+1} = \{x \in \mathbb{F}_{q^2}^* : x^{q+1} = 1\}$. Então, $f(x)$ é dado por

$$f(x) = x^{-1}\gamma(1 + \beta x^{kq-k} + \beta^{-1}x^{-kq+k}). \tag{1}$$

Ao analisar involuções, é necessário considerar a quantidade de pontos fixos que essas funções possuem, especialmente no contexto da criptografia, em que tal característica desempenha um papel crucial na segurança do algoritmo. Um ponto fixo em uma permutação, representado por $f(a) = a$, é um elemento que permanece inalterado sob a aplicação de uma função. Experiências práticas conduzidas por [28] revelam uma correlação entre as propriedades criptográficas de algoritmos e a quantidade de pontos fixos, sugerindo que uma S-box deva conter poucos ou nenhum ponto fixo. Em [19], é feito um estudo de pontos fixos dos polinômios de permutação propostos; o caso específico da involução em (1) apresenta apenas 2 pontos fixos.

2.3 Uma Transformada do Cosseno sobre Corpos Finitos de Característica Dois

Em [14], são introduzidas quatro transformadas do cosseno sobre corpos finitos de característica 2, isto é, sobre $\mathbb{F}_{2^r}$, $r \in \mathbb{N}^*$. Para isso, é primeiramente definida a função cosseno sobre um corpo de característica 2:

Definição 2.1. *Seja $\zeta \in \mathbb{F}_{2^r}$ um elemento com ordem multiplicativa denotada por* $\mathrm{ord}(\zeta)$*. A função cosseno de corpo finito relacionada a ζ é definida como*

$$\cos_\zeta(n) := \zeta^n + \zeta^{-n}, \tag{2}$$

em que $n \in \mathbb{Z}$ [14].

A função em (2) possui a propriedade de que todos os valores distintos possíveis para $\cos_\zeta(n)$ são encontrados para $n \in \{0, 1, ..., \mathrm{ord}(\zeta) - 1\}$ [14].

A transformada do cosseno sobre corpos finitos do tipo 1 (FFCT-I, do inglês *finite field cosine transform*) é então definida por:

Definição 2.2. *(FFCT-I) [14] Seja $\zeta \in \mathbb{F}_{2^r}$ um elemento tal que* $\mathrm{ord}(\zeta) = 2N - 1$*. A FFCT-I do vetor $\mathbf{x}$, cujas componentes são $x[n] \in \mathbb{F}_{q^2}$, $n = 1, 2, \ldots, N-1$ é o vetor $\mathbf{X}$, cujas componentes $X[k] \in \mathbb{F}_{2^r}$, $k = 1, 2, \ldots, N-1$, são dadas por*

$$X[k] := \sum_{n=1}^{N-1} x[n] \cos_\zeta(kn). \tag{3}$$

A forma matricial de FFCT-I é dada por $\mathbf{X} = \mathbf{C}\mathbf{x}$, em que $\mathbf{C}$ é uma matriz de dimensões $(N-1) \times (N-1)$, cujos elementos são dados por $C[k, n] = \cos_\zeta(kn)$. Além disso, como a FFCT-I é uma involução [14], a matriz da transformada direta $\mathbf{C}$ é igual a matriz da transformada inversa, ou seja, $\mathbf{C} = \mathbf{C}^{-1}$.

DOI: 10.5540/03.2025.011.01.0492

3 Definição de um Novo Bloco de Permutação com Comprimento $b = (2 + n) \times 32$

O bloco de permutação proposto é descrito no Algoritmo 1, em que $f(.)$ é uma função de permutação que emprega um polinômio de permutação dado por (1) e $\mathbf{C}$ é a matriz de transformação da FFCT-I, sendo ambas definidas sobre $\mathbb{F}_{2^8}$. A entrada do bloco é um vetor binário $\mathbf{I}$ de comprimento $b = (2 + n) \times 32$, $n \in \mathbb{N}^*$. Logo, a função $f(.)$ permuta uma palavra de 8 bits em outra palavra de 8 bits, e a matriz $\mathbf{C}$ é utilizada para transformar um bloco de $8 \times 8 = 64$ bits. A sobreposição de 32 bits a cada aplicação de $\mathbf{C}$ é feita para que a informação seja difundida entre os blocos e aplicação de $f(.)$ escolhida para adicionar confusão e não-linearidade ao sistema [10].

Algoritmo 1: Bloco de Permutação $P(.)$ Proposto

input : $\mathbf{I} \rightarrow$ vetor binário de comprimento b.
output: $\mathbf{I}' \rightarrow$ vetor binário de comprimento b.
begin
  $\mathbf{I}' \leftarrow \mathbf{I}$
  for $n \leftarrow 0$ **to** $(b/8 - 1)$ **do**
    $\mathbf{I}'[8n : 8n + 7] \leftarrow f(\mathbf{I}'[8n : 8n + 7])$
  end
  for $n \leftarrow 0$ **to** $(b/32 - 1)$ **do**
    $\mathbf{I}'[32n : 32n + 63 \,(\mathrm{mod}\, b)] \leftarrow \mathbf{C} \times (\mathbf{I}'[32n : 32n + 63 \,(\mathrm{mod}\, b)])$
  end
end

A partir da especificação do corpo finito, neste caso $\mathbb{F}_{2^8}$, são encontrados os β's e seus respectivos k's que obedecem ao Corolário 2.1. Ou seja, é necessário achar os pares β e k para os quais a relação $\beta^2\gamma^{(q-1)k} = 1$ é satisfeita, em que $\beta \in \mu_{17}$ e $1 \leq k \leq 8$. O elemento primitivo utilizado é $\gamma = \alpha^8 + \alpha^4 + \alpha^3 + \alpha^2 + 1$, em que α é um elemento gerador do corpo. Para calcular $\mathbf{C}$, é usado $\zeta = \alpha^{15} = \alpha^5 + \alpha^2 + \alpha$, com ordem $ord(\zeta) = 2N - 1 = 17$. Dessa forma, é gerada uma matriz quadrada de comprimento $N - 1 = 8$.

4 Implementações e Resultados

Para testar o bloco de permutação proposto, foi substituído o bloco de permutação em cada um dos seguintes algoritmos: SHA3-224, SHA3-256, SHA3-384, SHA3-512, S-Quark, ASCON-*hash*, e SPONGENT-128/256/128, SPONGENT-160/320/160, SPONGENT-224/448/224, SPONGENT-256/256/128, SPONGENT-256/512/256, chamados doravante de SPONGENT-1, SPONGENT-2, SPONGENT-3, SPONGENT-4 e SPONGENT-5, respectivamente [2, 4, 9, 13, 27]. Os algoritmos escolhidos foram selecionados de forma que o comprimento do estado interno b seja um múltiplo de 32, permitindo a aplicação do bloco proposto.

Os experimentos foram realizados utilizando a linguagem de programação Python 3.10.12 e a biblioteca Sagemath, versão 9.5, para operações em corpos finitos. Os códigos base utilizados são: SHA3 [12], SPONGET [22], S-Quark [24] e ASCON [11].

Como teste preliminar para ajustar o bloco de permutação proposto a cada umas das funções *hash*, a função de permutação foi testada para cada β. Para calcular o *hash* de cada algoritmo, foi utilizada a mensagem de entrada "EMBARCADOS", que possui 80 bits. Em seguida, invertendo um bit por vez, a *hash* é calculada novamente, totalizando em 80 *hashes* diferentes para cada algoritmo. Durante esses testes, foi aplicada a função de efeito avalanche da mensagem original a

Tabela 1: β's escolhidos para cada algoritmo.

n^o	k	β	**Algoritmos**
0	7	$\alpha^3+\alpha^2+\alpha+1$	SHA3-384, SPONGENT-2
1	5	$\alpha^7+\alpha^6+\alpha^4+\alpha^3+\alpha^2+\alpha+1$	
2	3	$\alpha^4+\alpha^3+\alpha$	SPONGENT-3, SPONGENT-4
3	1	$\alpha^5+\alpha^4+\alpha^3+\alpha+1$	SPONGENT-1
4	6	$\alpha^6+\alpha^4+\alpha^3+1$	
5	4	$\alpha^5+\alpha^2$	SHA3-256, SHA3-512, S-Quark, ASCON-*hash*
6	2	$\alpha^5+\alpha^3+\alpha^2$	SHA3-224, SPONGENT-5

cada *hash* resultante das alterações de um único bit. O efeito avalanche é dado por:

$$\text{Efeito Avalanche} = \frac{\text{Hamming}(H(\mathbf{x}), H(\mathbf{x}'))}{n}, \tag{4}$$

em que Hamming() é a distância de *Hamming*, que conta os pares de bits diferentes entre duas *strings*, $H()$ é a função *hash*, n é o tamanho da saída do *hash*, e $\mathbf{x}$ e $\mathbf{x}'$ são duas *strings* de bits que só diferem em um bit. Em (4), é esperado um valor próximo a $0,50$ para indicar um efeito de avalanche adequado. De acordo com [5], um resultado na faixa de $0,45-0,60$ é considerado bom. Na Tabela 1 são mostrados os valores de β, k e para quais algoritmos eles foram utilizados.

Uma vez escolhidos os valores de β, foram novamente realizados os testes de efeito avalanche, desta vez, utilizando uma mensagem de entrada de 80 bits, gerada utilizando uma função pseudoaleatória. O cálculo da *hash* foi realizado para cada algoritmo considerando o algoritmo original e utilizando o bloco de permutação proposto. Porém, quando utilizado o bloco proposto, são empregadas apenas duas rodadas de aplicação de $P(.)$ por ciclo. Em seguida, para avaliar a sensibilidade da *hash*, um bit por vez foi invertido, resultando em um total de 80 *hashes* distintas para cada algoritmo e calculado o efeito avalanche. Este procedimento foi repetido 100 vezes para cada algoritmo, totalizando 8000 *hashes* analisadas por algoritmo. Os resultados são apresentados na Tabela 2. Na tabela, é apresentado o valor médio, desvio padrão σ, valor máximo e mínimo do efeito avalanche para cada um dos algoritmos testados. É possível observar que, independentemente do algoritmo de *hash* utilizado e do uso do bloco de permutação original ou do proposto, os valores ficam muito próximos aos desejado.

Tabela 2: Testes de efeito avalanche comparando o bloco de permutação proposto com o algoritmo original.

Algoritmo	Com o Bloco Proposto				Algoritmo Original			
	Média	σ	Máximo	Mínimo	Média	σ	Máximo	Mínimo
ASCON-*hash*	0,4998	0,0311	0,6055	0,3867	0,5003	0,0309	0,6250	0,3867
S-Quark	0,5004	0,0312	0,6055	0,3906	0,4992	0,0314	0,6172	0,3672
SHA3-224	0,4994	0,0359	0,6250	0,3795	0,4996	0,0335	0,6250	0,3750
SHA3-256	0,4999	0,0343	0,6250	0,3594	0,5002	0,0311	0,6370	0,3906
SHA3-384	0,5003	0,0284	0,5938	0,3802	0,4994	0,0252	0,6020	0,3906
SHA3-512	0,4995	0,0241	0,5859	0,4121	0,4995	0,0222	0,5780	0,4199
SPONGENT-1	0,5003	0,0439	0,6641	0,3438	0,5003	0,0443	0,6800	0,3125
SPONGENT-2	0,5000	0,0392	0,6438	0,3625	0,4999	0,0393	0,6563	0,3687
SPONGENT-3	0,5004	0,0334	0,6205	0,3750	0,5001	0,0332	0,6161	0,3839
SPONGENT-4	0,5005	0,0320	0,6172	0,3828	0,5003	0,0313	0,6055	0,3867
SPONGENT-5	0,4999	0,0309	0,6094	0,3789	0,5005	0,0311	0,6094	0,3867

Além do teste do efeito avalanche, foram realizados testes para avaliar a robustez dos algoritmos de *hash* frente aos ataques de pré-imagem, segunda pré-imagem e colisão [25, 26]. Para cada um dos três testes foram calculados 10000 *hashes* para cada um dos algoritmos em suas versões originais e utilizando o bloco de permutação proposto. Todos os algoritmos, tanto aqueles que utilizam o novo bloco de permutação quanto os originais, demonstraram sucesso no conjunto de testes.

5 Considerações Finais

Neste trabalho, foi proposto um novo bloco de permutação visando sua utilização em construções esponja. Este bloco de permutação emprega um polinômio de permutação juntamente com a transformada do cosseno do tipo 1 sobre corpos finitos de característica 2. Quando se emprega o bloco proposto, em substituição ao bloco de permutação, em algoritmos de *hash* encontrados na literatura, apenas duas aplicações do bloco proposto por rodada são suficientes para entregar resultados semelhantes aos dos algoritmos originais. Entre trabalhos futuros estão a realização de mais testes de segurança sobre o bloco proposto, a avaliação do seu uso em algoritmos de cifragem que utilizam construções esponja e a procura por algoritmos de cálculo eficiente da estrutura proposta em implementações utilizando linguagens de programação de mais baixo nível e o uso de hardware dedicado.

Referências

[1] Amir Akbary, Dragos Ghioca e Qiang Wang. "On constructing permutations of finite fields". Em: **Finite Fields Appl.** 17.1 (2011), pp. 51–67. ISSN: 1071-5797.

[2] M. Alawida, A. Samsudin, N. Alajarmeh, J. S. Teh, M. Ahmad e W. H. Alshoura. "A Novel Hash Function Based on a Chaotic Sponge and DNA Sequence". Em: **IEEE Access** 9 (2021), pp. 17882–17897.

[3] A. Alfrhan, T. Moulahi e A. Alabdulatif. "Comparative study on hash functions for lightweight blockchain in Internet of Things (IoT)". Em: **Blockchain: Research and Applications** 2.4 (2021), p. 100036. ISSN: 2096-7209.

[4] R. AlTawy, Raghvendra Rohit, Morgan He, Kalikinkar Mandal, Gangqiang Yang e Guang Gong. "Towards a Cryptographic Minimal Design: The sLiSCP Family of Permutations". Em: **IEEE Trans. Comput.** 67.9 (2018), pp. 1341–1358.

[5] NRDP Astuti, I Arfiani e E Aribowo. "Analysis of the security level of modified CBC algorithm cryptography using avalanche effect". Em: **IOP Conference Series: Materials Science and Engineering**. Vol. 674. 1. IOP Publishing. 2019, p. 012056.

[6] G. Bertoni, J. Daemen, M. Peeters e G. Van Assche. "Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications". Em: **Selected Areas in Cryptography**. Springer Berlin Heidelberg, 2012, pp. 320–337.

[7] G. Bertoni, J. Daemen, M. Peeters e G. Van Assche. "Sponge functions". Em: **ECRYPT hash workshop**. Vol. 2007. 9. 2007.

[8] J. R. de Oliveira Neto, J. B. Lima e D. Panario. "The Design of a Novel Multiple-Parameter Fractional Number-Theoretic Transform and Its Application to Image Encryption". Em: **IEEE Trans. Circuits Syst. Video Technol.** 30.8 (2020), pp. 2489–2502.

[9] C. Dobraunig, M. Eichlseder, F. Mendel e M. Schläffer. "Ascon v1.2: Lightweight Authenticated Encryption and Hashing". Em: **Journal of Cryptology** 34.3 (jun. de 2021).

[10] C. Douligeris e D. N. Serpanos. "Appendix A: Cryptography Primer: Introduction to Cryptographic Principles and Algorithms". Em: **Network Security: Current Status and Future Directions**. 2007, pp. 459–479.

[11] Maria E., Vladimir V. e Armando F. **Python implementation of Ascon**. https://github.com/meichlseder/pyascon. 2023.

[12] V. A. Gilles. **XKCP: eXtended Keccak Code Package**. https://github.com/XKCP/XKCP. 2024.

[13] M. A. Jimale, M. R. Z'aba, M. L. B. M. Kiah, M. Y. I. Idris, N. Jamil, M. S. Mohamad e M. S. Rohmad. "Parallel Sponge-Based Authenticated Encryption With Side-Channel Protection and Adversary-Invisible Nonces". Em: **IEEE Access** 10 (2022), pp. 50819–50838.

[14] J. B. Lima, M. Barone e R. M. Campello de Souza. "Cosine transforms over fields of characteristic 2". Em: **Finite Fields Appl.** 37 (2016), pp. 265–284. ISSN: 1071-5797.

[15] J. B. Lima, Edmar S. da Silva e R.M. Campello de Souza. "Cosine transforms over fields of characteristic 2: fast computation and application to image encryption". Em: **Signal Process. Image Commun.** 54 (2017), pp. 130–139.

[16] V.S. Lima, F. Madeiro e J. B. Lima. "Encryption of 3D medical images based on a novel multiparameter cosine number transform". Em: **Comput. Biol. Med.** 121 (2020), p. 103772.

[17] A. Maetouq e S. M. Daud. "HMNT: Hash Function Based on New Mersenne Number Transform". Em: **IEEE Access** 8 (2020), pp. 80395–80407.

[18] N. Nabeel, M. H. Habaebi e M. D. R. Islam. "Security Analysis of LNMNT-LightWeight Crypto Hash Function for IoT". Em: **IEEE Access** 9 (2021), pp. 165754–165765.

[19] T. Niu, K. Li, L. Qu e W. Qiang. "New constructions of involutions over finite fields". Em: **Cryptogr. Commun.** 12 (mar. de 2020), pp. 165–185.

[20] A. Pedrouzo-Ulloa, J. R. Troncoso-Pastoriza e F. Pérez-González. "Number Theoretic Transforms for Secure Signal Processing". Em: **IEEE Trans. Inf. Forensics Secur.** 12.5 (mai. de 2017), pp. 1125–1140.

[21] J. M. Pollard. "The fast Fourier transform in a finite field". Em: **Math. Comput.** 25.114 (abr. de 1971), pp. 365–374.

[22] Joost R. **Readable-crypto**. https://github.com/joostrijneveld/readable-crypto. 2014.

[23] I Reed e Treiu-Kien Truong. "The use of finite fields to compute convolutions". Em: **IEEE Trans. Inf. Theory** 21.2 (1975), pp. 208–213.

[24] Ayoub S. **QuarkPython**. https://github.com/ayoubSoussi/QuarkPython. 2020.

[25] W. Stallings. **Pearson etext cryptography and network security**. 8ª ed. Pearson, 2019.

[26] H. C. A. Van Tilborg e S. Jajodia. **Encyclopedia of Cryptography and Security**. 2ª ed. Germany: Springer, 2011. ISBN: 978-1-44195905-8.

[27] S. Windarta, S. Suryadi, K. Ramli, A. A. Lestari, W. Wildan, B. Pranggono e R. W. Wardhani. "Two New Lightweight Cryptographic Hash Functions Based on Saturnin and Beetle for the Internet of Things". Em: **IEEE Access** 11 (2023), pp. 84074–84090.

[28] A. M. Youssef, S. E. Tavares e H. M. Heys. "A new class of substitution-permutation networks". Em: **Workshop on Selected Areas in Cryptography, SAC**. Vol. 96. 1996, pp. 132–147.

