Análise de Metodologias de Implementação e Desempenho em FPGA dos Algoritmos Criptográficos Leves Simon e Speck

Claudio Roberto Costa[1], Fábio Dacêncio Pereira[2], Edward David Moreno[3], Fernanda Mayumi Ohnuma Tachibana[4]

Centro Universitário Eurípides de Marília (UNIVEM) - Computing and Information Systems Research Lab (COMPSI) - Marília-SP, Brasil[1,2,4], Universidade Federal de Sergipe (UFS) - Departamento de Computação (DCOMP) - Aracaju-SE, Brasil[3]

[1]claudiocostaa84@gmail.com, [2]prof.fabiopereira@gmail.com, [3]edwdavid@gmail.com, [4]mayumi@univem.edu.br

Abstract. *This work consists of research, implementation and comparison of two algorithms of lightweight block ciphers Simon and Speck regarding the area and performance when projected on FPGA. Were adopted different architectures for these algorithms, in a first implementation is considered the smallest area, the second has a higher frequency and the third has a lower frequency. At the end is generated a relation throughput/area to assess the best cost benefit among architectures.*

Resumo. *Este trabalho consiste na pesquisa, implementação e a comparação de dois algoritmos de criptografia de cifras de blocos leves Simon e Speck em relação a área e desempenho quando projetados em FPGA. Foram adotadas diferentes arquiteturas para estes algoritmos, sendo que em uma primeira implementação é considerada a menor área, segunda possui maior frequência e a terceira possui menor frequência. Ao final é gerada uma relação vazão/área para avaliar o melhor custo benefício entre as arquiteturas.*

1. Introdução

A quantidade de informações que estão em transmissão ou armazenadas em sistemas computacionais é significantemente crescente. Técnicas de segurança como a criptografia, assegura alguns serviços de proteção sobre essas informações, como: confidencialidade, integridade e autenticidade. [Menezes, Oorschot and Scott 1996]

No entanto existem vários algoritmos de criptografia, com graus de complexidade diferente e exigências distintas relacionado ao esforço computacional. Sendo assim, como adotar o método mais adequado para proteger as informações?

Para isso é usual separar as informações de dados, onde as informações são definidas como objetos, contendo valor ou significado para que sejam protegidas, assim, os demais objetos não necessitam de um grau de proteção por meio da criptografia.

Atualmente as informações são classificadas conforme os critérios de cada organização ou pessoa. O tipo de serviço de proteção mais utilizado para a segurança desses dados é a confidencialidade, resultando em um esquema de classificação por níveis críticos de proteção sendo elas: informação confidencial, restrita, uso interno e pública. [ISSO/IEC 27001 2005]

No esquema de classificação acima, apenas uma parte do conteúdo digital são classificados como informações e são protegidas por algoritmos de criptografia. Contudo, basta que no processo de identificação dos dados e informações ocorram erros, equívocos ou descuido para que estas sejam categorizadas erroneamente e consequentemente não recebam a proteção adequada por meio da criptografia.

Ao longo dos anos surgiram vários algoritmos criptográficos de cifras de blocos leves para atender a necessidade de proteção dessa grande massa de dados em transmissão ou armazenados em sistemas computacionais sem degradar o desempenho computacional, tais como: Simon [Beaulieu et al. 2013], Speck [Beaulieu et al. 2013], Tea [Wheeler and Needham 1995], Twine [Sukaki et al. 2011], LBlock [Wu and Zhang 2011], LED [Guo, Peyrin, Poschmann and Robshaw 2011], entre outros.

Neste artigo apresentamos dois algoritmos de cifras de blocos leves, Simon e Speck [Beaulieu et al. 2013], ambos se adéquam a uma grande variedade de aplicações como: *Internet of Things* – IoT (Internet das coisas), computação pervasiva e/ou ubíqua, sistemas embarcados ou em cenários onde o hardware disponível é limitado em relação a capacidade de processamento, memória e consumo de energia.

Os algoritmos Simon e Speck são explorados neste trabalho abordando diferentes metodologias de implementação em circuitos programáveis FPGA com o objetivo de analisar aspectos que circundam a relação entre desempenho e área.

2. Contextualização (Lightweight Block Ciphers)

No ano 1977, o NBS (*National Bureau of Standards*) estabeleceu o primeiro algoritmo criptográfico padrão com base em cifras simétricas de blocos, o *Data Encryption Standard* – DES. Esse algoritmo possui características de cifra em bloco, baseados na função de Feistel e em redes de substituição-permutação, atualmente base de algoritmos criptográficos leves.

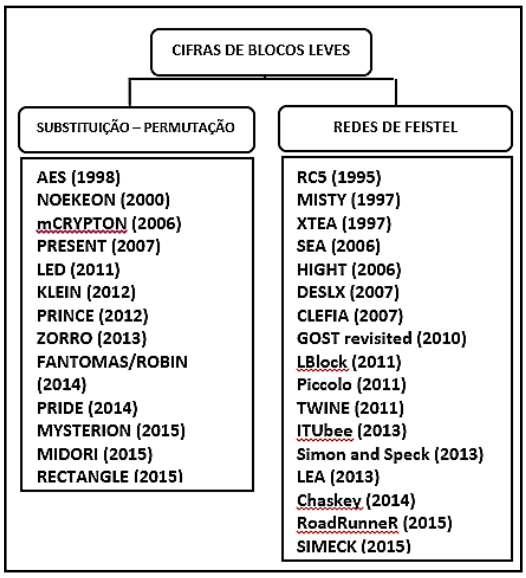

Figura 1. Classificação por estrutura dos algoritmos considerados como cifras leves

A cifra de bloco é uma função que mapeia blocos de n-bits de texto não cifrado para blocos de n-bits de texto cifrado, sendo n o comprimento do bloco. A função é parametrizada por k-bits da chave K contendo o mesmo tamanho do bloco, assim evitando a expansão dos dados. [Handbook of Applied Cryptography, 2001]

As cifras de blocos leves operam em blocos de tamanhos fixos, onde a variação depende do algoritmo. Geralmente de tamanho 64 e 128 bits, elas podem ser classificadas como: "Redes de Substituição-Permutação" e "Redes de Feistel", o segundo é utilizado nos algoritmos Simon e Speck, como pode ser observado na figura 1.

Com pode ser observado na figura 1, recentemente, novos algoritmos de cifra leves foram criados, como é o caso do Simon e Speck, e também outros como: Simeck, RoadRunnner, Rectangle, Pride, Chaskey. Isso realça a demanda deste tipo de proteção no cenário atual dos sistemas computacionais.

2.1. Cifras Feistel (base Simon e Speck)

A maioria dos algoritmos de cifras de blocos possuem uma estrutura descrita em 1973 por Horst Feistel, desenvolvedor de cifras da IBM [Feistel 1973]. Os passos do bloco de Feistel é descrito abaixo:

- As entradas dos algoritmos são divididas em duas partes iguais Left e Right;
- Para gerar o texto cifrado, as duas partes sofrem processamento de *n* iterações;
- Cada iteração recebe Left-1 e Right-1, elas são derivadas da iteração anterior, além disso, uma sub-chave Ki é derivada da chave inicial K;
- Através de um algoritmo gerador de chaves, sub-chaves distintas são geradas;
- Todas as iterações têm a mesma estrutura e são chamadas de *round* (rodadas);
- Para cada round, aplica-se uma função denominada *round function*, esta realiza operações de XOR bit a bit, como demonstra a figura 2:

3. Trabalhos Correlatos

Esta seção tem como objetivo abordar artigos encontrados no estado da arte, os quais tratam sobre implementações dos algoritmos de cifra leve Simon e Speck, ambos com ênfase na plataforma FPGA. São apresentados três trabalhos, onde todos eles apresentam resultados de suas implementações, onde são comparados o desempenho dos algoritmos por meio do impacto de área e vazão de cada um.

3.1 "Simon and Speck Block Ciphers for the Internet of Things"

No trabalho de BEAULIEU et al. (2015) foram realizadas implementações dos algoritmos Simon e Speck em plataformas como circuitos integrados de aplicação especifica (ASICs), FPGAs e microcontroladores, com a finalidade de mensurar o desempenho destes em diversos cenários para aplicação deste tipo de segurança direcionado à Internet das coisas (IoT).

A implementação feita em ASICs do algoritmo Simon utiliza menor área quando comparadas com outras cifras de bloco leve contendo o mesmo tamanho do bloco e chave. A lógica necessária para calcular um bit por rodada é pequena, quando a escala da

implementação é igual ou superior a dois bits, podem ser atualizadas em apenas um ciclo de relógio com impacto mínimo sobre a área.

Por conseguinte, a implementação do algoritmo Speck nesta mesma plataforma não traz tanta diferença relacionado ao desempenho de Simon. Uma das principais diferenças entre estes é que Speck substitui as portas lógicas AND por somador completo e adiciona multiplexadores para atualização dos estados.

Segundo BEAULIEU et al. O artigo coloca em destaque os resultados obtidos nas implementações através de simulações com linguagem VHDL, relacionando-os a outros algoritmos encontrados no estado da arte. Com a velocidade do relógio à 100 kHz, foram comparados com outros algoritmos que correspondem com o mesmo tamanho de bloco, tamanho de chave, velocidade e testados em plataformas ASICs, os quais podem ser encontrados no estado da arte no cenário de cifras de bloco leve.

Nas implementações seguintes, realizadas nas plataformas FPGAs na linguagem VHDL e microcontroladores na linguagem Assembly e C, foram comparadas novamente com outros algoritmos criptográficos contidos no estado da arte, onde ao observar os resultados deixa evidente sua eficiência em relação aos outros. Apesar dos algoritmos serem implementados em diversas plataformas, tanto em hardware quanto em software, Simon e Speck não apresentaram dificuldades em adaptação, diferentemente, como por exemplo, o algoritmo de cifra leve PRESENT.

Entre as implementações realizadas em três plataformas diferentes no trabalho de BEAULIEU et al., destacamos as implementações em FPGA relacionadas na tabela 1 com o intuito de parametrizar os resultados com o nosso trabalho.

Tabela 1.Comparações de performance entre algoritmos de diferentes autores em FPGA

Tamanho	Álgoritmo	Área (Slice LUTs)	Vazão (Mbits/s)
64/128	**Simon**	**24**	**9.6**
	Simon	**138**	**512**
	Speck	**34**	**7.0**
	Speck	**153**	**416**
128/128	**Simon**	**28**	**5.7**
	Simon	36	3.6
	Simon (DPA)	87	3.0
	Simon	**197**	**567**
	Simon	**375**	**867**
	Speck	**36**	**5.0**
	Speck	**232**	**455**
	Speck	**401**	**920**

Logo os autores, concluem que os algoritmos Simon e Speck possuem vantagens não apenas comparado ao desempenho em plataformas diferentes, mas também, pela sua flexibilidade. Possuindo versatilidade, além de sua simplicidade, faz com que esses dois algoritmos sejam ideais para uso em redes heterogêneas, onde permitem ser otimizados para uma aplicação especifica.

3.2. "Simple SIMON FPGA implementations of the Simon 64-128 block cipher"

No trabalho de WETZELS J. e BOKSLAG W. (2015) foram realizadas implementações do algoritmo de criptografia de cifra leve Simon 64/128 na plataforma Xilinx Spartan-6 FPGA series com diferentes tipos de arquitetura em circuitos combinacionais, são elas:

- Round Function: A arquitetura apresentada é um circuito combinacional que expressa a rodada de função como um componente independente;
- Arquitetura iterativa: Trata-se de uma arquitetura iterativa, onde o circuito combinacional da rodada de função projetada anteriormente, é acoplada a um único registrador e multiplexador, conectados a um sinal no qual alimenta a chave da rodada. Devido a interligação desses componentes, a execução da encriptação completa sobre o texto plano torna-se possível, pois todas as rodadas necessárias são calculadas de forma iterativa;
- Arquitetura de desdobramento de laço: É conhecido como uma técnica onde há a redução de instruções dentro dos laços para otimizar a velocidade de execução do programa. Este tipo de arquitetura dá a liberdade para escolher entre ser parcial ou completa, onde deve-se considerar a vazão e a área utilizada. Afim de alcançar o rendimento máximo que o FPGA pode oferecer, os autores optaram por utilizar a arquitetura completa;
- Arquitetura Inner-round pipelining: Possui como base a arquitetura iterativa, a sua rodada de função é dividida em subfunções independentes, onde registradores são utilizados para armazenar resultados parciais entre essas subfunções;
- Arquitetura Outer-round pipelining: Possui como base a arquitetura de desdobramento de laço, onde o designer determina a quantidade de rodadas que vão utilizar a técnica de desdobramento de laço sem ultrapassar o limite de área e circuitos combinacionais que a arquitetura possui;
- Arquitetura Mixed Inner-Outer-round pipelining: Possui como base as arquiteturas Outer-round e Inner-round pipelining, onde o designer pode escolher entre usar parcialmente ou totalmente a arquitetura outer-round pipelining, substituindo a rodada da função por outra otimizada encontrada na arquitetura inner-round pipelining.

O artigo expõe a tabela 2, resultados das implementações obtidas através de simulações no FPGA, com linguagem VHDL, relacionando área e vazão consumida por cada algoritmo.

Tabela 2. Comparações de área e vazão entre algoritmos em FPGA

Arquitetura	Vazão (Mbits/s)	Área (Slice LUTs)	Área/Vazão Slice LUTs/(Mbits/s)
Round Function	24	0	32
Iterativa	148	201	353
Iterativa	105	202	329
Iterativa	113	199	328
Desdobramento de laço (K = 44)	2952	0	8096
Outer-round pipelining (K = 44)	1149	2752	4096
Mixed Inner-Outer-round pipelining (K_i = 2, K_0 = 44)	2161	5568	6459

Em conclusão os autores acentuam que devido ao prazo de projeto, não foi possível a validação correta das arquiteturas Inner-round pipelining, Mixed Pipelining, assim considera-se provisoriamente o resultado exposto no artigo.

Nenhuma arquitetura proposta foi otimizada relacionado ao desempenho e confiabilidade como foi realizado em trabalhos correlatos citados no artigo, logo, foi proposto para trabalhos futuros pelos autores.

3.3. "SpecTre: A Tiny Side-Channel Resistant Speck Core for FPGAs"

No trabalho de CHEN et al.(2015) Foram realizadas duas implementações do algoritmo de criptografia de cifra leve Speck 128/128 na plataforma Xilinx Spartan-3 FPGA series, utilizando a linguagem de descrição de hardware Verilog para aplicação deste tipo de criptografia no cenário de Internet das coisas (IoT).

As duas implementações baseiam-se em trabalhos de criptografias onde os registradores são implementados na forma de serialização de bits na transmissão de dados para outros componentes que compõe arquitetura, sendo apenas a chave disposta paralelamente durante a execução das rodadas.

Porém, a diferença entre os dois algoritmos é que a segunda implementação, afim de aumentar sua confiabilidade, acrescentou a criptografia limiar. Esse criptosistema aplica os conceitos de partilha XOR-secreto baseado em computação multipartidária comprovadamente seguro contra-ataques de canal lateral. Foram feitos análise de consumo de energia e a resistência à ataques de canal lateral.

Os resultados das implementações na plataforma FPGA são relacionados na tabela 3, associados a resultados de desempenho de outros algoritmos encontrados no estado da arte, onde a implementação dos autores encontra-se em destaque.

Tabela 3. Comparações de área e vazão entre algoritmos encontrados no estado da arte em FPGA

Cifra	Área (Slice LUTs)	Vazão (Mbits/s)	Plataforma
Criptografia limiar;			
TI-Speck128/128	**99**	**9.68**	**Xc3s50**
TI-Simon128/128	87	3.0	Xc3s50
Cifras de bloco desprotegido;			
Speck128/128	**43**	**10.05**	**Xc3s50**
Simon128/128	36	3.6	Xc3s50

4. Arquiteturas Propostas Para Simon e Speck

Nesta seção é discutido os tipos de arquiteturas e suas especificações utilizadas e o que é esperado nas criptografias leves Simon e Speck. As arquiteturas foram projetadas com *Field Programmable Gate Array* (FPGA), particularmente na família Xilinx Artix7 - XC7A100T, Xilinx ISE 14.2, utilizando a linguagem de descrição VHDL. As arquiteturas implementadas são categorizadas em: *Sync Single Rounds*, *Sync Full Rounds* e *Async Full Rounds*.

- *Sync Single Rounds* (SSR): Na arquitetura composta por sinais, a utilização de laços repetitivos não é possível, pois os valores dos sinais só são atualizados ao término de cada iteração (*clock*), resultando no aumento significativo de iterações, porém tem-se um ganho em relação à área utilizada dentro do circuito e a frequência de operação;

- *Sync Full Rounds* (SFR): Na arquitetura composta por variáveis, os valores dos cálculos necessários para executar o algoritmo são atualizados instantaneamente, isso possibilita a utilização de laços repetitivos (*for*), fazendo com que apenas uma única iteração do circuito (*clock*) seja realizado na geração das chaves e nas operações de cifrar e decifrar. No entanto, esse tipo de metodologia consome uma maior área do circuito digital;
- *Async Full Rounds* (AFR): A arquitetura é composta de circuitos combinacionais onde ocorre a associação de portas lógicas e suas operações podem ser especificadas por meio de um conjunto de equações Booleanas, esta é utilizada para realizar as operações do *round function* e da geração das chaves. Esta arquitetura também contém uma memória onde são armazenadas as chaves geradas e os resultados das operações realizadas por cada round realimentando os rounds seguintes. O tempo de iteração de cara round está associado diretamente ao tempo de atraso de propagação dos valores entre as portas lógicas, das entradas até a saída.

Para a avaliação da vazão de cada arquitetura em ambos os algoritmos, é realizado o cálculo a partir da divisão do tempo base (1 segundo), pelo período de execução que é o tempo de propagação da lógica do circuito implementado, multiplicado pela quantidade iterações necessárias para cada implementação e por fim multiplicado pelo tamanho da palavra (32 bits). Esse cálculo é representado na equação 1.

$$Vazão = \frac{1}{Período * Qtde\ Iterações} * 32 \quad (1)$$

$$VA = \frac{Área}{vazão} \quad (2)$$

Obtido a vazão, uma relação entre vazão e área (equação 2) é realizada para comparar o desempenho entre os algoritmos. A variável área é situada pela a quantidade de SliceLUTs consumida por cada implementação.

Nas seções 5 e 6 são apresentados e especificados os algoritmos Simon e Speck implementando utilizando as arquiteturas propostas, assim como a análise de desempenho de área e vazão.

5. Algoritmo SIMON

O bloco de cifra do algoritmo Simon é composto por uma palavra de *n* bits (portanto um bloco 2*n*-bits), este é denotado como Simon2n, onde n pode ser 16, 24, 32, 48 ou 64. Simon2*n* possui m palavras chaves (*mn*-bit) que se refere a como Simon2*n*/*mn*. Por exemplo, Simon64/128 refere-se à versão de Simon em blocos de texto simples de 64 bits, usando uma chave de 128 bits. [Beaulieu et al. 2013]

Esse algoritmo utiliza as operações XOR (⊕) bit a bit, AND (&) bit a bit e descolamento circular de bits (S^n) nas suas rodadas de encriptação (3) e decriptação (4), onde *l* e *r* são as duas partes que compõem o texto plano (esquerda e direita respectivamente) e *k* é a chave da rodada.

A função de encriptação pode ser expressada como:

$$R(l, r, k) = ((S1(l) \,\&\, S8(l)) \oplus S2(l) \oplus r \oplus k, l) \quad (3)$$

E a função de decriptação:

$$R^{-1}(l, r, k) = (r, (S1(r) \,\&\, S8(r)) \oplus S2(r) \oplus l \oplus k) \quad (4)$$

Para as operações de cifração e decifração são necessárias T (*T* depende do tamanho de *n*) rodadas, e consequentemente a geração das chaves para cada rodada a partir da expansão da chave inicial. Exceto a chave da rodada, todas as rodadas de Simon são exatamente as mesmas, e as operações são perfeitamente simétricas em relação ao mapa de deslocamento circular em n-bits palavras.

Portanto, neste artigo é apresentado a implementação deste algoritmo em VHDL de Simon32/64, com T = 32 (quantidade de rodadas), em três vertentes, utilizando as arquiteturas SFR, SSR e AFR apresentadas na seção 4, sucedendo a comparação destes sobre a área e desempenho.

Na tabela 5 tem-se informações sobre desempenho, área e a quantidade de iteração das arquiteturas descritas em VHDL e implementadas em FPGA, destacando-se frequência máxima do circuito e seu consumo de área medido em SliceLUTs.

Tabela 5. Estatística: Área e Desempenho de Simon

SIMON	Desempenho		Área	N° de Iterações
	Período (ns)	Frequência (MHz)	Slice LUTs	Cifração
SFR	41,499	24,096	1504	1
SSR	2,719	367,796	973	65
AFR	26,087	38,333	2465	1

Como pode ser observado na tabela 6, a área consumida pela arquitetura SFR é 35,30% maior que SSR. Já na arquitetura SSR, a frequência é 15 vezes maior que SFR. A arquitetura AFR consome uma área 39% maior que SFR e 60% maior que SSR; sua frequência é 1,5 menor que a arquitetura SFR e 9 vezes maior que SSR. Para gerar uma comparação entre as arquiteturas, é proposta a análise da relação área/vazão, apresentada na tabela 6.

Tabela 6. Relação Área / Vazão

SIMON	Vazão	Área/Vazão
	Mbits/s	Slice LUTs/(Mbits/s)
SFR	771	1,95
SSR	175	5,56
AFR	1.226	2,01

O resultado exibido na tabela 6, mostra que a relação da arquitetura SFR possui área/vazão 275% melhor que SSR. A arquitetura AFR possui uma relação área/vazão 267% melhor que SSR e 3% inferior a SFR.

6. Algoritmo SPECK

O algoritmo de Speck foi otimizado para desempenho em implementações em software. Representado usualmente pelo tamanho do bloco (2*n*) e o tamanho da chave (*mn*), onde *n* é o tamanho da palavra. A notação para as diferentes variantes Speck é análoga a utilizada por Simon. Por exemplo, quando Speck24/48 refere-se à versão de Speck com blocos de texto simples de tamanho 48 bits e utilizando uma chave de 48 bits. [Beaulieu et al. 2013]

Esse algoritmo utiliza em suas operações de criptografia (6) e geração de chaves as operações de XOR (+) bit a bit, adição modular de 2^n (+) e descolamento circular de

bits (S^n) durante suas rodadas. Na sua operação de decriptografia (7) ao invés de adição modular é utilizado a subtração modular. Cada rodada de Speck depende do mapa (5):

$$R_k : GF(2)^n \times GF(2)^n \rightarrow GF(2)^n \times GF(2)^n \quad (5)$$

Definido por:

$$R_k(x, y) = ((S^{-\alpha}x + y) \oplus k, S^{\beta}y \oplus (S^{-\alpha}x + y) \oplus k) \quad (6)$$

A função inversa para a decriptografia é:

$$R_k^{-1}(x, y) = (S^{\alpha}((x \oplus k) - S^{-\beta}(x \oplus y)), S^{-\beta}(x \oplus y) \quad (7)$$

Para os blocos de tamanho 32, os valores de rotação são α=7 eβ=2, para os outros tamanhos de blocos α=8 e β= 3.

Neste artigo é apresentado a implementação em VHDL de Speck32/64, com T = 22 (quantidade de rodadas), utilizando o mesmo cenário em que ocorreu a implementação do algoritmo Simon, foram coletados os dados referentes ao número de iterações, a quantidade de SliceLUTs, o período mínimo e a frequência máxima, para gerar uma comparação entre as implementações, é proposta a análise da relação área/vazão. Estas informações foram relacionadas na tabela 8.

Tabela 8. Análise Área e Desempenho de Speck

SPECK	Desempenho		Área	N° de Iterações
	Período (ns)	Frequência (MHz)	SliceLUTs	Cifração
SFR	75,746	13,202	2238	1
SSR	4,012	249,258	752	45
AFR	36,522	27,38	485	1

Como pode ser observado na tabela 8, a área consumida pela arquitetura SFR é 297% maior que SSR e 461% maior que AFR. A frequência da arquitetura SSR é 18 vezes maior que SFR e 9 vezes maior que AFR.

A tabela 9 demostra a relação entre vazão e área, para comparar o desempenho dos algoritmos. Para a área é utilizado a quantidade de SliceLUTs.

Tabela 9.Relação Área/ Vazão

SPECK	Vazão	Área/Vazão
	Mbits/s	Slice LUTs/(Mbits/s)
SFR	422	5,3
SSR	177	4,24
AFR	876	0,55

O resultado exibido na tabela 9, expõe que a arquitetura SFR contém uma relação de área/vazão 25% inferior quando comparado com a arquitetura SSR. A arquitetura AFR é 963% e 770% superior a SFR e SSR respectivamente.

7. Conclusão

Conclui-se que ambos os algoritmos utilizando a arquitetura AFR, possuem uma relação área/vazão melhor quando comparados com as arquiteturas SFR e SSR, com exceção de Simon (SFR).

Ainda é possível comparar os dados estatísticos dos algoritmos Simon e Speck. A implementação do algoritmo Simon utilizando SFR, tem uma relação de área/vazão de 1,95, o qual é 271,79% melhor do que Speck com 5,3. A arquitetura de Speck (SSR), tem relação de área/vazão de 4,24, o qual é 26% melhor que a versão Simon com 5,37. A arquitetura de Speck com AFR tem uma relação de área/vazão de 0,55, o qual é 365% melhor do que Simon com 2,01, como pode ser observado na tabela 10.

Tabela 10. Análise Simon e Speck

Algorit.	Freq.	LUTs	Vazão	Área/Vazão
Simon(SFR)	24,096	1504	771	1,95
Simon(SSR)	367,796	973	175	5,56
Simon(AFR)	38,333	2465	1.226	2,01
Speck(SFR)	13,202	2238	422	5,3
Speck(SSR)	249,258	752	177	4,24
Speck(AFR)	27,38	485	876	0,55

No entanto, dependendo do cenário onde será realizada a aplicação da solução, pode-se escolher entre a melhor implementação conforme os requisitos relacionadas ao desempenho ou área. Por ter uma vazão relativamente alta é possível cifrar dados e informações em sistemas computacionais, assim mesmo que alguma informação seja classificada como dados, ela estará protegida por criptografia. E como próximos passos será gerado consumo de energia dessas arquiteturas na plataforma FPGA.

Referências

Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., and Wingers, L. The Simon and Speck Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404 (2013). https://eprint.iacr.org.

Handbook of Applied Cryptography, 2001 www.cacr.math.uwaterloo.ca/hac/

H. Feistel, "Cryptography and Computer Privacy," Scientific American, v. 228, n. 5, May 73, pp. 15-23.

Wheeler, D., Needham, R. TEA, a Tiny Encryption Algorithm. In Preneel, B. (ed.) FSE 1994, LNCS vol. 1008, pp. 363{366, Springer, Heidelberg (1995).

Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E. TWINE: A Lightweight,Versatile Block Cipher.

Wu, W., Zhang, L. LBlock: A Lightweight Block Cipher. In ACNS 2011, LectureNotes in Compute Science, No. 6715, 327-344. Springer-Verlag (2011).

Guo, J., Peyrin, T., Poschmann, A., Robshaw, M. The LED Block Cipher. In Preneel, B. and Takagi, T. (eds.) CHES 2011, LNCS vol. 6917, pp. 326{341, Springer, Heidelberg (2011).

Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, Handbook of Applied Cryptography, CRC Press, ISBN: 0-8493-8523-7, 1996.

ISO/IEC 27001, Information Security Management System (ISMS), 2005.

Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., and Wingers, L. Simon and Speck: Block Ciphers for the Internet of Things. NIST Lightweight Cryptography Workshop, 20-21 July 2015.

Wetzels J., Wouter B. Simple SIMON FPGA implementations of the Simon 64/128 block cipher. Cryptographic Engineering – Kerckhoffs Institute, 2015.

Chen C., Inci M.S., Taha M., Eisenbarth T. SpecTre A Tiny Side-Channel Resistant. Cryptology ePrint Archive, Report 2015/691 (2015). https://eprint.iacr.org/2015/691.