Avaliação de Desempenho dos Algoritmos Criptográficos Skipjack e RC5 para Redes de Sensores sem Fio

Tiago M. Cavalcante[1,3]*, Fernando P. Garcia[2,3,4], Danielo G. Gomes[1,3], Rossana M. C. Andrade[1,2,3]†

Universidade Federal do Ceará (UFC)
[1]Programa de Pós-Graduação em Engenharia de Teleinformática (PPGETI)
[2]Mestrado e Doutorado em Ciência da Computação (MDCC)
[3]Grupo de Redes de Computadores, Engenharia de Software e Sistemas (GREat)
[4]Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)
Fortaleza,CE, Brasil

{tiagocavalcante,fernandogarcia,dgomes,rossana}@great.ufc.br

Resumo. *O aumento de aplicações em Redes de Sensores sem Fio (RSSF) faz com que a preocupação com a segurança da informação nesse tipo de redes torne-se mais relevante. Levando em consideração que a seleção do algoritmo de criptografia apropriado é essencial para prover a segurança dos dados em RSSF, a nossa proposta é avaliar o desempenho dos algoritmos Skipjack e RC5, dois algoritmos muito utilizados em RSSF, com o objetivo de orientar um projeto de RSSF quanto à escolha do algoritmo criptográfico mais adequado à sua aplicação. Experimentos reais foram realizados na plataforma MicaZ com o auxílio de um osciloscópio digital e os resultados das medições indicam que o algoritmo RC5 requer menos ROM (7%), tempo (8%) e energia (8%) na encriptação. Em contrapartida, o Skipjack requer menos memória RAM (9%), tempo (5%) e energia (5%) na decriptação.*

Abstract*. The increase in the number of applications for wireless sensor networks (WSN) makes more relevant the concern about information security in such networks. Taking in consideration that the selection of the appropriate encryption algorithm is essential for providing data security in WSN, the goal of this paper is to evaluate the performance of RC5 and Skipjack algorithms aiming to direct the design of WSN in the choice of cryptographic algorithm. Experiments were carried out in a real platform MicaZ, with the help of a digital oscilloscope. The measurement results indicate that the RC5 algorithm requires less ROM (7%), time (8%) and energy (8%) for encryption. In contrast, the Skipjack requires less RAM (9%), time (5%) and energy (5%) in the decryption.*

* Bolsista de mestrado da CAPES
† Bolsista de produtividade DT 2 do CNPq

1. Introdução

Os avanços recentes da microeletrônica proporcionaram o desenvolvimento de microsensores que, acoplados a dispositivos com capacidade de processamento e recursos de computação limitados e providos de comunicação sem fio, formam um nó sensor [Akyildiz et al. 2002]. Dentre as inúmeras aplicações das Redes de Sensores Sem Fio (RSSF), destacam-se o monitoramento ambiental, a aquisição de dados na indústria e aplicações de tempo real [Borges Neto et al. 2010].

Se por um lado, o aumento das aplicações de RSSF em diversas áreas enseja a preocupação com a segurança dos dados coletados, por outro, a segurança é um aspecto que vem sendo considerado um desafio em RSSF devido suas limitações de memória, processamento e energia [Kaps, Gaubatz and Sunar 2007]. Além disso, o custo da segurança para o projeto de RSSF ainda se apresenta como uma variável desconhecida [Lee, Kapitanova and Son 2010].

Como os algoritmos de criptografia estão entre os principais elementos utilizados para prover a segurança dos dados em um processo de comunicação, a seleção do algoritmo apropriado é essencial para prover a segurança dos dados em RSSF [Simplício Jr e Barreto 2010]. Portanto, a avaliação de algoritmos de criptografia é de fundamental importância para orientar o projeto de aplicações de RSSF quanto à escolha do algoritmo criptográfico mais adequado à sua aplicação.

Diante desse contexto, nossa proposta é avaliar o desempenho, por meio de experimentos reais, dos algoritmos Skipjack [Nist 1998] e RC5 [Rivest 1997], dois algoritmos muito utilizados em RSSF, com o objetivo de orientar um projeto de RSSF. As métricas consideradas na avaliação de cada algoritmo são a quantidade de memória requerida (ROM e RAM), o tempo de execução e o consumo de energia nas fases de encriptação e decriptação dos dados.

2. Algoritmos de Criptografia

Nesta seção são descritas as principais características dos algoritmos de criptografia avaliados neste trabalho, quais são Skipjack e RC5. A escolha desses algoritmos é justificada pelo fato de que ambos foram desenvolvidos para dispositivos com capacidade limitada de processamento e memória, além de fazer parte da camada de segurança TinySec [Karlof, Sastry and Wagner 2004], utilizada em aplicações de RSSF, sendo que o Skipjack também é a cifra padrão do protocolo de segurança para RSSF MiniSec [Luk et al. 2007].

2.1 Skipjack

O Skipjack é um algoritmo de cifra de bloco desenvolvido na década de 80 pelo U.S *National Security Agency* (NSA), tornando-se público em 1998. Ele foi concebido especialmente para dispositivos com capacidade de processamento limitada, utilizando apenas operações de bit para a criptografia e decriptografia das mensagens. O Skipjack possui as seguintes características principais: tamanho do bloco de 64 bits, tamanho da chave criptográfica de 80 bits e 32 *rounds* [Nist 1998].

Em 2002, foi apresentada a primeira criptoanálise completa dos 32 *rounds* do Skipjack, utilizando *slide attack* [Phan 2002]. Assim, observa-se a vulnerabilidade do

algoritmo Skipjack, sendo a limitação da chave um dos fatores principais que tornam o algoritmo inseguro.

2.2. RC5

O RC5 é um algoritmo de cifra de bloco desenvolvido em 1994 por Ronald Rivest. Ele foi concebido para, entre outros objetivos, requerer pouca quantidade de memória e possuir alto nível de segurança [Rivest 1997]. O RC5 possui as seguintes características principais: tamanho do bloco de 32, 64 ou 128 bits, tamanho da chave criptográfica variável de 0 a 2040 bits (0 a 255 bytes) e número de *rounds* variável de 0 a 255 [Rivest 1997].

O nível de segurança do algoritmo RC5 é estabelecido pela escolha de seus parâmetros. Rivest (1997) definiu a configuração RC5 64/12/16 (tamanho do bloco de 64 bits, chave de 16 bytes e 12 *rounds*) como padrão. Porém, para ser considerado seguro contra ataques de segurança, o número de *rounds* deve ser maior do que 16 [Potlapally et al. 2005]. Assim, a configuração RC5 64/18/16 foi escolhida para a realização dos experimentos neste trabalho.

3. Metodologia

Nesta seção são apresentadas a aplicação desenvolvida e as métricas escolhidas para a avaliação do desempenho dos algoritmos estudados neste trabalho, além da descrição dos experimentos realizados.

3.1. Aplicação Utilizada na Avaliação

Para avaliar o desempenho dos algoritmos Skipjack e RC5 foi necessário o desenvolvimento de uma aplicação[‡] (chamada de Criptotest) utilizando a linguagem NesC para o sistema operacional TinyOS, versão 2.1.

Algoritmo 1 CriptoTest

Entrada: *textoClaro*, o bloco a ser criptografado; *chave*, a chave criptográfica da cifra de bloco;
Saída: *textoCifrado*, o bloco criptografado; Alteração lógica do bit 4 da porta E do microcontrolador.

```
1:  //A cifra de bloco é definida em tempo de compilação
2:  Altere o estado do bit 4 da porta E para 0
3:  se segurança então
4:     inicializaChave(chave)
5:     altere o estado do bit 4 da porta E para 1
6:     textoCifrado ← criptografaBloco(textoClaro, chave)
7:     altere o estado do bit 4 da porta E para 0
8:     textoClaro ← decriptografaBloco(textoCifrado, chave)
9:  fimse
10: escreva(textoClaro, textoCifrado)
```

[‡] Códigos da aplicação disponíveis em: http://www.great.ufc.br/malveira/.

A aplicação desenvolvida para a avaliação está descrita no Algoritmo 1. Esse algoritmo fornece condições para a medição do tempo de execução da operação de encriptação (linhas 5 e 7) pelo processo que será descrito mais adiante. A instrução entre as linhas 5 e 7 do algoritmo determinam a operação a qual seu tempo será medido. Assim, com uma pequena alteração no Algoritmo 1 (posição das linhas 5 e 7) é possível avaliar as operações de inicialização da chave e de decriptação, de forma análoga à encriptação.

Ao criptografar ou decriptografar uma mensagem, a aplicação envia para um microcomputador, via USB, o conteúdo da mensagem, permitindo a depuração durante o desenvolvimento do código. O algoritmo de criptografia a ser utilizado na aplicação pode ser selecionado em tempo de compilação, podendo ser escolhido o RC5, o Skipjack ou nenhum algoritmo de criptografia (sem segurança).

Os algoritmos de criptografia contidos na TinySec estão disponíveis apenas para a versão do TinyOS 1.x. Dessa forma foi necessário portar os algoritmos de criptografia para a versão atual do TinyOS, a 2.X. Esse processo exigiu alteração do código fonte original e mudanças na configuração do TinyOS, por meio da inclusão da interface BlockCipher que é implementada pelos componentes RC5 e Skipjack da TinySec.

3.2. Métricas Utilizadas na Avaliação

Foram definidas as seguintes métricas para a avaliação do desempenho dos algoritmos Skipjack e RC5: quantidade de memória ROM requerida, quantidade de memória RAM requerida, tempo de execução e consumo de energia. As métricas foram escolhidas levando-se em consideração as principais limitações de recursos computacionais em RSSF.

3.3. Descrição dos Experimentos

Todos os experimentos foram realizados em um nó sensor MicaZ, uma plataforma muito utilizada em aplicações de RSSF, sendo bastante popular [Liu and Ning 2008]. Ela foi desenvolvida pela Crossbow Technology e possui como principais características: 4KB de memória RAM, 128KB de memória ROM, conversores analógico-digitais de 10 bits e transceptor de radio frequência CC2420 [Crossbow 2006].

A quantidade de memória RAM e ROM requerida por cada algoritmo foi obtida a partir do próprio processo de compilação do código fonte no TinyOS. O resultado do processo de compilação com a indicação da quantidade de memória requerida é ilustrado na captura de tela da Figura 1.

Figura 1. Ilustração do processo de compilação, destacando a quantidade de memória ROM e RAM requerida pela aplicação.

As medições do tempo de execução foram realizadas com o auxílio de um osciloscópio digital Tektronix TDS 210. O pino E4 do microcontrolador ATMega128L da plataforma MicaZ foi conectado a um canal do osciloscópio. Com a utilização da interface GeneralIO do TinyOS, o nível lógico desse pino foi alterado de 0 para 1 antes e de 1 para 0 após as operações de criptografia, decriptografia e inicialização da chave. O tempo no qual o pino permanece com nível lógico 1 representa o tempo de execução de cada operação que pode ser medido no osciloscópio. O procedimento utilizado para nas medições do tempo de execução é mostrado na Figura 3 e uma ilustração real do uso do osciloscópio para efetuar as medições é mostrada na Figura 4. O consumo de energia foi obtido a partir das seguintes equações:

$$potência = tensãoBateria * correnteOperação \quad (1)$$

$$energiaConsumida = potência * tempoExecução \quad (2)$$

Os valores de tensão da bateria e de corrente de operação foram obtidos no documento de especificação da plataforma MicaZ (*datasheet*), que são 3V e 8mA, respectivamente, e o tempo de execução foi medido com um osciloscópio, conforme descrito anteriormente. Assim, a potência foi calculada, utilizando-se a Equação 1 da seguinte forma*:*

$$potência = 3*8*10^{-3} = 24\ mW.$$

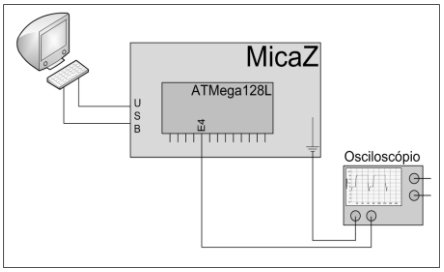

Figura 3. Esquema simplificado de medição do tempo de execução.

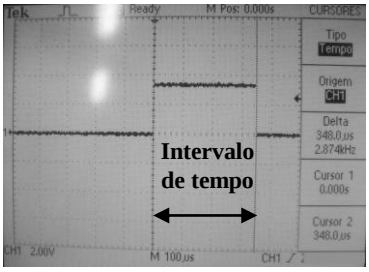

Figura 4. Imagem ilustrativa da medição do tempo de execução no osciloscópio.

A partir da potência calculada, a quantidade de energia requerida por cada algoritmo é obtida de acordo com a Equação 2, sendo que o tempo de execução é o intervalo de tempo medido com o osciloscópio.

$$energiaConsumida = 24*10^{-3} * tempoExecução$$

Com essa estratégia de medição, obtém-se a energia gasta por cada algoritmo de forma aproximada, uma vez que não são utilizados os valores reais de tensão e corrente. Porém, Lee, Kapitanova e Son (2010) verificaram em seus trabalhos que a diferença entre os valores reais e teóricos não é significativa, não afetando a comparação do desempenho dos algoritmos estudados neste trabalho.

4. Resultados

Nesta seção, os resultados obtidos na avaliação de cada métrica são apresentados e discutidos.

4.1. Quantidade de Memória Requerida

As quantidades de memória ROM e RAM requeridas por cada algoritmo são apresentadas nas Figuras 5 e 6, respectivamente. Pode-se observar na Figura 5 que o Skipjack requer mais memória ROM do que o RC5, aumentando a quantidade em quase 20% quando comparado com a aplicação sem segurança. O Skipjack requer cerca de 7% mais memória ROM que o RC5. A quantidade de memória ROM requerida pela aplicação desenvolvida, utilizando o Skipjack, representa 16% do total disponível na MicaZ (128KB), enquanto a quantidade, utilizando o RC5, representa apenas 15%. Por outro lado, com relação à memória RAM, o RC5 requer cerca de 9% a mais que a quantidade requerida pelo Skipjack. A quantidade requerida pelo RC5 representa quase 15% a mais que a quantidade requerida pela aplicação sem segurança. A quantidade de memória RAM requerida pela aplicação desenvolvida, utilizando o RC5, representa 25% do total disponível na MicaZ, enquanto a quantidade, utilizando o Skipjack, representa apenas 23%.

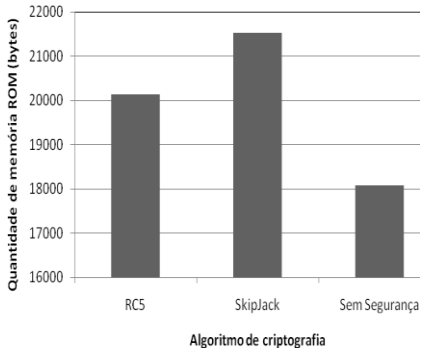

Figura 5. Quantidade de memória ROM requerida.

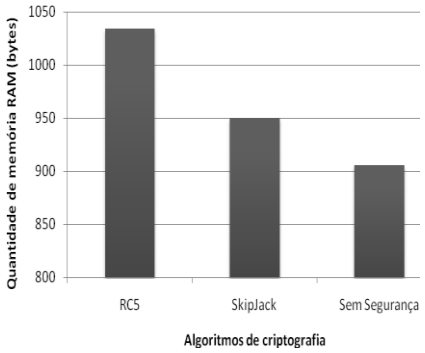

Figura 6. Quantidade de memória RAM requerida

4.2. Tempo de Execução

O tempo de execução de cada algoritmo na criptografia e decriptografia de um bloco de 64 bits é mostrado na Figura 7. Os números representam a média dos valores obtidos em 10 medições. A Tabela 1 apresenta os valores medidos com intervalo de confiança de 95%. Como o número de experimentos é menor do que 30 e o desvio padrão populacional é desconhecido, a tabela t-Student foi utilizada para o cálculo do intervalo [Feldman and Valdez-Flores 1995]. Devido à pequena magnitude do intervalo de confiança, preferimos apresentá-lo apenas na Tabela 1 para facilitar a visualização dos resultados.

Na Figura 7 é possível perceber que a operação de *Key Setup* (inicialização e expansão da chave criptográfica) do RC5 requereu significativamente mais tempo que do Skipjack – cerca de 3,5 vezes mais tempo. Porém essa fase não é decisiva, pois ela é realizada apenas uma vez durante a inicialização da aplicação. A operação de encriptação com o algoritmo RC5 demorou 0,32 ms, sendo cerca de 8% mais rápida do que com o Skipjack. Por outro lado a operação de decriptação demorou 0,34 ms, sendo cerca de 5% mais lenta do que com o Skipjack. Considerando a soma do tempo de execução da operação de encriptação com a decriptação, o RC5 apresentou tempo de execução 1% menor que o Skipjack.

4.3. Consumo de Energia

O consumo de energia de cada algoritmo é mostrado na Figura 8. Conforme mencionado anteriormente, o consumo de energia foi obtido a partir do tempo de execução dos algoritmos e através da aplicação da Equação 2, uma vez que foram utilizados valores teóricos para o cálculo da potência. A média dos valores de energia calculados a partir das 10 medições de tempo de processamento com intervalo de confiança de 95% é mostrada na Tabela 2. Da mesma forma que o tempo de execução, a tabela *t-Student* foi utilizada para o cálculo do intervalo de confiança e preferimos apresentá-lo apenas na Tabela 2 para facilitar a visualização dos resultados.

Na Figura 8 é possível verificar que, na etapa de *Key Setup*, o RC5 consome significativamente mais energia que o Skipjack – cerca de 3,6 vezes. Na operação de encriptação, o RC5 apresentou um consumo de energia de 7,7 µJ, sendo cerca de 8% menor do que o Skipjack. Porém, na operação de decriptação, o RC5 apresentou um consumo de 8,8 µJ, sendo cerca de 5% maior do que o Skipjack. Considerando a soma da energia gasta na operação de encriptação com a decriptação, o RC5 apresentou uma quantidade 1% menor que o Skipjack.

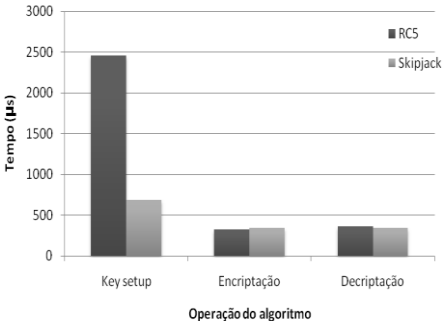

Figura 7. Tempo de execução.

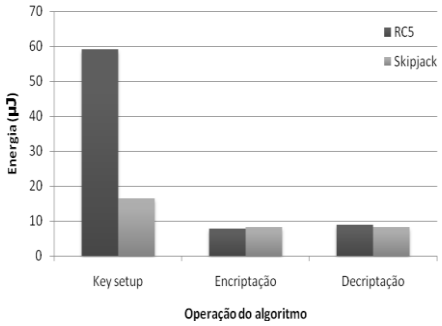

Figura 8. Consumo de energia.

Tabela 1. Tempo de execução em µs com intervalo de confiança de 95%.

Algoritmo	*Key setup (µs)*	*Encriptação (µs)*	*Decriptação (µs)*
Skipjack	683,1 ± 0,8297	347,4 ± 0,5272	348,6 ± 0,5272
RC5	2460,7 ± 0,6207	321,0 ± 0,7108	368,0 ± 0,5026

Tabela 2. Quantidade de energia requerida em µJ com intervalo de confiança de 95%.

Algoritmo	*Key setup (µJ)*	*Encriptação (µJ)*	*Decriptação (µJ)*
Skipjack	16,394 ± 0,0199	8,337 ± 0,0126	8,366 ± 0,0126
RC5	59,056 ± 0,0148	7,704 ± 0,0170	8,832 ± 0,0120

4.4. Discussão dos resultados

Com base nos resultados apresentados neste trabalho, pode-se deduzir o nível de adequabilidade dos algoritmos Skipjack e RC5 às aplicações de RSSF. Primeiramente, a partir do exposto na descrição dos dois algoritmos na Seção 3, pode-se afirmar que o RC5 é mais seguro do que o Skipjack, sob certas configurações de tamanho do bloco,

tamanho da chave e número de *rounds*. Dessa forma, em aplicações que exigem prioritariamente um alto nível de segurança, como aplicações militares e médicas, o algoritmo RC5 é o mais apropriado.

Para aplicações em que a prioridade é a eficiência de memória ROM, o algoritmo mais adequado é o RC5. Por outro lado, o Skipjack é o mais apropriado para aplicações em que a eficiência de memória RAM é mais importante. Para aplicações com restrições de tempo, como aplicações de tempo real em geral, o algoritmo RC5 é a melhor opção, uma vez que ele requer menos tempo de processamento do que o Skipjack. Finalmente, em aplicações como o monitoramento ambiental, nas quais a sobrevivência dos nós é o fator determinante, o RC5 é mais apropriado por requerer um menor consumo de energia.

5. Trabalhos Relacionados

Diversos trabalhos têm apresentado estudos sobre o impacto de algoritmos de criptografia em RSSF [Law, Doumen and Hartel 2004, Guimarães et al. 2005, Simplício Jr 2008, Koo et al. 2008, Jinwala, Patel e Dasgupta 2009, Casado and Tsigas 2009, Lee, Kapitanova and Son 2010], procurando avaliar o desempenho de alguns algoritmos e descobrir qual o custo de prover segurança em tais redes.

Law, Doumen and Hartel (2006) estudaram e avaliaram o desempenho dos algoritmos de criptografia RC5, RC6, Rijndael, MISTY1, KASUMI e Camelia em microcontroladores MSP430F149. Os autores utilizaram apenas duas métricas, quantidade de memória requerida e tempo de processamento, ao contrário desta proposta.

Guimarães et al (2005) avaliaram o desempenho dos algoritmos Skipjack, RC5, RC6, TEA e DES em RSSF, verificando o impacto em CPU e a quantidade de memória requerida por cada algoritmo. No entanto, os autores utilizaram nós sensores Mica2 para a realização dos experimentos, enquanto que na nossa proposta foi utilizada a plataforma MicaZ. Ressalta-se que, de acordo com [Lee, Kapitanova and Son 2010], os resultados para a Mica2 não podem ser usados como referência para a MicaZ.

Simplício Jr (2008) desenvolveu uma nova cifra de bloco para dispositivos com recursos limitados, Curupira, e comparou seu desempenho com os algoritmos Skipjack e AES implementados em plataformas de 8 e 32 bits.

Koo et al (2008) analisaram o desempenho do algoritmo de criptografia HIGHT e compararam com os algoritmos RC5 e Skipjack. Os autores utilizaram nós sensores Mica2 para a realização dos experimentos, contudo o consumo de energia foi avaliado apenas através de simulações. Da mesma forma, Jinwala, Patel e Dasgupta (2009) utilizaram simulações para verificar o impacto da família de algoritmos de criptografia TEA em RSSF e compararam com o desempenho do Skipjack quanto à quantidade de memória requerida, *throughput* e consumo de energia. Diferentemente desses trabalhos, nós realizamos experimentos reais para avaliar os algoritmos Skipjack e RC5.

Casado and Tsigas (2009) avaliaram os algoritmos AES, RC5, Skipjack, Triple-DES, Twofish e XTEA quanto à quantidade de memória requerida, tempo de execução e consumo de energia. Eles realizaram os experimentos na plataforma MSB-430, utilizando o sistema operacional Contiki.

Lee, Kapitanova and Son (2010) realizaram um trabalho denso sobre segurança em RSSF. Os autores avaliaram algoritmos de criptografia, modos de operação e algoritmos de autenticação em plataformas MicaZ e TelosB. Os algoritmos de criptografia analisados foram o Skipjack, RC5, AES e XXTEA. Porém, os autores não apresentaram intervalos de confiança dos dados nem a quantidade de experimentos realizados. Além disso, de acordo com seus resultados, o RC5 consome mais energia que o Skipjack, diferentemente do que nossos resultados indicam. Por fim, a diferença entre os resultados dos autores e os nossos, em relação à quantidade de memória RAM e ROM se deve a possível inclusão do modo de operação ou outras rotinas aos algoritmos originais.

6. Conclusão e Trabalhos Futuros

O trabalho apresentou uma avaliação de desempenho dos algoritmos de criptografia RC5 e Skipjack para RSSF. A quantidade de memória requerida, o tempo de execução e o consumo de energia foram as métricas analisadas para avaliar o desempenho de cada algoritmo.

Os experimentos foram realizados na plataforma MicaZ com o auxílio de um osciloscópio digital. Os resultados indicam que o RC5 requer menos memória ROM, menos tempo de execução e menos energia para a encriptação de dados do que o Skipjack. Em contrapartida, o Skipjack levou vantagem em relação à quantidade requerida de memória RAM, ao tempo de execução e ao consumo de energia durante a decriptação de dados.

Portanto, nosso objetivo foi alcançado, uma vez que a análise dos algoritmos e a descrição do processo detalhado de medição são capazes de orientar o projeto de RSSF quanto à escolha do algoritmo de criptografia mais adequado a sua aplicação, considerando-se os dois algoritmos avaliados. Como trabalhos futuros, pode-se citar a avaliação do impacto e verificação da viabilidade de outros algoritmos de cifra de bloco em RSSF.

Referências

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: A survey. *Computer networks*, 38: 393-422.

Borges Neto, J. B., Gomes, D. G., Ribeiro Neto, P. F., and Andrade, R. M. C. (2008). *In Proceedings of Euro American Conference on Telematics and Information Systems*.

Casado, L., and Tsigas, P. (2009). ContikiSec: a secure network layer for wireless sensor networks under the contiki operating systems. *In Proceedings of Nordsec*, pp. 133-147.

Crossbow Technology. (2006). MPR-MIB user manual. http://www.cs.ucsb.edu/~nchohan/docs/moteManual.pdf, December.

Feldman, R. M., and Valdez-Flores, C. (1995). Applied probability and stochastic processes. New York, Springer.

Guimarães, G., Souto, E., Kelner, J., and Sadok, D. (2005). Evaluation of security mechanisms in wireless sensor networks. *In Proceedings of 10th Simpósio Brasileiro*

de em Segurança da Informação e Sistemas Computacionais (SBSEG), pages 152-165.

Jinwala, D., Patel, D., and Dasgupta, K. S. (2009). Investigating and analyzing the light-weight ciphers for wireless sensor networks", *INFOCOMP Journal of Computer Science,* Vol 8, Issue 2, pp 39-50.

Karlof, C., Sastry, N., and Wagner, D. (2004). TinySec: A link layer security architecture for wireless sensor networks. *In Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys),* pages 162-175.

Kaps, J. P., Gaubatz, G., Sunar, B. (2007). Cryptography on a speck of dust. *Computer*, 40: pp. 38-44.

Koo, W. K., Lee, H., Kim, Y. H., and Lee, D. H. (2008). Implementation and analysis of new lightweight cryptographic algorithm suitable for wireless sensor networks. *In Proceedings Of International Conference On Information Security And Assurance* 2008, Busan, Korea, IEEE Computer Society, pages 73-76.

Law, Y. W., Doumen, J., and Hartel, P. (2006). Survey and benchmark of block ciphers for wireless sensor networks. *ACM Transactions on Sensor Networks TOSN*, vol 2, issues 1.

Lee, J., Kapitanova, K., and Son, S. H. (2010). The price of security in wireless sensor networks. *Computer network*, 54: 2967-2978.

Liu, A., and Ning, P. (2008). TinyECC: A cofigurable library for elliptic curve cryptography in wireless sensor networks. *In Proceedings of 7th International Conference on Information Processing in Sensor Networks (IPSN)*, pages 245-256.

Luk, M., Mezzour, G., Perrig, A., Gligor, V. D. (2007). MiniSec: a secure sensor network communication architecture". *In Proceedings of IEEE International Conference on Information Processing in Sensor Networks (IPSN)*.

Potlapally, N. R., Ravi, S., Raghunathan, A., Jha, N K. A study of the energy consumption characteristics of cryptographic algorithms and security protocols, *IEEE TMC 2005*, pp. 128–143.

Nist, National Institute of Standards and Technology. (1998). Skipjack and kea algorithm specifications. http://cryptome.org/jya/skipjack-spec.htm, December.

Phan, R. C. (2002). Cryptanalysis of full Skipjack block cipher. *Electronics Letters*, 38, 2: 69-71.

Rivest, R. L. The RC5 encryption algorithm. (1994). *In Proceedings of the Second International Workshop on Fast Software Encryption (FSE), pages* 86–96.

Simplício Jr, M. A., Barreto, P. S. L. (2010). Algoritmos criptográficos para redes de sensores. *In Proceedings of 10th Simpósio Brasileiro de em Segurança da Informação e Sistemas Computacionais (SBSEG)*, pages 523-530.