Capítulo 4

Segurança em Redes de Sensores Sem Fio

Cíntia Borges Margi[1], Marcos Simplício Jr[2], Paulo S.M.L. Barreto[2], Tereza C.M.B. Carvalho[2]

Abstract

The main goal for this chapter is to present an overview of security in Wireless Sensor Networks (WSN). First, we introduce general concepts of WSN. Next, main vulnerabilities and attacks are described, as well as security architectures for WSN. Following, problems related to routing, localization ad data aggregation are discussed and some secure solutions are presented. Then symmetric and asymmetric encryption algorithms, Message Authentication Codes (MACs), and key distribution mechanisms are discussed and analyzed with WSN context. Finally, issues related to security mechanisms deployment and tests in WSN testbeds are discussed.

Resumo

O objetivo deste capítulo é apresentar a visão geral da área de segurança em Redes de Sensores sem Fio (RSSF). Primeiro é feita uma apresentação de diversos conceitos sobre RSSF. Em seguida, as principais vulnerabilidades e ataques são descritos, bem como algumas arquiteturas de segurança para RSSF. Com estes conceitos, os problemas relacionados a roteamento, localização e agregação de dados são discutidos e soluções seguras são apresentadas. Então, algoritmos de criptografia simétrica e assimétrica, códigos de autenticação de mensagens (MACs), e mecanismos de distribuição de chaves são discutidos e analisados no contexto de RSSF. Finalmente, questões relacionadas a implantação e testes de mecanismos de segurança em testbed de RSSF são discutidas.

4.1. Introdução

A tecnologia de Redes de Sensores Sem Fio (RSSF) é uma tecnologia que tem propulsionado o desenvolvimento de aplicações em diversas áreas. Como exemplos podem

[1]Escola de Artes, Ciências e Humanidades (EACH) da Universidade de São Paulo (USP) – Brasil. email: cintia@usp.br

[2]Departamento de Engenharia de Computação e Sistemas Digitais (PCS) da Escola Politécnica da Universidade de São Paulo (EPUSP) – Brasil. email: {mjunior, pbarreto, carvalho}@larc.usp.br

ser citadas aplicações que hoje incluem monitoramento ambiental, monitoramento de estruturas de construção civil, controle de temperatura e umidade de ambientes fechados, mapeamento de temperatura em amplas áreas, gerenciamento de desastres naturais, entre outros. Nestes cenários, geralmente os dados coletados pelos nós sensores são enviados a um nó sorvedouro, que por sua vez irá encaminhá-los a um servidor conectado à Internet, de modo que os usuários das aplicações possam acessar os dados coletados em tempo real.

As RSSF podem ser definidas como uma classe especial de redes *ad hoc* de múltiplos saltos (MANETs - Multihop Ad Hoc Networks), pois ambas possuem muitas características comuns. As MANETs são redes sem fio, que não possuem qualquer infraestrutura fixa. Os nós participantes normalmente utilizam baterias como fonte de energia e, portanto, têm acesso a um suprimento limitado de energia. Os equipamentos em redes de sensores tipicamente possuem recursos limitados, como baixa capacidade de processamento, pouca memória para armazenamento, sistemas de comunicação de baixa largura de banda, e principalmente têm fonte de energia com capacidade limitada [31].

Outra característica importante das redes de sensores é o fato de que, para várias aplicações, a implantação da rede é feita sem nenhuma forma de monitoramento dos nós durante toda sua vida operacional. O projeto de protocolos de comunicação que levam em consideração a otimização do consumo de energia pelos nós é essencial. Consequentemente, a compreensão sobre as fontes de consumo de energia devido à execução de tarefas referentes a processamento, sensoreamento e comunicação (as três principais fontes de consumo de energia em um nó) proporciona o conhecimento necessário para definir os ciclos de operação dos nós sensores. O uso de ciclos de operação é um método comum para economizar energia em implantações de redes de sensores [77, 111]. Este método permite que o nó sensor alterne entre períodos de atividade, inatividade (*idle*) e modo de baixo consumo (*sleep*), economizando, assim, energia.

A maioria da pesquisa desenvolvida atualmente em redes de sensores foca em aplicações que requerem baixa taxa de amostragem dos sensores e pouca largura de banda da rede, como aplicações que fazem uso de sensores de umidade ou temperatura. Além disso, a principal hipótese é que os custos de comunicação (em termos de energia) dominam em redes de sensores. Porém o que acontece se sensores mais sofisticados, como acelerômetros ou magnetômetros, forem utilizados? O trabalho desenvolvido por Doherty et al. [37] apresenta três aplicações diferentes: monitoramento de ambiente, detecção de terremotos e rastreamento, sendo que cada uma delas apresenta um componente diferente que domina os custos de energia. No caso do cenário de monitoramento de ambiente, o objetivo é monitorar as condições ambientais de um prédio de escritórios, e os sensores utilizados incluem sensores de temperatura, umidade e luminosidade. Nesse caso, todos esses sensores possuem taxa de amostragem baixa e os nós estão a um salto de distância do sorvedouro, porém a comunicação domina o consumo de energia. No caso do cenário de detecção de terremoto, monitora-se a frequência de oscilação em diversos pontos da estrutura. Para tanto, utilizam-se acelerômetros de 3 eixos e magnetômetros de 3 eixos, e a taxa de amostragem típica é de 100 Hz. Neste caso, os custos de energia relacionados a sensoreamento e comunicação são equivalentes. O terceiro caso apresentado é o cenário de rastreamento, onde se detectam e rastreiam veículos. Os sensores usados incluem microfones, acelerômetros e magnetômetros. Vários nós comportam-se somente

como encaminhadores de dados, não executando sensoreamento algum. Neste cenário, o sensoreamento consume mais energia do que a comunicação.

Um cenário ainda mais complexo são as Redes de Sensores Visuais sem Fio (RSV), que incluem câmeras como sensores [78, 24, 97]. Este tipo de redes de sensores possui requisitos de comunicação mais complexos do que redes de sensores compostos por sensores de temperatura e umidade por exemplo. Dentre estes requisitos, podemos citar: suporte da rede para múltiplos fluxos com prioridades diferentes; gerenciamento de recursos adaptável como largura de banda e potência do sinal de transmissão; garantia de baixa latência e variação de atraso para comunicação entre nós sensores; conhecimento do estado atual da rede (largura de banda, energia disponível nos nós vizinhos); auto-organização e auto-gerenciamento. Além de terem requisitos de comunicação mais complexos, as RSVs exigem maior capacidade de processamento do nó. A aquisição de vídeo (ou de sequência de imagens) é uma operação intensa para a CPU (*Control Process Unit*), pois os dados obtidos precisarão ainda ser comprimidos. O uso de algoritmos de computação visual pode minimizar a quantidade de dados a ser transmitido e/ou tornar o nó inteligente. Um nó sensor visual inteligente deve ser capaz de tomar decisões sobre quando rastrear um objeto, quando passar a responsabilidade de rastrear um objeto para um nó vizinho em melhor posição, e quando e que tipo de dado enviar ao nó sorvedouro. Dependendo da quantidade de energia disponível no nó e dos recursos de rede disponíveis, o nó sensor pode decidir se envia o vídeo (ou uma sequência de imagens) com ou sem compressão, ou alguma forma de representação de alto nível do objeto sendo rastreado.

Conforme já mencionado, a utilização de ciclos de operação é comum em implantações de redes de sensores. Geralmente, os ciclos de operação são definidos antes da implantação. Por exemplo, no trabalho desenvolvido por Doherty et al. [37], as tarefas são determinadas tomando como base a energia disponível. No caso da redes de sensores implantada em *Great Duck Island* [77] para monitoramento de habitat, a vida útil da rede deveria ser de nove meses, de modo a capturar as variações de plantas e animais com a mudança das estações do ano. Considerando este fato, a energia disponível nos nós por dia, e o custo estimado das tarefas a serem executadas (sensoreamento, transmissão de dados, sistema operacional, tarefas de manutenção), determinou-se o conjunto diário de tarefas e, portanto, o ciclo de operação.

Em outra abordagem possível, o nó sorvedouro determina as tarefas a serem executadas por um nó e seu intervalo de inatividade, e consequentemente o seu ciclo de operação. Como exemplo desta abordagem podemos citar o TinyDB [76], que foi implementado no TinyOS [52] e provê acesso de maneira similar a SQL (*Structured Query Language*), onde o usuário especifica os dados que quer extrair da rede de sensores, juntamente com parâmetros adicionais, como por exemplo a taxa de amostragem. Uma vez feita a requisição, TinyDB irá coletar os dados dos sensores, filtrar, agregar e rotear para o nó sorvedouro. Levis et al. [68] propõem a utilização de *Máquinas Virtuais Dependentes de Aplicação (ASVM - Application Specific Virtual Machines)* para reprogramar nós sensores. Esse tipo de abordagem centralizada provê mais flexibilidade do que o uso de ciclos de operação pré-determinados porém, além de gerar tráfego adicional na rede, depende de uma entidade central para otimizar a coleta de dados.

Com a utilização de redes de sensores com nós inteligentes, pode-se adotar uma terceira abordagem: o próprio nó determina como será seu ciclo de operação baseado nos requisitos da aplicação (sequência de tarefas a serem executadas, frequência mínima e máxima de amostragem), na quantidade de energia disponível na sua bateria, no estado da rede (perda de pacotes, atraso médio, largura de banda disponível), e nas informações recebidas de seus vizinhos. A partir dessas informações, os nós sensores podem decidir quando enviar dados, quando mudar para o estado de baixo consumo de energia, quando coletar dados, e assim por diante. Para tratar deste problema, torna-se necessário um modelo de consumo de energia que considere a comunicação, o processamento e o sensoreamento. Este modelo poderia ser usado tanto por projetistas de redes de sensores como de protocolos. Outra aplicação para este modelo seria na definição de um *gerente de recursos* a ser implementado no nó sensor, que tomaria as decisões relativas às atividades a serem executadas. Lachenmann et al. [64] propõem uma abstração de programação para aplicações de RSSF conscientes de energia, onde não existe redundância e nenhum nó deve falhar. Utilizando informações de consumo de energia de blocos de código obtidas em simuladores, os autores propõem níveis de consumo de energia, que devem ser escolhido de acordo com o tempo de vida desejado.

Os protocolos de camada de controle de acesso ao meio (MAC - *Medium Access Control*) específicos para RSSF incluem: S-MAC [113], TRAMA [98] e TMAC [112]. Mais recentemente, o padrão IEEE 802.15.4 [108] foi proposto, englobando uma especificação das camadas de enlace e física (as duas camadas inferiores da pilha de protocolos OSI) com as seguintes características: conexão sem fio de curto alcance com baixa taxa de transferência e baixa latência, mobilidade e baixo consumo de energia. É classificado como um padrão de rede *Wireless Personal Area Network* (WPAN) e opera tipicamente no Espaço de Operação Pessoal (ou POS - *Personal Operating Space*) de 10 metros e sua taxa de transmissão varia entre 20 e 250 kb/s. O mecanismo de Controle de Acesso ao Meio utilizado é o CSMA/CA (*Carrier Sense Multiple Access - Collision Avoidance*) com alocação de *slots* de tempo opcional, reconhecimento de mensagem e uma estrutura de sinalização conhecida como *beacon* [17].

Dada as características das RSSF, onde são importantes o uso eficiente de energia e a colaboração entre os nós para atingir o objetivo da aplicação sendo executada, protocolos da pilha TCP/IP (como o IP, TCP ou UDP) não são utilizados. Dentre os diversos protocolos propostos, temos exemplos de protocolos de coleta de dados, como o Diffusion [55] e SPIN (*Sensor Protocols for Information via Negotiation*) [63], agregação de dados [106], controle de topologia [21], *clustering* [50]. Observa-se que estes protocolos acabam sendo extremamente dependentes da aplicação, não existindo padrões.

4.1.1. Nós Sensores

As RSSF são compostas por dispositivos de baixo custo, que possuem recursos limitados de processamento, armazenamento, comunicação, energia, além dos sensores propriamente ditos [31]. Os nós sensores TelosB e MicaZ da Crossbow têm sido amplamente utilizados em *testbeds* e implantações de RSSF.

O nó sensor TelosB [29] possui processador RISC de 16 bits com 8 MHZ de clock, 48 Kilobytes de memória flash programável, 10 Kilobytes de memória RAM, EEPROM

de configuração de 16 Kilobytes, interface USB v1.1 ou mais alta, e 3 LEDs. Ele consome 1,8 mA no modo ativo e 5,1 μA no modo *sleep*, utilizando duas pilhas AA como fonte de energia. Seu rádio é compatível com o padrão IEEE802.15.4 [108], operando na faixa de frequência ISM de 2,4 a 2,4835 GHz, e taxa de transmissão de 250 Kbps. O TelosB possui os seguintes sensores: luz visível (320 a 730 nm); luz visível a IR (320 a 1100nm); umidade e temperatura.

O Crossbow MicaZ [27] também é compatível com o padrão IEEE 802.15.4 e utiliza o mesmo rádio do Crossbow TelosB. No entanto, utiliza o microcontrolador MPR2400 Atmel e possui 128 Kilobytes de memória flash programável, 512 Kilobytes de memória para medidas, EEPROM de configuração de 4 Kilobytes, e 3 LEDs. Ele consome 8 mA no modo ativo e menos de 15 μA no modo *sleep*, utilizando duas pilhas AA como fonte de energia. A placa de sensores MTS400CA inclui sensores de temperatura, umidade, luz visível e acelerômetro de 2 eixos.

4.2. Segurança em RSSF: Visão Geral

As diferentes aplicações de RSSF possuem diferentes requisitos de segurança. Imagine uma aplicação de monitoramento climático em uma floresta com a finalidade de obter dados para estudos biológicos. Os pesquisadores que utilizarão os dados recebidos no sorvedouro, esperam que estes reflitam a realidade do que foi medido nos diversos pontos da floresta. Ou seja, a integridade dos dados coletados e transmitidos é importante.

Os dados obtidos com a monitoração de aspectos climáticos de uma floresta ou características de animais em uma região não são sigilosos. Contudo, a divulgação do estado de máquinas e/ou do ambiente em uma planta fabril poderia levar a prejuízos ou vantagem a concorrência. Assim, o sigilo desses dados monitorados é extremamente importante. Além da confidencialidade dos dados, também é necessário garantir o sigilo das informações de roteamento para evitar que estas sejam usadas em ataques de negação de serviço.

Se considerarmos a automação residencial com RSSF ou aplicações ligadas a área de saúde, além da confidencialidade dos dados, é importante garantir a autenticidade dos dados que são coletados e transmitidos, bem como a autenticidade dos nós. No caso de tarefas de administração e gerenciamento da rede, como no envio de consultas via TinyDB [76] ou a reprogramação de nós sensores [68], a autenticidade do sorvedouro precisa ser verificada, pois caso contrário a operação da RSSF fica comprometida.

Ainda, é importante garantir a disponibilidade da aplicação de RSSF as partes autorizadas. Portanto, são necessários mecanismos de sobrevivência a ataques de negação de serviço (como a injeção de pacotes inúteis para aumentar o consumo de energia dos nós e diminuir o tempo de vida da RSSF) que poderiam ocorrer em qualquer camada da rede.

Além dos serviços de segurança citados (integridade dos dados, confidencialidade, autenticidade do nó e dos dados e disponibilidade), também é importante garantir que os dados enviados são recentes (*data freshness*), ou seja, que nenhum intruso está replicando dados antigos. Este mesmo conceito também pode se aplicar às chaves em uso na RSSF (*key freshness*). E, assim, os mecanismos de estabelecimento de chaves devem garantir

que as chaves em uso são recentes, impedindo o uso de chaves antigas que poderiam ter sido obtidas após o comprometimento de um nó [54].

Dadas as características das RSSF, que se baseiam em nós com recursos de processamento, armazenamento, comunicação e energia limitados, os mecanismos de segurança empregados devem ser escaláveis (em termos de energia e atraso). Os desafios para a implementação de mecanismos de segurança em RSSF, de acordo com Hu e Sharma [54], incluem:

- **Conflito entre minimizar consumo de recursos e maximizar a segurança**: neste contexto, recursos são energia, ciclos de CPU, memória. Para garantir o sigilo de mensagens é necessário criptografar os dados, e para garantir a autenticidade são adicionados *tags* de autenticação, o que pode aumentar o tamanho da mensagem a ser transmitida. A criptografia e o cálculo dos códigos de autenticação de mensagens são operações que serão executadas no nó origem e em cada nó que precise ler o conteúdo da mensagem cifrada ou verificar a autenticidade da mensagem. Além disso, o encaminhamento de mensagens maiores irá custar mais energia a todos os nós na sua rota. Os mecanismos de estabelecimento e distribuição de chaves também consomem energia, e devem ser otimizados para reduzir o processamento e minimizar o número de mensagens trocadas. O uso de ciclos de trabalho (*dutycycles*) com período de inatividade dos nós (que mudam para modo *sleep* para reduzir o consumo de energia), pode levar a falhas na sincronização ou atualização de chaves. Ainda, se expandirmos o conceito de recursos para incluir o custo do nó, mecanismos de proteção física do nó (*tamper protection*) aumentam o custo.

- **Topologia de RSSF suscetível a ataques ao enlace**: diferentemente das redes cabeadas, as redes sem fio não possuem proteção física ao enlace ou aos equipamentos. Assim, ataques passivos (monitoramento do canal) e ativos (interferência) são possíveis. Além disso, existe uma grande quantidade de nós, com a possibilidade de mobilidade em alguns casos, e estes normalmente não são monitorados. Desta forma, alguns nós podem ser capturados e substituídos por nós comprometidos, ou informações críticas podem ser obtidas.

- **Características da comunicação sem fio**: o alcance da transmissão limitado (menor do que 100 metros, chegando a 20m em algumas situações), a largura de banda limitada (menos do 250 Kbps), os enlaces não são confiáveis e os canais são unidirecionais, que requerem mecanismos diferentes daqueles usados em redes cabeadas.

4.2.1. Vulnerabilidades e Ataques

Conforme descrito pelos desafios indicados por Hu e Sharma [54], muitas das vulnerabilidades das RSSF existem devido a comunicação sem fio e o fato de que os nós sensores ficam em locais sem segurança física ou não são monitorados.

As principais vulnerabilidades relacionados a camada física do modelo OSI incluem a interferência do sinal de comunicação transmitido, e o dano de nós sensores. A interferência do sinal de comunicação transmitido por um nó (*signal jamming*) ocorre quando um nó intruso gera sinais aleatórios para impedir que a comunicação entre os nós

da RSSF ocorra corretamente. Uma maneira de evitar este tipo de interferência com as frequências em uso é através do uso de espalhamento espectral para a codificação dos sinais [54]. Porém, os rádios com suporte a codificação por espalhamento espectral são mais complexos, mais caros e consomem mais energia, o que pode inviabilizar seu uso em RSSF.

A segunda vulnerabilidade física é oriunda do fato dos nós sensores ficarem em locais sem segurança física ou não monitorados, e contempla formas de *node tampering*. Um intruso poderia danificar um nó sensor, de modo que este não efetuaria as suas funções de coleta de dados e/ou roteamento, prejudicando o funcionamento da aplicação sendo executada pela RSSF. Ainda, o nó poderia ser substituído por um nó malicioso para gerar ataques a rede ou obter informações sendo transmitidas. Uma terceira possibilidade é que as informações armazenadas em um nó sensor capturado sejam extraídas, permitindo a um atacante obter chaves de criptografia ou autenticação. Para evitar que esta vulnerabilidade seja explorada são necessários circuitos ou mecanismos para proteção dos dados, capas de proteção ou selos.

Protocolos de Controle de Acesso ao Meio baseados em contenção podem representar uma vulnerabilidade em RSSF. Basta induzir colisões de um octeto para que um quadro completo seja corrompido, e que a retransmissão seja necessária. Corromper um quadro de confirmação (ACK) poderia fazer com que nós entrem em *backoff* exponencial, e as transmissões de dados sejam atrasadas ou perdidas (devido ao número máximo de tentativas). Mecanismos de correção de erro mais complexos poderiam evitar que a colisão de um octeto descartasse a mensagem toda. Observe que explorar esta vulnerabilidade pode fazer com que os nós em uma região fiquem completamente sem energia, o que poderia levar a um particionamento da rede e/ou sua indisponibilidade.

As vulnerabilidades na camada de rede provêm de problemas associados ao roteamento de dados, uma vez que, em RSSF, todos os nós são roteadores. A forma mais direta de ataque a um protocolo de roteamento é alterar, repetir ou falsificar (*spoof*) pacotes de controle do mesmo, de forma a criar *loops*, desvios, buracos negros ou partições [59]. Dentre os ataques, podemos enumerar [60, 54, 3]:

- **Buracos Negros** (*Black or sink holes*): um nó malicioso introduz a melhor rota a vários um destino passando por ele, possibilitando ao nó malicioso descartar ou modificar pacotes. Ainda, a medida que mais pacotes são roteados e mais nós disputam o acesso ao meio de transmissão, os nós sensores que encontram-se nessa rotas terão sua energia consumida mais rapidamente, o que poderia levar a um particionamento da rede com morte desse nós.

- **Inundação da rede** (ou *flooding*): um nó malicioso inunda a rede com mensagens falsas, causando congestionamento e consumo excessivo de energia pelos nós sensores. Além disso, o envio de mensagens de roteamento falsas pode criar rotas erradas, causando descarte de pacotes.

- **Desvios e *loops***: nós maliciosos alteram o roteamento dos pacotes, de modo a direcionar o tráfego através de desvios ou *loops* por nós com pouca energia ou congestionados. Isso atrasa a entrega dos pacotes, ou provoca a sua perda.

- ***Wormholes***: dois nós maliciosos em diferentes partições da rede criam um túnel entre si, utilizando frequência de rádio diferente daquela utilizada na RSSF. Através deste túnel, enviam pacotes de uma partição da rede para a outra, fazendo com que nós em diferentes partições acreditem serem vizinhos, e consequentemente introduzindo problemas de convergência no roteamento.

- **Sequestro de nós**: um grupo de nós maliciosos cercam um nó sensor da RSSF, e o sequestram ao recusar o envio de suas mensagens, descartando-os ou injetando pacotes inválidos.

- **Nós irmãos** (*Sybil Nodes*): um nó malicioso assume múltiplas identidades (fabricadas ou roubadas), que são chamadas de nós irmãos. Além de ataques a protocolos de roteamento, os "nós irmãos´´ podem ser usados para atacar protocolos de protocolos de armazenamento distribuído, agregação de dados, eleições e algoritmos de detecção de mau-comportamento.

A Figura 4.1 ilustra alguns ataques possíveis na camada de rede em RSSF.

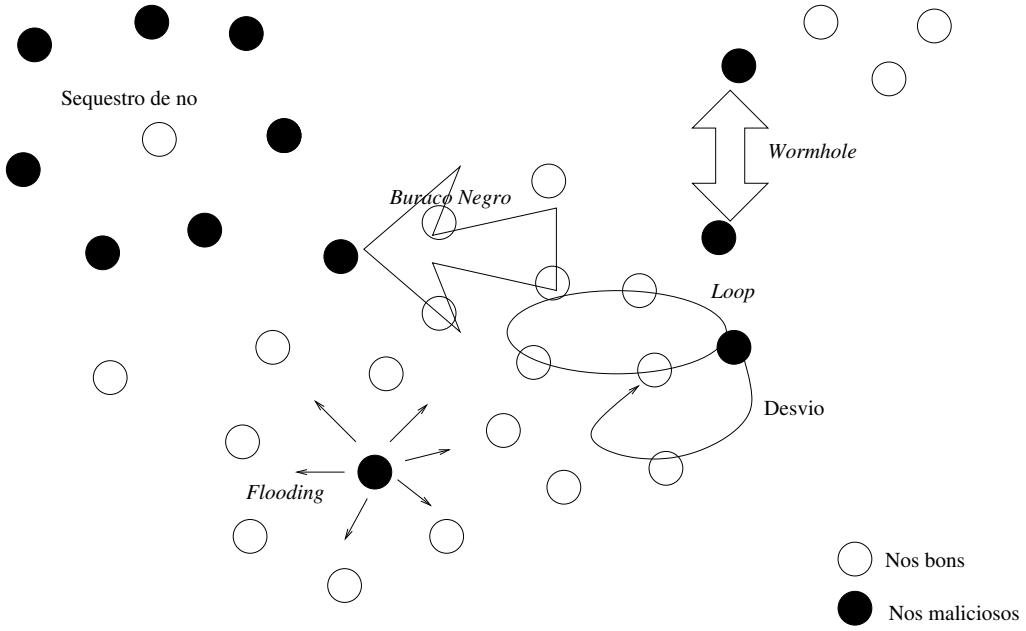

Figura 4.1. Ataques na camada de rede em RSSF.

Do ponto de vista da camada de aplicação, **Buracos na Cobertura** ocorrem se um número insuficiente de nós, de acordo com os requisitos da aplicação, realiza o sensoreamento em uma dada região. Se a densidade da distribuição de nós não for suficiente, e/ou se muitos nós "morrerem" em uma região, estes buracos aparecem.

Outro aspecto a ser considerado em várias camadas, é o envio de mensagens de reconhecimento falsas (*acknowledgment spoofing*). Diversos protocolos de RSSF utilizam mensagens de reconhecimento, e a falsificação destas pode fazer com que um nó acredite que um vizinho sem energia está funcionando corretamento, ou que um enlace ruim é bom.

Portanto, os principais ataques a RSSF envolvem a captura de nós, o manuseio indevido dos mesmos (*tampering*) e negação de serviço (através da criação de buracos negros, *wormholes*, introdução de desvios ou *loops*, sequestro de nós, criação de problemas na cobertura da aplicação) [94].

4.2.2. Arquiteturas de Segurança

Dadas as necessidades de segurança caracterizadas pelos serviços de integridade dos dados, confidencialidade, autenticidade do nó e dos dados, disponibilidade, e dados recentes (*freshness*), descritos na Seção 4.2, alguns autores propuseram arquiteturas de segurança para atender a estes serviços. Dentre estas arquiteturas podemos destacar: SPINS [95], TinySec [58] e MiniSec [75]. Ainda, o padrão IEEE802.15.4 inclui um *framework* de segurança para atender os serviços de integridade dos dados, confidencialidade, autenticidade.

É importante lembrar que os nós sensores nas RSSF geralmente possuem recursos de processamento, armazenamento, comunicação e energia limitados e, portanto, os mecanismos de segurança empregados devem ser escaláveis (em termos de energia e atraso).

SPINS

Perrig et al. [95] propuseram o SPINS (*Security Protocols for Sensor Networks*), que é um conjunto de protocolos de segurança para RSSF. Ele consiste de dois blocos básicos: SNEP (*Secure Network Encription Protocol*) e o μTESLA (*micro Timed Efficient Stream Loss-tolerant Authentication*).

SNEP provê confidencialidade, autenticação e integridade da mensagem, além de evitar ataques de repetição, através de criptografia e códigos de autenticação de mensagem (MAC - *Message Authentication Code*). São adicionados 8 bytes por mensagem. A segurança semântica é obtida através de um contador que é incrementado a cada mensagem. A autenticação dos dados é obtida usando o mesmo algoritmo de criptografia no modo CBC-MAC [85]. μTESLA emula assimetria através da disponibilização atrasada de chaves simétricas, e funciona como o serviço de autenticação de *broadcasts* para o SNEP.

TinySec

Uma das arquiteturas de segurança de RSSF mais conhecida é o TinySec [58], projetado e implementado no sistema operacional TinyOS. Segundo seus autores, a maior motivação para implementá-lo foi o fato do SPINS [95] não ter sido nem projetado nem implementado completamente. É um mecanismo para prover confidencialidade, integridade e autenticidade na camada de enlace de dados, além de proteção a repetição de mensagens. Porém, não provê mecanismos para gerenciamento e estabelecimento de chaves, utilizando uma única chave para toda a rede, o que sacrifica o seu nível de segurança.

O TinySec permite dois modos de operação, que refletem em dois formatos de mensagens: TinySec-Auth, para mensagens autenticadas, e TinySec-AE, para mensagens

autenticadas e cifradas. No modo TinySec-AE, a carga útil é de até 29 bytes, que é cifrada, e o cabeçalho do pacote é de 8 bytes, sendo que o pacote inteiro (dados e cabeçalho) é autenticado. No modo TinySec-Auth, o pacote inteiro (dados e cabeçalho) é autenticado, sendo a carga útil também de até 29 bytes, mas o cabeçalho do pacote é de 4 bytes.

O pacote TinySec-AE é constituído pelos campos de: destino, controle de camada de rede, tamanho da mensagem transmitida, origem, contador, dados e MAC. Os cinco primeiros campos do pacote são utilizados como vetor de inicialização (IV), sendo que o campo contador (Ctr) é utilizado para que o IV não se repita e a cada mensagem enviada, é incrementado. Ao combinar o contador com os outros campos, o TinySec torna o IV mais complexo e, consequentemente, menos previsível. O TinySec-Auth utiliza menos campos que o TinySec-AE, mantendo quatro campos do formato padrão do pacote do TinyOS, eliminando dois deles e adicionando o campo de MAC. A eliminação do campo CRC (*Cycle Redundance Check*) é justificada pelo fato de que o MAC adicionado permite a detecção de erros de transmissão da mensagem.

Em relação ao conjunto de algoritmos, o mecanismo de criptografia adotado pelo TinySec é o modo de operação CBC que pode ser combinado com várias cifras de bloco, dentre as quais RC5 e Skipjack [90]. Para autenticação e integridade de mensagens, o TinySec utiliza o algoritmo CBC-MAC [85].

MiniSec

O MiniSec [75] é um protocolo de camada de segurança para RSSF, sendo que sua abordagem é parecida com a abordagem do TinySec [58] e se propõe a resolver alguns pontos fracos deste. A principal mudança de mecanismos em relação ao TinySec é a utilização de um modo de operação de cifra de bloco diferente, o OCB (*Offset Codebook*), que elimina a necessidade de adicionar enchimento aos blocos de texto claro. Desta forma, a mensagem cifrada fica do mesmo comprimento do texto original, economizando na transmissão de menos bytes. Outra vantagem apontada pelos autores, é que o OCB calcula o texto cifrado e o MAC da mensagem conjuntamente, diferentemente do CBC que necessita de uma execução para cifrar a mensagem e outra para gerar o MAC.

Em relação ao formato de pacotes, o MiniSec elimina o campo contador (CTR) presente TinySec para compor o vetor de inicialização (IV), diminuindo 2 bytes do cabeçalho. Isto é possível porque o modo de operação OCB não necessita que o IV seja aleatório, e também porque o MiniSec utiliza somente alguns bits dos campos tamanho (LEN) e destino (DST) como parte do IV.

O MiniSec também adiciona proteção contra ataques repetição, utilizando mecanismos de sincronização e estruturas de dados para armazenar contadores de mensagens. Esses mecanismos também contribuem para eliminar o campo contador (CTR) do pacote de transmissão de dados, pois os contadores para sincronização são utilizados como parte do IV. Essa abordagem é interessante, porém se um nó da RSSF receber mensagens de muitos nós diferentes as estruturas de dados podem passar a ocupar uma grande de bytes de memória, outro recurso limitado nas RSSF. Segundo Luk et al. [75], existe a tendência de aumento no espaço de memória nos nós sensores, enquanto as fontes de energia permanecem restritas, o que torna viável o aumento do consumo de memória em favor da

economia de energia.

Padrão IEEE 802.15.4

O padrão IEEE 802.15.4 [108] provê os seguintes serviços de segurança na camada de enlace: (1) controle de acesso; (2) integridade da mensagem; (3) confidencialidade da mensagem; e (4) proteção contra repetição. Estes serviços de segurança são providos através de um dos oito conjuntos de segurança (*security suites*) definidos no padrão. No entanto, a configuração padrão é a de nenhum serviço habilitado. O segundo modo provê somente criptografia, utilizando o AES no modo CTR, enquanto o terceiro provê somente autenticação (AES-CBC-MAC). O quarto modo provê criptografia e autenticação, utilizando o AES-CCM. Observe que o código de autenticação de mensagem gerado pode ter 128, 64 ou 32 bits, e esta combinação é permite os oito modos de operação [16].

Considerações

Tanto o TinySec como o MiniSec, assim como o *framework* de segurança do padrão IEEE 802.15.4, operam na camada de enlace de dados. Assim, mensagens transmitidas pelos nós da RSSF precisam ser cifradas/decifradas a cada salto na comunicação, adicionando *overhead*. Da mesma forma, a autenticidade e integridade das mensagens é verificada através dos códigos de autenticação de mensagem. Observe que se uma única chave for usada na RSSF toda (hipótese assumida pelos autores do TinySec [58]), só será realizada uma única operação. Porém, a criptografia na camada de enlace não protege funções da camada de rede, como roteamento de múltiplos saltos, pois nós comprometidos ainda poderiam gerar mensagens autenticas.

Observe que dependendo do padrão de comunicação entre os nós da RSSF, isto é, se ocorrem entre pares de nós vizinhos, ou numa vizinhança, ou do nó ao sorvedouro, diferentes conjuntos de chaves podem ser necessárias. Desta forma, os mecanismos de segurança (criptografia e códigos de autenticação de mensagem) deveriam ser empregados na própria camada de aplicação.

4.2.3. Roteamento Seguro

O roteamento de dados em RSSF possui alguns desafios característicos deste ambiente. O principal foco é a coleta de informações em uma região, não considerando um nó único. A própria topologia de RSSF é um fator complicador, pois existe um grande número de nós (tipicamente considera-se implantações com centenas a milhares de nós); não existe infra-estrutura (energia, monitoramento dos nós); podem ocorrer mudanças de topologia devido a nós sem energia, ou a adição de novos nós, ou acidentes com os nós existentes, ou mobilidade dos nós sensores.

De acordo com Karlof e Wagner [59], os protocolos de roteamento de RSSF tornam-se mais suscetíveis a ataques devido a sua simplicidade e, em alguns casos, uma única mensagem poderia desabilitar uma RSSF toda. A Tabela 4.1 resume alguns dos principais protocolos de roteamento para RSSF e como estes podem ser atacados.

Os protocolos mencionamos na Tabela 4.1 incluem o *TinyOS beaconing*, que é distribuído com o sistema operacional TinyOS [52], o *Direct Diffusion* [55], exemplos de protocolos de roteamento geografico (como o GEAR [115]), de *clustering* (como o LEACH [49]), de conservacao de energia (como o SPAN [23]), entre outros. Nesta seção, detalhamos três destes protocolos (*TinyOS beaconing*, *Direct Diffusion* e roteamento geográfico), e em seguida, discutimos algumas alternativas seguras.

Tabela 4.1. Resumo de ataques contra alguns protocolos de roteamento de RSSF [59].

Protocolo	**Ataques Relevantes**
TinyOS beaconing	Informação de roteamento falsa, encaminhamento seletivo de mensagens, *sinkholes*, nós irmãos, *wormholes*, inundação de mensagens de controle.
Directed diffusion	Informação de roteamento falsa, encaminhamento seletivo de mensagens, *sinkholes*, nós irmãos, *wormholes*, inundação de mensagens de controle.
Roteamento Geográfico (GPSR, GEAR)	Informação de roteamento falsa, encaminhamento seletivo de mensagens, nós irmãos.
Encaminhamento de custo mínimo	Informação de roteamento falsa, encaminhamento seletivo de mensagens, *sinkholes*, *wormholes*, inundação de mensagens de controle.
Protocolos baseados em *clustering* (LEACH, TEEN, PEGASIS)	Encaminhamento seletivo de mensagens, inundação de mensagens de controle.
Rumor routing	Informação de roteamento falsa, encaminhamento seletivo de mensagens, *sinkholes*, nós irmãos, *wormholes*.
Manutenção de topologia para conservação de energia (SPAN, GAF, CEC, AFECA)	Informação de roteamento falsa, nós irmãos, inundação de mensagens de controle.

TinyOS Beaconing

O protocolo *TinyOS Beaconing* constrói uma árvore de cobertura por largura (*breadth first spanning tree*) com raiz no nó sorvedouro (ou estação base). Periodicamente, o sorvedouro envia uma atualização de rota via *broadcast*. Os nós que recebem esta atualização marcam o sorvedouro como pai, e reenviam a atualização. O algoritmo continua recursivamente, com cada nó marcando como seu pai aquele de quem recebe a atualização primeiro naquele período. Cada pacote recebido ou gerado por um nó, é encaminhado a seu pai, até que alcancem o sorvedouro.

Karlof e Wagner [59] discutem como o protocolo *TinyOS Beaconing* pode ser atacado. Observe que, como os pacotes de atualização de rotas não são autenticados, qualquer nó pode se passar pelo sorvedouro, enviando estas informações a RSSF. A Figura 4.2

ilustra como a árvore de roteamento muda após um ataque com informações falsas.

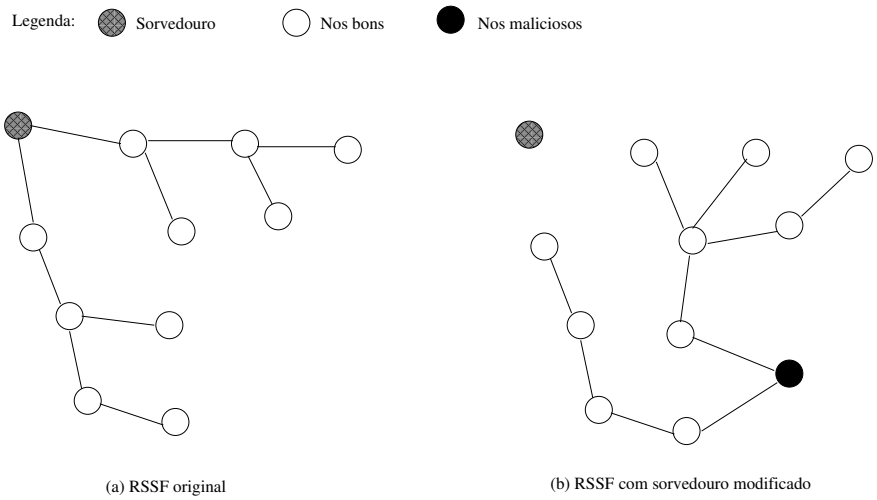

Figura 4.2. Alteração do sorvedouro através de informações de roteamento falsa.

Mesmo se as informações de atualização fossem autenticadas, ainda seria possível que um atacante monitorasse, removesse ou modificasse pacotes em uma determinada região combinando ataques de *wormhole* e *sinkhole*. Por exemplo, se forem utilizados dois nós maliciosos, um próximo ao sorvedouro e outro em uma parte mais distante da rede, é possível criar um *wormhole*, conforme ilustrado na Figura 4.3. Observe que o alcance da transmissão destes nós maliciosos deve ser maior do que o alcance de um nó sensor normal, ou então a geografia do local precisa favorecer o particionamento da rede. Note que é possível criar o *wormhole* porque o nó malicioso próximo ao sorvedouro encaminha as atualizações autenticadas ao segundo nó malicioso, que as encaminha a seus vizinhos. Como a mensagem de atualização de rota através do segundo nó malicioso chega a seus vizinhos mais rapidamente que a mensagem originalmente enviada pelo sorvedouro, este nó se torna a raiz da árvore de roteamento desta partição da rede, conforme observamos na Figura 4.3. A partir deste ponto, o nó malicioso pode decidir quais pacotes encaminhar ou não, interferindo no funcionamento correto da RSSF.

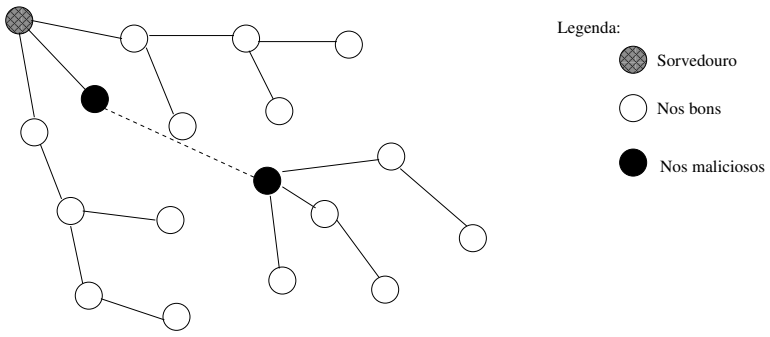

Figura 4.3. Criação de *wormhole* para modificar o roteamento de mensagens na RSSF.

Dependendo da potência de transmissão do nó malicioso, este pode inundar a rede com mensagens de controle de roteamento (por exemplo, mensagem de atualização

de rota ou de identificação de vizinho), eventualmente cobrindo todos os nós na RSSF. Todos os nós no alcance desta mensagem consideram o nó malicioso seu pai na arvore de roteamento. A partir disto, a rede fica comprometida, pois seus pacotes são encaminhados ao nó malicioso. Dada a simplicidade do protocolo *TinyOS Beaconing*, nenhum mecanismo de recuperação desta situação parece viável.

Loops de roteamento podem ser criados através da apropriacão de atualizacões de roteamento alheias (*spoofing*). Por exemplo, se um nó malicioso determina que dois nós A e B possuem alcance de radio. O nó malicioso envia uma mensagem de roteamento com o endereco de A como origem para B, que marca o nó A como seu pai. Quando o nó A receber a atualizacao do nó B, ira marcar B como seu pai. A partir disso, as mensagens ficam sendo enviadas de A para B e vice-versa, em um *loop*.

Direct Diffusion

O *Direct Diffusion* [55] é um algoritmo de roteamento orientado a dados para extrair informações dos nós sensores. O sorvedouro envia `interesses` para tipos de dados nomeados, criando `gradientes` na rede destinados a obter os eventos (isto é, dados combinando com os `interesses`). Os nós que possam atender aos `interesses`, disseminam informações no caminho reverso a propagação do `interesse`. Nós que recebam o mesmo `interesse` de múltiplos nós vizinhos, podem propagar os eventos nos múltiplos enlaces correspondentes. Os `interesses` fluem a uma taxa baixa no início, mas a medida que alcançam o sorvedouro, este pode reforçar uma vizinhança, solicitando uma taxa de dados mais alta. Isto continua recursivamente ate alcançar os nós que geram os eventos, fazendo com que estes aumentem a taxa de dados. Da mesma forma, caminhos também podem ser reforçados negativamente, fazendo com que diminua a taxa de envio de dados.

Observa-se que o *Direct Diffusion* baseia-se na inundação de mensagens para atingir seus objetivos de disseminação de dados. A inundação de mensagens possui natureza robusta, já que múltiplas cópias de uma mesma mensagem são enviadas. Portanto, é difícil para um nó malicioso impedir que os `interesses` alcancem os nós capazes de atendê-los. Assim, neste caso, um adversário tem quatro possíveis objetivos [59]: (1) supressão de fluxo; (2) clonagem de fluxo; (3) influência no caminho; e (4) encaminhamento seletivo e manipulação de dados.

A supressão de fluxo é um ataque de negação de serviço. A maneira simples de atingir o objetivo é através da repetição de mensagens copiadas de reforço negativo (*negative reinforcement spoofing*).

A clonagem de fluxos permite a monitoração dos mesmos. Um nó malicioso pode repetir o envio de um `interesse` de um sorvedouro legítimo, listando-o como sorvedouro. Assim, o nó malicioso também ira receber os dados deste fluxo.

Um nó adversário pode influenciar o caminho ao repetir (*spoof*) mensagens de reforço positivo e negativo, e eventos de dados falsos. Por exemplo, se o nó malicioso quiser desviar o fluxo de dados de modo a estar na rota, ele irá reforçar o evento positivamente para os nós que enviou o `interesse`, enquanto envia reforços negativos

aquele de quem recebeu o `interesse`. Dessa forma, um fluxo de dados legítimo será desviado através do adversário, pois este reforçou o caminho com taxas de dados mais altas. Assim, através da manipulação dos reforços positivos e negativos, um nos malicioso consegue determinar quais mensagens serão encaminhadas e manipular os dados.

Roteamento Geográfico

Geographic and Energy Aware Routing (GEAR) [115]) é um protocolo de roteamento que utiliza informação de posição dos nós para destinos geográficos para os pacotes para disseminar consultas e respostas de rotas. O GEAR decide o próximo salto na rota utilizando duas métricas: distância do alvo e energia disponível nos próximo nó. Assim, um fluxo não é roteado por um único caminho, drenando a energia dos nós nesta rota, mas está num "feixe" da origem ao sorvedouro.

De acordo com Karlof e Wagner [59], o GEAR pode sofrer influência na rota, se a informação de localização for informada erroneamente. Independente da localização de um nó malicioso, este pode anunciar a sua posição como sendo parte um fluxo conhecido. Ainda, o nó malicioso pode anunciar que possui energia máxima, o que o tornaria a primeira escolha para encaminhar pacotes.

Um nó malicioso pode criar um ataque de nós irmãos anunciando a presença de múltiplos nós falsos perto de um nó real, todos eles anunciando máxima energia. Assim, o nó malicioso é escolhido como próximo salto, e pode manipular as mensagens e/ou monitorar seu conteúdo.

Contra-medidas

De acordo com Karlof e Wagner [59], os mecanismos para prover proteção a ataques de falsificação de mensagens, de nós irmãos, inundação de mensagens e repetição de reconhecimentos, são: (1) autenticação e criptografia na camada de enlace; (2) roteamento multi-caminho; (3) verificação de identidade; (4) *broadcast* autenticado. Estes mecanismos podem ser aplicados a protocolos existentes. No entanto, ataques usando *wormholes* e *sinkholes* são mais complexos e devem ser abordados quando do projeto dos protocolos de roteamento.

Hu e Sharma [54] recomendam os seguintes itens para obter roteamento seguro:

- **Uso de criptografia simétrica para autenticação de mensagens de roteamento e descoberta de rotas**: a criptografia simétrica consome menos recursos de processamento e memória que a criptografia assimétrica, conforme apresentado nas seções 4.3.1 e 4.4.1.
- **Integrar roteamento com múltiplos caminhos e roteamento seguro**: O uso de rotas redundantes provê tolerância a falhas e disseminação de dados confiável.
- **Esquemas de roteamento devem ser *tolerantes a intrusão* e não *detectores de intrusão***: Determinar se existem nós intrusos em uma RSSF rapidamente não é

simples. Krontiris et al. [61] apresentam um dos primeiros trabalhos de detecção de intrusão cooperativa em RSSF, mostrando a viabilidade do esquema para um nó intruso. Se os protocolos de roteamento forem tolerantes a intrusão, o dano causado por ataques deveria ficar contido em uma região e não se propagar pela rede.

- **Uso de modelo de confiança localizado ao invés de gerenciamento de segurança centralizado**: Modelos de confiança globais e centralizados são caros e complexos para RSSF. Devido a característica da comunicação em RSSF (colaboração entre nós vizinhos, *broadcasts*), tipicamente um nó precisa confiar em seus vizinhos, e assim um modelo localizado seria mais eficiente.

- **Reduzir o *overhead* de roteamento**: a redução de mensagens de controle de roteamento reduz o custo de comunicação entre os nós, consumindo menos energia. Ainda, como estas mensagens devem utilizar mecanismos de segurança, seu uso excessivo implica em mais processamento para autenticar e/ou cifrar seu conteúdo.

4.2.4. Localização de nós segura

A capacidade de integrar informações espaciais ao dados coletados pelos nós sensores é importante para aplicações como rastreamento (*tracking*) de objetos ou a monitoração de características de um ambiente. Para maioria das aplicações, a localização melhora o valor da informação obtida, sendo necessária para o roteamento geográfico. Se os nós sensores possuírem um módulo GPS (*Global Positioning System*), é trivial obter a informação de localização. Da mesma forma, se os nós foram colocados manualmente em coordenadas pré-determinadas, a sua localização também é conhecida. Caso contrário, é necessário utilizar outros mecanismos de localização.

Segundo Savvides et al. [1], podemos dividir os mecanismos de localização em duas classes: ativa e passiva. Na localização ativa, estão as técnicas que emitem sinais no ambiente para medir alcance ao nó ou alvo, como por exemplo sistemas de radar e sonares refletores, ou elementos da infra-estrutura que emitem sinais que o nó pode receber, como o GPS. Na localização passiva estão as técnicas que monitoram sinais em um determinado canal, como por exemplo elementos conhecidos de *beacon* na infra-estrutura permitem ao nó determinar a sua localização. Após obter estes sinais, o nó deve proceder ao cálculo de sua localização, utilizando triangulação ou trilateração, por exemplo.

Os serviços de segurança necessários a localização segura [2] são:

- **Autenticação**: informação de localização deve ser fornecida somente por fontes autorizadas, e portanto é necessário autenticá-las.

- **Integridade**: as informações fornecidas pelas fontes autorizadas não podem ter sido alteradas para que os nós descubram a sua localização.

- **Disponibilidade**: as informações devem estar disponíveis quando o nó precisar calcular a sua localização.

- **Irretratabilidade**: nem a fonte que provê informação de localização, nem o nó sensor que recebe essa informação deveriam ser capazes de negar a troca de informações posteriormente.

- **Privacidade**: um dos maiores requisitos de segurança é manter sigilo sobre a localização de um nó. A fonte deve ajudar o nó a obter a sua localização, mas nem a posição do nó nem da fonte deveriam se tornar públicas.

Alguns exemplos de técnicas de localização segura são: SeRLoc, SPINE (*Secure Positioning in Sensor Networks*) e DRBTS (*Distributed Reputation and Trust-based Security Protocol*) [2].

O SeRLoc é uma técnica de localização segura, distribuída, resistente a ataques de *wormholes*, nós irmãos e comprometimento de nós. Esta técnica considera que os nós sensores da RSSF possuem antenas omnidirecionais, e que existe um conjunto de localizadores que possuem antenas direcionais. Cada localizador transmite *beacons* que contém duas partes de informação: as coordenadas do localizador e o ângulo das linhas de fronteira da antena em relação a um eixo global. O SeRLoc possui restrições de setor exclusivo e alcance de comunicação, que ajudam a evitar ataques de *wormhole*. Assim, para comprometer o processo de localização, atacantes precisariam se passar por vários localizadores.

SPINE (*Secure Positioning in Sensor Networks*) é um sistema de posicionamento baseado em alcance e com multi-lateração verificável, permitindo o cálculo seguro da posição e a verificação de posições de nós moveis. Os nós utilizam relógios com precisão de nano-segundos para obter limites da sua distância a pontos de referência próximos. Dada a estimativa de distância a pelo menos três pontos de referência, é possível determinar a localização por multi-lateração. Porém SPINE é centralizado e pode criar gargalos em uma autoridade central ou sorvedouro.

O DRBTS (*Distributed Reputation and Trust-based Security Protocol*) é um protocolo que pretende prover localização segura. Os nós *beacon* monitoram uns aos outros, fornecendo informações aos nós sensores sobre quais *beacons* são confiáveis, usando uma abordagem de eleição. Para que seja considerado confiável um nó *beacon* precisa receber votos de pelo menos metade dos seus vizinhos comuns. À medida que aumenta a densidade de nós, o sistema torna-se mais robusto.

4.2.5. Agregação de dados segura

Um dos principais componentes no consumo de energia de um nó sensor é a comunicação. Assim, uma das principais estratégias é a agregação de dados na RSSF, de modo a minimizar a transmissão de dados redundantes. Solis e Obraczka [106] discutem aspectos importantes na agregação de dados na RSSF, mostrando como a determinação dos períodos de agregação dos dados nos nós pode levar ao atraso do envio dos dados ao sorvedouro. Além desse aspecto, que pode ser observado pelo aspecto de que o sorvedouro deve receber dados recentes (*data freshness*), é importante garantir a integridade dos dados. Dependendo da aplicação sendo executada na RSSF, a confidencialidade também é importante.

Alzaid et al. [4] fazem uma pesquisa sobre segurança na agregação de dados, e propõem uma classificação dos esquemas existentes, apresentada na Figura 4.4. A primeiro classificação é com relação ao número de nós que realizam a agregação de dados dentro da RSSF, antes dos dados atingirem o sorvedouro. São duas as possibilidades: um

único nó agregador ou múltiplos nós agregadores. A segunda classificação é com relação a existência ou não da fase de verificação, que está associada a integridade dos dados sendo agregados.

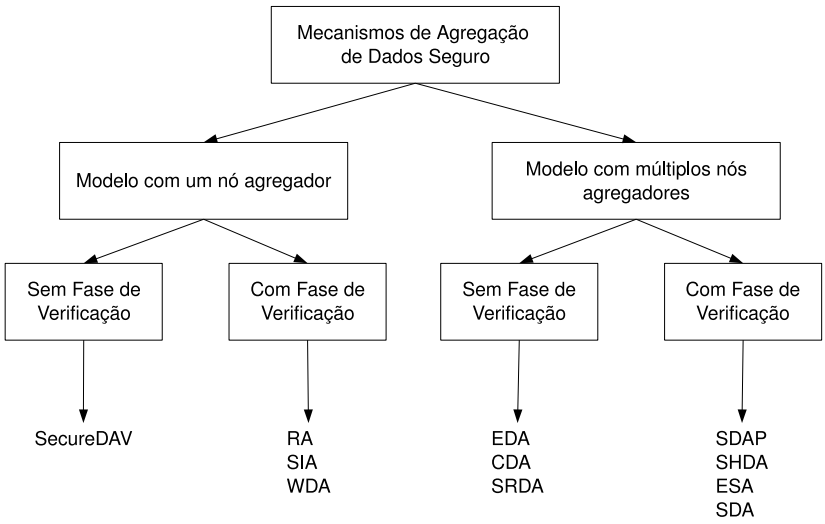

Figura 4.4. Classificação de mecanismos de agregação de dados seguros.

Esquemas com um único nó agregador se aplicam a RSSF pequenas, ou caso o resultado de uma consulta deva ser enviado a um nó externo a RSSF. Este nó agregador precisa de recursos computacionais e mais energia que os demais nós para agregar os dados neste modelo. O caso de múltiplos nós agregadores se aplica a RSSF maiores, onde vários ou todos os nós podem realizar a agregação de dados. A fase de verificação permite que o nó que realizou a consulta a RSSF consiga verificar se todos os dados agregados são válidos. Esta verificação é mais complexa quando existem múltiplos nós agregadores na RSSF.

De acordo com a classificação apresentada na Figura 4.4, os principais esquemas de agregacao segura incluem: SecureDAV, RA (*Resilient Aggregation*), SIA (*Secure Aggregation Information*), WDA (*Witness Based Data Aggregation*), EDA (*Encryption Data Aggregation*), CDA (*Concealed Data Aggregation*), SRDA (*Secure Refrence-based Data Aggregation*), SDAP (*Secure hop-by-hop Data Aggregation Protocol*), SHDA, ESA e SDA (*Secure Data Aggregation*) [4]. As principais características de alguns destes esquemas são descritos a seguir.

SDA (*Secure Data Aggregation*) utiliza múltiplos nós agregadores com fase de verificação. As medidas obtidas por um nó são encaminhadas aos próximos saltos, e são verificadas e agregadas no segundo salto. O nó sensor precisa armazenar as medidas para autenticá-lo com uma chave compartilhada com o sorvedouro. O esquema provê integridade de dados, autenticação e dados recentes. Porém, se nós pais e filho forem comprometidos, a integridade pode ser quebrada.

O ESA é um extensão do SDA, onde os autores propõe o uso de chaves para a co-

municação entre pares de nós vizinhos (um salto) e chaves para pares de nós a dois saltos de distância. Isso elimina a necessidade de armazenar dados para verificar a autenticidade com a chave do sorvedouro, além de permitir o uso de criptografia na comunicação entre os nós, garantindo a confidencialidade.

O SecureDAV melhora a garantia de integridade do SDA e EDA ao assinar os dados agregados. Este esquema baseia-se em *cluster*, onde existe uma chave secreta compartilhada para verificação dos dados agregados. Se o nó concordar com o media divulgada pelo nó agregador, que é o *cluster-head*, ele assina o valor agregado. O nó agregador combina as assinaturas para gerar uma assinatura total do *cluster* do dado agregado, que é enviado ao sorvedouro.

SIA (*Secure Aggregation Information*) é um *framework* com três fases: (1) coletar os dados dos sensores e agregar localmente, (2) confirmar os dados e reportar o resultado da agregação ao sorvedouro e (3) provar que o resultado está correto. Nesta arquitetura, cada nó compartilha uma chave secreta com o sorvedouro e com o nó agregador, que permite verificação de autenticidade e confidencialidade.

O WDA (*Witness Based Data Aggregation*) é um esquema para garantir a validação dos dados enviados por um nó agregador. Para esta prova, o nó agregador envia diversas provas de testemunhas ao sorvedouro. As testemunhas também são nós agregadores, mas que não enviam seus resultados ao sorvedouro. Ao invés disso, elas calculam o MAC (código de autenticação de mensagem) do resultado dos dados agregados, e enviam o MAC ao nós agregador como prova. Este esquema garante somente a integridade dos dados agregados. Observe que tanto o o WDA e o SecureDAV não garantem que os dados são recentes (*data freshness*) [4].

Mais recentemente, Castelluccia et al. [20] propõe um esquema de criptografia que permite a agregação aditiva de dados cifrados. A agregação baseada neste esquema pode ser usada para cálculo de valores estatísticos, como a média, variância e desvio padrão dos dados coletados pelos nós sensores.

Observe que, assim como no caso do roteamento seguro e dos mecanismos de localização segura, são necessários esquemas que façam uso de autenticação e criptografia dos dados, verificação de identidade das partes envolvidas, e garantia de que os dados são recentes para obter a agregação segura dos dados.

4.3. Criptografia Simétrica

Uma *cifra simétrica* (ou simplesmente *cifra*) é um tipo de algoritmo criptográfico reversível utilizado para garantir confidencialidade aos dados. Durante o processo de *encriptação*, uma cifra transforma uma mensagem legível M em uma mensagem cifrada C usando uma chave secreta K; o processo inverso é denominado *decriptação*.

Cifras de bloco operam apenas sobre conjuntos de dados tendo um tamanho definido, n, que é denominado o *tamanho de bloco*. Para processar mensagens menores do que n, técnicas de *padding* [87] são necessárias, pelas quais alguns bits são adicionados à mensagem até que um bloco seja completado. Mensagens maiores do que n são processadas bloco a bloco, de acordo com um determinado modo de operação; e.g., no modo CBC [87], cada bloco cifrado é combinado com o seguinte por meio de uma operação

de ou-exclusivo (XOR), e apenas então o resultado é encriptado novamente. É comum encontrar cifras de bloco que adotam uma estrutura interativa, i.e., compostas por sub-operações (denominadas *rounds*) que se repetem um certo número de vezes. Este é o caso, por exemplo, do Rijndael [34], escolhido como o atual AES (*Advanced Encryption Standard* – ou "Padrão Avançado de Encriptação") [86] e adotado mundialmente. Cada round utiliza uma sub-chave gerada a partir da chave secreta inicial, no processo conhecido como *escalonamento de chaves*.

Cifras de fluxo, por outro lado, produzem uma sequência pseudo-aleatória de bits que são combinados com a mensagem legível via XOR ou outra operação simples. Assim, não há restrição no tamanho das mensagens que podem ser processadas, nem necessidade de *padding* ou um modo de operação. Apesar de cifras de fluxo serem potencialmente mais rápidas do que cifras de bloco, a arte de projetar cifras de bloco é atualmente melhor dominada, o que costuma motivar uma mais ampla adoção destas últimas. De fato, quando um comportamento de fluxo é desejado (e.g., para prevenir a expansão das mensagens causada por *padding*), é comum usar uma cifra de bloco em um modo de operação que emule o comportamento de cifras de fluxo (e.g., CTR [87]).

4.3.1. Uso de cifras em RSSF

Redes de sensores possuem diversas peculiaridades que devem ser consideradas quando da escolha de algoritmos criptográficos. Isto acontece porque a maioria das cifras consideradas robustas atualmente foram desenvolvidos para propósitos gerais, adaptando-se a diversos cenários. Assim, elas não necessariamente respondem de forma otimizada às limitações de memória, energia e capacidade de processamento inerentes aos dispositivos utilizado em RSSF.

Um ponto importante diz respeito à possível expansão de mensagens causada por técnicas como *padding*. É altamente recomendado que tal expansão seja evitada, mesmo que isto signifique um aumento na quantidade de processamento necessário, já que a transmissão de 1 bit requer uma quantidade de energia equivalente à execução de 800-1000 instruções [57]. Portanto, a adoção de uma cifra de fluxo – ou, equivalentemente, de um modo de operação em fluxo para cifras de bloco – ou de técnicas que previnam o aparecimento deste problema (e.g., *Ciphertext Stealing* [82]) devem ser considerados.

No caso de cifras de bloco, o próprio tamanho de bloco deve ser escolhido com cuidado, já que as mensagens trocadas entre nós – bem como entre nós e centrais de processamento – costumam ser inferiores a 60 bytes [26], sendo 24 bytes muitas vezes considerado um tamanho típico [84]. Desta forma, a adoção de cifras que operam sobre blocos muito grandes acaba levando a um "desperdício" de energia decorrente da maior quantidade de processamento envolvida e da possível expansão das mensagens encriptadas (dependendo do modo de operação utilizado). De fato, o tamanho de bloco do AES, de 128 bits, é por vezes considerado muito grande para uso em RSSF [99].

Por outro lado, devido à reduzida largura de banda apresentada pelos sensores, o número de mensagens disponíveis para um atacante é bem menor do que o normalmente encontrado em redes convencionais, o que permite a adoção de algoritmos criptográficos um pouco menos conservadores [57].

Atualmente, diversas soluções de segurança voltadas a RSSF (e.g., TinySec [57], Sensec [69] e Minisec [74]) adotam como padrão o Skipjack, uma cifra de bloco convencional que apresenta um bom desempenho mas fornece uma reduzida margem de segurança[3]. No entanto, em razão das particularidades inerentes a estas redes, bem como à sua crescente importância, diversos trabalhos têm sido desenvolvidos tanto no sentido de identificar as cifras mais adequadas para este contexto quanto visando ao desenvolvimento de soluções dedicadas.

A seguir, serão apresentadas algumas cifras desenvolvidas especificamente para redes com recursos limitados. Em seguida, serão discutidos alguns resultados da literatura que avaliam o desempenho destas e de outras cifras.

4.3.2. Cifras Dedicadas: Estado-da-Arte

Existem atualmente diversas soluções criptográficas dedicadas a sistemas embarcados. Exemplos de propostas recentes incluem CURUPIRA [104], PRESENT [15], HIGHT [53], SEA [107], mCrypton [70], Trivium [36] e Grain [51], cujas características são resumidas na Tabela 4.2.

Tabela 4.2. Características de algumas cifras dedicadas.

Cifra	Tipo	Tamanho de Bloco (bits)	Tamanho de Chave (bits)
CURUPIRA	Bloco	96	96/144/192
PRESENT	Bloco	64	80/128
HIGHT	Bloco	64	128
$SEA_{n,b}$	Bloco	$n = (6b)^{\alpha}, \alpha > 0$	n
mCrypton	Bloco	64	64/96/128
Grain	Fluxo	1	80/128
Trivium	Fluxo	1	80

O projeto destes algoritmos leva em consideração as restrições de recursos inerentes a estas plataformas. Assim, é comum o uso de operações simples, fazendo com que estas cifras sejam potencialmente mais eficientes e compactas do que algoritmos de uso geral. Adicionalmente, todos os algoritmos desta tabela são considerados seguros, no sentido que a forma mais eficiente de ataque contra eles é por meio de força bruta. De fato, todos levam em consideração formas avançadas de criptanálise em seu projeto, apesar de ainda não terem sido tão amplamente analisados como o AES e outros algoritmos amplamente utilizados. Por outro lado, o nível de segurança oferecido por elas costuma ser menor do que o usualmente encontrado em redes convencionais, já que o objetivo é fornecer segurança suficiente para suprir as necessidades de RSSF. Conforme pode ser observado na Tabela 4.2, os tamanhos de bloco e chave são em sua maioria menores do que aqueles do AES (bloco: 128 bits, chave: 128, 192 ou 256 bits), por exemplo.

O CURUPIRA é uma cifra de bloco de 96 bits projetada de acordo com metodologia conhecida como Estratégia de Trilha Larga (ETL) [33], a mesma utilizada no AES, a qual é reconhecida pela sua resistência a diversas modalidades de criptanálise moderna (e.g., ataques linear [79] e diferencial [12]). De fato, apesar do menor tamanho de bloco,

[3]O Skipjack usa chaves criptográficas de 80 bits, por vezes considerado o tamanho mínimo para cifras modernas, e possui baixa margem de segurança contra ataques, dado que 31 de seus 32 rounds podem ser criptanalisados com sucesso [11].

a estrutura de sua função de round é bem semelhante à do AES, apresentando o mesmo tipo de transformações lineares, não-lineares e de adição de chave. As operações básicas do algoritmo são orientadas a bytes, incluindo fundamentalmente XOR, shift de bits e indexação de tabelas (pode ser usada uma única tabela de 256 bytes para maior eficiência em software, ou então duas tabelas de 16 bytes para implementações rápidas e compactas em hardware). Um ponto importante é que todas as transformações utilizadas na função de round são involuções (i.e., auto-inversas), de modo que a encriptação e decriptação são idênticas exceto pela sequência de chaves utilizadas. Deste modo, o mesmo algoritmo pode ser usado em ambos os processos, levando a uma estrutura mais compacta tanto em software quanto em hardware.

A operação do CURUPIRA envolve 10, 14 ou 18 rounds, respectivamente para os tamanhos de chave de 96, 144 ou 192 bits. Duas versões do algoritmo de escalonamento de chaves foram propostas: o CURUPIRA-1 [6] é mais conservador, adotando um processo de escalonamento também baseado na ETL; já no CURUPIRA-2, as sub-chaves são geradas por meio de um LFSR (*Linear Feedback Shift Register*, ou "Registrador de Deslocamento Linear com Retro-alimentação"), tendo como foco um desempenho superior e menor tamanho de código. Em ambos os casos, as sub-chaves podem ser calculadas sob demanda, de modo a reduzir o uso de memória RAM pelo algoritmo.

O PRESENT é uma cifra de bloco com 32 rounds, tamanho de bloco de 64-bits e tamanho de chave de 80 ou 128 bits. O seu projeto tem como objetivo principal prover uma solução altamente compacta em hardware e, ao mesmo tempo, com um desempenho comparável àquele de cifras de fluxo modernas. De fato, esta cifra se utiliza de algumas estruturas inspiradas no Serpent [10], uma cifra de bloco de uso geral reconhecida pelo seu excelente desempenho em hardware. Em cada round, são executadas operações bastante básicas em hardware, como XOR, indexação de tabelas de nibbles (estruturas de 4 bits) e permutação bit-a-bit; por outro lado, estas duas últimas operações não costumam ser tão eficientes em software [72].

HIGHT é uma cifra de bloco que opera sobre blocos de 64 bits e adota chaves de 128 bits. Ela foi desenvolvida para uso em dispositivos altamente limitados, como etiquetas de RFID, devido ao reduzido número de componentes necessários para sua implementação em hardware. No entanto, sua estrutura é bastante simples para implementação eficiente em software, contando com operações como XOR, adição módulo 256 e rotação bit-a-bit. Basicamente, a cifra consiste em uma transformação inicial, 32 rounds usando 4 sub-chaves cada, uma transformação final e um algoritmo de escalonamento de chaves responsável por produzir as 128 sub-chaves necessárias. Um ponto importante com relação a este algoritmo de escalonamento é que o mesmo apresenta um comportamento cíclico, i.e., a chave original é recuperada ao final da operação de encriptação/decriptação; esta característica facilita bastante a computação das sub-chaves sob demanda.

A cifra de bloco $\text{SEA}_{n,b}$ (de *Scalable Encryption Algorithm* – "Algoritmo de Encriptação Escalável") opera sobre blocos de tamanho variável, n, tendo como única restrição que n seja múltiplo de b, o tamanho das palavras sobre as quais trabalha o processador (e.g., $b = 8$ no caso de bytes). O tamanho de chave utilizado também é n, enquanto o número de rounds mínimo para evitar ataques criptanalíticos modernos é calculado como o menor número ímpar n_r tal que $n_r \geqslant 3n/4 + n/b + 2\lfloor n/2 \rfloor$. As operações de encriptação e

decriptação adotam rotinas compactas e destinadas a processadores com uma conjunto de instruções limitado (basicamente, funções AND, OR, XOR, rotação e adição modular). Além disto, o escalonamento de chaves adotado pelo $SEA_{n,b}$ é cíclico e idêntico tanto na encriptação quanto na decriptação, de modo que o mesmo algoritmo de escalonamento pode ser utilizado em ambos os processos. No entanto, em comparação com outras cifras, o número de rounds necessário para prover um nível razoável de segurança é bastante elevado.

A cifra de bloco mCrypton [70] opera sobre blocos de 64 bits e aceita chaves de 64, 96 ou 128 bits. A operação do algoritmo envolve 12 rounds compostos de transformações bastante simples e orientadas a nibbles, como indexação de tabelas (usando quatro tabelas não-lineares, de 16 nibbles cada), XOR e AND. Já o algoritmo de escalonamento de chaves consiste basicamente em indexação de tabelas de nibbles (usando apenas uma das quatro tabelas disponíveis) e rotações (tanto sobre palavras de 16 bits quanto bit-a-bit). Algumas das transformações usadas nos rounds da cifra são involuções, permitindo o reuso de partes do algoritmo de encriptação durante o processo de decriptação e, desta forma, reduzindo o espaço ocupado pela cifra completa.

Grain e Trivium são duas cifras de fluxo orientadas a hardware. Ambos utilizam chaves de 80 bits e apresentam uma estrutura bastante simples, tendo sido escolhidos para fazer parte do conjunto de cifras de fluxo selecionadas pelo projeto europeu ECRYPT[4]. O Grain consiste basicamente na interação entre dois registradores de deslocamento com retro-alimentação (um linear e outro não-linear) de 80 bits, os quais geram a sequência pseudo-aleatória de bits a partir da chave inicial e de um vetor de inicialização (IV) de 64 bits. Já o Trivium requer um IV de 80 bits, o qual é utilizado juntamente com a chave para inicializar um registrador interno de 288 bits; durante a operação da cifra, os bits deste registrador são iterativamente selecionados tanto para a geração da sequência de bits para encriptação/decriptação de mensagens, quanto para a atualização do próprio registrador.

4.3.3. Análise de Desempenho

A literatura inclui diversos trabalhos cujo objetivo é identificar as cifras mais adequadas para uso em RSSF. Estes trabalhos são muitas vezes bastante distintos em termos de metodologia, plataforma, métricas e foco da análise, o que dificulta uma comparação direta entre os resultados obtidos. Entretanto, o estudo dos mesmos é bastante instrutivo no sentido de "filtrar" grupos de algoritmos com potencial para satisfazer aos requisitos de segurança e eficiência de uma aplicação específica. Com este intuito em mente, discutiremos a seguir alguns trabalhos voltados à avaliação tanto a cifras convencionais quanto dedicadas.

Um estudo bastante completo sobre cifras de fluxo convencionais é apresentado em [45]. Os dados medidos são o tempo necessário e energia consumida pelos processos de inicialização e encriptação de mensagens de diferentes tamanhos, bem como a quantidade memória (RAM e flash) ocupada pelos algoritmos. A plataforma escolhida é uma placa de desenvolvimento equipada com um processador ARM922T[5]. A análise dos resultados obtidos para as 12 cifras estudadas permite identificar que algumas apresentam

[4]http://www.ecrypt.eu.org/

[5]http://www.arm.com/products/CPUs/ARM922T.html

um bom potencial para uso em RSSF, em especial o Snow v2.0 [43].

Já em [66], Law et al. desenvolve um estudo bastante completo de cifras de bloco convencionais. Neste caso, as medidas são realizadas em microcontrolador MSP430F149 [110], da Texas Instruments. A discussão concentra-se principalmente na análise de segurança das mesmas e na avaliação da memória ocupada e do desempenho considerando diferentes modos de operação. Como resultado desta análise, os autores propõem o uso de diferentes cifras para diferentes combinações de requisitos de segurança e memória. Mais especificamente, os autores concluem que o Skipjack [90] é o mais recomendado para aplicações com reduzida necessidade de segurança; já em cenários onde o nível de segurança deve ser elevado, os autores sugerem o uso do MISTY1 [80] e o AES quando há, respectivamente, reduzida e elevada disponibilidade de memória.

Um outro estudo interessante é apresentado por Strydis et al. em [109], cujo foco principal é o uso de criptografia em aplicações biomédicas usando dispositivos limitados (e.g., implantes dotados de sensores biométricos). Neste trabalho, um total de 13 cifras de bloco são avaliadas de acordo com as seguintes métricas: tamanho de código, velocidade do processo de encriptação, segurança, consumo máximo e médio de energia e de potência. Os dados são medidos usando a ferramenta XTREM [25], um simulador de desempenho e consumo de energia para o processador embarcado XScale, da Intel [56]. Os resultados indicam o MISTY1 como a cifra com melhor pontuação considerando as métricas utilizadas, enquanto o IDEA [65, 81] e o RC6 [100] também se mostraram interessantes. Por outro lado, os autores parecem ter utilizado uma versão muito pouco otimizada do Skipjack, já que os resultados do mesmo em termos de desempenho e eficiência energética mostraram-se muito inferiores àqueles das outras cifras analisadas, contrariando análises similares em plataformas limitadas [66, 72].

Um outro estudo de interesse é aquele apresentado por Großschädl et al. em [47]. Este trabalho também é voltado a cifras convencionais, mas tem como objetivo avaliar a eficiência de implementações altamente compactas de alguns algoritmos modernos. Como resultado, os autores concluem que é possível otimizar cifras como AES e RC6 de modo a obter um bom desempenho e reduzido consumo de energia em plataformas limitadas. Todavia, uma limitação deste trabalho é que não são consideradas algumas cifras reconhecidas pelo seu alto desempenho e reduzida necessidade de memória (e.g., Skipjack, IDEA ou MISTY1), dificultando uma avaliação mais abrangente dos resultados obtidos.

Já com relação a cifras dedicadas, uma análise bastante relevante é apresentada em [42], no qual são consideradas implementações em software e hardware de diversos algoritmos. Dentre as cifras consideradas, aquelas que ocupam uma menor área de chip são Grain e PRESENT e a que provê maior vazão é o mCrypton; em comparação, o HIGHT ocupa cerca de duas vezes mais espaço do que PRESENT, fornecendo uma vazão ligeiramente inferior a este último. Nos testes com as implementações em software, a vazão obtida pelo HIGHT é elevada, apesar de não superar a do IDEA; entretanto, os autores afirmam não terem sido capazes de obter uma implementação compacta do HIGHT, de modo que o espaço ocupado por ele chega a ser o dobro daquele usado pelo AES. Já o PRESENT apresenta um código bastante compacto e baixo desempenho, enquanto o SEA não se sobressai em nenhum destes dois quesitos.

Outro trabalho recente que aborda cifras dedicadas, comparando-as com algoritmos convencionais, é aquele apresentado em [72]. Neste estudo, PRESENT, HIGHT e SEA mostram-se menos interessantes do que o Skipjack em termos de velocidade e consumo de energia; no entanto, o tamanho de código destas cifras dedicadas é bastante inferior ao do Skipjack.

O artigo no qual o PRESENT é apresentado [15] também inclui uma seção que avalia o desempenho de algumas cifras dedicadas, apesar de restringir-se a implementações em hardware. Esta análise mostra que o mCrypton atinge um bom desempenho, superando consideravelmente cifras como HIGHT, AES e o próprio PRESENT; por outro lado, o PRESENT apresenta-se como uma solução mais compacta.

Finalmente, em [105], desenvolvemos nossa análise do desempenho do CURUPIRA em diferentes plataformas. O resultado é que, especialmente em sua segunda versão (CURUPIRA-2), esta cifra dedicada chega a ser mais rápida do que implementações bastante otimizadas do Skipjack e do AES em plataformas limitadas, além de apresentar um custo reduzido de memória.

4.4. Mecanismos de Autenticação de Mensagens

Algoritmos de *MAC* (*Message Authentication Code*, ou "Códigos de Autenticação de Mensagens") são funções irreversíveis que geram uma saída de tamanho fixo, o chamado *tag de autenticação*, a partir de uma entrada de tamanho arbitrário e uma chave secreta. Seu uso é feito da seguinte forma: em primeiro lugar, o emissor da mensagem gera o tag de autenticação usando a chave K compartilhada com o destinatário da mensagem. Ao receber o conjunto mensagem+tag, o receptor calcula localmente o tag da mensagem e compara com o valor recebido. Caso os valores sejam idênticos, a mensagem é aceita. Caso contrário, a mensagem ou o tag recebido foi modificado durante a transmissão, ou então o par foi gerado por um usuário que desconhece K e tentou "forjar" uma mensagem. Portanto, o uso de algoritmos de MAC garante que apenas usuários autorizados criem e verifiquem a autenticidade das mensagens trocadas.

Quanto maior o tamanho do tag utilizado, mais difícil se torna a tarefa de forjá-lo, mas também maior é o tamanho das mensagens trocadas. Assim, a escolha do tamanho mais adequado depende dos requisitos da aplicação: há normalmente um limite superior para o tamanho do tag que pode ser produzido por um determinado algoritmo de MAC, enquanto tags menores podem ser obtidos truncando o tag original.

Em muitas aplicações, confidencialidade e autenticação são ambos serviços essenciais. Uma forma simples e segura de prover ambos os serviços é simplesmente usar um MAC e uma cifra seguros de forma independente, com chaves distintas[6]. Por outro lado, existem algoritmos que combinam ambos os serviços sob uma única chave secreta, simplificando as tarefas de gerenciar, distribuir e armazenar chaves. Estas soluções são conhecidos genericamente como esquemas de *AE* (*Authenticated Encryption*, ou "Encriptação Autenticada"). A maior parte destes esquemas permite a autenticação de dados encriptados (confidenciais) e não-encriptados (e.g., um cabeçalho que deve ser mantido

[6]É importante ressaltar que a utilização de uma mesma chave para ambos os processos costuma levar a problemas de segurança graves [13].

às claras devido a seu uso durante o roteamento de pacotes). Algoritmos com esta característica são conhecidos como esquemas de *AEAD – Authenticated Encryption with Associated Data*, ou "Encriptação Autenticada com Dados Associados", onde os "dados associados" correspondem à porção da mensagem deixada às claras.

O projeto de esquemas de AE normalmente envolve um MAC e uma cifra internos, instanciados com a mesma chave e com um certo vetor de inicialização (IV). Apesar da grande variedade de estratégias de construção, existem essencialmente duas classes de algoritmos de AE: quando duas passagens são feitas pela mensagem (uma para encriptá-la e a outra para autenticá-la), o esquema é chamado *Two-Pass* ("Duas Passagens"); quando uma única passagem é feita, tem-se um esquema *One-Pass* ("Uma Passagem").

A seguir, vamos discutir o uso destas soluções em RSSF, e apresentaremos algumas construções de MACs e esquemas de AEAD especialmente promissoras para uso nestas redes.

4.4.1. Autenticação de Mensagens e Encriptação Autenticada em RSSF

A autenticação de mensagens costuma ser um requisito importante em diversas aplicações de RSSF, já que a introdução de dados falsos na rede podem levar a ações equivocadas e potencialmente graves (e.g., um diagnóstico incorreto da saúde um paciente em uma aplicação médica). Por esta razão, a maioria das arquiteturas de segurança voltadas a RSSF proveem algum tipo de mecanismo de autenticação, mesmo quando a encriptação dos dados não é necessária. Por exemplo, o TinySec usa uma variante do CBC-MAC [85, 8] para autenticação de dados, que podem ou não ser encriptados usando uma cifra de bloco em modo CBC (perceba que duas chaves distintas são necessárias neste caso); já o Minisec adota o Offset CodeBook (OCB) [62], um esquema de AEAD. Em ambos os casos, o tamanho do tag de autenticação é 4 bytes.

Em plataformas com recursos limitados, uma estratégia econômica na construção de algoritmos de MAC (e, por extensão, de AEAD) é reutilizar a estrutura de alguma primitiva criptográfica, como um algoritmo de hash ou, mais comumente, uma cifra de bloco. Nestes casos, o espaço extra necessário para implementar a autenticação acaba sendo bastante reduzido (e.g., 10% do espaço usado pela cifra).

Algoritmos convencionais baseados em cifras de bloco costumam chamar a cifra subjacente uma vez para cada bloco da mensagem autenticada, de modo que o custo da autenticação é semelhante ao de encriptar a mensagem; este é o caso de algoritmos de MAC como CBC-MAC, CMAC [88] e PMAC [14], e de esquemas de AEAD como o EAX [9].

Em comparação, a operação de esquemas baseados na estrutura Carter-Wegman [91] faz intenso uso das chamadas "funções de hash universais", estruturas dotadas de propriedades mais simples do que aquelas de primitivas criptográficas. O resultado é que algoritmos que seguem esta estrutura (e.g., GMAC e GCM [89]) podem ser implementados de forma mais eficiente do que soluções convencionais. De fato, é possível chegar a um custo por bloco processado tão baixo quanto 10%–20% de uma encriptação. No entanto, isto ocorre quando o algoritmo é implementado usando tabelas relativamente grandes que, por dependerem da chave utilizada, devem ser armazenadas em RAM sempre que a aplicação

alvo envolver mecanismos de troca de chaves. Em consequência, este tipo de otimização torna-se inviável em dispositivos com baixa disponibilidade de memória RAM, como é o caso da maioria dos sensores comerciais. Por outro lado, implementações mais compactas geralmente não apresentam um desempenho muito superior àquele de algoritmos convencionais, o que faz com que esta estratégia seja pouco atrativa em RSSF.

Já em esquemas de AEAD do tipo *One-Pass*, o custo do processo de autenticação é bastante reduzido, o que torna estas soluções altamente interessantes para uso em RSSF. Apesar disso, todos os esquemas *One-Pass* desenvolvidos até o momento são cobertos por algum tipo de patente, um grande empecilho para sua ampla adoção.

A despeito desta falta de opções de autenticação otimizadas (i.e., com baixo custo de processamento e memória, e livres de patentes) de uso geral que pudessem ser aplicadas em RSSF, existe atualmente um número consideravelmente reduzido de alternativas específicas para este problema. Dentre as soluções existentes, duas são de especial interesse: a estrutura ALRED [35] e o modo CS [103]. A Figura 4.5 mostra um desenho esquemático de ambas as soluções, que são discutidas com mais detalhes a seguir.

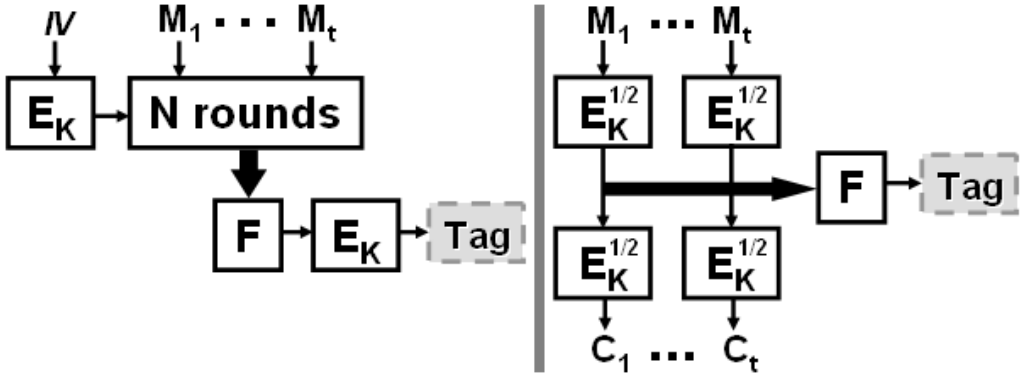

Figura 4.5. A estrutura ALRED (esquerda) e o modo CS (direita). Notação: E_K: encriptação completa; $E_K^{1/2}$ meia encriptação; F: função para combinar resultados parciais; M: mensagem legível (ou cifrada, no caso do ALRED; C: mensagem cifrada.

4.4.1.1. A estrutura ALRED e o Cipher-State

A estrutura ALRED [35] foi originalmente proposta por Daemen e Rijmen para a construção eficiente de MACs baseados em cifras de bloco, especialmente (mas não exclusivamente) quando a cifra escolhida possui uma estrutura semelhante à do SQUARE [32] – e.g., AES e CURUPIRA. Nestes casos, o custo por bloco autenticado gira em torno de 25%–40% do custo de uma encriptação completa. Em plataformas com memória abundante, isto ainda é mais lento do que usar um algoritmo baseado na estrutura Carter-Wegman; entretanto, como a estrutura ALRED requer pouco espaço adicional na memória, ela mostra-se mais adequada para uso em RSSF.

A ideia por trás da construção ALRED é efetuar uma encriptação completa apenas no início e no final do processo de autenticação, enquanto os blocos da cifra são processados por meio de apenas alguns rounds (sem chave) da cifra subjacente. O número

específico de rounds necessários desta forma depende da cifra utilizada, mas costuma ser 4 no caso de cifras da família do SQUARE. Como desvantagem, o número total de mensagens que podem ser autenticadas usando esta estratégia costuma ser inferior ao obtido com algoritmos convencionais, para uma mesma cifra de bloco subjacente.

Um exemplo de MAC baseado na estrutura ALRED é o algoritmo denominado MARVIN, desenvolvido recentemente para atender às particularidades inerentes a plataformas com recursos restritos. Em linhas gerais, este algoritmo consiste na geração de um valor secreto inicial (o resultado da encriptação de uma constante) que alimenta um LFSR; as saídas deste LFSR são combinadas com os blocos da mensagem e passam por alguns rounds da cifra subjacente; finalmente, os resultados parciais são combinados entre si via XOR e encriptados, gerando o tag de autenticação.

Apesar de não ter sido criada explicitamente com este objetivo em mente, a estrutura ALRED pode também ser utilizada na construção de algoritmos de AEAD. Este é o caso do LETTERSOUP, que também é voltado a plataformas limitadas e pode ser visto como uma extensão do MARVIN. De fato, o algoritmo consiste basicamente na encriptação da mensagem usando a cifra subjacente em um modo de operação em fluxo baseado em um LFSR, e na autenticação do resultado (e, possivelmente, de dados associados) usando o MARVIN. Além dos benefícios diretamente herdados da estrutura ALRED, esta solução apresenta alguns atrativos adicionais para uso em RSSF, com destaque para a não-expansão das mensagens, e o uso apenas da do algoritmo de encriptação da cifra subjacente tanto para encriptar quanto para decriptar os dados.

Um detalhe adicional com relação ao MARVIN e ao LETTERSOUP é que ambos apresentam uma sinergia particularmente interessante com o CURUPIRA-2, já que os três algoritmos utilizam um mesmo tipo de LFSR em sua estrutura. Desta forma, é possível reduzir ainda mais o tamanho de código ocupado por estes algoritmos de autenticação quando esta é a cifra subjacente utilizada.

Já o modo *Cipher State* ("Estado da Cifra"), ou simplesmente modo CS [103], desperta interesse por permitir a construção de soluções de AE (não de AEAD) cujo custo de autenticar dados encriptados é extremamente reduzido. A ideia neste caso é explorar a informação parcial do processo de encriptação das mensagens para autenticá-las. Mais especificamente, o valor resultante da aplicação de metade do número total de rounds (e.g., 5 no caso do AES) sobre cada bloco da mensagem é combinado com a saída de um LFSR e adicionado a um acumulador; o tag de autenticação é então calculado a partir da encriptação deste acumulador. Em cenários em que as mensagens trocadas são sigilosas, devendo ser encriptadas, o custo total da autenticação usando o modo CS é mínimo, resumindo-se ao custo de operação do LFSR mais duas encriptações adicionais (uma para inicializar o LFSR e a outra aplicada ao acumulador no final do processo).

Entretanto, o esquema não apresenta apenas vantagens. Primeiramente, ele foi concebido como uma solução de AE, mas não de AEAD. Uma possível expansão do mesmo seria lidar com os dados não encriptados da mesma forma como é feito com os dados sigilosos; neste caso, o custo de processamento dos dados associados seria metade de uma encriptação completa. Isto corresponde a 50% do custo de um algoritmo convencional, porém não chega a ser mais eficiente do que usar uma solução baseada na estrutura ALRED. Além disto, para mensagens cujo comprimento não é múltiplo do

tamanho de bloco, não é previsto qualquer mecanismo que dispense o uso de *padding*, de forma que o modo CS conforme originalmente especificado pode levar à expansão de mensagens. Finalmente, esquemas baseados no modo CS requerem a implementação da função de decriptação da cifra subjacente. O impacto deste fato sobre o tamanho de código do algoritmo poderia ser pouco significativo caso a cifra de bloco adotada possuísse uma estrutura involutiva, como é o caso do CURUPIRA e de outras cifras projetadas para plataformas restritas. Contudo, de acordo com os autores do modo CS, este esquema não deve ser utilizado em conjunto com este tipo de cifra, apesar de não serem dadas explicações detalhadas que esclareçam esta limitação.

Portanto, as vantagens do modo CS quando comparado à estrutura ALRED dependem bastante das características da aplicação alvo, havendo normalmente um compromisso entre tamanho de código e desempenho obtido.

4.4.2. Análise de Desempenho

A literatura inclui um número reduzido de trabalhos que avaliam mecanismos de autenticação no contexto de RSSF.

Em um artigo recente, Bauer et al. [7] faz uma análise do desempenho e da ocupação de memória RAM de quatro esquemas de AEAD: OCB, EAX, GCM e CCFB+H [73], todos usando o AES como cifra de bloco subjacente. A plataforma escolhida para os testes foi um sensor MICAz [28]. Assumindo que a aplicação em questão usa tags de 4 bytes e dados associados de 8 bytes, os autores recomendam a adoção do CCFB+H devido ao seu melhor desempenho e menor uso de memória para diferentes tamanhos de mensagem. Como vantagens adicionais, o algoritmo previne a expansão das mensagens encriptadas, não é coberto por patentes e, segundo os autores da análise, é fácil de implementar. Já o OCB é considerado uma solução atrativa para processar pacotes grandes. Entretanto, este AEAD requer a implementação do algoritmo de decriptação da cifra subjacente e, sendo um AEAD do tipo *One-Pass*, é coberto por patentes (ao menos nos EUA). Finalmente, o EAX mostra-se pouco menos eficiente do que CCFB+H e OCB, permanecendo como alternativa possível, enquanto a adoção do GCM é fortemente desaconselhada pelos autores da análise em razão do seu desempenho muito inferior.

Tendo em vista a dificuldade de encontrar trabalhos semelhantes avaliando o desempenho de algoritmos de MAC, decidimos desenvolver nossa própria análise cobrindo CMAC, PMAC, GMAC e MARVIN. A plataforma de testes escolhida foi um sensor TelosB [30] rodando o TinyOS [67] como sistema operacional, e a cifra de bloco subjacente para todos os algoritmos foi o CURUPIRA-2. Os testes incluíram duas implementações de cada MAC, uma com otimizações voltadas a reduzir o tamanho do código e a outra visando o aumento de desempenho.

Em termos de desempenho, os resultados do PMAC e do CMAC foram semelhantes, sendo ambos os mais eficientes para a autenticação de mensagens pequenas, com até 12 bytes (i.e., um único bloco). Para mensagens maiores do que 12 bytes, o MARVIN mostrou-se mais eficiente. Finalmente, o tempo necessário para a autenticação com o GMAC foi cerca de 10 vezes inferior àquele dos outros algoritmos testados, mais uma vez comprovando a baixa eficiência da estrutura Carter-Wegman em cenários com me-

mória limitada[7]. Estes resultados mantiveram-se consistentes para ambas as versões de otimização utilizadas.

O comportamento dos algoritmos com relação a consumo de energia foi semelhante ao de seu desempenho, porém não exatamente igual: PMAC e CMAC tiveram um consumo de energia inferior ao do MARVIN para mensagens com até 24 bytes, sendo finalmente superados por este último para mensagens maiores. O consumo do GMAC, por outro lado, foi bem acima do consumo dos seus pares (e.g., cerca de 8 vezes o consumo do Marvin para mensagens de 24 bytes).

Em suas versões mais compactas, todos os algoritmos apresentaram um tamanho de código semelhante, por volta de 2.5 KiB, com ligeira vantagem para o MARVIN, com 2.4 KiB. Já para as versões com desempenho otimizado, o tamanho de código foi próximo de 3.2 KiB para todos os algoritmos exceto pelo MARVIN, cujo tamanho de código foi de apenas 2.7 KiB. Em ambas as versões, a quantidade de memória RAM utilizada pelos algoritmos ficou entre 130 e 150 bytes, valores correspondentes ao PMAC e ao MARVIN, respectivamente.

4.4.3. Sobre Segurança de Esquemas de AEAD

Uma questão importante com relação aos esquemas de AEAD discutidos anteriormente (e, na verdade, da maioria das soluções de AEAD existentes) é a segurança dos mesmos depende da não-repetição dos IVs sob uma mesma chave.

Os efeitos da repetição de IVs é especialmente grave quando adota-se um processo de encriptação em fluxo. Este é o caso do EAX, GCM e LETTERSOUP, para os quais a encriptação de duas mensagens M_1 e M_2 sob um mesmo IV e chave resulta em textos cifrados C_1 e C_2 satisfazendo a relação $C_1 \oplus C_2 = M_1 \oplus M_2$. Para o modo CCFB+H, que adota uma combinação dos modos CFB e CTR [87], a mesma relação entre mensagens claras e cifradas existe para o primeiro bloco das mensagens, persistindo para blocos subsequentes apenas se todos os blocos anteriores forem idênticos. Já no OCB, apenas o último bloco é encriptado em modo de fluxo (com o intuito de prevenir a expansão do mesmo), de modo que esta relação se mantém para mensagens tendo o mesmo tamanho; para o restante dos blocos, o OCB usa uma variante do modo ECB [87] semelhante àquela adotada pelo CS, de modo que o efeito da repetição de IVs em ambos os casos se restringe à geração de blocos cifrados idênticos a partir blocos claros idênticos. Finalmente, a repetição de IVs em um modo de operação do tipo CBC (como o adotado no TinySec) revela uma quantidade mínima de informação: apenas o comprimento do maior prefixo compartilhado entre eles, em blocos.

A repetição de IVs pode ser evitada por meio do uso de IVs suficientemente grandes. Todavia, esta estratégia deve ser considerada com cuidado em aplicação de RSSF, já que a adição de IVs grandes em cada pacote transmitido levaria inevitavelmente ao um maior consumo de energia e à redução da vida útil dos sensores.

Para evitar este problema, algumas técnicas foram desenvolvidas. Ao invés de enviar o IV completo em cada pacote, emissor e receptor poderiam manter um contador

[7]Na implementação do GMAC, não foram utilizadas tabelas pré-calculadas pelos motivos discutidos na seção 4.4.1.

sincronizado, o qual seria incrementado a cada pacote recebido, e do qual o valor do IV seria obtido. Esta é a estratégia adotada pelo SNEP (Secure Network Encryption Protocol) [96], no qual o valor do contador corresponde àquele do IV utilizado e, portanto, nenhum IV é trocado pelas partes comunicantes. Isto também é feito no MiniSec, cujos pacotes incluem apenas alguns bits do IV com o intuito de facilitar a ressincronização entre contadores quando há perda de pacotes. Também é possível reutilizar alguns campos do cabeçalho dos pacotes como parte dos IVs, como é feito no TinySec e no Sensec: ambos reaproveitam 4 bytes do cabeçalho na construção dos IVs. Finalmente, antes que os IVs sejam repetidos, mecanismos de substituição de chaves devem ser empregados.

4.5. Distribuição de chaves e criptografia assimétrica em RSSF

Um aspecto crucial para o uso de algoritmos baseados em chaves secretas, como é o caso de cifras, MACs e esquemas de AEAD, é a forma como estas chaves são distribuídas. Em dispositivos modernos, isto é feito normalmente por meio de protocolos consideravelmente complexos (e.g., envolvendo diversas trocas de mensagens, mensagens grandes, ou operações com elevado custo computacional), inadequados para uso em RSSF. Por esta razão, o desenvolvimento de soluções eficientes para estes cenários sempre foi considerado um grande desafio.

A literatura conta atualmente com diversas propostas para distribuição de chaves voltadas a RSSF (para uma lista bastante completa, veja [83]). Em função de suas características, estas soluções podem ser agrupadas em três tipos principais [19]: *Esquemas Auto-Regulados*, *Esquemas Arbitrados* e *Esquemas de Pré-distribuição*. A seguir, vamos discutir as características de cada uma destes grupos, dando uma visão geral de como os mesmos funcionam. Antes disto, entretanto, apresentaremos os requisitos que devem ser levados em consideração na avaliação destes esquemas.

4.5.1. Requisitos e Métricas

Esquemas de gerenciamento de chaves costumam ser avaliados por meio de diversas métricas. De acordo com os diferentes (e geralmente conflitantes) requisitos associados a estas métricas, elas podem ser classificadas em três grupos distintos: segurança, eficiência e flexibilidade.

As métricas de segurança estão associadas à capacidade de resistir a ataques. Desta forma, o esquema de gerenciamento de chaves deve não apenas prevenir que entidades monitorando a rede descubram as chaves utilizadas pelos nós, mas também garantir que a captura física de alguns nós tenha um impacto reduzido (ou, idealmente, nulo) nas comunicações envolvendo apenas nós não capturados, fator conhecido como *resiliência* à captura de nós. Além disto, mesmo que alguns nós sejam capturados e a informações em sua memória sejam extraídas, o esquema deve prevenir estes dados sejam usados para criar nós na rede. Uma forma de fazer isto é por meio de mecanismos pelos quais um nó malicioso é identificado e revogado (i.e., os nós legítimos param de comunicar-se com o nó malicioso). Entretanto, para que isto seja possível, o esquema de gerenciamento de chaves deve prover mecanismos que permitam verificar a identidade dos nós de forma segura. Finalmente, quando necessário, entidades confiáveis (e apenas elas) devem ser capazes de atualizar as chaves de alguns ou de todos os nós da rede.

Métricas de eficiência estão diretamente relacionadas com as limitações de hardware dos nós da rede. Idealmente, todos os sensores fisicamente capazes de se comunicar (i.e., que estejam dentro da área de alcance de suas antenas) devem ser capazes de estabelecer uma chave entre si para que esta comunicação se dê de forma segura. Portanto, esquemas realistas de gerenciamento de chaves devem resultar em redes com um bom nível de conectividade e, ao mesmo tempo, não devem causar um impacto significativo no uso de processamento, memória, banda e energia.

Por fim, a flexibilidade destes esquemas é medida pela sua capacidade de ser implementado em uma ampla gama de cenários. Desta forma, em geral é desejável que a eficácia dos mesmos não dependam de informações difíceis de se obter antes que a rede seja efetivamente montada. Por exemplo, em aplicações nas quais os sensores são distribuídos sobre o terreno alvo de forma aleatória, é difícil predizer o posicionamento final de um certo nó, ou mesmo quais nós farão parte de sua vizinhança. Além disto, é importante considerar a escalabilidade do esquema já que, ao longo da vida útil da rede, seu tamanho pode variar de forma bastante dinâmica; portanto, as soluções mais flexíveis são aquelas capazes de suportar redes com um grande número de sensores e, ao mesmo tempo, que permitem a introdução dinâmica de nós na rede sem impactos na sua segurança.

4.5.2. Esquemas de Pré-distribuição

Em esquemas de pré-distribuição, as chaves criptográficas usadas por todos os sensores são carregadas na memória dos mesmos antes que a rede seja montada. Esta estratégia costuma ser capaz de evitar o uso de protocolos complexos para o estabelecimento chaves entre nós, garantindo uma boa eficiência. Além disto, ela costuma levar a uma rede pouco dependente de nós coordenadores, já que todo o material criptográfico necessário para garantir a segurança das comunicações já está presente nos sensores. Por outro lado, a troca de chaves nestes casos costuma ser uma tarefa mais complexa, ou mesmo impossível de ser realizada.

Dois esquemas de pré-distribuição bastante simples são usar uma única chave global em todos os nós ou, inversamente, carregar cada nó com uma chave para cada outro nó da rede. Com a primeira solução, tem-se uma elevada eficiência, mas a segurança oferecida é muito reduzida já que a captura de um único nó comprometeria todas as comunicações da rede; já no segundo caso, obtém-se uma rede altamente segura, mas o custo de memória envolvido inviabiliza sua adoção em redes compostas por muitos sensores.

Por esta razão, foram desenvolvidas técnicas mais escaláveis e oferecendo um nível razoável de segurança. Uma estratégia comumente utilizada é adotar um intermediário entre as duas soluções acima: distribuir um conjunto reduzido de chaves para cada sensor. Um exemplo é o esquema proposto por Eschenauer et al. [44], no qual as chaves são selecionadas aleatoriamente a partir de um conjunto inicial, o que reduz o uso de memória ao custo de conectividade (pares de nós podem não possuir uma chave em comum) e resiliência (a captura de um nó revela diversas chaves sendo utilizadas pela rede). Outro exemplo é aquele proposto por Chan et al. [22], no qual apenas alguns pares de nós recebem chaves compartilhadas, o que reduz ainda mais a conectividade quando comparado ao esquema de Eschenauer et al., mas resolve o problema de resiliência do mesmo.

Existem também esquemas nos quais as chaves são geradas a partir da multipli-

cação de matrizes [40] ou avaliação de polinômios [71], e apenas os coeficientes destas estruturas matemáticas são carregados nos sensores Neste caso, pode-se obter uma rede com elevada conectividade e resiliência, mas a complexidade das operações envolvidas (e.g., multiplicações e exponenciações sobre números grandes) costuma ter um impacto considerável sobre sua eficiência.

Finalmente, a maioria das técnicas de pré-distribuição podem ser combinadas com informação sobre o posicionamento final dos (grupos de) nós, o que geralmente resulta em uma melhor conectividade com um mesmo custo de memória e processamento, apesar de reduzir a flexibilidade do esquema resultante. Exemplos incluem o esquema proposto por Du et al. [39], que se baseia no esquema probabilístico apresentando de Eschenauer et al. [44], e o trabalho de Canh et al. [18], baseado em soluções polinomiais.

4.5.3. Esquemas Arbitrados

Já os esquemas arbitrados dependem de nós especiais, com maior responsabilidade do que simples nós sensores, para estabelecer e gerenciar as chaves da rede. Apesar de este não ser o cenário mais geral, tal abordagem mostra-se bastante atrativa caso entidades com maior poder de processamento já estejam disponíveis na rede, pois elas podem concentrar o ônus de processamento de operações complexas (e.g., revogação de nós). Este costuma ser o caso de RSSF hierárquicas, nas quais as entidades em questão costumam ser estações-base ou *cluster heads* (“líderes de grupo”). O maior risco neste caso é que tais entidades costumam tornar-se um alvo preferencial de ataques, os quais podem afetar parcelas consideráveis da rede caso sejam bem sucedidos.

Um exemplo de esquema arbitrado é o SHELL [114], que se mostra um tanto complexo devido ao uso de diversos tipos de chave (pelo menos sete) e algumas entidades diferentes para o gerenciamento das chaves da rede; por outro lado, esta natureza distribuída das responsabilidades de gerenciamento leva a uma maior resiliência contra captura de nós.

Outra solução interessante é o esquema proposto por Panja et al. em [93], cujo foco são RSSF com vários níveis hierárquicos. Neste esquema, após os uso de uma chave global na inicialização da rede, cada chave é gerada a partir de diversas chaves parciais, de tamanho reduzido. O menor tamanho das chaves usadas desta maneira traz alguns benefícios: facilita tarefas de troca de chaves, reduz a quantidade de processamento e memória utilizada nos nós sensores, etc. Por outro lado, o uso de uma chave global nos estágios iniciais da rede, bem como o reduzido tamanho das chaves nos nós sensores que ficam na base da hierarquia, podem trazer impactos negativos em termos de segurança da rede.

4.5.4. Esquemas Auto-Regulados

Esquemas auto-regulados, por sua vez, utilizam-se de algoritmos criptográficos assimétricos para estabelecer chaves entre pares de sensores de forma dinâmica, após a formação da rede. Esta característica faz com que os mesmos sejam atrativos principalmente em cenários nos quais a topologia da rede altera-se com frequência, por exemplo devido à mobilidade dos nós ou à adição de novos sensores. Além disto, a quantidade de informação armazenada em cada sensor é reduzida (e.g., apenas uma chave pública e privada), a

resiliência é elevada, e pode-se reduzir ou eliminar a existência de pontos centrais confiáveis que se tornariam alvos preferenciais de ataques.

Apesar destas vantagens, o maior desafio neste tipo de estratégia diz respeito à eficiência da maioria dos algoritmos assimétricos conhecidos atualmente. Por exemplo, o uso de protocolos bastante difundidos como os baseados no RSA [101] são reconhecidamente inviáveis, já que sua execução costuma levar vários minutos e consumir uma quantidade considerável de recursos [38]. Por esta razão, durante muito tempo os esquemas auto-regulados foram considerados como uma possibilidade teórica, mas não aplicáveis prática.

Com o recente desenvolvimento de soluções assimétricas bem mais leves do que algoritmos tradicionais, como é o caso da criptografia de curvas elípticas (ECC) [48]), os esquemas auto-regulados passaram a receber uma maior atenção. De fato, de acordo com a recente análise apresentada em [5], o tempo necessário para calcular um emparelhamento (o processo geralmente mais dispendioso neste tipo de solução) em um sensor MICAz pode ser tão baixo quanto 2 segundos.

O potencial do uso de ECC em RSSF é especialmente acentuado quando estas soluções são combinadas como esquemas criptográficos baseados em identidades, nos quais a chave pública do nó corresponde ao seu ID (e.g., seu endereço físico). Este tipo de abordagem dispensa a necessidade de certificados para a autenticação dos nós [92], além de permitir a construção de soluções não-interativas[8] para o gerenciamento de chaves. Por exemplo, usando um protocolo não interativo (e.g., o SOK [102]) diretamente, nós cujas chaves privadas tenham sido geradas por um mesmo *Centro de Distribuição de Chaves (CDC)* podem calcular uma chave compartilhada simplesmente sabendo a identidade de seu interlocutor. Já em cenários cujos nós são gerenciados por diferentes CDCs, soluções apresentando uma eficiência semelhante podem ser obtidas, por exemplo por meio de uma organização hierárquica entre os CDCs [46].

É importante ressaltar que ainda é cedo para dizer que todas as questões de eficiência relacionadas aos esquemas auto-regulados foram resolvidas. No entanto, os avanços recentes nesta área deixa claro o grande potencial desta abordagem, que tende a ocupar um espaço cada vez mais importante dentre as soluções preferenciais de gerenciamento de chaves em RSSF.

4.6. Testes em Plataformas de RSSF

Tipicamente algoritmos e mecanismos de segurança para RSSF são avaliados em função do seu tempo de execução e do seu consumo de energia. O tempo de execução determina o atraso que será inserido na aplicação de RSSF. Por exemplo, se a leitura obtida através do sensor de temperatura for cifrada antes do seu envio, o tempo necessário para a criptografia representa um atraso de processamento. Além disso, a energia consumida com a criptografia deve ser somada a energia necessária para executar as tarefas no nó sensor (por exemplo, leitura do sensor de temperatura e processamento da mesma). Caso a criptografia dos dados necessite de enchimento, causando um aumento no tamanho do

[8]Uma solução não-interativa não exige trocas de mensagens por parte dos nós que desejam estabelecer uma chave.

bloco de dados a ser transmitido pela RSSF, faz-se necessário analisar o impacto que isso causará na transmissão e recepção de mensagens na RSSF. Uma análise mais simples, focada no nó sensor, medirá o tempo e consumo de energia da transmissão e recepção de um pacote maior.

Porém, analisar o impacto de algoritmos e mecanismos de segurança para a RSSF como um todo não é trivial. Esta análise é dependente da aplicação, dos protocolos (enlace, roteamento, sincronização, localização, vizinhança) em uso na RSSF, e em qual camada os mecanismos estão sendo aplicados. Por exemplo, no caso do TinySec ou do *framework* de segurança do padrão IEEE 802.15.4, toda mensagem transmitida será autenticada e/ou cifrada. Nesse caso, todo nó sensor na rota da mensagem irá verificar a autenticidade e/ou decifrá-la, decidir sobre roteamento, cifrá-la novamente e enviá-la ao próximo salto.

Por outro lado, se os algoritmos e mecanismos de segurança para a RSSF são implementados a nível de aplicação, a verificação da autenticidade e/ou decifragem é feita somente no destino final. No entanto, se a mensagem ficou maior por conta dos mecanismos aplicados, todos os nós no seu caminho sofrerão com roteamento de mensagens maiores.

4.6.1. Demonstração de Segurança em RSSF

Esta demonstração trata de mecanismos de segurança implementados a nível de aplicação, ou seja, o cálculo e a verificação da autenticidade e/ou cifragem e decifragem da mensagem é feita somente na origem e no destino final. O seu foco está nos efeitos do mecanismo nos nós sensores, e não na rede como um todo. O nó sensor utilizado é o Crossbow TelosB [29], executando o sistema operacional Contiki 2.2.1 [41].

Para demonstrar o impacto de algoritmos de segurança em aplicações de RSSF, criamos quatro redes independentes, cada uma com dois nós. A primeira RSSF transmite e recebe mensagens claras, ou seja, sem qualquer mecanismo de segurança. Na segunda RSSF, os dados são autenticadas utilizando o MARVIN, enquanto a terceira RSSF cifra os dados com o with CURUPIRA. A quarta RSSF combina criptografia e autenticação, usando o CURUPIRA e o MARVIN para atingir seus objetivos. Além disso, existe um nó espião que monotora todas as mensagens sendo transmitidas por estas RSSF, conforme ilustrado na Figura 4.6.

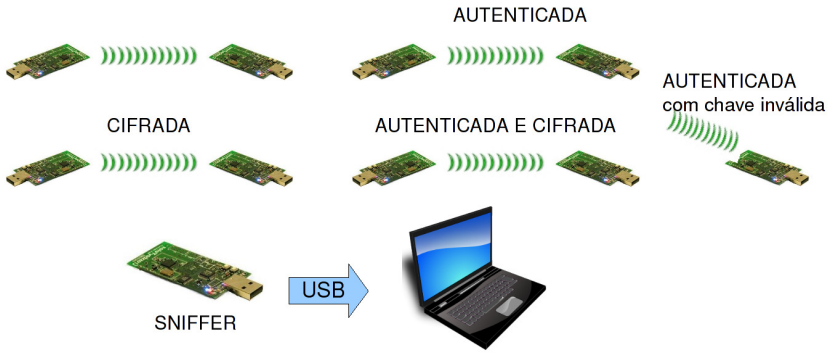

Figura 4.6. Configuração da *testbed*.

Os dados capturados pelo nó espião, que está conectado a um computador portátil através da porta USB, são mostrados na console do mesmo. O nó espião consegue entender os dados transmitidos através da RSSF sem mecanismos de segurança, e também da RSSF com dados autenticados. Porém, o nó espião não compreende os dados cifrados e/ou cifrados e autenticados das outras duas RSSF.

As mensagens possuem 12 bytes tanto na RSSF sem mecanismos de segurança como RSSF que encripta os dados. Por outro lado, as mensagens das RSSF com criptografia e criptografia/autenticação possuem ambas 16 bytes, devido a adição da *tag* de autenticação.

Dado que todo os nós estão no mesmo alcance, os nós receptores das quatro RSSF também receberão os dados das outras três redes. Porém, como as redes possuem diferentes tamanhos e formatos de mensagens, uma mensagem enviada na RSSF com dados cifrados e autenticados pode causar erros na rede sem mecanismos de segurança, por exemplo. Para mostrar se a mensagem é compatível ou causa algum tipo de erro, usamos os LEDs dos nós sensores. Quando o formato da mensagem e o seu valor é aceitável, o LED verde é aceso; já o LED vermelho é aceso se o formato da mensagem for inválido ou se uma mensagem não-autenticada for recebida (se for esperada uma mensagem autenticada). Se o formato da mensagem estiver correto, mas o seu valor for inesperado, o LED azul acende. A Tabela 4.3 resume que LEDs acendem quando os diferentes dados são recebidos em cada RSSF. Observe que dada a combinação de tipos de RSSF e dos vários formatos de mensagens transmitidas/recebidas, é possível mostrar o comportamento dos vários níveis de segurança de rede.

Tabela 4.3. Comportamento dos LEDs para uma dada RSSF e tipo de mensagem recebida.

	Rede			
Mensagem	**Clara**	**Cifrada**	**Autenticada**	**Cifrada/Autenticada**
Clara	Verde	Azul	Vermelho	Vermelho
Cifrada	Azul	Verde	Vermelho	Vermelho
Autenticada	Vermelho	Vermelho	Verde	Azul
Cifrada e Autenticada	Vermelho	Vermelho	Azul	Verde

O processo de transmissão de dados (e criptografia e/ou autenticação, se necessário) é disparado ao pressionar um dos botões do nós sensor TelosB. Antes de efetuar a transmissão da mensagem, um LED é aceso. No nó receptor, depois que a mensagem é recebida e processada, o LED de status acende.Assim, é possível observar as velocidades de funcionamento dos nós, que são praticamente as mesmas.

A última etapa na demonstração é a validação da autenticação. Um outro nó sensor envia uma mensagem autenticada com formato correto, porém utilizando uma chave inválida. Os nós sensores da rede com autenticação de mensagens, ao verificarem a *tag*, detectam que esta é inválida na rede.

4.7. Considerações Finais

Mecanismos de segurança inevitavelmente causam sobrecarga de processamento a aplicação de uma RSSF, e possivelmente também causam sobrecarga na comunicação, devido ao aumento no tamanho das mensagens. Porém, para algumas aplicações, esta sobrecarga é aceitável devido as suas necessidades de segurança.

Tipicamente, os serviços de segurança de integridade dos dados, confidencialidade, autenticidade do nó e dos dados e disponibilidade, precisam ser garantidos. Ainda é importante garantir que os dados enviados são recentes (*data freshness*), ou seja, que nenhum intruso está replicando dados antigos. Este mesmo conceito também pode se aplicar às chaves em uso na RSSF (*key freshness*). E, assim, os mecanismos de estabelecimento de chaves devem garantir que as chaves em uso são recentes, impedindo o uso de chaves antigas que poderiam ter sido obtidas após o comprometimento de um nó [54]. Dadas as características das RSSF, que se baseiam em nós com recursos de processamento, armazenamento, comunicação e energia limitados, os mecanismos de segurança empregados devem ser escaláveis (em termos de energia e atraso).

Neste capitulo, apresentamos a visão geral da área de segurança em Redes de Sensores sem Fio (RSSF). Observa-se que a pesquisa nesta área tem mostrado resultados recentes, fazendo com que o emprego de mecanismos de segurança seja viável.

Agradecimentos

Parte deste trabalho foi financiado pelo Centro de Pesquisa e Desenvolvimento, Ericsson Telecomunicações S.A., Brasil.

A implantação da *testbed* de demonstração de segurança em RSSF foi realizada com o valioso auxílio de Bruno Trevizan de Oliveira e Gustavo Tejada de Sousa.

Referências

[1] *Wireless Sensor Networks*, chapter Localization in Sensor Networks. Springer, 2004.

[2] *Encyclopedia of Wireless and Mobile Communications*, chapter A Survey on Secure Localization in Wireless Sensor Networks. CRC Press, Taylor and Francis Group, 2007.

[3] Nadeem Ahmed, Salil S. Kanhere, and Sanjay Jha. The holes problem in wireless sensor networks: a survey. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(2):4–18, 2005.

[4] Hani Alzaid, Ernest Foo, and Juan Manuel Gonzalez Nieto. Secure data aggregation in wireless sensor network: a survey. In *Sixth Australasian Information Security Conference (AISC2008)*, volume 81 of CRPIT, page 93?105, 2008.

[5] D. Aranha, L. Oliveira, J. López, and R. Dahab. NanoPBC: Implementing cryptographic pairings on an 8-bit platform. In *Conference on Hyperelliptic curves, discrete Logarithms, Encryption, etc. – CHiLE'09*, 2009.

[6] P. Barreto and M. Simplicio. CURUPIRA, a block cipher for constrained platforms. In *Anais do 25º Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2007*, volume 1, pages 61–74. SBC, 2007. http://www.larc.usp.br/~mjunior/files/en/Curupira1-extended.pdf.

[7] G. Bauer, P. Potisk, and S. Tillich. Comparing block cipher modes of operation on MICAz sensor nodes. In *PDP'09: Proc. of the 2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, pages 371–378, Washington, DC, USA, 2009. IEEE Computer Society.

[8] M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, 61(3):362–399, 2000.

[9] M. Bellare, P. Rogaway, and D. Wagner. The EAX mode of operation: A two-pass authenticated-encryption scheme optimized for simplicity and efficiency. In *Fast Software Encryption 2004*, pages 389–407, February 2004. http://www.cs.ucdavis.edu/~rogaway/papers/eax.pdf.

[10] E. Biham, R. Anderson, and L. Knudsen. Serpent: A new block cipher proposal. In *FSE*, pages 222–238, 1998.

[11] E. Biham, A. Biryukov, and A. Shamir. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In *Advances in Cryptology – Eurocrypt'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 55–64. Springer, 1999.

[12] E. Biham and A. Shamir. Differential cryptanalysis of DES-like cryptosystems. In *Crypto'90: Proc. of the 10th Annual International Cryptology Conference on Advances in Cryptology*, pages 2–21, London, UK, 1991. Springer-Verlag.

[13] J. Black. Authenticated encryption, 2004. http://www.cs.colorado.edu/~jrblack/papers/ae.pdf.

[14] J. Black and P. Rogaway. A block-cipher mode of operation for parallelizable message authentication. In *Advances in Cryptology - EUROCRYPT'02. Lecture Notes in Computer Science*, pages 384–397. Springer-Verlag, 2002.

[15] A. Bogdanov, L.R. Knudsen, G. Leander, C. Paar, A. Poschmann, M.J.B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In *Cryptographic Hardware and Embedded Systems – CHES'2007*, Lecture Notes in Computer Science, Heidelberg, Germany, 2007. Springer.

[16] D. Boyle and T. Newe. Security protocols for use with wireless sensor networks: A survey of security architectures. In *ICWMC '07: Proceedings of the Third International Conference on Wireless and Mobile Communications*, page 54, Washington, DC, USA, 2007. IEEE Computer Society.

[17] Edgar H. Callaway. *Wireless Sensor Networks: Architectures and Protocols*. CRC Press, Inc., Boca Raton, FL, USA, 2003.

[18] N. Canh, Y.-K. Lee, and S. Lee. HGKM: A group-based key management scheme for sensor networks using deployment knowledge. *6th Annual Communication Networks and Services Research Conference. CNSR'08*, pages 544–551, May 2008.

[19] D. Carman, P. Kruus, and B. Matt. Constraints and approaches for distributed sensor network security. Technical Report 00-010, NAI Labs, September 2000.

[20] Claude Castelluccia, Aldar C-F. Chan, Einar Mykletun, and Gene Tsudik. Efficient and provably secure aggregation of encrypted data in wireless sensor networks. *ACM Trans. Sen. Netw.*, 5(3):1–36, 2009.

[21] Alberto Cerpa and Deborah Estrin. Ascent: Adaptive self-configuring sensor networks topologies. In *Proceedings of the Twenty First International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, New York, NY, USA, June 2002.

[22] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks. In *SP'03: Proc. of the 2003 IEEE Symposium on Security and Privacy*, page 197, Washington, DC, USA, 2003. IEEE Computer Society.

[23] Benjie Chen, Kyle Jamieson, Hari Balakrishnan, and Robert Morris. Span: an energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. *Wireless Networks*, 8(5):481–494, 2002.

[24] Wu chi Feng, Brian Code, Ed Kaiser, Mike Shea, Wu chang Feng, and Louis Bavoil. Panoptes: scalable low-power video sensor networking technologies. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pages 562–571, New York, NY, USA, 2003. ACM Press.

[25] G. Contreras, M. Martonosi, J. Peng, G-Y. Lueh, and R. Ju. The XTREM power and performance simulator for the Intel XScale core: Design and experiences. *ACM Trans. Embed. Comput. Syst.*, 6(1):4, 2007.

[26] C.M. Cordeiro and D.P. Agrawal. *Ad Hoc & Sensor Networks: Theory And Applications*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2006.

[27] Crossbow. Micaz datasheet. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf`, 2008.

[28] Crossbow. MICAz datasheet. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf`, 2008.

[29] Crossbow. Telosb datasheet. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf`, 2008.

[30] Crossbow. TelosB datasheet. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf`, 2008.

[31] David Culler, Deborah Estrin, and Mani Srivastava. Overview of sensor networks. *Computer Magazine*, 37(8):41–49, 2004.

[32] J. Daemen, L. R. Knudsen, and V. Rijmen. The block cipher SQUARE. In *Fast Software Encryption – FSE'97*, volume 1267 of *Lecture Notes in Computer Science*, pages 149–165, Haifa, Israel, 1997. Springer.

[33] J. Daemen and V. Rijmen. The wide trail design strategy. *Lecture Notes in Computer Science*, 2260:222–239, 2001. `http://link.springer-ny.com/link/service/series/0558/papers/2260/22600222.pdf`.

[34] J. Daemen and V. Rijmen. *The Design of Rijndael: AES – The Advanced Encryption Standard*. Springer, Heidelberg, Germany, 2002.

[35] J. Daemen and V. Rijmen. A new MAC construction ALRED and a specific instance ALPHA-MAC. In *FSE*, pages 1–17, 2005.

[36] C. DeCannière. Trivium: a stream cipher construction inspired by block cipher design principles. *Information Security*, 4176:36–55, 2006.

[37] L. Doherty, B. A. Warneke, B. Boser, and K. S. J. Pister. Energy and performance considerations for smart dust. *International Journal of Parallel and Distributed Systems and Networks*, 4(3):121–133, 2001.

[38] B. Doyle, S. Bell, A. Smeaton, K. McCusker, and N. O'Connor. Security considerations and key negotiation techniques for power constrained sensor networks. *The Computer Journal*, 49(4):443–453, 2006.

[39] W. Du, J. Deng, Y. Han, S. Chen, and P. Varshney. A key management scheme for wireless sensor networks using deployment knowledge. *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, 1:–597, March 2004.

[40] W. Du, J. Deng, Y. Han, P. Varshney, J. Katz, and A. Khalili. A pairwise key predistribution scheme for wireless sensor networks. *ACM Trans. Inf. Syst. Secur.*, 8(2):228–258, 2005.

[41] Adam Dunkels, Björn Grönvall, and Thiemo Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proc. of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, November 2004.

[42] T. Eisenbarth, S. Kumar, C. Paar, A. Poschmann, and L. Uhsadel. A survey of lightweight-cryptography implementations. *IEEE Design and Test of Computers*, 24(6):522–533, 2007.

[43] P. Ekdahl and T. Johansson. A new version of the stream cipher SNOW. In *Selected Areas in Cryptography*, pages 47–61, 2002. `http://www.it.lth.se/cryptology/snow/snow20.pdf`.

[44] L. Eschenauer and V. Gligor. A key-management scheme for distributed sensor networks. In *CCS'02: Proc. of the 9th ACM conference on Computer and communications security*, pages 41–47, New York, NY, USA, 2002. ACM.

[45] N. Fournel, M. Minier, and S. Ubéda. Survey and benchmark of stream ciphers for wireless sensor networks. In *WISTP*, volume 4462 of *Lecture Notes in Computer Science*, pages 202–214. Springer, 2007.

[46] R. Gennaro, S. Halevi, H. Krawczyk, T. Rabin, S. Reidt, and S. Wolthusen. Strongly-resilient and non-interactive hierarchical key-agreement in MANETs. In *ESORICS'08: Proc. of the 13th European Symposium on Research in Computer Security*, pages 49–65, Berlin, Heidelberg, 2008. Springer-Verlag.

[47] J. Großschädl, S. Tillich, C. Rechberger, M. Hofmann, and M. Medwed. Energy evaluation of software implementations of block ciphers under memory constraints. In *DATE'07: Proc. of the conference on Design, automation and test in Europe*, pages 1110–1115, San Jose, CA, USA, 2007. EDA Consortium.

[48] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2003.

[49] Wendi Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *33rd Hawaii International Conference on System Sciences (HICSS '00)*, Hawaii, January 2000.

[50] W.R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *Proceedings of the Hawaii International Conference on System Sciences*, pages 3005–14. IEEE, January 2000.

[51] M. Hell, T. Johansson, and W. Meier. Grain: a stream cipher for constrained environments. *Int. J. Wire. Mob. Comput.*, 2(1):86–93, 2007.

[52] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System architecture directions for networked sensors. *SIGPLAN Not.*, 35(11):93–104, 2000.

[53] D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, and S. Chee. HIGHT: A new block cipher suitable for low-resource device. In *CHES*, pages 46–59, 2006.

[54] Fei Hu and Neeraj K. Sharma. Security considerations in wireless sensor networks. *Ad Hoc Networks*, 3(1):69–89, Jan. 2005.

[55] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Sixth Annual International Conference on Mobile Computing and Networking, (MobiCom 2000)*, pages 56–67. ACM, August 2000.

[56] Intel Corp. *Intel XScale Core: Developer's Manual*, 2000. Order No. 273473-001.

[57] C. Karlof, N. Sastry, and D. Wagner. TinySec: a link layer security architecture for wireless sensor networks. In *2nd International Conference on Embedded Networked Sensor Systems – SenSys'2004*, pages 162–175, Baltimore, USA, 2004. ACM.

[58] Chris Karlof, Naveen Sastry, and David Wagner. Tinysec: A link layer security architecture for wireless sensor networks. In *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, November 2004.

[59] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. In *First IEEE International Workshop on Sensor Network Protocols and Applications*, pages 113–127, 2002.

[60] Ifrah Farrukh Khan and Muhammad Younas Javed. A survey on routing protocols and challenge of holes in wireless sensor networks. *Advanced Computer Theory and Engineering, International Conference on*, 0:161–165, 2008.

[61] Ioannis Krontiris, Zinaida Benenson, Thanassis Giannetsos, Felix C. Freiling, and Tassos Dimitriou. Cooperative intrusion detection in wireless sensor networks. In *EWSN*, pages 263–278, 2009.

[62] T. Krovetz and P. Rogaway. Internet draft: The OCB authenticated-encryption algorithm. `http://www.cs.ucdavis.edu/~rogaway/papers/ocb-id.htm`, March 2005.

[63] J. Kulik, W.R. Heinzelman, and H. Balakrishnan. Negotiation-based protocols for dissemination information in wireless sensor networks. In *Submited to ACM Wireless Networks*, 2001.

[64] Andreas Lachenmann, Pedro José Marrón, Daniel Minder, and Kurt Rothermel. Meeting lifetime goals with energy levels. In *SenSys '07: Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 131–144, New York, NY, USA, 2007. ACM.

[65] X. Lai and J. Massey. A proposal for a new block encryption standard. In *EUROCRYPT'90: Proc. of the workshop on the theory and application of cryptographic techniques on Advances in Cryptology*, pages 389–404, New York, NY, USA, 1991. Springer-Verlag.

[66] Y. W. Law, J. Doumen, and P. Hartel. Survey and benchmark of block ciphers for wireless sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 2(1):65–93, 2006.

[67] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. *TinyOS: An operating system for wireless sensor networks*. Springer-Verlag, 2004.

[68] Philip Levis, David Gay, and David Culler. Active sensor networks. In *2nd USENIX/ACM Symposium on Network Systems Design and Implementation (NSDI)*, 2005.

[69] T. Li, H. Wu, X. Wang, and F. Bao. SenSec design. Technical report, InfoComm Security Department, February 2005.

[70] C.H. Lim and T. Korkishko. mCrypton – a lightweight block cipher for security of low-cost RFID tags and sensors. In *WISA*, pages 243–258, 2005.

[71] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *CCS'03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 52–61, New York, NY, USA, 2003. ACM.

[72] W. Liu, R. Luo, and H. Yang. Cryptography overhead evaluation and analysis for wireless sensor networks. *Communications and Mobile Computing, International Conference on*, 3:496–501, 2009.

[73] S. Lucks. Two-pass authenticated encryption faster than generic composition. In *Fast Software Encryption 2005*, pages 284–298, 2005. `http://www.iacr.org/cryptodb/archive/2005/FSE/3123/3123.pdf`.

[74] M. Luk, G. Mezzour, A. Perrig, and V. Gligor. MiniSec: A secure sensor network communication architecture. In *IPSN'07: Proc. of the 6th international conference on Information processing in sensor networks*, pages 479–488, New York, NY, USA, 2007. ACM.

[75] Mark Luk, Ghita Mezzour, Adrian Perrig, and Virgil Gligor. MiniSec: a secure sensor network communication architecture. In *IPSN '07: Proceedings of the 6th international conference on Information processing in sensor networks*, pages 479–488, New York, NY, USA, 2007. ACM.

[76] Sam Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TinyDB: An acqusitional query processing system for sensor networks. *ACM Transactions on Database Systems*, 2005.

[77] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, and John Anderson. Wireless sensor networks for habitat monitoring. In *First ACM Workshop on Wireless Sensor Networks and Applications*, Atlanta, GA, September 2002.

[78] Cintia Borges Margi, Xiaoye Lu, Gefan Zhang, Ganymed Stanek, Roberto Manduchi, and Katia Obraczka. A power-aware, self-managing wireless camera network for, wide area monitoring. In *First Workshop on Distributed Smart Cameras (DSC 2006)*, Boulder, Colorado, USA, October 2006.

[79] M. Matsui. Linear cryptanalysis method for DES cipher. In *Advances in Cryptology - Eurocrypt'93*, volume 765 of *Lecture Notes in Computer Science*, pages 62–73, Lofthus, Norway, 1993. Springer-Verlag. `http://homes.esat.kuleuven.be/~abiryuko/Cryptan/matsui_des.PDF`.

[80] M. Matsui. New block encryption algorithm MISTY. In *Fast Software Encryption – FSE'97*, volume 1267 of *Lecture Notes in Computer Science*, pages 54–68. Springer, 1997.

[81] MediaCrypt AG. The IDEA block cipher – submission to the NESSIE project. http://cryptonessie.org, 2000.

[82] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Boca Raton, USA, 1999.

[83] Johann Van Der Merwe, Dawoud Dawoud, and Stephen McDonald. A survey on peer-to-peer key management for mobile ad hoc networks. *ACM Comput. Surv.*, 39(1):1, 2007.

[84] R. Müller, G. Alonso, and D. Kossmann. SwissQM: Next generation data processing in sensor networks. In *CIDR*, pages 1–9, 2007.

[85] NIST. *Federal Information Processing Standard (FIPS PUB 113) – Standard on Computer Data Authentication*. National Institute of Standards and Technology, U.S. Department of Commerce, May 1985. http://www.itl.nist.gov/fipspubs/fip113.htm.

[86] NIST. *Federal Information Processing Standard (FIPS 197) – Advanced Encryption Standard (AES)*. National Institute of Standards and Technology, November 2001.

[87] NIST. *Special Publication SP 800-38A – Recommendations for Block Cipher Modes of Operation, Methods and Techniques*. National Institute of Standards and Technology, December 2001. http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf.

[88] NIST. *Special Publication 800-38B Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication*. National Institute of Standards and Technology, U.S. Department of Commerce, May 2005. http://csrc.nist.gov/publications/PubsSPs.html.

[89] NIST. *Special Publication 800-38D – Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. National Institute of Standards and Technology, U.S. Department of Commerce, November 2007. http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf.

[90] NSA. *Skipjack and KEA Algorithm Specifications, version 2.0*. National Security Agency, May 1998.

[91] M. N.Wegman and J. L. Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22:265–279, 1981.

[92] L. Oliveira, R. Dahab, J. Lopez, F. Daguano, and A. Loureiro. Identity-based encryption for sensor networks. In *PERCOMW'07: Proc. of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops*, pages 290–294, Washington, DC, USA, 2007. IEEE Computer Society.

[93] B. Panja, S. Madria, and B. Bhargava. Energy-efficient group key management protocols for hierarchical sensor networks. *Int. J. Distrib. Sen. Netw.*, 3(2):201–223, 2007.

[94] Adrian Perrig, John Stankovic, and David Wagner. Security in wireless sensor networks. *Commun. ACM*, 47(6):53–57, 2004.

[95] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar. SPINS: security protocols for sensor netowrks. In *Proceedings of the seventh annual international conference on Mobile computing and networking*, pages 189–199. ACM Press, 2001.

[96] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar. SPINS: security protocols for sensor netowrks. In *Proc. of the 7th Annual International Conference on Mobile Computing and Networking*, pages 189–199. ACM Press, 2001.

[97] Mohammad Rahimi, Rick Baer, Obimdinachi I. Iroezi, Juan C. Garcia, Jay Warrior, Deborah Estrin, and Mani Srivastava. Cyclops: In situ image sensing and interpretation in wireless sensor networks. In *SenSys 2005)*, 2005.

[98] V. Rajendran, K. Obraczka, and J.J. Garcia-Luna-Aceves. Energy-efficient, collision-free medium access control for wireless sensor networks. In *ACM SenSys 03*, Los Angeles, CA, November 2003.

[99] S. Rinne, T. Eisenbarth, and C. Paar. Performance analysis of contemporary light-weight block ciphers on 8-bit microcontrollers. `http://www.lightweightcrypto.org/papers.php`, 2007. Ecrypt workshop SPEED - Software Performance Enhancement for Encryption and Decryption.

[100] R.L. Rivest, M. Robshaw, R. Sidney, and Y. Yin. The RC6 block cipher. In *in First Advanced Encryption Standard (AES) Conference*, page 16, 1998.

[101] R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[102] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Symposium on Cryptography and Information Security-SCIS'2000*, pages 26–28, 2000.

[103] Sandia. Submission to NIST: Cipher-state (CS) mode of operation for AES. `http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/cs/cs-spec.pdf`, 2004.

[104] M. Simplicio. Algoritmos criptográficos para redes de sensores. Master's thesis, Escola Politécnica at the University of São Paulo, April 2008. `http://www.teses.usp.br/teses/disponiveis/3/3141/tde-30092008-182545/`.

[105] M. Simplicio, P. Barreto, T. Carvalho, C. Margi, and M. Näslund. The CURUPIRA-2 block cipher for constrained platforms: Specification and benchmarking. In *Proc. of the 1st International Workshop on Privacy in Location-Based Applications - PiLBA'08*, volume 397. CEUR-WS, 2008. `http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-397/`.

[106] Ignacio Solis and Katia Obraczka. The impact of timing in data aggregation for sensor networks. In *The 2004 International Conference on Communications (ICC 2004)*, June 2004.

[107] F.X. Standaert, G. Piret, N. Gershenfeld, and J.J. Quisquater. SEA: A scalable encryption algorithm for small embedded applications. In *Proc. of Smart Card Research and Applications (CARDIS'06)*, LNCS, pages 222–236. Springer-Verlag, 2006.

[108] IEEE Standard. IEEE 802.15.4: Wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs), 2006.

[109] C. Strydis, D. Zhu, and G. Gaydadjiev. Profiling of symmetric-encryption algorithms for a novel biomedical-implant architecture. In *CF'08: Proc. of the 5th conference on Computing frontiers*, pages 231–240, New York, NY, USA, 2008. ACM.

[110] Texas Instruments, Inc. *MSP430x13x, MSP430x14x Mixed Signal Microcontroller – Datasheet*, 2001.

[111] Gilman Tolle, Joseph Polastre, Robert Szewczyk, Neil Turner, Kevin Tu, Phil Buonadonna, Stephen Burgess, David Gay, Wei Hong, Todd Dawson, and David Culler. A macroscope in the redwoods. In *Third ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2005.

[112] Tijs van Dam and Koen Langendoen. An adaptive energy-efficient MAC protocol for wireless sensor networks. In *ACM SenSys 03*, Los Angeles, CA, November 2003.

[113] Wei Ye, John Heidemann, and Deborah Estrin. An energy-efficient MAC protocol for wireless sensor networks. In *Proceedings of the 21st International Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2002)*, New York, NY, USA, June 2002.

[114] M. Younis, K. Ghumman, and M. Eltoweissy. Location-aware combinatorial key management scheme for clustered sensor networks. *IEEE Trans. Parallel Distrib. Syst.*, 17(8):865–882, 2006.

[115] Yan Yu, Ramesh Govindan, and Deborah Estrin. Geographical and energy aware routing: a recursive data dissemination protocol for wireless sensor networks. Technical Report TR-01-0023, UCLA - Computer Science Department, 2001.