Avaliação de Desempenho de Mecanismos de Segurança para Redes de Sensores Sem Fio

Tiago M. Cavalcante[1,3,i], Fernando P. Garcia[2,3,4], Rossana M. C. Andrade[1,2,3,ii]

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)

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

Resumo. *O crescimento do número de aplicações para Redes de Sensores sem Fio (RSSF) bem como a diversificação das mesmas fez com que a segurança da informação nesse tipo de rede se tornasse uma preocupação mais constante. Entretanto, prover segurança em RSSF é um grande desafio devido às severas limitações de recursos de comunicação, processamento, memória e energia dos dispositivos utilizados nessas redes. Além disso, o custo para prover a segurança dos dados em RSSF ainda não foi precisamente determinado. Assim, este trabalho propõe uma avaliação de desempenho de mecanismos de segurança para prover confidencialidade, integridade e autenticação em RSSF. Experimentos reais foram realizados na plataforma MicaZ com o auxílio de um osciloscópio digital e os resultados oferecem um direcionamento quanto à escolha dos mecanismos de segurança mais apropriados para RSSF.*

Abstract. *The number and diversity of applications for Wireless Sensor Networks (WSN) has grown what makes more relevant the concern about information security in such networks. However, providing security in WSN faces challenges because of the severe limitations of processing, communication, power and memory resources of the devices used in these networks. In addition, the cost to provide data security in WSN has not yet been precisely determined. Thus, this paper proposes a performance evaluation of security mechanisms to provide confidentiality, integrity and authentication in WSN. Real experiments, which are performed in the MicaZ platform with the help of a digital oscilloscope, are also presented. The results provide a direction in relation to the choice of the most appropriate security mechanisms for WSN.*

1. Introdução

Como já bastante discutido na literatura [Akyildiz et al. 2002], os avanços recentes da microeletrônica proporcionaram o desenvolvimento de pequenos sensores que, em

[i] Bolsista de mestrado da CAPES

[ii] Bolsista de produtividade DT 2 do CNPq

conjunto com dispositivos com capacidades de processamento e recursos de computação limitadas e providos de comunicação sem fio, formam um nó sensor. E uma coleção desses nós sensores trabalhando cooperativamente forma uma Rede de Sensores Sem Fio (RSSF). Esse tipo de rede possui um grande potencial de aplicações, sendo utilizada geralmente para coletar dados em um determinado ambiente com o objetivo de realizar tarefas de monitoramento e/ou de rastreamento. Nos últimos anos, várias pesquisas têm buscado resolver os problemas relacionados às severas limitações de recursos em RSSF, principalmente as limitações de processamento, armazenamento e energia. Entretanto, o surgimento de novos campos de aplicação, como a computação ubíqua, introduziu novos problemas relacionados a fatores como a alta mobilidade dos nós sensores e requisitos de *quality of service* (QoS) [Borges Neto et al. 2010].

Por outro lado, a segurança da informação em RSSF é um dos principais problemas nesse tipo de rede, sendo o foco de diversos pesquisadores da área nos últimos anos. Assim como os sistemas de comunicação sem fio em geral, as RSSF utilizam o ar como meio de propagação do sinal. Essa característica torna essas redes vulneráveis a diversos tipos de ataques, evidenciando a necessidade do emprego de mecanismos de segurança em RSSF [Cirqueira et al. 2011]. Se por um lado o aumento das aplicações de RSSF, bem como a diversificação das mesmas, 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 [Cavalcante et al. 2011].

Além do mais, para garantir a segurança dos dados em RSSF, as aplicações podem necessitar de serviços básicos de segurança**,** tais como confidencialidade, integridade e autenticação. Como os algoritmos de criptografia e de autenticação estão entre os principais elementos para prover esses serviços, a seleção dos mecanismos apropriados é essencial para prover a segurança dos dados em RSSF [Simplício Jr e Barreto 2010]. Portanto, a avaliação dos mecanismos de segurança possui fundamental importância para orientar a escolha dos mecanismos mais apropriados para RSSF.

Diante desse contexto, nossa proposta é avaliar o desempenho dos algoritmos criptográficos AES [Dworking 2001], Skipjack [Nist 1998], Klein [Gong et al. 2010], RC5 [Rivest 1994], Present [Bogdanov 2007], além dos modos de operação CBC, CFB, OFB, CTR [Dworking 2005] e OCB [Rogaway 2011] e os algoritmos de autenticação CBCMAC [Bellare et al. 2000], CMAC [Dworking 2005], HMAC-MD5, HMAC-SHA1 [Nist 2002], com o objetivo de orientar a escolha dos mecanismos de segurança mais apropriados para RSSF. As métricas consideradas na avaliação de cada algoritmo foram a quantidade de memória requerida e o consumo de energia.

Este trabalho está estruturado da seguinte forma. A Seção 2 apresenta os principais conceitos envolvidos com segurança em RSSF. Em seguida, na Seção 3, são discutidos os principais trabalhos relacionados. A Seção 4 aborda a metodologia aplicada e a Seção 5 apresenta os resultados obtidos. Por fim, a Seção 6 conclui o artigo.

2. Segurança em Redes de Sensores Sem Fio

Assim como os sistemas de comunicação sem fio em geral, as RSSF utilizam o ar como meio de propagação do sinal, tornando as mensagens trocadas na rede susceptíveis à escuta, interceptação ou modificação de seu conteúdo por um adversário. Logo, as

aplicações podem necessitar dos seguintes serviços de segurança: a) confidencialidade – serviço que garante que uma informação será acessada somente por dispositivos autorizados; b) integridade – serviço que fornece condições de identificar se o conteúdo de uma mensagem não foi alterado após o seu envio; e c) autenticação – serviço que fornece condições de garantir que um determinado participante da rede é autêntico.

É bom salientar que existem outros serviços de segurança, mas que ou são muito improváveis ou exigem mecanismos muito dispendiosos para o contexto de RSSF. Um exemplo é o serviço de irretratabilidade, que consiste na capacidade de impedir que um dos participantes da rede, envolvidos em uma comunicação, negue ter participado de toda ou parte da comunicação [Braga et al. 1998]. Esse serviço exige protocolos complexos com grande *overhead* de comunicação e processamento que são inviáveis para contexto de RSSF atual.

Além do mais, os principais mecanismos utilizados para prover o serviço de confidencialidade são os algoritmos de criptografia e para prover autenticação e integridade são os *Message Authentication Codes* (MAC) e as funções de *hash*, respectivamente, sendo que MAC também fornece a integridade dos dados.

2.1.1. Algoritmos de criptografia

Os algoritmos de criptografia são divididos em duas categorias [Simplício e Barreto 2010]: algoritmos de chave simétrica e algoritmos de chave assimétrica. Uma vez que os algoritmos de chave assimétrica apresentam, em geral, uma maior complexidade e consequentemente um baixo desempenho quando comparados com os algoritmos de chave simétrica [Saraogi 2004], esses últimos são mais adequados para prover confidencialidade em RSSF, sendo divididos em cifras de bloco[iii] e cifras de fluxo. A segurança das cifras de fluxo depende fortemente da geração de um fluxo pseudoaleatório de chaves. Dessa forma, elas são pouco recomendadas para aplicações nas quais a repetição de chaves após um curto período de tempo seja uma realidade, como geralmente é o caso das RSSF. Assim, as cifras de bloco são mais atrativas para RSSF [Simplício e Barreto 2010]. Elas dividem a mensagem em blocos de tamanho fixo e realizam a operação de cifragem em cada bloco de forma independente. As cifras avaliadas neste trabalho estão caracterizadas na Tabela 1.

Tabela 1. Características das cifras de bloco.

Cifra de bloco	Tamanho do bloco (bits)	Tamanho da Chave (bits)	Número de rounds
AES	128	128, 196 ou 256	10, 12 ou 14
Skipjack	64	80	32
Klein	64	64, 80 ou 96	12, 16 ou 20
RC5	32, 64 ou 128	0 a 2040	0 a 255
Present	64	80 ou 128	32

O *Advanced Encryption Standard* (AES) é o algoritmo de criptografia de domínio público aprovado pelo NIST [Dworking 2001], sendo amplamente usado em sistemas de segurança. Conforme publicado em [Cryptrec 2011], uma criptoanálise com complexidade ligeiramente menor do que o ataque de força bruta foi apresentada, sendo

[iii] Os termos "algoritmo de criptografia" e "cifra de bloco" serão utilizados como sinônimos no restante deste trabalho.

que essa criptoanálise não é considerada realista, uma vez que exige uma grande quantidade de dados. Este trabalho avaliou a configuração AES-128 (chave de 128 *bits*).

Por outro lado, o Skipjack é um algoritmo de criptografia desenvolvido na década de 80 pela NSA [Nist 1998] e tornado público em 1998. Ele foi concebido especialmente para dispositivos com capacidade de processamento limitada. Em 2002, foi apresentada a primeira criptoanálise completa dos 32 *rounds* do Skipjack [Phan 2002]. Assim, observa-se a vulnerabilidade do algoritmo Skipjack, sendo a limitação da chave um dos fatores principais que reduzem o nível de segurança da cifra.

O Klein é uma cifra de bloco de domínio público voltado para RSSF e seus criadores apresentaram um estudo mostrando a robustez da cifra a diversos tipos de ataques [Gong et al. 2010], revelando seu nível de segurança adequado para aplicações como RSSF. Para obter o maior nível de segurança, avaliamos a configuração Klein-96.

O RC5 é uma cifra de bloco patenteada e concebida para, entre outros objetivos, requerer pouca quantidade de memória com alto nível de segurança [Rivest 1994]. Esse nível é definido pela escolha de seus parâmetros. Rivest (1994) definiu a configuração RC5 64/12/16 (tamanho do bloco de 64 bits, chave de 16 bytes e 12 *rounds*) como padrão. Entretanto, 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.

Finalmente, o algoritmo de criptografia Present é uma cifra de bloco de domínio público projetada como alternativa ao AES para dispositivos com capacidade de processamento limitada, como as RSSF. Os autores da cifra recomendam o uso do Present como somente encriptação para redução da quantidade de memória requerida. Borghoff et al. (2011) apresentaram uma criptoanálise contra até 28 dos 31 rounds do Present. Dessa forma, este trabalho considerou apenas o algoritmo de encriptação e, para obter o maior nível de segurança, avaliamos a configuração Present-128.

2.2. Modos de operação

Os modos de operação são utilizados com as cifras de bloco para suportar a encriptação de mensagens com tamanhos maiores do que o tamanho de bloco padrão da cifra. Eles geralmente utilizam um *initialization vector* (IV), gerado frequentemente de forma pseudoaleatória para garantir segurança semântica, evitando que uma mesma mensagem criptografada seguidas vezes resulte em textos cifrados idênticos.

O modo CBC realiza uma operação *xor* (ou exclusivo) entre o bloco a ser cifrado e o bloco cifrado anteriormente. Assim, cada bloco de texto cifrado depende dos demais cifrados anteriormente. O primeiro bloco a ser criptografado deve ser somado (*xor*) com um IV para que blocos iguais gerem blocos cifrados diferentes. A mensagem cifrada no modo CBC possui tamanho múltiplo do tamanho do bloco padrão da cifra. Assim, caso o último bloco seja menor do que o tamanho padrão da cifra, ele será preenchido.

O modo CFB transforma uma cifra de bloco em um gerador de números pseudoaleatórios. O texto cifrado realimenta a cifra de bloco, sendo esse processo repetido para produzir um fluxo de *bits* pseudoaleatório, de forma semelhante a uma cifra de fluxo. Além disso, o modo CFB requer um parâmetro *s*, tal que *s* indica o número de *bits* da saída do modo. Frequentemente, esse parâmetro é incorporado ao

nome do modo, por exemplo, CFB-8. Neste trabalho, foi escolhida a configuração CFB-128 devido ao tamanho padrão da cifra AES. As operações de encriptação e decriptação nesse modo utilizam apenas o algoritmo de encriptação da cifra de bloco.

Já o modo OFB é muito semelhante ao modo CFB. A principal diferença está na realimentação da saída da função de criptografia em vez do texto cifrado (saída da encriptação somado – *xor* – com o texto claro). De outra forma, o modo CTR utiliza um contador o qual deve ser utilizado como entrada na encriptação e decriptação de cada bloco. Esse contador deve possuir um valor diferente a cada mensagem criptografada. Além do contador, geralmente é utilizado um número chamado de *nonce*, o qual é similar ao IV nos demais modos.

Por outro lado, o modo OCB, além de fazer o papel de um modo de operação na encriptação dos dados, fornece também o serviço de autenticação. Ele recebe como parâmetros um conjunto de blocos de texto claro, uma chave criptográfica (a mesma utilizada pela cifra de bloco adjacente) e um *nonce*, e devolve um conjunto de blocos de texto cifrado e uma *tag* de autenticação. Geralmente, um contador é utilizado como *nonce*, não sendo necessário que ele seja secreto ou imprevisível.

Vale ressaltar que os modos de operação CFB, OFB, CTR e OCB funcionam de forma semelhante a uma cifra de fluxo. Assim, o tamanho da mensagem criptografada será o mesmo da mensagem não criptografada. Por outro lado, o modo CBC pode produzir textos cifrados com tamanho superior ao texto claro correspondente, uma vez que o tamanho da mensagem criptografada será múltiplo do tamanho da mensagem clara. No contexto de RSSF, isso pode resultar em envio de *bytes* desnecessário pela rede e consequente perda de energia. Uma técnica conhecida por *ciphertext stealing* [Nist 2007] pode ser utilizada para evitar esse problema. É bom lembrar ainda que os modos CFB, OFB e CTR utilizam apenas o algoritmo de encriptação tanto na encriptação como também na decriptação dos blocos, ao contrário dos demais modos. Essa característica traz benefícios com relação à quantidade de memória requerida, uma vez que é necessária uma quantidade relativamente menor de código.

Finalmente, vale salientar que a geração dos IVs é uma tarefa crítica na implantação de mecanismos de criptografia, uma vez que a utilização indevida de IVs pode comprometer a segurança do sistema. Os modos CBC e CFB exigem a imprevisibilidade dos IVs, mas permitem a sua repetição, ao contrário dos demais modos. Além disso, os participantes da rede ou devem possuir o IV ou devem possuir informações de como calculá-lo, uma vez que é necessário o uso do mesmo IV na encriptação e na decriptação de uma determinada mensagem.

2.3. *Message Authentication Code* (MAC)

Um MAC é um algoritmo irreversível que recebe como entrada uma chave e uma mensagem a ser autenticada e fornece como saída um código, muitas vezes chamado de *tag,* que possui uma relação unívoca com a entrada. Dessa forma, caso uma mensagem seja alterada após sua transmissão, o receptor poderá verificar a autenticidade da mensagem simplesmente aplicando o MAC e comparando a *tag*, possibilitando, de acordo com o resultado dessa comparação, validar ou rejeitar a mensagem.

O CBCMAC é um algoritmo de autenticação baseado no modo CBC. A mensagem é criptografada no modo CBC utilizando uma cadeia de zeros como IV e o

último bloco criptografado é a *tag* de autenticação. Diferentemente do modo CBC, o CBCMAC devolve apenas um bloco de dados (a *tag*), cujo tamanho é definido pelo tamanho de bloco padrão da cifra adjacente. O algoritmo CBCMAC apresenta vulnerabilidades para autenticar mensagens de tamanho variável [Bellare et al. 2000]. Esta avaliação utilizou a variação do CBCMAC [Bellare et al. 2000] mais atrativa para RSSF, a qual encripta o tamanho da mensagem e soma (*xor*) com o primeiro bloco de texto claro, resolvendo o problema.

De outra forma, o algoritmo CMAC foi desenvolvido para suprir as deficiências do CBCMAC, sendo recomendado pelo NIST. O CMAC utiliza um algoritmo que gera duas sub-chaves K1 e K2 a partir da chave principal, as quais são utilizadas para autenticar mensagens com tamanho seja múltiplo ou não do tamanho de bloco padrão da cifra de bloco adjacente, respectivamente.

Por outro lado, o HMAC é um algoritmo de autenticação baseado em funções de *hash*. Assim, ele utiliza uma função de *hash*, em vez de uma cifra de bloco, e uma chave para autenticar uma mensagem. O tamanho da chave deve ser igual ou maior que *L/2*, em que *L* representa o tamanho da saída da função de *hash* adjacente. O HMAC gera uma *tag* de autenticação cujo tamanho depende da função de *hash* utilizada. Caso o *Message Digest 5* (MD5) seja utilizado, o tamanho da *tag* gerada será 128 *bits*. De outra forma, caso *Secure Hash Algorithm* 1 (SHA1) seja usado, o tamanho da *tag* será 160 *bits*. Esta avaliação considerou o HMAC com essas duas funções de *hash*. Embora tenha sido encontradas vulnerabilidades no MD5 e SHA1, o HMAC geralmente não é afetado por elas [Schneier 2005].

Vale salientar que o tamanho *len* da *tag* de um MAC é definido de tal forma que um adversário possa forjar um texto cifrado válido com probabilidade 2^{-len}. Além disso, caso necessário, o tamanho da *tag* poderá ser truncado, considerando apenas os *n* primeiros *bits* mais significativos. Porém, é recomendado o uso de pelo menos 32 *bits* da *tag* nas aplicações em geral. Considerando o contexto atual de RSSF, no qual os dispositivos possuem largura de banda em torno de 250 kbps, uma *tag* de 32 *bits* – tamanho padrão da maioria dos protocolos de segurança para RSSF [Karlof et al. 2004, Luk et al. 2007] – poderá ser forjada por um adversário em apenas 90 dias de contínuas tentativas. Assim, utilizar uma *tag* maior pode ser uma alternativa para aumentar o nível de segurança em RSSF.

3. Trabalhos Relacionados

Diversos trabalhos têm avaliado o impacto de mecanismos de segurança em RSSF, entre os quais se destacam [Law et al. 2006, Casado e Tsigas 2009, Lee et al. 2010, Cavalcante et al. 2011].

Law et al. (2006) estudaram e avaliaram o desempenho dos algoritmos de criptografia RC5, RC6, Rijndael, MISTY1, KASUMI e Camelia em microcontroladores MSP430F149, ao contrário desta proposta que utilizou a plataforma MicaZ. Por outro lado, Casado e Tsigas (2009) avaliaram as cifras de bloco AES, RC5, Skipjack, Triple-DES, Twofish e XTEA, os modos de operação CBC e OCB, além do algoritmo de autenticação CMAC. As métricas analisadas foram quantidade de memória requerida, tempo de execução e consumo de energia. Eles realizaram os experimentos na

plataforma MSB-430, utilizando o sistema operacional Contiki, ao contrário desta proposta que utiliza a MicaZ e o sistema operacional TinyOS.

Lee et al. (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. Os modos de operação avaliados foram o CBC, CFB, OFB, CTR e OCB. Os algoritmos de autenticação avaliados foram o CBCMAC, XMAC, CMAC e HMAC-MD5. Entretanto, os autores não apresentaram intervalos de confiança dos dados nem a quantidade de experimentos realizados. Além disso, encontramos diferenças significativas com relação aos resultados desses autores. De acordo com os resultados desses autores, o RC5 consome mais energia que o Skipjack, diferentemente do que nossos resultados indicam. Com relação aos modos de operação, o modo CTR apresentou o melhor desempenho dentre os demais modos, ao contrário dos resultados obtidos pelos autores, em que o CFB apresentou um desempenho melhor. Com relação aos algoritmos de autenticação, encontramos diferenças significativas com relação ao consumo de energia do CBCMAC, CMAC e HMAC-MD5.

Finalmente, Cavalcante et al. (2011) avaliaram o desempenho de dois algoritmos criptográficos para RSSF, Skipjack e RC5, e encontraram diferenças significativas com relação aos resultados dos trabalhos anteriores, o que motivou o desenvolvimento deste trabalho para, além de avaliar de forma mais rigorosa e com um maior número de experimentos, investigar outros mecanismos de segurança para RSSF: MACs e modos de operação. Vale ressaltar ainda que este trabalho avaliou uma versão otimizada da cifra Skipjack e utilizou a versão mais estável do TinyOS, obtendo alguns resultados diferentes dos apresentados em Cavalcante et al. (2011), conforme será visto na Seção 5.

4. Metodologia Aplicada

Todas as implementações dos mecanismos de segurança avaliados neste trabalho foram validadas por meio de exaustivos testes com entradas e saídas conhecidas (vetores de testes) disponíveis na literatura. Alguns algoritmos foram portados para a linguagem *nesC* do sistema operacional TinyOS e outros implementados a partir de sua descrição, conforme resumido na Tabela 2. Para avaliar o desempenho dos mecanismos de segurança foi necessário o desenvolvimento de uma aplicação, chamada *CryptoTest*[iv], utilizando a linguagem *nesC* para a versão mais estável do sistema operacional TinyOS (1.x). Ela fornece condições para a medição do tempo de execução das operações dos mecanismos pelo processo que será descrito mais adiante. O mecanismo de segurança a ser utilizado na aplicação pode ser selecionado em tempo de compilação.

Foram definidas as seguintes métricas para a avaliação do desempenho dos mecanismos: quantidade de memória ROM requerida, quantidade de memória RAM requerida e consumo de energia. As métricas foram escolhidas levando-se em consideração as principais limitações de recursos computacionais em RSSF. Todos os experimentos foram realizados em um nó sensor MicaZ, uma plataforma muito utilizada em aplicações de RSSF, sendo bastante popular. Ela foi desenvolvida pela Crossbow Technology e possui como principais características: 4KB de memória RAM, 128KB de memória ROM e transceptor de radio frequência CC2420. A quantidade de

[iv] Códigos da aplicação e dos mecanismos disponíveis em http://www.great.ufc.br/~tiagocavalcante.

memória RAM e ROM requerida por cada algoritmo foi obtida a partir do próprio processo de compilação do código fonte no TinyOS. Já o consumo de energia foi obtido a partir das Equações 1 e 2, sendo que os valores de tensão da bateria e de corrente de operação foram obtidos no documento de especificação da plataforma MicaZ (*datasheet*), quais são 3V e 8mA, respectivamente.

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

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

Tabela 2. Implementação e validação dos mecanismos de segurança.

Mecanismo	Implementação	Validação
AES	*nesc* [Iis 2011]	[Dworking 2001]
Skipjack	*nesc* [Luk et al. 2007]	[Nist 1998]
Klein	*nesc* [Gong et al. 2011]	[Gong et al. 2011]
RC5	*nesc* [Karlof et al. 2004]	[Rivest 1994]
Present	Portado do *c* [Cis lab 2011]	[Bogdanov et al. 2007]
CBC, CFB, OFB, CTR	Implem. de [Dworking 2001]	[Dworking 2001]
OCB	*nesc* [Luk et al. 2007]	[Rogaway 2011]
CBCMAC	Adaptado do *nesc* [Karlof et al. 2004]	[Dworking 2001]
CMAC	Implem. de [Dworking 2005]	[Dworking 2005]
HMAC-MD5, HMAC-SHA1	Portado do *c* [MD5deep 2011]	[Nist 2002]

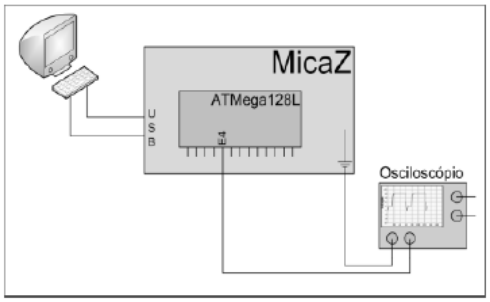

Figura 2. Esquema simplificado de medição do tempo de execução.

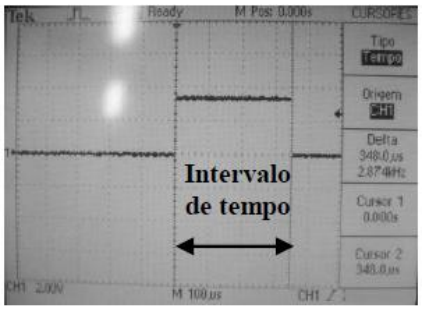

Figura 3. Imagem ilustrativa da medição do tempo de execução no osciloscópio.

Assim, a potência foi calculada, utilizando-se a Equação 1 da seguinte forma: *potência = 3*8*10^{-3} = 24 mW*. A partir da potência calculada, a quantidade de energia requerida por cada algoritmo foi obtida de acordo com a Equação 2. As medições do tempo de execução foram realizadas com o auxílio de um osciloscópio digital *Tektronix TDS 210*[v]. O pino E4 do microcontrolador *ATMega128L* da plataforma MicaZ foi conectado a um canal do osciloscópio. O nível lógico desse pino foi alterado de 0 para 1 antes e de 1 para 0 após cada operação do mecanismo de segurança. 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 nas medições do tempo de execução é mostrado na Figura 2 e uma ilustração real do uso do osciloscópio para efetuar as medições é mostrada na Figura 3. A conexão USB com o PC permitiu a depuração na etapa de implementação. Com essa estratégia de medição, obtém-se a energia gasta por cada algoritmo de forma aproximada, uma vez que não são utilizados

[v] Os experimentos com o osciloscópio foram realizados no LABOMICRO-IFCE.

os valores reais de tensão e corrente. Entretanto, Lee et al. (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.

5. Resultados

5.1. Quantidade de memória requerida

Os resultados relacionados à métrica quantidade de memória requerida são apresentados nas Figuras 4, 5, 6, 7, 8 e 9. Vale ressaltar que os valores levam em consideração também a aplicação desenvolvida para realizar as medições, o que não afeta a comparação, pois o único fator diferente na aplicação consiste apenas no mecanismo de segurança a ser utilizado. Pode-se observar na Figura 4 que o Skipjack é a cifra que requer a menor quantidade de ROM, enquanto o AES requer a maior quantidade com relação às demais cifras. As cifras Klein, RC5 e Present requerem cerca de 7%, 8% e 26% mais ROM que o Skipjack. Por outro lado, como mostra a Figura 5, o RC5 foi o algoritmo que apresentou a menor quantidade requerida de RAM, seguido pelo Skipjack (80% maior) e Klein (2 vezes maior). Já a quantidade de memória RAM requerida pelo AES representou quase 58% do total disponível na MicaZ (4KB).

As Figuras 6 e 7 apresentam a quantidade de memória ROM e RAM requeridas pelos modos de operação utilizando a cifra AES-128, respectivamente Conforme mostrado na Figura 13, os modos de operação CTR, OFB e CFB apresentaram resultados muito próximos com relação à quantidade de memória ROM requerida, com uma ligeira vantagem do modo OFB. Já os modos CBC e o OCB apresentaram uma quantidade requerida em torno de 60% e 47% maiores do que a quantidade requerida pelo modo OFB, respectivamente. Por outro lado, com relação ao consumo de memória RAM, os modos CTR, OFB e CFB também apresentaram resultados bastante próximos. Entretanto, o modo CTR foi o que apresentou a menor quantidade requerida. Já os modos CBC e OCB apresentaram quantidades requeridas de memória RAM em torno de 2,1 e 2,3 vezes maiores do que a quantidade requerida pelo CTR, respectivamente.

De acordo com a Figura 8, o CBCMAC foi o MAC que apresentou menor quantidade de ROM requerida, seguido pelo CMAC e OCB. Vale destacar que o HMAC-SHA1 apresentou uma quantidade significativamente superior aos demais, em torno de 21% do total disponível na MicaZ (128KB). Já com relação memória RAM, o OCB que apresentou a maior quantidade, sendo em torno de 13,7 vezes maior que a quantidade requerida pelo HMAC-MD5, que apresentou a menor quantidade.

5.2. Consumo de energia

O consumo de energia para realizar a inicialização da chave[vi] dos algoritmos criptográficos está mostrado na Figura 10 com intervalo de confiança de 95%. Observa-se que o RC5 apresentou um consumo de energia significativamente maior que as demais cifras na inicialização da chave, embora essa etapa não seja determinante, uma vez que é realizada apenas na inicialização da aplicação. O Skipjack, AES e Klein foram as cifras que apresentaram os menores valores, nessa ordem. O consumo de energia de

[vi] Essa tarefa também é chamada de expansão da chave, escalonamento da chave ou *key setup*.

cada mecanismo de segurança é mostrado nas Figuras 11, 12 e 13. Os gráficos apresentam a média dos valores de energia calculados a partir das 31 medições de tempo de processamento com intervalo de confiança de 95%. Devido à pequena magnitude do intervalo de confiança, alguns intervalos não são visíveis nos gráficos.

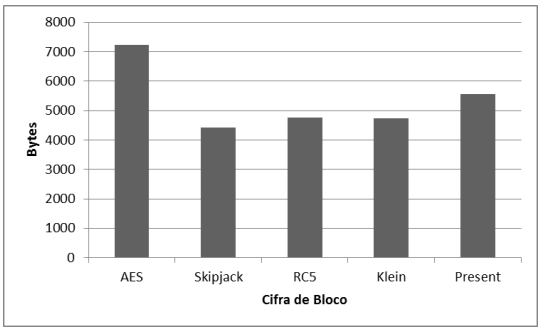

Figura 4. Quantidade de memória ROM requerida pelas cifras de bloco.

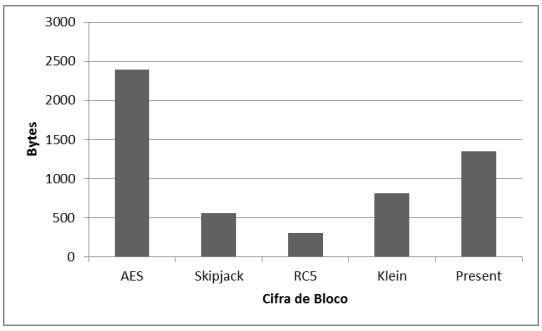

Figura 5. Quantidade de memória RAM requerida pelas cifras de bloco.

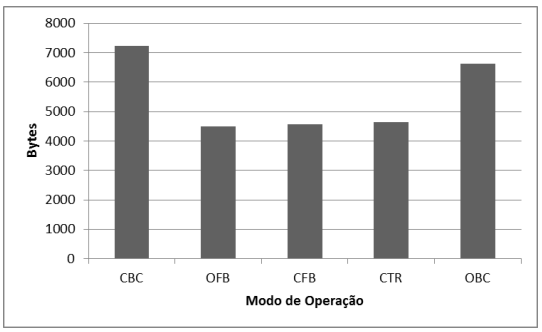

Figura 6. Quantidade de memória ROM requerida pelos modos de operação.

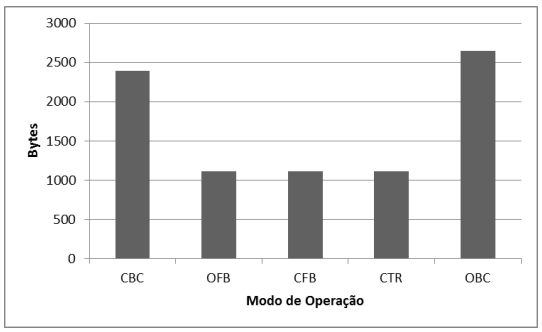

Figura 7. Quantidade de memória RAM requerida pelos modos de operação.

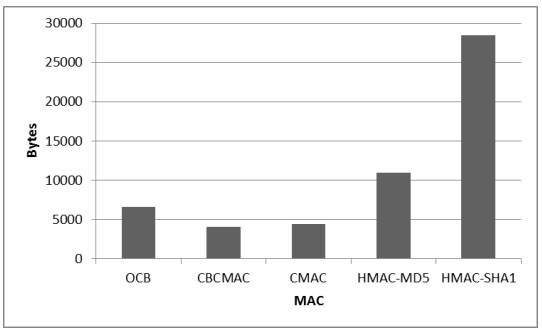

Figura 8. Quantidade de memória ROM requerida pelos MACs.

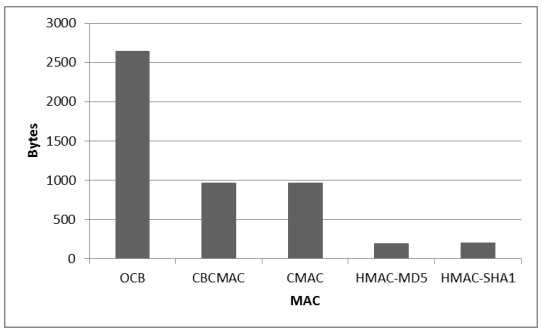

Figura 9. Quantidade de memória RAM requerida pelos MACs.

Na Figura 11 é possível observar que o AES é o algoritmo criptográfico que apresentou o pior desempenho, uma vez que consumiu a maior quantidade de energia, o que era esperado por ser o mais complexo. Por outro lado, o Present apresentou um consumo intermediário entre o AES e as demais cifras. Já o Skipjack e o RC5 foram os algoritmos que apresentaram o melhor desempenho em termos de consumo de energia. O Skipjack apresentou praticamente o mesmo desempenho nas duas operações e, por isso, apenas uma operação é mostrada. Já o RC5 apresentou um consumo cerca de 1% menor do que o Skipjack na encriptação e cerca de 4% maior do que o Skipjack na decriptação. Assim, o Skipjack levou ligeira vantagem sobre o RC5 no geral. Para

criptografar e decriptografar um bloco de 128 *bytes*, a cifra Klein requer em torno de 43% e 66% a mais que o Skipjack.

Com relação aos modos de operação, utilizando o AES-128, a Figura 12 mostra que o CBC e o OCB apresentaram os maiores consumos de energia, nessa ordem. Por outro lado, o OFB, o CFB e o CTR apresentaram desempenho bastante próximos, sendo que o CTR levou ligeira vantagem sobre os modos CFB e OFB, sendo o modo que apresentou o menor consumo de energia. Para criptografar um bloco de 128 *bytes*, o CTR apresentou desempenho 4% e 8% melhor do que o CFB e o OFB, respectivamente. Conforme mostrado na Figura 13, os algoritmos de autenticação CBCMAC e CMAC apresentaram os menores consumos de energia, nessa ordem. O HMAC-SHA1 apresentou consumo significativamente maior que os demais algoritmos, consumindo em torno de 5,4 vezes mais que o CBCMAC para autenticar uma mensagem de 128 *bytes*. Já o OCB, também considerando uma mensagem de 128 *bytes*, consumiu em torno de 26% mais energia para criptografar e autenticar a mensagem e 53% mais energia para decriptografar e verificar a *tag* do que o CBCMAC.

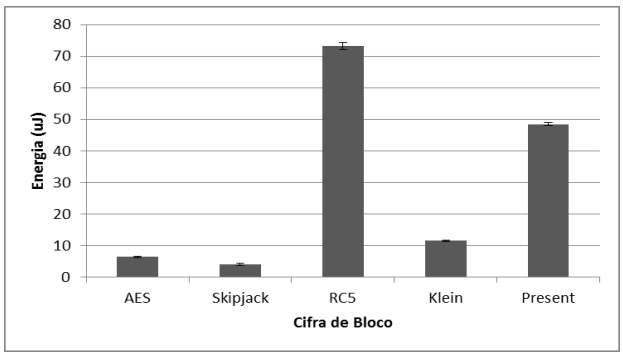

Figura 10. Consumo de energia na inicialização da chave criptográfica.

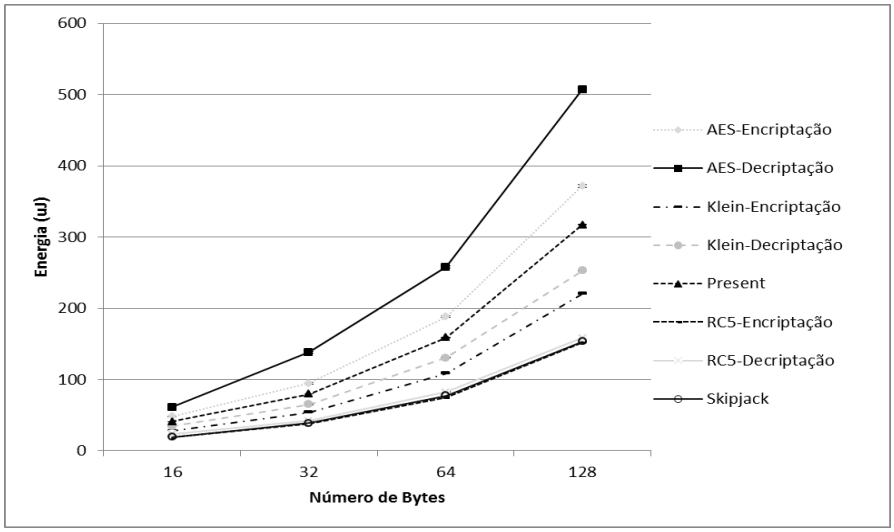

Figura 11. Consumo de energia das cifras de blocos no modo CBC.

5.3. Discussão dos resultados

Os resultados obtidos neste trabalho podem contribuir para a escolha apropriada de mecanismos de segurança a serem empregados em aplicações de RSSF, bem como no desenvolvimento de protocolos de segurança para tais redes. Primeiramente, com

relação aos algoritmos de criptografia, o RC5 (avaliado com a configuração RC5 64/18/16) mostrou-se como a melhor opção para uso em RSSF por apresentar alto nível de segurança e desempenho muito próximo do Skipjack. Este último deve ser evitado, pois, apesar de apresentar o melhor desempenho entre os demais, oferece um nível baixo de segurança, conforme mencionado na Seção 2.1. Além do mais, devido ao fato de o RC5 possuir registro de patente, o que dificulta o seu uso a longo prazo, o algoritmo Klein pode ser uma alternativa atrativa para RSSF.

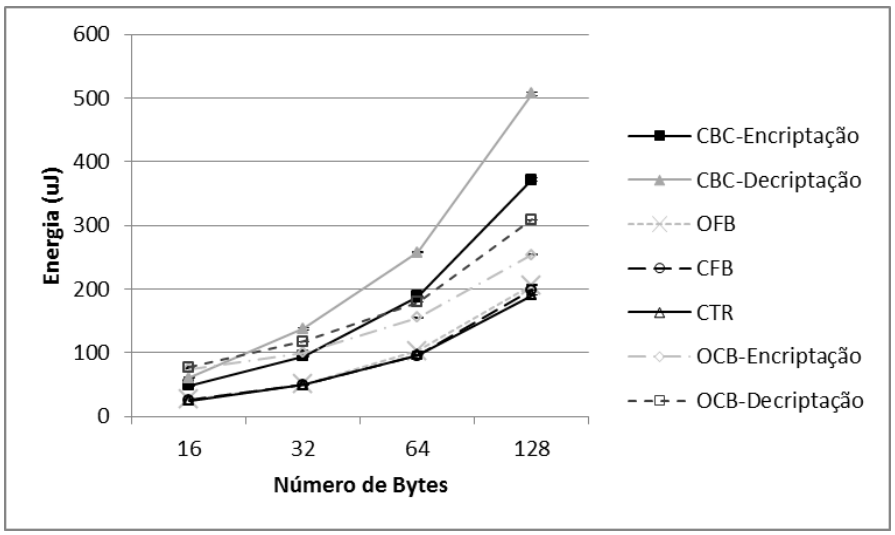

Figura 12. Consumo de energia dos modos de operação com o AES-128.

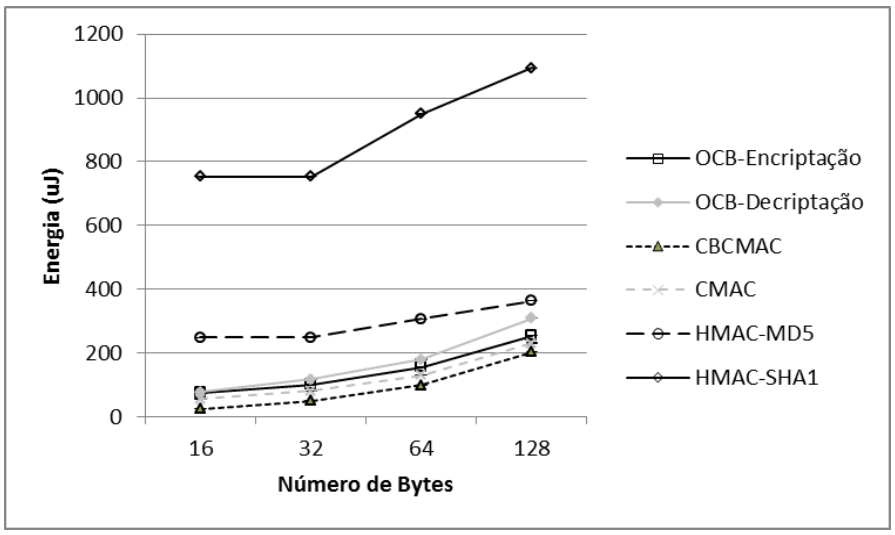

Figura 13. Consumo de energia dos MACs.

Os modos CTR, CFB e OFB apresentaram os melhores desempenhos tanto em termos de memória quanto de consumo de energia. Devido ao fato de que o CFB permite o reuso do IV, ele pode ser uma boa opção para RSSF. Entretanto, o uso do CTR apropriadamente apresentou o melhor desempenho dentre os demais modos em termos de consumo de energia. Além disso, embora o modo OCB apresente um consumo superior, ele é uma solução atrativa quando se deseja confidencialidade e autenticação ao mesmo tempo. Porém, vale lembrar que o OCB possui registro de patente e seu uso é condicionado à concessão de licença. Finalmente, o CBCMAC e o

CMAC apresentaram os melhores desempenhos dentre os algoritmos de autenticação avaliados neste trabalho, sendo os mais recomendados para RSSF. Vale ressaltar que os algoritmos HMAC-SHA1 e HMAC-MD5 são bastante eficientes em plataforma de 32 *bits*. Entretanto, como a arquitetura da MicaZ é de 8 *bits*, esses algoritmos apresentaram desempenho muito inferior aos demais.

6. Conclusão e Trabalhos Futuros

O trabalho apresentou uma avaliação de desempenho de mecanismos de segurança para RSSF. As métricas de desempenho analisadas foram a quantidade de memória ROM e RAM requerida e o consumo de energia. Os experimentos foram realizados na plataforma MicaZ com o auxílio de um osciloscópio digital. Os resultados indicam que o RC5, o CTR e o CBCMAC são as melhores opções para RSSF em termos de desempenho de memória e consumo de energia, dentre os mecanismos avaliados. Portanto, nosso objetivo foi alcançado, uma vez que a análise dos algoritmos e a descrição do processo detalhado de medição podem fornecer um direcionamento para a escolha apropriada de mecanismos de segurança para RSSF, bem como no desenvolvimento e evolução de protocolos de segurança para tais redes. Como trabalhos futuros, pode-se citar a avaliação de outros mecanismos de segurança e a avaliação de outras métricas, como a latência da rede em decorrência do uso de tais mecanismos.

Referências

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

Bellare, M., Kilian, J., and Rogaway, P. (2000). The security of the cipher block chaining message authentication code. In *JCSS*, vol 61.

Bogdanov, A., Knudsen, L. R., Leander, G., Poschmann, A., Robshaw, M. J. B., Seurin, Y., Vikkelsoe, C. (2007). Present: an ultra-lightweight block cipher. In CHES.

Borges Neto, J. B., Ribeiro Neto, P. F., e Andrade, R. M. C. (2010) "Wireless sensor networks advances for ubiquitous computing", Designing solutions-based ubiquitous and pervasive computing: new issues and trends, Mendes Neto, F. M., Ribeiro Neto, P. F. Hershey-PA: IGI Global, p. 175-189.

Borghoff, J., Knudsen, L. R., Leander, G., and Thomsen, S. S. (2011). Cryptanalysis of present-like ciphers with secret s-boxes. In *Fast Software Encryption*, v. 6733.

Braga, A. M., Rubina C. M. F., and Dahab, R. (1998). Tropyc: a pattern language for cryptographic software. In Proceedings of Pattern Languages of Programs.

Cirqueira, A. C., Andrade, R. M. C., e Castro, M. F. (2011). Um mecanismo de segurança com adaptação dinâmica em tempo de execução para dispositivos móveis. In Seminário Integrado de Software e Hardware, p. 1337-1351.

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

Cavalcante, T. M., Garcia, F. P., Gomes, D. G., e Andrade, R. M. C. (2011). Avaliação de desempenho dos algoritmos criptográficos skipjack e rc5 para redes de sensores sem fio. In *Simpósio Brasileiro de Computação Ubíqua e Pervasiva*, p.1103-1112.

Cis lab, (2011) “Cryptography and Information Security Lab”. http://bit.ly/snHm9V, November.

Cryptrec, Cryptography Research and Evaluation Commitees. (2011) “Security of 128-bit block cipher AES”, http://bit.ly/t0fJV8, Setember.

Dworking, M. (2001) “Recommendation for block cipher modes of operation”, http://1.usa.gov/gu49gK, October.

Dworking, M. (2005) “Recommendation for block cipher modes of operation: the cmac mode for authentication”, http://1.usa.gov/uqxrRA, October.

Gong, Z., Nikova, S.I., and Law, Y.W. (2011). Klein: A new family of lightweight block ciphers. In *Workshop on RFID Security*.

Iis, Indian Institute of Science. (2011) “Cryptography algorithms for tinyos”, http://bit.ly/rwzDJT, October.

Karlof, C., Sastry, N., and Wagner, D. (2004). TinySec: A link layer security architecture for wireless sensor networks. In *Sensys,* p. 162-175.

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

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

Luk, M., Mezzour, G., Perrig, A., and Gligor, V. D. (2007). MiniSec: a secure sensor network communication architecture. In *IEEE IPSN*.

MD5deep, MD5deep and hashdeep. (2011). http://bit.ly/1sMRu2, November.

Nist, National Institute of Standards and Technology. (1998) “Skipjack and kea algorithm specifications”, http://bit.ly/uCWxLY, December.

_____. (2002) “The keyed-hash message authentication code (hmac)”, FIPSP 198, http://1.usa.gov/gjg0co, October.

_____. (2007) “Proposal to extend cbc mode by ciphertext stealing”, http://1.usa.gov/sXAIy4, October.

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

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

Rivest, R. L. The RC5 encryption algorithm. (1994). In *FSE,* p. 86–96.

Rogaway, P. (2011). “OCB mode”, http://bit.ly/tzHeom, November.

Saraogi, M. (2004). Security in Wireless Sensor Networks. In *ACM SenSys*.

Schneier, B. (2005) “Sha-1 broken”, http://bit.ly/rqS8Nc, February.

Simplício, M. A. J., e Barreto, P. S. L. (2010). Algoritmos criptográficos para redes de sensores. In *SBSEG*, p. 523-530.