Análise comparativa de protocolos de segurança para redes de sensores sem fio

Mateus Augusto Silva Santos

Dissertação apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Ciências

Programa: Ciência da Computação
Orientador: Prof. Dr. Routo Terada

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro do CNPq

São Paulo, Outubro de 2009

Análise comparativa de protocolos de segurança para redes de sensores sem fio

Este exemplar corresponde à redação
final da dissertação devidamente corrigida
e defendida por Mateus Augusto Silva Santos
e aprovada pela Comissão Julgadora.

Banca Examinadora:

- Prof. Dr. Routo Terada (orientador) - IME-USP.
- Prof. Dr. Alfredo Goldman vel Lejbman - IME-USP.
- Profa. Dra. Cíntia Borges Margi - EACH-USP.

Agradecimentos

Este espaço é dedicado àqueles que deram a sua contribuição para que essa dissertação fosse realizada. A todos eles deixo aqui o meu agradecimento sincero.

À minha amada esposa Cynthia, por apoiar e compreender a minha difícil decisão de deixar uma boa colocação no mercado de trabalho para me dedicar aos estudos. Além disso, agradeço pela sua paciência nos momentos mais difíceis e principalmente pelo seu amor.

À minha querida filha Isabela, por ser uma grande fonte de motivação e alegria para a minha vida.

Aos meus queridos pais Eduardo e Wilma, pelo grande incentivo e suporte financeiro. Agradeço também pelos inúmeros conselhos e por me receberem em sua casa com paciência e amor.

À minha irmã Ana Cláudia, pelos conselhos e por sempre torcer pelo meu sucesso.

Ao meu orientador Dr. Routo Terada, pela escolha do tema deste trabalho e pela orientação. Agradeço também por sua paciência em me atender e por me colocar em contato com a interessante área da criptografia computacional.

Aos professores da Universidade Federal de Itajubá, em especial ao professor Edison de Jesus, pelo excelente trabalho com os cursos de graduação desta instituição, o que foi base para meus estudos no mestrado.

Aos professores do IME-USP, pelo compromisso com ensino e pesquisa, em especial aos professores Leônidas Brandão e Carlos Eduardo, os quais também me deram excelentes recomendações.

À professora Dra. Cíntia Margi, pelas interessantes sugestões.

Aos colegas da pós-graduação, pelas produtivas conversas e principalmente pela companhia.

Resumo

As redes de sensores sem fio (RSSF) são compostas por pequenos dispositivos distribuídos em uma região geográfica com a finalidade de monitorar ou interagir com o ambiente. Esse tipo de rede tem sido alvo de grande atenção da comunidade acadêmica e empresarial, dados os avanços das produções científicas e aplicações comerciais. Além disso, há grande potencial para esse modelo de rede, pois há diversos benefícios em se ter muitos dispositivos de baixo custo trabalhando em cooperação e que ainda podem interagir com o mundo real.

As RSSFs apresentam novos desafios, até então inexistentes na maioria das redes modernas. A baixa capacidade de processamento dos dispositivos, a limitação do tamanho de um pacote, a baixa taxa de transferência de pacotes, a baixa capacidade da bateria de um dispositivo e o alcance limitado do rádio dificultam ou até inviabilizam muitas implementações de segurança. Neste sentido, há uma grande variedade de protocolos de segurança, os quais tentam fornecer o máximo de propriedades desejadas, como por exemplo autenticidade e confidencialidade de mensagens.

Nesta dissertação, analisamos e comparamos dois pares de protocolos de segurança que possuem grande atenção da comunidade. Os protocolos foram analisados com base em seus mecanismos criptográficos e propriedades oferecidas. Além disso, com o uso de um simulador de RSSFs, realizamos experimentos que ajudam a entender o comportamento de dois protocolos, principalmente no que se relaciona com o consumo de energia dos dispositivos sensores.

Palavras-chave: redes de sensores, criptologia, protocolos de segurança.

Abstract

Wireless Sensor Networks (WSNs) are formed by many tiny devices deployed for monitoring or interacting with the environment in a geografic area. This kind of network has received great attention in the academic and corporate community, given the advances of comercial applications and scientific developments. The benefits of having many low cost devices cooperating and interacting with the real world explain the great potential of such networks.

The WSNs come with new challenges, so far not faced in modern networks. The low-power, low-processing devices, limited bandwidth and radio range make many security implementations difficult or even unfeasible. Within this context, there is a diversity of security protocols, intended to provide as many security properties as possible, like confidentiality and authentication of messages.

In this master thesis, we compare and evaluate two pairs of security protocols that enjoy significant attention in the community. These protocols are evaluated based on their given properties and cryptographic primitives. Using a WSNs simulator, we performed experiments that help to understand the behavior of these two protocols, mainly for energy consumption purposes.
Keywords: sensor networks, cryptology, security protocols.

Sumário

Lista de Abreviaturas **xi**

Lista de Figuras **xiii**

Lista de Tabelas **xv**

1 Introdução **1**
1.1 Motivações e Desafios 1
1.2 Objetivos 2
1.3 Organização do Documento 2

2 Conceitos **3**
2.1 Redes de Sensores Sem Fio 3
2.1.1 Características 4
2.1.2 Aplicações 4
2.1.3 Modelos de Comunicação 6
2.1.4 Segurança em Redes de Sensores Sem Fio 6
2.1.5 Ataques e Vulnerabilidades em Sensores 7
2.2 Criptografia Simétrica e Assimétrica 8
2.3 Propriedades de Segurança Desejadas em RSSFs 9
2.3.1 Confidencialidade de Informação e Segurança Semântica 9
2.3.2 Autenticidade e Integridade de Mensagem 9
2.3.3 *Freshness*: Proteção Contra Ataques do Tipo *Replay* 10
2.3.4 Resistência a Comprometimento de Nós da Rede 10
2.4 Demais Propriedades Desejadas em RSSFs 10
2.4.1 Baixo Sobrecusto Computacional e de Comunicacão 10
2.4.2 Resistência a perda de pacotes da rede 11
2.4.3 Escalabilidade 11
2.5 Protocolos de Segurança 11

3 Criptografia Simétrica em RSSFs **13**
3.1 Notações 13
3.2 Message Authentication Codes (MAC) 13

3.2.1 Cálculo do valor MAC 14
3.2.2 Propriedades de um algoritmo MAC 14
3.3 Cifras Simétricas 14
3.3.1 Esquema de Cifragem 15
3.3.2 Cifras de Fluxo 16
3.3.3 Cifras de Bloco 16
3.3.4 Modos de Operação 17
3.3.5 A Cifra Skipjack 21

4 Protocolos de Segurança para Comunicação *Unicast* 25
4.1 Protocolos Baseados em Esquemas Simétricos 25
4.1.1 Protocolos TinySec e Minisec-U 27
4.2 Protocolo TinySec 27
4.2.1 Confidencialidade 27
4.2.2 Autenticação e Integridade 28
4.2.3 Formatos de Pacote 28
4.2.4 Esquema de Chaves 28
4.3 Protocolo MiniSec-U 29
4.3.1 Confidencialidade 29
4.3.2 Autenticação e Integridade 29
4.3.3 Formato de Pacote 30
4.3.4 Esquema de Chaves 30
4.4 Cifras de Bloco Alternativas para os Protocolos MiniSec-U e TinySec 30

5 Protocolos de Segurança para Comunicação *Broadcast* 31
5.1 Protocolos *Unicast* também podem ser utilizados em comunicação *Broadcast*? 31
5.1.1 Adaptações de Protocolos *Unicast* para Comunicação *Broadcast* 32
5.2 Protocolos *Broadcast* Simétricos e Assimétricos 32
5.2.1 Protocolos *Broadcast* com Esquemas Simétricos Predominantes: μTESLA e MiniSec-B 32
5.3 O Protocolo μTESLA 33
5.3.1 Considerações 33
5.3.2 Funcionamento do μTESLA 33
5.3.3 Geração das Chaves 34
5.3.4 Exemplo de funcionamento do protocolo 34
5.3.5 Implementação da Função Unidirecional 35
5.3.6 Confidencialidade, Autenticação e Integridade 35
5.4 O Protocolo MiniSec-B 35
5.4.1 Considerações 36
5.4.2 Funcionamento do MiniSec-B 36

5.4.3 Sliding Windows: Proteção Contra Replay Com Janela de Vulnerabilidade . . 36
5.4.4 Proteção Contra Replay Sem Janela de Vulnerabilidade 38
5.4.5 Confidencialidade, Autenticação e Integridade 40
5.4.6 Formatos de Pacote 40

6 Análise Comparativa dos Protocolos TinySec e MiniSec-U 41
6.1 Análise Teórica: Metodologia 41
6.2 Considerações Iniciais sobre Segurança 42
6.3 Confidencialidade e Segurança Semântica 43
6.3.1 Modos de Operação CBC e OCB 43
6.3.2 Condições de Segurança Semântica 47
6.3.3 Valor Inicial Previsível 49
6.3.4 Cifra de Bloco 49
6.4 Autenticidade e Integridade de Mensagem 50
6.4.1 Considerações Iniciais 50
6.4.2 Tentativa de fraudar o algoritmo CBC-MAC 50
6.4.3 Tentativa de fraudar o algoritmo OCB 51
6.4.4 Vantagem de adversários contra os algoritmos CBC e OCB 52
6.4.5 Autenticação de Mensagens de Tamanho Fixo e Variável 53
6.5 *Freshness*: Proteção Contra Ataques do Tipo Replay 54
6.6 Resistência a Comprometimento de Nós da Rede 54
6.7 Demais Propriedades 54
6.7.1 Autenticação Imediata de Mensagens 54
6.7.2 Baixo Sobrecusto Computacional e de Comunicação 55
6.7.3 Resistência a perda de pacotes da rede 55
6.7.4 Escalabilidade 56
6.8 Análise Experimental: Implementação e Simulação 56
6.8.1 TinyOS 56
6.8.2 TOSSIM: Um Simulador de Eventos para TinyOS 58
6.8.3 Metodologia de Implementação 60
6.8.4 Resultados dos Experimentos 62

7 Análise Comparativa dos Protocolos μTESLA e MiniSec-B 67
7.1 Metodologia 67
7.2 Confidencialidade 68
7.3 Autenticidade e Integridade de Mensagem 69
7.4 Freshness 69
7.5 Resistência a Comprometimento de Nós 70
7.6 Autenticação Imediata 70
7.7 Baixo Sobrecusto Computacional e de Comunicação 71

7.8 Resistência a perda de pacotes da rede 72
7.9 Escalabilidade 73

8 Conclusões **75**
8.1 Comparação dos Protocolos *Unicast* MiniSec-U e TinySec 75
8.2 Comparação dos Protocolos *Broadcast* MiniSec-B e μTESLA 76
8.3 Contribuições 76
8.4 Trabalhos Futuros 76

A Esquema de Autenticação e Cifragem **79**

B Dados Experimentais **81**
B.1 Consumo de Energia 81
B.2 Espaço Utilizado 84

C Listagem de Códigos **87**
C.1 Funções da Aplicação Surge 87
C.2 Protocolos MiniSec e TinySec 88

Referências **89**

Lista de Abreviaturas

RSSF	Redes de Sensores Sem Fio (*Wireless Sensor Networks*).
OCB	*Offset Code Book*
CBC	*Cipher Block Chainning*
CRC	*Cyclic Redundant Check*
CTR	*Counter*
AES	*Advanced Encryption Standard*
MAC	*Message Authentication Code*
CBC-MAC	*Cipher Block Chainning Message Authentication Code*
SPINS	*Security Protocols for Sensor Networks*
μTESLA	*micro Time Efficient Streaming Loss-tolerant Authentication*
SNEP	*Secure Network Encryption Protocol*
WEP	*Wired Equivalent Privacy*
ECC	*Elliptic Curve Cryptography*
PBC	*Pairing-Based Cryptography*
CPA	*Chosen-Plaintext Attack*
IND-CPA	*Indistinguishability under Chosen-Plaintext Attack*
SEM-CPA	*Semmantic Security under Chosen-Plaintext Attack*
VI	Valor Inicial

Lista de Figuras

2.1 Arquitetura representativa de uma RSSFs [40] 4
2.2 Arquitetura de um Sistema de Monitoração Ambiental [54]. 5
2.3 Cifragem e Decifragem em Criptografia Simétrica 8

3.1 Autenticação com Message Authentication Code 14
3.2 Cifragem e Decifragem no modo CTR. 17
3.3 Cifragem e Autenticação no modo OCB. 18
3.4 Cifragem no modo CBC. 19
3.5 Cálculo do valor MAC através de uma cifra de bloco no modo CBC. 21
3.6 Diagrama da permutação G [60]. 22

4.1 Formatos de Pacote do TinySec-AE e TinyOS. 29
4.2 Formato de Pacote do MiniSec-U. 30

5.1 Revelação de Chaves por Tempo para Autenticação. 35
5.2 Linha do Tempo. 38
5.3 Geração de um Filtro de Bloom [25]. 39
5.4 Formato de Pacote do MiniSec-B. 40

6.1 Oráculo de Cifragem 44
6.2 Experimento com a versão 0 do oráculo 46
6.3 Experimento com a versão 1 do oráculo 46
6.4 Oráculo de Verificação para o algoritmo CBC. $\mathrm{GAuthCBC}_K(M)$ é um algoritmo que devolve a tag de autenticação correspondente a M e K 51
6.5 Oráculo de Verificação para o algoritmo OCB. 51
6.6 Arquitetura do TOSSIM [46]. 58
6.7 Arquitetura do PowerTOSSIM [80]. 60
6.8 PowerTOSSIM: Consumo Total de Energia em uma RSSF com 10 (a), 50 (b) e 100 (c) Nós após 1200 segundos. 64
6.9 PowerTOSSIM: Média do Consumo de Energia dos Dispositivos em RSSFs com 10, 50 e 100 Nós após 1200 segundos com (a) e sem (b) Picos de Consumo de cada Protocolo. 64
6.10 Exemplo de Árvore de Roteamento para uma Rede com Seis Nós Sensores. 65

6.11 PowerTOSSIM: Média do Consumo de Energia dos Dispositivos em RSSFs com 10, 30, 50, 70 e 100 Nós após 1200 segundos. 65
6.12 PowerTOSSIM: Média do Consumo de Energia dos Dispositivos em RSSFs com 10, 30, 50, 70 e 100 Nós após 1200 segundos. As Simulações Consideram Possibilidades de Perda de Pacotes. 66
6.13 Espaço de Memória ROM e RAM, em KBytes, ocupados pela aplicação compilada com cada protocolo de segurança (plataforma Mica2) 66

Lista de Tabelas

3.1 Regras A e B da Cifra Skipjack 21
3.2 Tabela F. A S-box do Skipjack 23

4.1 Protocolos de Segurança para Comunicação *Unicast*. 26

6.1 Principais Propriedades de Segurança dos Protocolos TinySec e MiniSec-U. 41
6.2 Aumento do tamanho de pacote, em bytes, dos protocolos de segurança MiniSec-U e TinySec em relação à versão original de pacote do sistema TinyOS. 55
6.3 Exemplos de Comandos, Eventos e Tarefas Implementadas em linguagem nesC para o sistema TinyOS. 57
6.4 Versões de Software Utilizados em Simulações. 60
6.5 Parâmetros de Implementação dos Mecanismos dos Protocolos TinySec e MiniSec-U 61

7.1 Aumento do tamanho de pacote, em bytes, dos protocolos de segurança μTESLA e MiniSec-B em relação à versão original de pacote do sistema TinyOS. 72

Capítulo 1

Introdução

O desenvolvimento tecnológico na área da eletrônica e das redes de comunicação permitiu a produção de dispositivos como os nós sensores. Esses pequenos equipamentos de baixo custo possuem diversas funções (sensoreamento, processamento, armazenamento e comunicação) e formam o principal ingrediente das redes de sensores sem fio (RSSFs).

Protocolos de segurança para RSSFs foram desenvolvidos para suportar comunicações *unicast*, ou seja, troca de mensagens entre apenas um par de nós (remetente e destinatário). Adaptações destes protocolos foram criadas para atender comunicações *broadcast*, principalmente aquelas em que a estação base envia mensagens para os nós sensores que compõem a rede. A diversidade dos protocolos existentes se explica pela dificuldade de se implementar um conjunto de mecanismos que ofereça todas as possíveis propriedades desejadas. Neste sentido, cada protocolo, seja ele *broadcast* ou *unicast*, apresenta pelo menos um elemento de desconforto que deve ser analisado para que um usuário interessado possa entender as possíveis implicações em determinadas aplicações.

1.1 Motivações e Desafios

O crescimento de aplicações para RSSFs e a limitada capacidade de processamento, armazenamento e energia dos dispositivos da rede criam um desafio para manter a segurança contra ataques de adversários com amplo poder computacional. Isto torna a escolha de um protocolo de segurança um processo complexo, já que aumentar a segurança implica em reduzir o tempo de funcionamento de dispositivos devido ao aumento do consumo de energia, além de outras possíveis implicações como queda de desempenho de dispositivos e limitação do número de participantes da rede.

A escolha de um protocolo de segurança pode ser auxiliada por uma análise teórica dos seus mecanismos, contemplando possíveis ataques, vulnerabilidades e limitações. Além disso, um ambiente de testes com sensores interconectados pode ajudar a entender o comportamento de um protocolo. Entretanto, a interação dos sensores com o ambiente pode não ser reproduzida de forma precisa, seja por utilização de uma quantidade insuficiente de equipamentos ou por dificuldade na inspeção de dados. Por isso, o uso de um simulador como um ambiente de testes pode também ser atrativo neste processo de escolha de um protocolo de segurança.

1.2 Objetivos

O objetivo deste trabalho é criar um guia na escolha e adaptação de protocolos de segurança de RSSFs, expondo vantagens e possíveis limitações. Quatro protocolos de grande atenção acadêmica foram escolhidos para análise, sendo que dois são esquemas de comunicação *unicast*, e o restante, *broadcast*. A simulação será feita apenas para protocolos *unicast*, restando aos demais protocolos uma comparação através da análise de seus mecanismos.

1.3 Organização do Documento

No Capítulo 2, são apresentadas as principais características de uma RSSF, incluindo exemplos de aplicações e questões de segurança. No Capítulo 3, estão descritos os principais mecanismos criptográficos utilizados em RSSFs. Os principais protocolos de comunicação *unicast* e *broadcast* são apresentados, respectivamente, nos Capítulos 4 e 5.

Os capítulos 6 e 7 compõem a parte principal deste trabalho, pois neles estão as análises comparativas dos protocolos de comunicação *unicast* e *broadcast*, respectivamente.

Finalmente, no Capítulo 8, apresentamos as considerações finais.

Capítulo 2

Conceitos

Antes de apresentarmos alguns protocolos de segurança e seus mecanismos criptográficos, é importante discutir sobre as redes de sensores sem fio. Neste capítulo, apresentamos as características de uma rede de sensores sem fio e algumas das principais propriedades desejadas para este tipo de rede, inclusive no que diz respeito à segurança e no bom funcionamento dos dispositivos que participam da rede.

2.1 Redes de Sensores Sem Fio

Uma rede de sensores sem fio (RSSF) é composta por um grande número de dispositivos distribuídos em uma região geográfica com a finalidade de monitorar ou afetar algum ambiente [1,76]. Esses dispositivos são chamados nós sensores e não possuem apenas capacidade sensorial, mas também recursos como processamento, memória e energia. Esses recursos são extremamente escassos e por isso algoritmos para esses equipamentos geralmente são desenvolvidos com o objetivo de minimizar o consumo, principalmente no que diz respeito à energia. Os nós de uma RSSFs são extremamente pequenos e se comunicam diretamente em distâncias curtas. A comunicação é realizada normalmente através de um rádio transceptor.

No contexto deste trabalho, iremos considerar apenas as RSSFs que possuem, além dos nós sensores, pelo menos uma *estação base*[1] ou *nó base*. Este nó funciona como uma porta de comunicação da rede com o mundo exterior, pois pode receber informações dos nós sensores e enviá-las para computadores pessoais e estações de trabalho distantes. A estação base possui recursos diferenciados e portanto não apresenta escassez de memória, processamento e energia. Além disso, a estação base pode enviar mensagens para os sensores. Na literatura é possível encontrar a definição de estação base equivalente à definição de *nó sorvedouro*[2], apesar de haver artigos que tratem esses conceitos como distintos [76]. Neste trabalho utilizaremos os conceitos nó sorvedouro, estação base e nó base de forma intercambiável, ou seja, consideramos todos esses conceitos como equivalentes. A figura 2.1 apresenta uma ilustração de uma RSSFs com diversos nós sensores (pontos) e uma estação base (quadrado).

[1] Do inglês, *base station*
[2] Do inglês, *sink node*

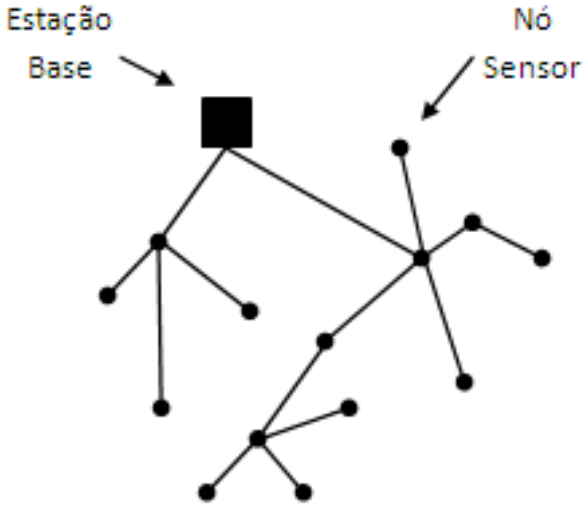


Figura 2.1: Arquitetura representativa de uma RSSFs [40]

2.1.1 Características

Uma RSSF possui muitas restrições se comparada a outros tipos de rede, como por exemplo as redes ad-hoc [1]. Além do grande número de dispositivos com poucos recursos, uma RSSF é tolerante a falhas, possui uma topologia que pode variar com muita frequência e possui nós que disseminam informações para outros nós em um esquema de comunicação multi-saltos[3]. A questão da tolerância a falhas é essencial, pois os dispositivos são de baixo custo e portanto estão sujeitos a problemas como falha de energia e danos físicos. Assim, os protocolos das RSSFs, sejam eles de segurança ou roteamento, são projetados em vista deste fato.

2.1.2 Aplicações

As RSSFs têm sido utilizadas para diversos propósitos. Algumas aplicações são o monitoramento de ambientes e estruturas [8, 54], vigilância em campos de batalha [17] e repostas a emergências médicas [92]. Muitas vezes os nós sensores são dispostos para funcionarem por um longo período de tempo, sendo que em alguns casos não há sequer a possibilidade de intervenção humana para recarregamento de baterias ou movimentação física [36]. A seguir, apresentamos dois exemplos de aplicações com alguns detalhes interessantes.

Monitoração Ambiental

Há uma dificuldade em se utilizar equipamentos para obter medidas em pequenas demarcações geográficas sem influenciar o ambiente. Por exemplo, a presença humana pode alterar comportamentos e comprometer resutados. Evitar este tipo de impacto e reduzir custos filtrando dados antes do envio de informações ou da tomada de ações pode ser feito através de uma RSSFs. Em [54], há detalhes de uma experiência como esta, em que o monitoramento do comportamento de pássaros

[3] Do inglês, *multihop*

em seus ninhos é feito através de 32 nós sensores (dispostos em Great Duck Island[4]), que processam e enviam informações diversas para uma estação base, a qual por sua vez está conectada à Internet. Os nós cooperam para efetuar tarefas complexas, como amostragem estatística e consolidação dos dados. Desta forma, é possível eliminar redundâncias e permitir que um usuário desta aplicação receba uma grande quantidade de informações pré-processadas, reduzindo o esforço para inferir os resultados finais.

Neste exemplo, o objetivo é obter informações como padrões de escolha de ninhos, entrada e saída do ninho em período de horas ou dias, alteração do ambiente ao redor em questão de meses e diferenças no ambiente antes e após a existência de ninhos, entre outras. Cada informação deve ser obtida a partir de um conjunto de medições de diferentes formas. Por exemplo, a presença de um pássaro no ninho e a existência de ovos são detectadas através de temperatura, a taxa de medição pode ser em horas ou meses, sensores de humidade podem ser utilizados para saber se o pássado saiu do ninho e teve acesso ao mar.

Um cientista pode também interagir com a rede de sensores através do envio de mensagens para a estação base. Isto pode ser feito a partir de um dispositivo móvel, se o cientista estiver na localidade, ou de um terminal conectado à Internet. Pode haver interesse em se interagir com a rede a partir da análise de resultados de medições iniciais, reajustando a forma com que os dados são obtidos.

A figura 2.2 apresenta a arquitetura geral do sistema de monitoração ambiental que utiliza RSSFs. Nesta arquitetura há duas RSSFs com diferentes frequências e alcance de rádio, as quais estão ligadas a uma estação base através de uma rede com largura de banda diferenciada, chamada de *Transit Network*. A estação base está conectada à rede mundial de computadores e a uma estrutura que inclui armazenamento com persistência dos dados e interfaces com o usuário.

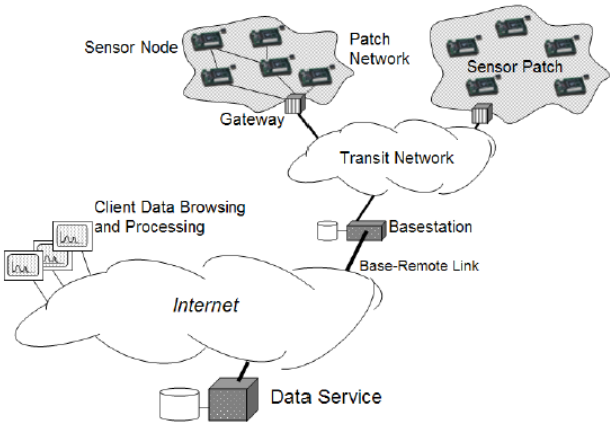


Figura 2.2: Arquitetura de um Sistema de Monitoração Ambiental [54].

[4]Uma pequena ilha entre os Estados Unidos e o Canadá.

Vigilância de Campos de Batalha

Em [17], é apresentado um projeto de detecção e classificação de objetos como movimentos de tropas e veículos em campos de batalha. Há o envio de informações para um centro de comando, o qual fornece suporte à sincronia de tempo dos nós sensores. É utilizada a troca de mensagens no modelo *multihop* (veja seção 2.1.3).

Este exemplo requer o uso de uma diversidade de protocolos para resolver questões como segurança, roteamento, difusão e consolidação da informação.

2.1.3 Modelos de Comunicação

Os nós sensores operam com potências extremamente baixas. Por isso, os pacotes transmitidos pela rede não podem alcançar longas distâncias. O padrão ZigBee [3], por exemplo, recomenda um alcance de transmissão de até 100 metros. Desta forma, para que um pacote chegue ao seu nó de destino muitas vezes é necessário que diversos saltos sejam dados, ou seja, entre um par de nós há um conjunto de nós distintos que permite o tráfego do pacote em grandes distâncias. Este esquema é chamado de transmissão ou comunicação *multihop*, ou ainda, *multi-saltos*.

Os três padrões de comunicação das RSSFs são os seguintes [40]:

- *Broadcast*: um único nó, tipicamente a estação base, envia mensagens para todos os demais nós da rede. Este padrão também é conhecido como um-para-muitos;

- *Unicast*: um par de nós realiza a troca de mensagens. O nó que envia as mensagens é chamado de remetente, sendo o outro chamado de receptor ou destinatário. Dizemos que dois nós são vizinhos se estão em locais de alcance do rádio de ambos, sem necessidade de saltos adicionais durante a comunicação. Neste padrão de comunicação não há necessidade dos nós serem vizinhos. O padrão *unicast* também é conhecido como um-para-um;

- Muitos-para-um: diversos nós sensores enviam mensagens para a estação base. Essas mensagens normalmente são informações processadas pelo nó a partir de dados obtidos através de sensores.

O esquema *multihop* pode ser utilizado em qualquer um dos padrões apresentados para que um pacote chegue ao seu nó destinatário. Outros modelos de comunicação não serão considerados neste trabalho.

2.1.4 Segurança em Redes de Sensores Sem Fio

Implementações de segurança implicam em sobrecustos que podem ser elevados em uma rede com equipamentos microeletrônicos. Alguns algoritmos criptográficos baseados em assinaturas digitais, por exemplo, podem requerer um sobrecusto de comunicação de 1000 bytes por pacote [66], o que seria impraticável em RSSFs, onde cada pacote possui aproximadamente 30 bytes de tamanho útil. Mesmo algoritmos muito utilizados atualmente como o RSA [70] e Diffie-Hellman [23] não seriam atrativos em RSSFs porque requerem um espaço de memória muito grande para os dispositivos da rede [66]. Além disso, o tamanho dos pacotes da rede, o consumo de espaço em

disco e o número de operações elementares dos dispositivos exercem grande influência no consumo de energia. Portanto a escolha de algoritmos eficientes em consumo de tempo e espaço é extremamente importante em uma RSSFs e torna o quesito segurança um desafio característico destas redes.

Uma forma de implementar segurança em RSSFs é através do fornecimento de propriedades como por exemplo confidencialidade e integridade de informações. As propriedades de segurança são obtidas através do uso de algoritmos ou mecanismos criptográficos. Esses algoritmos também são chamados de primitivas de segurança ou simplesmente primitivas. Chamaremos de propriedades desejadas todas as características ou propriedades que são atrativas em RSSFs, seja pelo nível de segurança ou desempenho. Antes de listar algumas propriedades desejadas em RSSFs iremos apresentar alguns ataques e vulnerabilidades em sensores e conceitos básicos de criptografia.

2.1.5 Ataques e Vulnerabilidades em Sensores

As RSSFs podem ser alvo de ataques que podem comprometer o funcionamento da rede como um todo. Isto pode acontecer em casos como ataques a algoritmos de roteamento que não implementam rotinas de segurança. Daí a importância de protocolos de segurança que eliminam vulnerabilidades e forneçam serviços para outros tipos de protocolos e até aplicações. A seguir, apresentamos alguns tipos de ataques em sensores.

Ataque *Sybil*

Um dispositivo que participa de uma RSSF pode ser ilegítimo, ou seja, não possuir autorização para trocar mensagens com os nós que compõem a rede. Quando um dispositivo ilegítimo assume múltiplas identidades, se fazendo passar por outros nós da rede, há um ataque *Sybil* [59]. Cada identidade do nó malicioso é chamada de *nó Sybil*. Este tipo de ataque pode ser utilizado principalmente contra algoritmos de roteamento em RSSF [40], os quais geralmente são vulneráveis porque não possuem uma camada de segurança.

Ataque Sorvedouro

Este ataque acontece quando há um nó comprometido ou malicioso que recebe grande parte das mensagens que trafegam por uma RSSF [19]. Este nó é disposto geograficamente de forma estratégica para receber as mensagens, podendo até alterá-las e replicá-las para a estação base.

A arquitetura de uma RSSF facilita este tipo de ataque porque geralmente todas as mensagens são direcionadas para um nó sorvedouro. Além disso, adversários podem alterar as mensagens de roteamento para tornar o nó malicioso atrativo, ou seja, fazer com que este nó seja um caminho comum para as mensagens da rede com destino ao nó sorvedouro.

Ataque *Loop*

Podem ser introduzidos *loops* na rede para que as mensagens fiquem circulando e não cheguem ao destinatário final. Isto pode gerar um maior consumo de energia dos dispositivos sensores [19].

Ataques Físicos e Comprometimento de Nós

Nós sensores não são resistentes a ataques físicos e podem ser de fácil acesso, pois geralmente estão dispostos em uma região geográfica para interação com um ambiente. A obtenção (física) do hardware de um dispositivo sensor pode ser feita com o objetivo de se descobrir as chaves criptográficas armazenadas [22]. Assim, um nó é comprometido e deixa de ser um participante legítimo da rede.

2.2 Criptografia Simétrica e Assimétrica

Dois modelos de troca de informações são extremamente importantes na área da criptografia computacional. São eles a criptografia simétrica e assimétrica. A criptografia simétrica se baseia na existência de uma chave secreta, a qual é compartilhada previamente entre todos os participantes da rede que desejam se comunicar de forma segura. Desta forma, uma mesma chave é utilizada para cifragem e decifragem. A figura 2.3 ilustra um esquema de cifragem e decifragem baseado em criptografia simétrica em que os participantes são Alice e Beto. O capítulo 3 explica o funcionamento de alguns algoritmos simétricos que são utilizados em RSSFs.

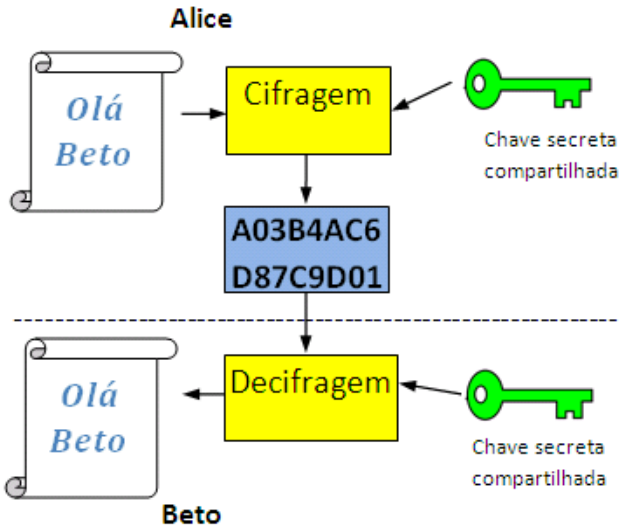


Figura 2.3: Cifragem e Decifragem em Criptografia Simétrica

Em 1976, Diffie e Hellman apresentaram um modelo que inspirou os algoritmos baseados em criptografia assimétrica ou de chave pública [23]. Os autores criaram o primeiro algoritmo que forneceu uma forma segura de distribuição de chaves. Assim, não havia a necessidade de um compartilhamento prévio das chaves como no caso dos algoritmos simétricos. Posteriormente, em 1978, Ron Rivest, Adi Shamir e Len Adleman utilizaram diferentes chaves para os processos de cifragem e decifragem [70]. Desta forma, uma chave é distribuída para todos de forma pública (chave pública), sendo utilizada para cifragem. Uma outra chave é guardada de forma segura (chave secreta) para uso no processo de decifragem. Este algoritmo é conhecido como RSA, nome derivado das iniciais de seus autores.

2.3 Propriedades de Segurança Desejadas em RSSFs

Os algoritmos criptográficos simétricos e assimétricos podem fornecer diversas propriedades de segurança que são desejadas em RSSFs. A existência dessas propriedades é um certificado de que alguns ataques e vulnerabilidades não são ignorados. As principais propriedades são apresentadas a seguir.

2.3.1 Confidencialidade de Informação e Segurança Semântica

Uma rede pode estar sujeita a atividades de participantes não autorizados, os quais chamaremos de adversários. Particularmente em RSSFs, adversários podem interceptar pacotes da rede em busca de informações secretas. A confidencialidade de informação é justamente a propriedade que evita a leitura destas informações, ou seja, mesmo que um adversário obtenha um pacote da rede ele não conseguiria recuperar a informação em tempo polinomial. Esta propriedade será mencionada simplesmente como confidencialidade.

A confidencialidade normalmente é obtida através de um esquema de cifragem. Quando a confidencialidade não apenas evita a recuperação total da informação, mas também protege textos cifrados de uma recuperação parcial dizemos que há também a propriedade de segurança semântica [55]. Isto quer dizer que um adversário não possui chance maior que 50% em responder uma pergunta do tipo sim ou não em relação a um texto cifrado [39]. Normalmente essa propriedade é oferecida através do uso de um ingrediente adicional em conjunto com um esquema de cifragem (veja seções 3.3.4 e 3.3.4).

O conceito de segurança semântica em criptografia assimétrica foi introduzido em [31], o qual inspirou uma adaptação interessante para criptografia simétrica em [9].

2.3.2 Autenticidade e Integridade de Mensagem

Um adversário pode participar da rede de várias formas. Além do recebimento de mensagens, nós sem autorização podem enviar pacotes a nós legítimos da rede. Um receptor de mensagens deve possuir mecanismos para saber se a informação recebida é realmente de uma fonte confiável e, em caso contrário, rejeitar a informação. Esta propriedade é chamada de autenticidade de mensagem ou simplesmente autenticidade.

O conceito de integridade de mensagem está relacionado com o de autenticidade: um adversário pode interceptar mensagens e efetuar alterações para posteriormente redirecioná-las a nós receptores legítimos, os quais por sua vez devem possuir mecanismos para detectar a existência dessas alterações. A integridade e a autenticidade podem ser fornecidas através de um mesmo mecanismo. Em RSSFs este mecanismo geralmente é um *Message Authentication Code* (MAC), o qual é descrito no capítulo 3.

Autenticação Imediata

Quando a autenticação acontece imediatamente após o recebimento de uma mensagem dizemos que a propriedade de autenticação imediata é satisfeita. É possível que mensagens recebidas sejam armazenadas para posterior autenticação. Por exemplo, um nó pode armazenar as mensagens rece-

bidas até que um parâmetro secreto (chave), necessário para a autenticação em tempo polinomial, seja revelado. Após a revelação da chave o nó realiza a autenticação das mensagens. O tempo de atraso, ou seja, o intervalo entre o recebimento das mensagens e a autenticação, representa a ausência da propriedade de autenticação imediata.

2.3.3 *Freshness*: Proteção Contra Ataques do Tipo *Replay*

Nós sensores são dispositivos que produzem uma resposta mensurável. Por exemplo, uma mudança de temperatura poderia ser detectada por um dispositivo sensor, o qual enviaria mensagens para um nó sorvedouro relatando o acontecimento. Portanto as mensagens que trafegam pela rede podem conter medidas que representam condições de um ambiente de acordo com o tempo. Se um adversário interceptar e armazenar mensagens da rede, este poderia enviá-las em qualquer momento para um nó sorvedouro com o objetivo de fraudar ou distorcer os dados de medidas dos sensores. O reenvio de mensagens legítimas por um adversário é chamado de ataque do tipo *replay*.

A autenticação e a confidencialidade não são suficientes para garantir a segurança contra ataques do tipo *replay*. Isto porque mensagens reenviadas serão decifradas e autenticadas, pois não foram alteradas após a interceptação. A principal forma de proteção contra ataques do tipo *replay* se baseia em atribuir um valor numérico para cada mensagem enviada. Assim, com o uso de um contador, é possível saber se uma mensagem faz parte de uma determinada sequência ou, informalmente, se uma mensagem é recente. Este tipo de proteção é chamada *Freshness* [39].

2.3.4 Resistência a Comprometimento de Nós da Rede

Os dispositivos das RSSFs são distribuídos em uma região geográfica para coleta de informações. Esta interação com o ambiente permite que um adversário possa realizar ataques físicos (veja seção 2.1.5). Desta forma, o dispositivo capturado se torna um nó comprometido. Como os nós sensores ainda não possuem proteção física contra estes ataques é interessante o uso de mecanismos de software para evitar o comprometimento da segurança de toda a rede, mesmo que um ou mais nós estejam comprometidos. Uma forma de proteção neste sentido é utilizar diferentes chaves para cada nó da rede. Isto requer um mecanismo de distribuição de chaves. O contrário disso seria utilizar uma única chave compartilhada entre todos os dispositivos da rede, ou seja, uma chave global. Neste caso, a obtenção de um único nó comprometeria toda a rede.

2.4 Demais Propriedades Desejadas em RSSFs

Além das propriedades de segurança, as quais são fornecidas principalmente por mecanismos criptográficos, outras três características são desejadas em uma RSSF. Essas características, apresentadas a seguir, não podem ser omitidas, pois poderiam impedir o fluxo de dados em uma RSSF com muitos dispositivos.

2.4.1 Baixo Sobrecusto Computacional e de Comunicacão

Como nós sensores possuem recursos computacionais limitados é ideal que os mecanismos de segurança tenham baixos sobrecustos computacionais. Além disso, um algoritmo com menor número de operações elementares representaria um consumo de energia menor. A redução de sobrecustos

computacionais é possível quando dois ou mais algoritmos com um mesmo propósito apresentam uma grande diferença de desempenho. Por exemplo, um processo de autenticação com o uso de assinatura digital é pelo menos duas vezes mais lento que com o uso de MAC [52]. Portanto a escolha de mecanismos para RSSFs não deve se basear somente em segurança, mas também em economia de recursos computacionais.

Sobrecustos de comunicação também representam um fator crítico. Cada bit transmitido pela rede pode consumir energia equivalente à execução de até 1000 instruções [35,39]. Logo a economia de energia também é feita através da redução do tamanho do pacote transmitido. Como o tamanho útil de cada pacote de uma RSSFs pode ter poucas dezenas de bytes não seria ideal o uso de mecanimos que requerem o envio de centenas de bytes, como por exemplo assinaturas de Merkle-Winternitz (MW) [52, 56, 57].

2.4.2 Resistência a perda de pacotes da rede

É possível que os dispositivos de uma RSSFs estejam em locais de difícil transmissão de dados. Isto poderia ocasionar uma alta taxa de perda de pacotes da rede. Se o projeto de um algoritmo de segurança não levar em consideração esta possibilidade é possível que se tenha impacto na segurança da rede. Por exemplo, o protocolo de autenticação HORS [52] utiliza algoritmos que são vulneráveis quando há uma determinada taxa de perda de pacotes da rede.

2.4.3 Escalabilidade

As RSSFs podem possuir um número muito elevado de dispositivos que enviam e recebem mensagens. É possível que um protocolo seja projetado a partir de restrições como número máximo de nós que podem enviar mensagens. Desta forma, a propriedade de escalabilidade não seria fornecida, pois o crescimento do número de nós tornaria inviável o uso do protocolo [66]. Algumas propriedades, como por exemplo *Freshness*, requerem um estado por remetente [51], ou seja, um determinado recurso computacional exclusivo para cada nó remetente da rede. Se este recurso adicional for constante em relação ao número de nós o protocolo permite que a rede seja escalável.

2.5 Protocolos de Segurança

Um protocolo é constituído por uma série de passos projetados para realizar uma tarefa, sendo que esses passos envolvem pelo menos duas partes [78]. A sequência de passos de um protocolo deve ser respeitada do começo ao fim para que se possa realmente fazer o que o protocolo promete. O envolvimento de pelo menos duas partes é essencial nesta definição. Isto quer dizer que dois ou mais participantes são necessários para completar os passos definidos por um protocolo. Como exemplo de participantes podemos citar pessoas e dispositivos eletrônicos.

Um protocolo de segurança utiliza algoritmos criptográficos para fornecer propriedades de segurança. Neste trabalho descreveremos os protocolos de segurança através da forma que cada propriedade de segurança é oferecida.

Capítulo 3

Criptografia Simétrica em RSSFs

Neste capítulo, apresentamos as primitivas de segurança que são utilizadas pelos protocolos estudados neste trabalho. Essas primitivas são compostas apenas por mecanismos criptográficos simétricos. Isto acontece porque os algoritmos de chave pública representam maiores sobrecustos computacionais e de comunicação [26, 39, 52] e portanto ainda não são tão atrativos em RSSFs.

3.1 Notações

Algumas notações utilizadas neste trabalho são apresentadas a seguir. Outras notações serão mostradas em momento oportuno.

- $(a \oplus b)$ denota a operação *ou-exclusivo* (bit a bit) a partir das cadeias a e b.
- $(a||b)$ representa a concatenação das cadeias a e b.
- $\{0,1\}^n$ representa o conjunto de cadeias binárias de comprimento n.

3.2 Message Authentication Codes (MAC)

Considere o seguinte problema:

Beto recebe uma mensagem que foi supostamente enviada por Alice. Como Beto pode ter garantias de que a mensagem realmente veio da Alice? Além disso, como Beto pode saber se a mensagem recebida foi alterada por um terceiro mal intencionado?

Estes problemas são resolvidos atualmente através de algoritmos de assinatura, principalmente com criptografia de chave pública [49, 89, 91]. Apesar disso, existe um mecanismo de melhor desempenho que utiliza criptografia simétrica [52], o qual é chamado de Message Authentication Code.

O MAC é um algoritmo de autenticação que recebe como entrada uma chave e uma mensagem, e devolve um valor numérico chamado *tag de autenticação* ou simplesmente *tag*. Em um algoritmo do tipo MAC a tag também é conhecida como *valor MAC*, o qual permite que a verificação de autenticidade de uma mensagem seja feita somente pelo possuidor da chave secreta. A figura 3.1 ilustra o processo. Considere que a chave K está compartilhada entre um remetente e um destinatário.

É interessante observar que o valor MAC calculado pelo remetente é enviado com a mensagem. Isto acontece porque um receptor também calcula este valor e o compara com aquele anexado

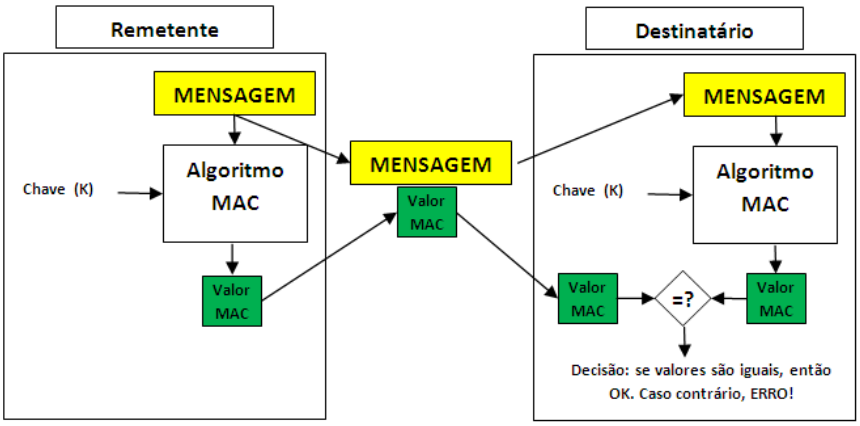


Figura 3.1: Autenticação com Message Authentication Code

pelo remetente na mensagem. Se os valores forem iguais a mensagem é aceita, caso contrário a mensagem é descartada.

3.2.1 Cálculo do valor MAC

O uso de uma cifra de bloco para calcular o valor MAC é um método muito estudado. O CBC-MAC, por exemplo, opera com uma cifra no modo CBC e é muito utilizado em RSSFs [39, 66]. Além disso, este algoritmo é mencionado em diversos padrões, incluindo [29, 62]. Na seção 3.3.4 apresentamos este algoritmo.

3.2.2 Propriedades de um algoritmo MAC

Seja uma chave simétrica k, um texto puro p_i e um valor MAC v_i, com i =1,2,3,... Um algoritmo MAC deve satisfazer as seguintes propriedades:

1. **o cálculo de v_i é computacionalmente fácil** com o conhecimento de k e p_i. Neste caso o algoritmo devolve v_i em tempo polinomial.

2. **o cálculo de v_i é computacionalmente difícil** para um adversário que não possui k, mas conhece um ou vários pares (p_i, v_i).

3. p_i possui tamanho arbitrário e v_i possui tamanho fixo.

3.3 Cifras Simétricas

O método mais utilizado para fornecer confidencialidade de informação é a cifragem. Em RSSFs a cifragem é realizada principalmente através de algoritmos simétricos, pois os algoritmos de chave pública apresentam menor desempenho [26]. O algoritmo de chave pública RSA [70], por exemplo, apresenta péssimos resultados experimentais em relação ao consumo de energia [28, 88] quando implementado em dispositivos de RSSFs. Chamamos os algoritmos simétricos para cifragem de cifras simétricas.

O funcionamento das cifras simétricas se baseia na aplicação de técnicas de substituição e transposição. Em [79], Claude Shannon descreve a substituição através do termo *confusão*, o qual tem como objetivo aumentar a complexidade da relação entre a chave e o texto cifrado, e a transposição através do conceito de *difusão*, que tem a função de eliminar redundâncias no texto cifrado.

As cifras simétricas são divididas em duas principais classes: cifras de fluxo[1] e cifras de bloco[2]. As cifras de bloco geralmente são mais lentas que as cifras de fluxo, mas são muito utilizadas em RSSFs por apresentarem bom nível de segurança [39, 84]. Rueppel [72] apresentou o seguinte resumo sobre a diferença entre cifras de bloco e fluxo:

> Cifras de bloco operam com transformações fixas em grandes blocos de texto puro. Cifras de fluxo operam com transformações que variam com o tempo em dígitos de texto puro.

A seguir, apresentamos com formalidade o significado de um esquema de cifragem (simétrico). Em seguida, descrevemos os conceitos de cifras de fluxo e cifras de bloco.

3.3.1 Esquema de Cifragem

Um esquema de cifragem é usado para evitar que pessoas sem autorização possam ler informações confidenciais. A autorização poderia ser concedida através da posse de uma chave, a qual pode ser um valor numérico combinado entre os participantes do esquema. Antes de passarem por este processo as informações são chamadas de texto puro, texto claro, ou texto legível[3](neste trabalho usamos estes termos de forma intercambiável) e após, texto cifrado. A seguir, definimos este esquema com base em [83, 86].

Definição 1. *Seja P um conjunto de textos puros, $\mathcal{K}$ um conjunto de chaves e C um conjunto de textos cifrados. Um esquema de cifragem simétrico $SE = (Gen, E, D)$ consiste de três algoritmos:*

1. *O algoritmo probabilístico Gen devolve uma chave $k \in \mathcal{K}$, o que denotamos como $k \xleftarrow{R} Gen()$.*
2. *O algoritmo E recebe uma chave $k \in \mathcal{K}$ e um texto puro $m \in P$ e devolve um texto cifrado $c \in C$, o que denotamos como $c \leftarrow E_k(m)$.*
3. *algoritmo D recebe uma chave $k \in \mathcal{K}$ e um texto cifrado $c \in C$ e devolve o texto puro $m \in P$ ou o símbolo $\perp$ que representa uma decifragem incorreta.*

É necessário que seja satisfeita a equação $D_k(E_k(m)) = m$ para todo $m \in P$, $k \in \mathcal{K}$.

Neste esquema, G é o algoritmo de geração de chaves. E é o algoritmo de cifragem e D, o de decifragem.

[1] Do inglês, *stream cipher*
[2] Do inglês, *block cipher*
[3] Do inglês, *plaintext*

3.3.2 Cifras de Fluxo

Há uma grande variedade de projetos de cifras de fluxo, os quais têm sido alvo de grande interesse pela comunidade européia. O projeto eSTREAM, por exemplo, foi uma iniciativa do European Network of Excellence for Cryptology (ECRYPT) para escolher cifras de fluxo com alto desempenho em hardware e software [7]. Além disso, a principal conferência européia de criptografia, o Eurocrypt, possui em seu histórico sessões dedicadas à apresentação de trabalhos sobre cifras de fluxo.

Esta popularidade das cifras de fluxo é principalmente atribuída a um trabalho de Shannon [79] sobre o *one-time-pad*, o qual é conhecido como **Cifra de Vernam** [85]. Neste trabalho, Shannon descreve o *one-time-pad* como um criptossistema incondicionalmente seguro, pois mesmo um adversário com capacidade computacional infinita não teria melhor chance de um simples palpite em descobrir o valor de um único bit do texto cifrado. Isto acontece porque o one-time-pad utiliza chaves aleatórias distintas e do mesmo tamanho do texto puro. Infelizmente a Cifra de Vernam é um criptossistema praticamente inviável. Além da dificuldade com o gerenciamento de chaves, os computadores podem gerar apenas números pseudo-aleatórios.

Uma cifra de fluxo se baseia na idéia da Cifra de Vernam, porém utiliza-se uma chave de tamanho relativamente pequeno para gerar um *keystream* pseudo-aleatório. A seguir, apresentamos a definição de *keystream*, que será utilizada posteriormente para definirmos as cifras de fluxo. As definições se baseiam em [55].

Definição 2. *Seja $\mathcal{K}$ o espaço de chaves utilizado por um esquema de cifragem. Uma sequência de símbolos $e_1e_2e_3...e_i \in \mathcal{K}$ é um keystream.*

Definição 3. *Seja E_e um algoritmo de cifragem que opera com blocos de tamanho 1, sendo $e \in \mathcal{K}$. Seja $m_1m_2m_3...$ uma cadeia de caracteres (texto puro) e $e_1e_2e_3...$ um* keystream *em $\mathcal{K}$. Uma cifra de fluxo é um algoritmo de cifragem que recebe duas entradas - o texto puro e o* keystream *- e devolve o texto cifrado $c_1c_2c_3...$, em que $c_i \leftarrow E_{e_i}(m_i)$.*

Como as cifras de fluxo operam com pequenos blocos (geralmente 1 bit ou 1 byte), os símbolos são tratados um por vez, ou seja, cada símbolo é combinado com um elemento do *keystream*.

3.3.3 Cifras de Bloco

Definição 4. *Uma cifra de bloco é uma função $E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n$ tal que: para cada $K \in \{0,1\}^k$, $E_K : \{0,1\}^n \rightarrow \{0,1\}^n$ é inversível e definida como $E_K(P) = E(K,P)$. E_K^{-1} é a função inversa de E_K tal que $E_K^{-1}(E_K(P)) = P$ e $E_K(E_K^{-1}(C)) = C$ para todo $P, C \in \{0,1\}^n$.*

Dizemos que E é a *função de cifragem*, a qual converte um texto puro de n bits em um texto cifrado do mesmo tamanho, com o controle de uma chave de k bits. A inversa de E, que denotaremos como D, é a *função de decifragem*, a qual converte um texto cifrado de n bits em um texto puro do mesmo tamanho, com o controle da mesma chave que foi utilizada na função de cifragem. Além disso, E e D devem ser facilmente computáveis.

Dizemos que n é o tamanho de bloco, sendo que n e k são parâmetros associados a uma cifra de bloco específica. A cifra Skipjack (veja seção 3.3.5), por exemplo, possui tamanho de bloco igual a 64 bits e tamanho de chave igual a 80 bits.

Muitas cifras de bloco são iterativas, ou seja, consistem de repetições (*rounds*) de uma mesma função. Neste caso, cada vez que a função é aplicada há o uso de uma nova chave (*sub-chave*), a qual é derivada de uma chave mestra através de um processo chamado *escalonamento de chaves*.

3.3.4 Modos de Operação

As cifras de bloco operam com cadeias de texto puro de tamanho igual a n bits, ou seja, com blocos de n bits. Quando o tamanho da entrada ultrapassa o de bloco há um particionamento da mensagem para que esta seja cifrada de forma iterativa. Os métodos utilizados para que uma cifra de bloco opere com mensagens longas são chamados de *modos de operação*. Quando a mensagem é menor que n bits deve ser feito um processo de preenchimento do espaço de mensagem até que se tenha o tamanho de bloco. Isto é conhecido como *padding* [24].

Nas próximas seções, apresentamos os modos de operação que são utilizados pelos protocolos de segurança estudados neste trabalho.

Modo Counter (CTR)

Apresentado em 1979 por Diffie e Hellman [48], o modo CTR é recomendado pelo National Institute of Standards and Technology (NIST) em uma publicação especial do ano de 2001 [24]. Este modo possui um comportamento semelhante ao de uma cifra de fluxo. A cifragem ocorre da seguinte forma: uma sequência de contadores e uma chave são utilizados como entrada para uma determinada cifra de bloco. Isto gera uma saída que é então somada (operação ou-exclusivo) com uma sequência de blocos de texto puro. Cada sequência de texto puro é cifrada separadamente e de forma independente. A figura 3.2 ilustra este processo, assim como a decifragem.

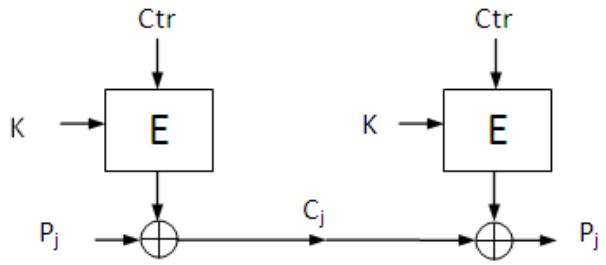


Figura 3.2: Cifragem e Decifragem no modo CTR.

Sejam $P_1, P_2, ..., P_n$ sequências de blocos de texto puro e $C_1, C_2, ..., C_n$, blocos cifrados. Sejam ainda $T_1, T_2, ..., Tn$ uma sequência de contadores. Denotamos a cifragem e decifragem no modo CTR da seguinte forma:

Cifragem: $O_j \leftarrow E_K(T_j),\ j = 1, 2, ..., n$
$C_j \leftarrow O_j \oplus P_j,\ j = 1, 2, ..., n$

Decifragem: $O_j \leftarrow E_K(T_j),\ j = 1, 2, ..., n$
$P_j \leftarrow O_j \oplus C_j,\ j = 1, 2, ..., n$

É interessante observar que a mesma função é utilizada para cifragem e decifragem e que o tamanho do texto cifrado é o mesmo do texto puro.

Modo OCB (Offset CodeBook)

Este modo de operação é um algoritmo que recebe três entradas - uma chave, um valor inicial (VI) e um texto puro - e devolve um texto cifrado e uma tag de autenticação. Isto significa que o modo OCB não é utilizado apenas para cifragem, pois oferece confidencialidade e autenticação em um só passo, ou seja, através de apenas um algoritmo. Neste modo, o VI é chamado de *Number Used Once* (NONCE), ou ainda, nonce, o qual não pode ser reutilizado durante as iterações do algoritmo. O remetente geralmente utiliza como NONCE um contador e não é necessário que este seja imprevisível ou secreto.

A seguir, apresentaremos o algoritmo de cifragem com o auxílio da figura 3.3. Esta ilustração deve ser interpretada da esquerda para a direita e de cima para baixo. Maiores detalhes sobre os parâmetros e operações podem ser encontrados em [75].

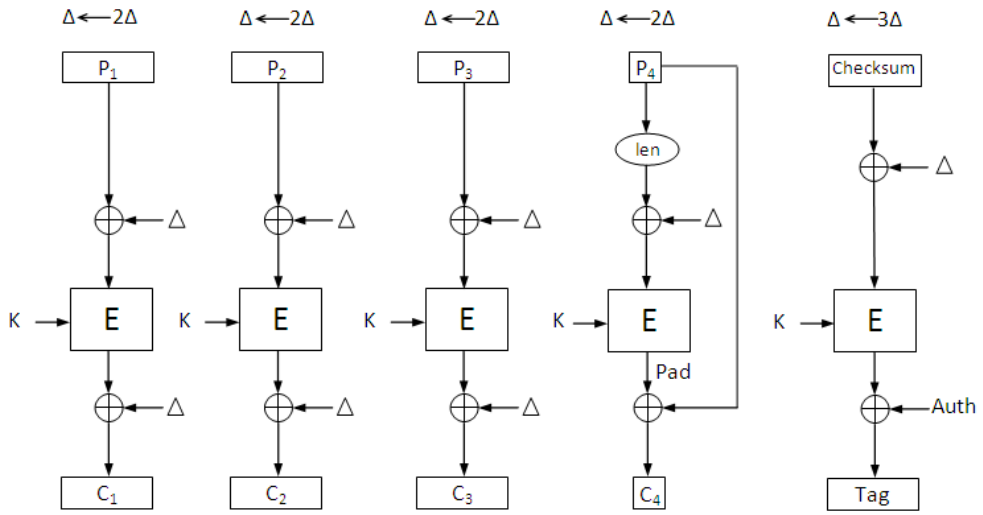


Figura 3.3: Cifragem e Autenticação no modo OCB.

Seja S uma cadeia binária de tamanho menor ou igual a n bits. Para exemplificar escolheremos $n = 128$. Definimos $S \ll 1$ como a retirada do primeiro bit de S e a inclusão de 0 como seu último bit[4], ou seja, o deslocamento à esquerda de S em 1 bit . Definimos $2S$ como a seguir:

$$2S = \begin{cases} S \ll 1 & \text{se o primeiro bit de } S \text{ é } 0 \\ (S \ll 1) \oplus \text{0x87} & \text{se o primeiro bit de } S \text{ é } 1 \end{cases}$$

[4]O primeiro bit é chamado de *bit mais significativo* e o último, *bit menos significativo*.

O valor 0x87 representa o número 87 em hexadecimal. Este valor pode variar de acordo com o tamanho máximo escolhido para S. Definimos $3S$ como $2S \oplus S$, e $len(x)$ como a cadeia binária que representa o tamanho de x. Se S é uma cadeia binária com até 128 bits de tamanho então definimos $S0^*$ como a cadeia binária com tamanho igual a 128 bits, obtida a partir da concatenação de um número suficiente de zeros para se obter tal tamanho de cadeia.

Sejam P um texto puro e C um texto cifrado. Sejam ainda H um *header* opcional associado a P, K uma chave e N um NONCE de 128 bits. A cifragem ocorre da seguinte forma:

Cifragem: Particione P nos blocos $P_1, P_2, ..., P_{m-1}$ de 128 bits e no bloco P_m de até 128 bits
$\Delta \leftarrow E_K(N)$
A cifragem ocorre como na figura 3.3
Checksum $\leftarrow P_1 \oplus P_2 \oplus ... \oplus P_{m-1} \oplus (C_m 0^* \oplus \text{Pad})$
O resultado é C, formado pelos blocos $C_1, C_2, ..., C_m$, e a tag de autenticação

Logo, $(C, Tag) \leftarrow OCB_K(N, P)$, em que (C, Tag) é enviado como mensagem por um remetente. É possível que se use apenas os primeiros bits da tag, ou seja, um prefixo T, enviando-se assim (C, T) como mensagem.
Por que H não foi utilizado? E o valor de $Auth$?
O cálculo de $Auth$ é feito através de um algoritmo que possui H como entrada. No contexto deste trabalho não utilizaremos um *header* e, desta forma, definimos $Auth$ como uma cadeia de zeros de tamanho 128 bits, ou ainda, 0^{128}. Este método geralmente é utilizado em protocolos de RSSFs. Um algoritmo para o cálculo de $Auth$ a partir de H pode ser encontrado em [75] e [73].

Modo CBC (Cipher Block Chaining)

No modo CBC [9, 24] cada bloco de texto puro é somado (operação ou-exclusivo) com o bloco anterior cifrado. Este modo requer que o primeiro bloco de texto puro seja somado com um valor inicial (VI) , o qual não precisa ser secreto, mas é importante que seja imprevisível. O uso de um simples contador, por exemplo, não é recomendável. A figura 3.4 ilustra a cifragem neste modo.

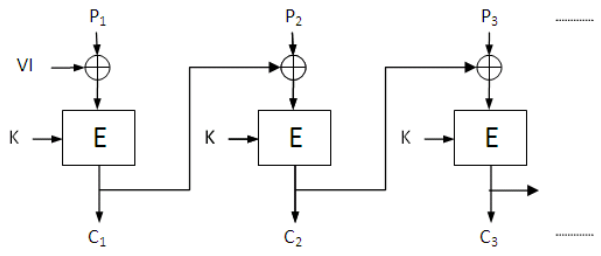


Figura 3.4: Cifragem no modo CBC.

Evitar a repetição do VI é uma prática importante no modo CBC, já que sem isso blocos de texto puro idênticos resultam em blocos de texto cifrado também iguais. Por outro lado, a alteração do VI, chave ou primeiros blocos do texto puro não é obrigatória, mas resulta em diferentes textos

cifrados, o que, aliás, é necessário para se obter segurança semântica (veja seção 2.3.1). A seguir, denotamos o modo CBC.

Sejam $P_1, P_2, ..., P_n$ sequências de blocos de texto puro e $C_1, C_2, ..., C_n$, blocos cifrados. Temos que:

$$\text{Cifragem:} \quad \begin{aligned} &C_1 \leftarrow E_K(P_1 \oplus VI) \\ &C_j \leftarrow E_K(P_j \oplus C_{j-1}),\ j = 2, ..., n \end{aligned}$$

$$\text{Decifragem:} \quad \begin{aligned} &P_1 \leftarrow D_K(C_1) \oplus VI \\ &P_j \leftarrow D_K(C_j) \oplus C_{j-1},\ j = 2, ..., n \end{aligned}$$

Ciphertext Stealing

Um algoritmo de cifragem no modo CBC devolve um texto cifrado que pode ser maior que o texto puro de entrada. Isto acontece porque o tamanho do texto cifrado é um múltiplo do tamanho de bloco da cifra. Impedir este aumento de tamanho pode ser interessante, mas requer o uso de uma técnica conhecida como *ciphertext stealing* [61].

Autenticação com CBC-MAC

Cifras de bloco no modo CBC são recomendadas para o cálculo de um valor MAC (veja seção 3.2). Isto é feito para que seja possível verificar a autenticidade de uma mensagem, por exemplo.

Assim como o CBC utilizado para cifragem, o CBC-MAC soma (ou-exclusivo) cada bloco de texto puro com o bloco de texto cifrado na iteração precedente. Porém, há duas diferenças principais: o CBC-MAC devolve um único bloco de dados (valor MAC) e o valor inicial (VI) é zero, ou ainda, uma cadeia de zeros. A figura 3.5 ilustra o cálculo do valor MAC através deste processo. Algumas notações serão apresentadas a seguir para facilitar futuras discussões.

Seja $f : \{0,1\}^n \rightarrow \{0,1\}^n$ uma função e m um inteiro positivo. Denotamos a função $f^{(m)} : \{0,1\}^{mn} \rightarrow \{0,1\}^n$ como a função de uma cadeia de entrada de mn-bit $M = M_1, M_2, ..., M_m$, com o tamanho de M_i igual a n, em uma cadeia de saída Y_m de n-bit, calculada como a seguir:

$$\begin{aligned} &Y_0 \leftarrow 0^n \\ &Y_i \leftarrow f(Y_{i-1} \oplus M_i) \text{ para } i = 1, ..., m \end{aligned}$$

Também utilizaremos os termos CBC-MAC e CMAC (Cipher Message Authentication Code) de forma intercambiável[5].

A escolha da cifra de bloco é extremamente importante para o desempenho de um algoritmo do tipo CBC-MAC. Uma cifra muito utilizada em RSSFs é o Skipjack, o qual é apresentado na próxima seção.

[5] O termo CMAC normalmente é utilizado para todos os mecanismos de autenticação baseados em cifras de bloco. Apesar disso, utilizaremos CMAC somente como referência ao algoritmo CBC-MAC.

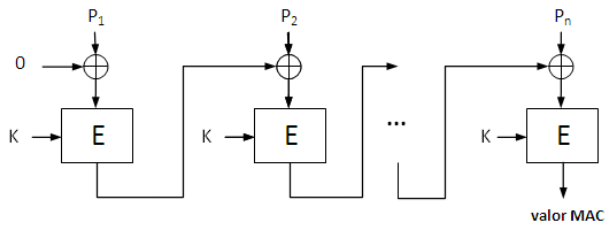


Figura 3.5: Cálculo do valor MAC através de uma cifra de bloco no modo CBC.

Confidencialidade e autenticação de mensagens em RSSFs podem ser fornecidas através do uso dos algoritmos CBC e CBC-MAC. Apesar da economia de código devido à semelhança entre esses algoritmos, há necessidade de dois passos de execução, sendo um para confidencialidade (CBC) e outro para autenticação (CBC-MAC).

3.3.5 A Cifra Skipjack

A cifra Skipjack [60], desenvolvida pela US National Security Agency (NSA)[6], é utilizada por alguns protocolos de segurança para RSSFs que são estudados neste trabalho. Trata-se de uma cifra de bloco com tamanho de chave igual a 80 bits, tamanho de bloco igual a 64 bits e número de *rounds* igual a 32. Cada *round* é composto por uma sequência de passos que é chamada de regra, sendo que existem somente duas regras distintas, A e B, descritas na tabela 3.1, em que $k+1$ é o número do *round* atual, w_1, w_2, w_3 e w_4 são sequências de blocos de 16 bits (texto puro) e G é a permutação ilustrada na figura 3.6. Os 32 *rounds* são formados por execuções alternadas entre as regras A e B da seguinte forma: 8 iterações da regra A seguidas de 8 iterações da regra B e, em seguida, repete-se esta execução.

Regra A	Regra B
$w_1^{k+1} \leftarrow G^k(w_1^k, \text{sub-chave}^k) \oplus w_4^k \oplus (k+1)$	$w_1^{k+1} \leftarrow w_4^k$
$w_2^{k+1} \leftarrow G^k(w_1^k, \text{sub-chave}^k)$	$w_2^{k+1} \leftarrow G^k(w_1^k, \text{sub-chave}^k)$
$w_3^{k+1} \leftarrow w_2^k$	$w_3^{k+1} \leftarrow w_1^k \oplus w_2^k \oplus (k+1)$
$w_4^{k+1} \leftarrow w_3^k$	$w_4^{k+1} \leftarrow w_3^k$

Tabela 3.1: Regras A e B da Cifra Skipjack

A Permutação G

Quatro *rounds* compõem a permutação G, também conhecida como Estrutura de Feistel ou Permutação de Feistel. Cada *round* utiliza uma sub-chave de 8 bits, denotada como cv(*cryptovariable*) na figura 3.6, e uma função F, chamada de *Tabela F*. Esta tabela também é conhecida como *Substitution Box* (S-box), um ingrediente muito utilizado em cifras de bloco. A tabela 3.2 é uma representação de F em notação hexadecimal. Esta tabela é indexada da seguinte forma: os 4 bits mais à esquerda de um dado de entrada representam o índice da linha e os 4 bits restantes (mais

[6] Agência Nacional de Segurança dos Estados Unidos

à direita) representam o índice da coluna. Por exemplo, $F(7a) = d6$.

A permutação G, ilustrada na figura 3.6, é representada a seguir:

$g_5 \| g_6 \leftarrow G^k(w = g_1 \| g_2)$, em que $g_i \leftarrow F(g_{i-1} \oplus \text{cv}_{4k+i-3}) \oplus g_{i-2}$ e k é o número do *round* e cv_{4k+i-3} é a sub-chave formada pelo $(4k + i - 3)$-ésimo byte da chave mestra. Assim:

$g_3 \leftarrow F(g_2 \oplus \text{cv}_{4k}) \oplus g_1$
$g_4 \leftarrow F(g_3 \oplus \text{cv}_{4k+1}) \oplus g_2$
$g_5 \leftarrow F(g_4 \oplus \text{cv}_{4k+2}) \oplus g_3$
$g_6 \leftarrow F(g_5 \oplus \text{cv}_{4k+3}) \oplus g_4$

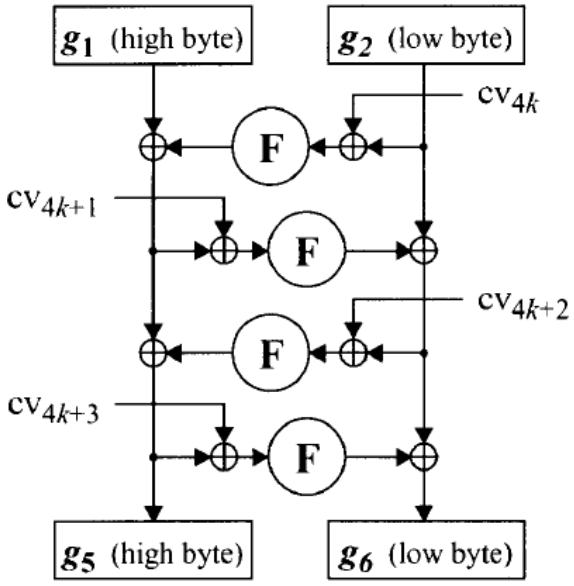


Figura 3.6: Diagrama da permutação G [60].

Chave e Sub-chaves

A chave mestra possui tamanho igual a 10 bytes. Como cada sub-chave possui tamanho igual a 1 byte, o cálculo da sub-chave cv, definido na permutação G como o $(4k + i - 3)$-ésimo byte da chave mestra, deve ser interpretado como uma operação *mod*-10, ou seja, como o resultado sendo um número inteiro no intervalo $[0, 9]$.

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
x0	a3	d7	09	83	f8	48	f6	f4	b3	21	15	78	99	b1	af	f9
x1	e7	2d	4d	8a	ce	4c	ca	2e	52	95	d9	1e	4e	38	44	28
x2	0a	df	02	a0	17	f1	60	68	12	b7	7a	c3	e9	fa	3d	53
x3	96	84	6b	ba	f2	63	9a	19	7c	ae	e5	f5	f7	16	6a	a2
x4	39	b6	7b	0f	c1	93	81	1b	ee	b4	1a	ea	d0	91	2f	b8
x5	55	b9	da	85	3f	41	bf	e0	5a	58	80	5f	66	0b	d8	90
x6	35	d5	c0	a7	33	06	65	69	45	00	94	56	6d	98	9b	76
x7	97	fc	b2	c2	b0	fe	db	20	e1	eb	d6	e4	dd	47	4a	1d
x8	42	ed	9e	6e	49	3c	cd	43	27	d2	07	d4	de	c7	67	18
x9	89	cb	30	1f	8d	c6	8f	aa	c8	74	dc	c9	5d	5c	31	a4
xA	70	88	61	2c	9f	0d	2b	87	50	82	54	64	26	7d	03	40
xB	34	4b	1c	73	d1	c4	fd	3b	cc	fb	7f	ab	e6	3e	5b	a5
xC	ad	04	23	9c	14	51	22	f0	29	79	71	7e	ff	8c	0e	e2
xD	0c	ef	bc	72	75	6f	37	a1	ec	d3	8e	62	8b	86	10	e8
xE	08	77	11	be	92	4f	24	c5	32	36	9d	cf	f3	a6	bb	ac
xF	5e	6c	a9	13	57	25	b5	e3	bd	a8	3a	01	05	59	2a	46

Tabela 3.2: Tabela F. A S-box do Skipjack

Capítulo 4

Protocolos de Segurança para Comunicação *Unicast*

Há uma diversidade de protocolos de segurança para RSSFs que foram desenvolvidos para comunicação *unicast*. Protocolos que adotam criptografia de chave pública não são tão eficientes quanto aqueles que utilizam esquemas criptográficos simétricos. Isto acontece porque esquemas assimétricos geralmente utilizam chaves de tamanho considerável ou operações custosas [15, 26, 49]. O tamanho de chave simétrica de 80 bits é equivalente a 1248 bits para esquemas baseados em RSA e 160 bits para aqueles baseados em curvas elípticas. Estes tamanhos são atualmente recomendados contra adversários computacionalmente fortes [6, 43]. O modelo de curvas elípticas com 160 bits de tamanho de chave seria bastante interessante se as operações com pontos da curva não fossem tão custosas [26]. É possível que novos métodos sejam descobertos de forma que os esquemas assimétricos atuais tornem-se bastante atrativos em redes de sensores. Alguns algoritmos baseados em criptografia de curvas elípticas já apresentam melhora significativa [63, 81].

Os esquemas simétricos utilizados em protocolos *unicast* também possuem suas limitações. Uma grande dificuldade é a distribuição de chaves, a qual pode ser essencial em algumas situações. Por exemplo, uma chave compartilhada entre um par de nós da rede poderia ser obtida através da captura de um dos dispositivos[1]. Se a chave estiver compartilhada entre muitos nós o problema da distribuição de chaves seria vital para a segurança da rede.

A seguir, apresentamos os principais protocolos de segurança para RSSFs que são baseados em esquemas simétricos. Depois detalharemos o funcionamento dos protocolos Minisec-U e TinySec, os quais são parte importante deste trabalho.

4.1 Protocolos Baseados em Esquemas Simétricos

A tabela 4.1 apresenta os principais protocolos para comunicação *unicast* que possuem apenas mecanismos simétricos.

O único protocolo apresentado na tabela 4.1 que não foi desenvolvido especificamente para RSSFs é o Wired Equivalent Privacy (WEP) [5, 18]. Este protocolo é de interessante estudo, pois foi pioneiro em utilizar uma cifra de fluxo para fornecer segurança em redes WLAN [5], em que a comunicação sem fio é feita entre computadores pessoais. Isto influenciou a criação dos demais protocolos mencionados na tabela.

[1] Isto pode acontecer em redes de sensores já que há alta interação dos dispositivos com o ambiente onde um fenômeno é observado

Protocolo	**Cifragem**	**Autenticação**
WEP	RC4	CRC
SPINS	CTR-RC5	CBC-MAC
TinySec	CBC-Skipjack	CBC-MAC
ZigBee	CTR-AES	CBC-MAC
MiniSec	OCB-Skipjack	OCB

Tabela 4.1: Protocolos de Segurança para Comunicação *Unicast*.

Como as mais rápidas cifras de fluxo são mais eficientes que as mais rápidas cifras de bloco [39, 84] podemos ter a forte intuição de que seu uso é altamente recomendado em RSSF. Uma rede com dispositivos com recursos muito limitados não poderia utilizar algoritmos considerados menos eficientes, já que desta forma o consumo de energia não seria minimizado. Apesar da eficiência em consumo de tempo e energia, algoritmos criptográficos baseados em cifras de fluxo não são interessantes porque há situações que podem causar grande vulnerabilidade. A repetição de Valores Iniciais (VI) representam um grande risco para as cifras de fluxo, principalmente se estes algoritmos forem executados em dispositivos com pouca memória disponível. Muitos ataques ao protocolo WEP foram apresentados neste sentido [18, 82].

O protocolo WEP também possui vulnerabilidades em seu método de autenticação. A utilização de um *Cyclic Redundant Check* (CRC) como *checksum* é interessante para verificar erros de transmissão, mas não é segura contra ataques de adversários. A ausência de uma chave para o cálculo do *checksum* é a principal causa dos problemas de segurança [18]. Portanto este protocolo não fornece segurança adequada no quesito autenticação e confidencialidade.

Outros protocolos apresentam melhorias de segurança em relação ao conceito do protocolo WEP. O Secure Network Encryption Protocol (SNEP) [66] é parte de um conjunto de protocolos da família *Security Protocols for Sensor Networks* (SPINS). O SNEP utiliza um mecanismo robusto de autenticação, o CBC-MAC, mas apresenta problemas de segurança ao utilizar a cifra de bloco RC5 [71] com o modo de operação CTR (veja também a seção 3.3.4)para fornecer confidencialidade. O modo CTR opera de forma semelhante a uma cifra de fluxo, pois utiliza um tamanho de bloco muito pequeno e portanto o mesmo problema do protocolo WEP pode acontecer, ou seja, a segurança diminui com o reuso de VIs, que são contadores no modo CTR. Se os VIs não forem reutilizados, o que ainda não é razoável em dispositivos de redes de sensores, então o nível de confidencialidade é tão bom quanto o de autenticação [9, 37].

O protocolo ZigBee [3] é apresentado neste trabalho apenas no contexto de segurança, pois trata-se de um conjunto de especificações para diversas finalidades como por exemplo o roteamento de pacotes, formatos de frames e serviços de aplicação. Neste contexto, o ZigBee é muito parecido com o SNEP, pois utiliza o mesmo modo de operação e a mesma forma de autenticação. A diferença é a escolha da cifra de bloco, já que o ZigBee utiliza o AES. As recomendações de uso do protocolo ZigBee se baseiam em impedir a repetição do contador através de um NONCE e utilizar um tamanho de chave simétrica de 128 bits. Isto forneceria um nível de segurança interessante se

comparado a outros protocolos [37], porém não há uma especificação de como implementar esta idéia de forma prática dentro do contexto das redes de sensores, já que não há recursos de armazenamento que sejam suficientes para que o contador não se repita. Mesmo se houvesse, o uso de um contador relativamente grande pode causar um sobrecusto de comunicação considerável, caso este seja transferido pela rede.

Portanto o protocolo WEP não é interessante já que apresenta muitas vulnerabilidades. Os protocolos SNEP e ZigBee são parecidos e apresentam o mesmo problema, mas poderiam ser melhor especificados e implementados para bom uso em RSSF. Resta observar o funcionamento dos protocolos TinySec e MiniSec-U, os quais são o foco deste capítulo e parte importante deste trabalho.

4.1.1 Protocolos TinySec e Minisec-U

O protocolo TinySec [39] possui grande atenção na comunidade e motivou a criação do protocolo MiniSec-U [51]. Estes protocolos possuem um bom nível de segurança, pois utilizam apenas mecanismos criptográficos sólidos, ou seja, algoritmos que já foram exaustivamente estudados e não apresentam vulnerabilidades relevantes se forem utilizados com o mínimo de recomendações [11, 14, 75].

4.2 Protocolo TinySec

O TinySec é o primeiro protocolo de segurança específico para RSSF que foi totalmente implementado. Antes da criação do TinySec havia muita atenção da comunidade para o SNEP, mas a falta de especificação e implementação deste protocolo auxiliou o crescimento da popularidade do TinySec, o qual foi utilizado até em equipamentos comercializados [4].

Algumas das prioridades de projeto do TinySec foram o aproveitamento de código e o uso de primitivas criptográficas sólidas e eficientes em consumo de espaço e tempo. Apesar disso, um fator de grande questionamento deste protocolo foi o aceite de algumas limitações em virtude do baixo consumo de energia. A ausência da proteção contra ataques do tipo replay é a maior limitação deste protocolo, porém os autores recomendam que isto seja feito em nível de aplicação, já que o TinySec é um protocolo específico para a camada de Enlace [39, 77].

O principal objetivo do TinySec é oferecer as seguintes propriedades básicas de segurança: confidencialidade, autenticação e integridade. A seguir, descrevemos os mecanismos utilizados pelo protocolo para oferecer cada propriedade.

4.2.1 Confidencialidade

A confidencialidade do TinySec se baseia nos seguintes mecanismos: um VI de 8 bytes e uma cifra de bloco no modo CBC. A cifra de bloco escolhida pelos autores do protocolo foi o Skipjack [60]. O VI é composto pelos seguintes campos: endereço do nó de destino do pacote (*Dest*), Active Message handler[2] (*AM*), tamanho do dado útil (*Len*), endereço do nó de origem do pacote (*Src*), contador de 16 bits (*Ctr*). Estes campos são anexados em cada pacote conforme ilustrado na figura 4.1. O contador é iniciado em 0 (zero) e incrementado depois de cada mensagem enviada.

[2] Pode ser comparado uma porta de comunicação

O modo CBC utilizado pelo TinySec possui uma peculiaridade. O VI é cifrado antes de sua utilização para evitar que o próximo VI seja previsível. Além disso, o TinySec utiliza a técnica ciphertext stealing (veja também a seção 3.3.4) para não haver diferença de tamanho entre o texto original e o cifrado. Isto é necessário já que uma cifra de bloco operada no modo CBC possui um tamanho de bloco fixo e portanto qualquer mensagem cifrada é um múltiplo da quantidade de blocos. A expansão de mensagem não é iteressante por questões de economia de energia.

A cifra de bloco utilizada pelo TinySec, o Skipjack, requer uma chave de 80 bits, a qual deve ser previamente compartilhada entre o remetente e o destinatário antes da troca de mensagens.

4.2.2 Autenticação e Integridade

O TinySec fornece autenticação e integridade através do uso de um CBC-MAC. O valor MAC é calculado em função da mensagem, da mesma chave utilizada para fornecer confidencialidade e de um VI.

O valor MAC é anexado na mensagem pelo remetente. Isto permite que o valor recalculado pelo destinatário possa ser comparado com aquele obtido a partir da mensagem enviada. Se os valores forem iguais a mensagem é aceita, caso contrário, a mensagem é descartada.

4.2.3 Formatos de Pacote

O TinySec possui duas versões de pacote: TinySec-AE e TinySec-Auth, sendo que a primeira é a versão mais completa, pois fornece autenticação, integridade e confidencialidade. A versão TinySec-Auth fornece apenas as propriedades de autenticação e integridade, ou seja, não utiliza a cifragem para proteger os dados úteis do pacote, chamados de carga útil[3]. Não há versões sem a propriedade de autenticação e integridade.

Os formatos de pacote do TinySec são baseados no formato do TinyOS (veja seção 6.8.1), porém há alterações significativas. O TinyOS utiliza um CRC de 2 bytes ao invés do CBC-MAC de 4 bytes do TinySec. Além disso, o TinyOS não possui os campos que definem o endereço de origem do pacote e o tamanho do contador, mas possui um campo para identificar o grupo de rede do remetente. A figura 4.1 é uma representação gráfica do formato de pacote do TinySec-AE e do TinyOS. Os números em cada campo representam o tamanho em bytes. Apenas o campo com preenchimento cinza representa dados cifrados. O formato de pacote do TinySec-Auth (não ilustrado) é semelhante ao TinySec-AE, sendo que a única diferença está no campo de carga útil, o qual não é cifrado.

4.2.4 Esquema de Chaves

Especificações de como uma chave pode ser distribuída e compartilhada não são parte do protocolo TinySec. Não há sequer recomendações de esquemas que possam ser utilizados com este protocolo. Apesar disso, o TinySec pode ser utilizado de forma transparente com esquemas como chaves globais e em grupos.

[3] Do inglês, *payload*

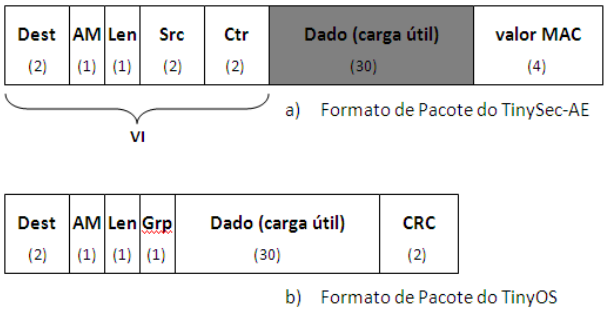


Figura 4.1: Formatos de Pacote do TinySec-AE e TinyOS.

4.3 Protocolo MiniSec-U

O protocolo MiniSec-U sofreu forte influência do TinySec. Isto fica claro quando se analiza alguns conceitos de projeto do MiniSec-U: aproveitamento de código e uso de primitivas criptográficas sólidas e eficientes em consumo de tempo e espaço. Além disso, o MiniSec-U utiliza um formato de pacote parecido com o do TinySec. Neste novo formato há pequenas alterações para minimizar sobrecustos.

Além das propriedades de confidencialidade, segurança e integridade, que também são oferecidas pelo TinySec, o MiniSec-U possui defesa contra ataques do tipo replay.

4.3.1 Confidencialidade

A confidencialidade do MiniSec-U se baseia em uma cifra de bloco no modo de operação OCB [75]. Assim como no TinySec, a cifra de bloco padrão deste protocolo é o Skipjack.

O processo de cifragem possui um ingrediente implementado de forma diferenciada dos demais protocolos de segurança específicos para RSSF. Há um contador compartilhado entre remetente e destinatário para variação do VI. Desta forma, o contador não precisaria ser enviado pela rede, pois a cada bloco do modo OCB o valor do contador é incrementado, o que representa uma sincronia entre os participantes da rede. Por outro lado, o MiniSec-U fornece a opção de se enviar alguns bits do contador através da mensagem. Isto é interessante em caso de perdas de pacotes na rede, o que requer um caro processo de sincronia do contador. O envio parcial do contador é chamado de Otimização LB (Last Bits).

O tamanho do contador utilizado pelo MiniSec-U é de 8 bytes, ou seja, quatro vezes maior que o tamanho do contador do TinySec. Isto acontece porque o MiniSec-U utiliza apenas o contador como VI. Além disso, o modo OCB requer que o VI seja do mesmo tamanho que o bloco definido para o Skipjack, que é de 64 bits.

4.3.2 Autenticação e Integridade

O modo de operação OCB também é utilizado pelo protocolo para fornecer autenticação e integridade. Este modo possui uma *tag* de tamanho definido, o qual foi adotado como 4 bytes no MiniSec-U. O cálculo desta *tag* é feito a partir do texto cifrado, do texto puro e de um *header*, sendo que este último consiste de uma cadeia de caracteres (string) de até 128 bits. Diz-se que o

header é omitido quando se usa o valor 0 (zero) como string. O MiniSec-U omite o *header*.

Assim como acontece com o valor MAC, a *tag* do modo OCB é recalculada pelo destinatário de uma mensagem, o qual compara este valor com o que foi enviado através da mensagem. Se existir qualquer diferença então a mensagem não é aceita; caso contrário, a mensagem é considerada como autêntica.

4.3.3 Formato de Pacote

A figura 4.2 apresenta o formato de pacote do MiniSec-U. Se comparado com o formato do TinySec, o Minisec-U possui duas diferenças: não utiliza um campo específico como contador e sobrescreve 3 bits do campo *Len* para envio parcial do contador. Isto é possível porque o tamanho máximo da carga útil é de 30 bytes, valor que pode ser representado com apenas 5 bits. Como o espaço reservado para o campo de tamanho da carga útil é de 8 bits, há 3 bits vagos.

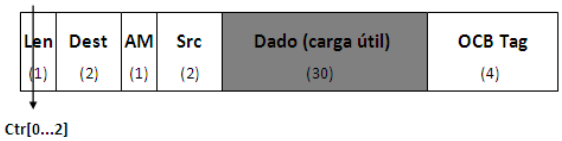


Figura 4.2: Formato de Pacote do MiniSec-U.

4.3.4 Esquema de Chaves

O MiniSec-U não inclui mecanismos para distribuição e gerenciamento de chaves.

4.4 Cifras de Bloco Alternativas para os Protocolos MiniSec-U e TinySec

Os protocolos MiniSec-U e TinySec especificam a cifra de bloco Skipjack como o padrão de cifragem dos dados. Embora este mecanismo tenha sido adotado, é possível que outras cifras sejam utilizadas, como por exemplo o AES [21] e o RC5. Há diferenças de desempenho e segurança entre as diversas cifras de bloco existentes, porém o foco deste trabalho é a comparação dos protocolos através de seus mecanismos padronizados. Comparações entre diversos mecanismos simétricos de cifragem para RSSF podem ser encontradas em [26, 33, 41].

Capítulo 5

Protocolos de Segurança para Comunicação *Broadcast*

A comunicacão *broadcast* é muito utilizada em RSSF na situação em que a estação base de uma rede envia mensagens para todos os demais nós. Estas mensagens muitas vezes são comandos de operação dos dispositivos, consultas à rede e atualização de software. Neste sentido é importante que as mensagens sejam devidamente autenticadas para que adversários mal intecionados não consigam comprometer os participantes da rede. Um exemplo de mensagem *broadcast* é a reinicialização de todos os nós da rede. Neste trabalho, consideramos que a confidencialidade das mensagens em *broadcast* não é tão importante quanto a autenticação. Desta forma, a cifragem torna-se uma propriedade prescindível e a autenticação, o objetivo principal de segurança.

5.1 Protocolos *Unicast* também podem ser utilizados em comunicação *Broadcast*?

Os protocolos de segurança desenvolvidos para troca de mensagens entre apenas um remetente e um destinatário podem não ser tão seguros em outros modelos de comunicação. Isto acontece principalmente com protocolos baseados em mecanismos simétricos, pois uma mesma chave compartilhada entre vários dispositivos representa um grande risco em caso de captura de nós. Nesta situação, toda a rede pode estar comprometida. Sem algumas considerações não seria seguro adotar os protocolos MiniSec-U e TinySec para o modelo de comunicação *broadcast*.

Não é somente a segurança que pode dificultar o uso de protocolos *unicast* em um modelo *broadcast*. O uso de contadores compartilhados, como no caso do MiniSec-U, pode implicar em uma necessidade de sincronia entre muitos nós. Isto iria gerar grande sobrecusto de comunicação e inviabilizar o uso deste conceito em comunicação *broadcast*. Portanto adaptações importantes são necessárias para tornar os protocolos *unicast* atrativos em outros modelos de comunicação. É comum um protocolo ter versões *unicast* e *broadcast* por este motivo. O TinySec possui apenas a versão *unicast* e o MiniSec-B é a versão *broadcast* do MiniSec-U.

A seguir, apresentamos exemplos de adaptações necessárias para que um protocolo *unicast* possa ser utilizado em modelos *broadcast*. Em seguida citamos alguns protocolos *broadcast* e suas limitações. Por fim, detalhamos o funcionamento dos dois protocolos *broadcast* que serão analisados posteriormente, o μTESLA e o MiniSec-B.

5.1.1 Adaptações de Protocolos *Unicast* para Comunicação *Broadcast*

As adaptações dos protocolos *unicast* normalmente utilizam mecanismos assimétricos para evitar o uso exclusivo de chaves privadas. Isso não implica que os protocolos *broadcast* são puramente assimétricos, ao contrário, são protocolos com mecanismos simétricos, porém com algumas funcionalidades adicionais que se caracterizam como assimétricas. Um exemplo de assimetria é um esquema de gerenciamento de chaves que contém uma chave mestre global (simétrica) e chaves secundárias reveladas de acordo com o tempo. Esta revelação de chaves agendada representa um tipo de assimetria, já que nem todos os participantes da rede têm posse de tais chaves em um determinado momento. Iremos considerar os protocolos com algumas características assimétricas como protocolos simétricos predominantes. Neste trabalho não há detalhamento dos protocolos para RSSF que utilizam esquemas puramente assimétricos, ou seja, aqueles que se baseiam exclusivamente em modelos criptográficos de chave pública.

Outros mecanismos normalmente utilizados para adaptações de protocolos *unicast* são estruturas de dados como Filtros de Bloom (veja seção 5.4.4). Isto mostra que os ingredientes de protocolos *broadcast* não são apenas os esquemas criptográficos.

5.2 Protocolos *Broadcast* Simétricos e Assimétricos

No contexto das RSSF, protocolos de segurança puramente assimétricos ainda não são tão atrativos quanto os simétricos, já que apresentam sobrecustos elevados. Assinaturas digitais, por exemplo, possuem alto sobrecusto computacional [52, 65, 67, 68] e one-time signatures, alto sobrecusto de comunicação [20, 52]. Alguns protocolos baseados em esquemas de chave pública que evitam estes sobrecustos foram criados [52, 64, 69], porém outras limitações não menos importantes emergem, como a falta de suporte a alta entropia de mensagens e a ausência de robustez a perda de pacotes da rede.

O destaque dos protocolos de segurança com esquemas simétricos predominantes relaciona-se com baixos sobrecustos e poucas limitações, como é o caso do μtesla [66] e MiniSec-B [51].

5.2.1 Protocolos *Broadcast* com Esquemas Simétricos Predominantes: μTESLA e MiniSec-B

Nas próximas seções são apresentados os protocolos *broadcast* μTESLA e MiniSec-B. Os dois protocolos consideram que todos os participantes da rede possuem sincronia de tempo com um erro máximo definido, o qual deve ser conhecido por todos os nós. Como o principal ingrediente dos protocolos é a autenticação, é importante ressaltar que a proteção contra ataques do tipo replay não pode ser omitida. Em caso contrário, um nó mal intencionado poderia armazenar pacotes para posterior reenvio, o que permitiria a autenticação dos pacotes em qualquer momento. A sincronia de tempo é utilizada justamente para permitir a propriedade de proteção contra ataques do tipo replay.

5.3 O Protocolo μTESLA

O protocolo μTESLA foi especificamente desenvolvido para comunicação *broadcast* em RSSF. Trata-se de uma extensão do protocolo TESLA [65, 67], o qual não se adapta a dispositivos com recursos escassos por utilizar assinatura digital [51].

5.3.1 Considerações

Além de considerar que os nós possuem uma sincronia de tempo com uma tolerância determinada, a qual chamaremos simplesmente de sincronia de tempo, o μTESLA requer que o tipo de comunicação tenha a estação base como único remetente possível e todos os demais nós como destinatários. Alterações no estilo de comunicação implicam em novas versões do protocolo, as quais não são foco de análise deste trabalho. O protocolo considera também que qualquer nó autorizado a receber mensagens deve conhecer uma agenda de revelação de chaves e que uma chave simétrica global é inicialmente compartilhada.

Portanto o resumo das considerações é o seguinte:

1. Existência uma chave simétrica global, estabelecida entre todos os participantes da rede
2. Existência de uma sincronia de tempo entre todos os nós da rede
3. Erro máximo da sincronia de tempo é um valor definido e conhecido por todos os nós da rede
4. Agenda de revelação de chaves conhecida por qualquer nó autorizado a receber mensagens
5. Modelo de comunicação *broadcast* em que a estação base é o único remetente (comunicação base-nós)

Todas as considerações acima serão chamadas de *requisitos do protocolo*.

A agenda de revelação de chaves permite que os nós receptores de mensagens possam saber em quais intervalos de tempo as chaves secundárias serão enviadas pela estação base.

5.3.2 Funcionamento do μTESLA

O protocolo se baseia no envio de chaves pela estação base para que os demais nós possam autenticar pacotes com mensagens úteis. O envio de chaves acontece de forma sincronizada. Para facilitar o entendimento, apresentaremos inicialmente o funcionamento básico do protocolo. Depois descreveremos os mecanismos com detalhes.

De uma forma geral, uma mensagem recebida por um nó é armazenada para posterior autenticação. Chaves secundárias serão enviadas pela estação base para que os demais nós possam utilizá-las para autenticar os pacotes armazenados em um determinado intervalo de tempo. Cada intervalo de tempo define a chave secundária específica que será utilizada para autenticar os pacotes.

Para enviar um pacote autenticado, a estação base calcula um valor MAC para o pacote com a chave secundária que ainda é desconhecida pelos demais nós. Assim que um nó recebe um pacote, verifica se a chave que foi utilizada para o cálculo do valor MAC deste pacote ainda não foi revelada pela estação base. Isto pode ser feito devido a sincronia de tempo, erro máximo de sincronia de

tempo e agenda de revelação de chaves. Se o nó receptor sabe que a chave correspondente ainda não foi revelada, então este considera que o pacote recebido não foi alterado por um adversário e o armazena em um buffer. No momento oportuno, a estação base envia a chave de verificação do pacote para todos os nós participantes da rede. Cada nó receptor da chave poderá verificar se esta é válida ou autêntica (veja exemplo na seção 5.3.4). Se a chave for considerada válida, o nó receptor pode portanto usá-la para autenticar o pacote armazenado.

5.3.3 Geração das Chaves

A geração das chaves é feita através de uma função unidirecional, a qual é um mapeamento de um Domínio D em uma Imagem I com duas propriedades:

1. A função é fácil de se calcular: há um algoritmo eficiente em consumo de tempo que, dado $x \in D$, devolve $y = F(x)$.

2. A função é difícil de se inverter: não é computacionalmente viável para um adversário, de posse de um $y \in I$, calcular o ponto x tal que $F(x) = y$.

Para uma definição formal consulte [10, 13].

Cada chave gerada pertence a uma cadeia de chaves, em que a geração completa é feita da seguinte forma: seja F uma função unidirecional e K_n a última chave de uma cadeia de chaves. A estação base escolhe K_n de forma pseudo-aleatória e aplica a função F repetidamente para calcular outras chaves de forma que $K_i = F(K_{i+1})$, em que $i = 0, 1, ..., n-1$.

Observamos que o uso de uma função unidirecional implica que é computacionalmente difícil para um adversário com posse de K_i calcular K_{i+1} (propriedade 2 da função unidirecional).

A escolha de K_n é feita com o uso de uma chave simétrica global. Seja K_{mestre} uma chave simétrica global compartilhada entre todos os participantes da rede, então K_n é escolhida da seguinte forma:

$$K_n \leftarrow F(K_{mestre})$$

A distribuição da chave mestre não é definida pelo protocolo.

5.3.4 Exemplo de funcionamento do protocolo

O exemplo apresentado a seguir considera os requisitos do protocolo (veja seção 5.3.1). Na figura 5.1 cada chave da cadeia de chaves está relacionada com um intervalo de tempo. Todos os pacotes enviados em um determinado intervalo de tempo serão autenticados com a mesma chave. Seja K_0 a chave simétrica global. Os pacotes P_1 e P_2, enviados no intervalo 1, contém um valor MAC gerado com a chave K_1. O pacote P_3 contém um valor MAC gerado com a chave K_2. Qualquer receptor ainda não pode autenticar os pacotes, pois as chaves ainda não foram reveladas. Suponha que todos os pacotes do intervalo 3 são perdidos por problemas com a rede, incluindo o pacote com a chave K_1. No intervalo 4 a estação base envia a chave K_2, que um nó receptor pode autenticar verificando a equação $K_0 = F(F(K_2))$. Como este nó sabe que $K_1 = F(K_2)$, é possível autenticar os pacotes P_1 e P_2 com K_1, e P_3 com K_2. Este exemplo mostra que não é necessário armazenar

todas as chaves enviadas pela estação base. Os nós receptores armazenam somente a última chave enviada. Outro detalhe é que o envio das chaves pela estação base é independente dos pacotes que contém as mensagens. Porém, a autenticação de um pacote depende da autenticação de sua chave correspondente. Um pacote é descartado se sua chave correspondente não for verificada pela função F. No exemplo, se a equação $K_0 = F(K_1)$ não for verdadeira, então K_1 não pode ser usada e portanto os pacotes P_1 e P_2 são descartados.

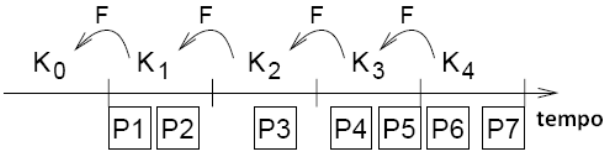


Figura 5.1: Revelação de Chaves por Tempo para Autenticação.

5.3.5 Implementação da Função Unidirecional

O μTESLA sugere a implementação da função unidirecional através do uso do CBC-MAC, como a seguir, em que K_{mestre} é a chave simétrica global e x, um número inteiro:

$$F(x) = \text{CBC-MAC}(K_{mestre}, x)$$

Neste caso, a função unidirecional possui as duas propriedades mencionadas na seção 5.3.3 e um ingrediente adicional: um segredo. Isso acontece porque o algoritmo CBC-MAC trabalha com uma chave criptográfica, a qual é utilizada por uma cifra de bloco.

5.3.6 Confidencialidade, Autenticação e Integridade

O μTESLA não fornece métodos de cifragem e omite a propriedade de confidencialidade. Apesar desta propriedade ser interessante em muitos casos, o tipo de comunicação considerado pelo protocolo (base-nós) requer maior atenção ao processo de autenticação de mensagens, conforme já discutido no início deste capítulo.

A integridade é obtida no mesmo passo da autenticação devido ao uso do MAC, o qual é um valor calculado por cada nó receptor de uma mensagem com o uso de uma chave secundária revelada pela estação base. É interessante observar que a estação base realiza o cálculo do valor MAC antes de enviar o pacote e a chave secundária.

5.4 O Protocolo MiniSec-B

O MiniSec-B foi criado especificamente para comunicação no modelo *broadcast* para RSSF. Possui as mesmas primitivas criptográficas que o MiniSec-U e oferece dois mecanismos de proteção contra ataques do tipo replay: um utiliza o conceito de Sliding Windows [51] e o outro, Filtros de Bloom [16, 51]. Um usuário do protocolo deve escolher somente um mecanismo entre estas duas opções.

Assim como o μTESLA, este protocolo evita o uso de contadores compartilhados, mas não apresenta um esquema dinâmico para o uso de chaves criptográficas. Na realidade o MiniSec-B

nem sequer possui um esquema de chaves como o μTESLA, mas isto não impede a adoção de um outro protocolo para o gerenciamento das chaves, assim como acontece com o MiniSec-U.

5.4.1 Considerações

Assim como o protocolo μTESLA, o MiniSec-B considera que todos os participantes da rede tenham uma sincronia de tempo com um erro máximo definido. Sempre que o valor máximo for ultrapassado deve-se iniciar um novo processo de sincronia entre os nós. Este processo é feito através da execução de um algoritmo que não pertence ao protocolo.

Outra consideração do protocolo é a existência uma chave simétrica compartilhada entre um remetente e um destinatário. Uma chave simétrica global pode ser compartilhada entre todos os nós, porém recomenda-se que cada remetente tenha uma chave diferente.

5.4.2 Funcionamento do MiniSec-B

A seguir, apresentaremos as técnicas utilizadas pelo MinSec-B para fornecer proteção contra ataques do tipo replay. Posteriormente mostraremos como o protocolo oferece confidencialidade, autenticação e integridade.

5.4.3 Sliding Windows: Proteção Contra Replay Com Janela de Vulnerabilidade

Este mecanismo é apresentado no MiniSec-B como uma forma de proteção contra ataques do tipo replay com uma janela de vulnerabilidade conhecida. Desta forma, todos os pacotes enviados durante um determinado período de tempo não estão protegidos contra este tipo de ataque. Portanto um pacote pode ser descartado por motivos de ataque do tipo replay somente se for enviado fora da janela de vulnerabilidade. A seguir, apresentamos o funcionamento deste mecanismo.

Definimos o tempo de latência como o intervalo de tempo que um pacote leva para chegar a um nó de destino, partindo de um nó de origem. Cada par de nós da rede possui um tempo de latência. Dizemos que o maior intervalo entre todos é a latência máxima da rede. Seja $E1, E2, E3, ...$ um conjunto de intervalos de tempo. Chamaremos cada intervalo deste conjunto de *época*. Seja Δ_T o erro máximo de sincronia entre dois nós da rede e Δ_N a latência máxima da rede. Definimos o tamanho de uma época como $2\Delta_T + \Delta_N$.

Cada época será utilizada como NONCE através da cifragem no modo OCB. Isto permite o descarte de pacotes do tipo replay, ou seja, enviados em épocas anteriores. Porém, o erro de sincronia de tempo entre os nós e a latência da rede podem implicar em grande número de descarte de pacotes legítimos durante a transição de épocas. Este problema é contornado com o uso de apenas duas épocas candidatas a NONCE.

A existência de apenas duas épocas que podem ter sido utilizadas como NONCE no modo OCB é a base de funcionamento da técnica de Sliding Windows. Isto é apresentado através do lema a seguir.

Lema 1. *Seja E_i a época atual e t_i o instante de tempo que marca o início de E_i. Se um nó recebe um pacote dentro do intervalo $(ti,\ ti + \Delta_T + \Delta_N)$, os dois candidatos a NONCE são E_i e E_{i-1}. Se um nó recebe um pacote dentro do intervalo $(t_i + \Delta_T + \Delta_N,\ t_{i+1})$, os dois valores candidatos são*

E_i e E_{i+1}.

Logo, pelo lema 1, há apenas duas épocas candidatas a NONCE para qualquer pacote recebido. A linha pontilhada da figura 5.1 mostra a fronteira mencionada, ou seja, a divisão entre os dois intervalos de recebimento de pacotes dentro da época E_i.

Prova do Lema 1

Baseado em [51], vamos mostrar que para qualquer pacote recebido há apenas duas épocas candidatas a NONCE.

Seja δ_n a latência real da rede tal que $0 < \delta_n < \Delta_N$, e δ_t o erro real de sincronia da rede tal que $-\Delta_T < \delta_t < \Delta_T$. Seja t_{s_i} o tempo local registrado pelo remetente no momento em que o pacote i é enviado e t_{r_i} o tempo local registrado pelo receptor no momento em que o pacote i é recebido. Temos que:

$$t_{s_i} + \delta_n + \delta_t = t_{r_i} \ (1)$$

No primeiro caso descrito no lema 1, consideramos os pacotes recebidos entre os instantes de tempo t_{r_1} e t_{r_2}, em que $t_{r_1} = t_i + \varepsilon$ e $t_{r_2} = t_i + \Delta_T + \Delta_N - \varepsilon$ (a figura 5.2 ilustra este caso). Para mostrar que os pacotes recebidos neste intervalo podem somente ter sido originados das épocas E_i ou E_{i-1} é utilizada a seguinte estratégia:

a. Cálculo do valor mais antigo de t_{s_1}. Portanto o menor valor de t_{s_1}.

b. Cálculo do valor mais recente de t_{s_2}. Portanto o maior valor de t_{s_2}.

Da equação (1) temos que $t_{s_1} = t_{r_1} - \delta_n - \delta_t$. Logo, o menor valor de t_{s_1} é $t_{r_1} - \Delta_N - \Delta_T$. Este instante está na época E_{i-1}. De forma similar, como $t_{s_2} = t_{r_2} - \delta_n - \delta_t$, temos que o maior valor de ts_2 é $t_{r_2} + \Delta_T$, o qual está em E_i.

No segundo caso do lema 1(não ilustrado) consideramos os pacotes recebidos entre os instantes t_{r_3} e t_{r_4}, em que $t_{r_3} = t_i + \Delta_T + \Delta_N + \varepsilon$ e $t_{r_4} = t_{i+1} - \varepsilon$. Novamente a partir da equação (1), temos que $t_{s_3} = t_{r_3} - \delta_n - \delta_t$. Logo, o menor valor de t_{s_3} é $t_{r_3} - \Delta_N - \Delta_T$, o qual está em E_i. O maior valor de t_{s_4} é $t_{r_4} + \Delta_T$, o qual está em E_{i+1}.

Portanto qualquer pacote recebido possui somente dois candidatos a NONCE, são eles: E_{i-1} e E_i, ou E_i e E_{i+1}.

O teste do valor de duas épocas determina rapidamente qual dos valores foi utilizado pelo remetente como NONCE.

Janela de Vulnerabilidade

Um pacote enviado no instante de início de uma época pode ser reenviado durante o restante da época atual mais $\delta_t + \delta_n$ da próxima época sem que seja considerado ilegítimo. Este intervalo, ou seja, $3\Delta_T + 2\Delta_N$, é chamado de janela de vulnerabilidade, pois não descarta qualquer pacote por motivos de ataques do tipo replay.

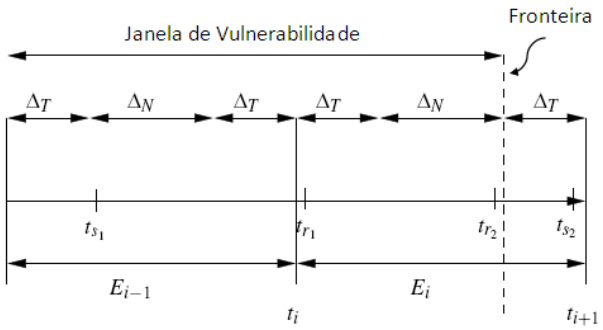


Figura 5.2: Linha do Tempo.

5.4.4 Proteção Contra Replay Sem Janela de Vulnerabilidade

O MiniSec-B utiliza uma estrutura conhecida como Filtros de Bloom para fornecer proteção contra ataques do tipo replay dentro da janela de vulnerabilidade apresentada em seções anteriores, a qual é um resultado da técnica Sliding Windows. Isto requer que cada nó participante da rede tenha um espaço dedicado a dois Filtros de Bloom e um contador. Desta forma, cada Filtro de Bloom está relacionado com uma época, pois pela técnica anterior sabe-se que há duas épocas possíveis para cada NONCE utilizado pelo nó remetente. O tamanho do contador é relativamente pequeno, pois este é reiniciado a cada transição de época. Antes de detalharmos o funcionamento desta técnica, mostraremos os conceitos relacionados com os Filtros de Bloom.

Filtros de Bloom

Filtros de Bloom são estruturas de dados eficientes em espaço que são utilizadas em testes de pertinência em conjuntos. Um teste de pertinência com o uso desta estrutura é um algoritmo que recebe um elemento e uma representação de um conjunto e devolve sim, se o elemento pertence ao conjunto, ou não, caso contrário. Como o teste se baseia em uma representação compacta de um conjunto de elementos há um custo envolvido. Um elemento externo ao conjunto pode ser reconhecido como um elemento legítimo do conjunto representado (falso-positivo). Apesar disso, não há possibilidade de falso-negativo, ou seja, um elemento legítimo não seria reconhecido como um elemento fora do conjunto. Portanto o teste de pertinência é probabilístico, já que não se armazenam todos os elementos do conjunto, mas somente uma representação.

Neste contexto, um algoritmo que utiliza esta estrutura possui duas operações: incluir um elemento na representação do conjunto e verificar se um elemento está no conjunto.

Geração de um Filtro de Bloom

Seja $A = \{a_1, a_2, ..., a_n\}$ um conjunto com n elementos e $h_1, h_2, ..., h_k$ funções *hash* distintas. Um Filtro de Bloom que representa o conjunto A é um vetor de m bits gerado como a seguir:

1. Cada posição do vetor é inicializada com 0.

2. Cada um dos elementos $a_i \in A$ é aplicado a cada uma das k funções *hash*.

3. Cada resultado do passo anterior representa uma posição do vetor, a qual deve ser inicializada com 1 (veja a figura 5.3).

Como verificar se um elemento a_j pertence ao conjunto A de elementos inseridos no filtro? Esta verificação é feita através da aplicação das k funções *hash* ao elemento a_j, ou seja, $h_1(a_j), h_2(a_j), ..., h_k(a_j)$. Se pelo menos um dos resultados corresponder a um bit de valor 0 no filtro então o elemento certamente não pertence ao conjunto A. Caso contrário, considera-se que o elemento pertence ao conjunto A, embora possa haver falso-positivo.

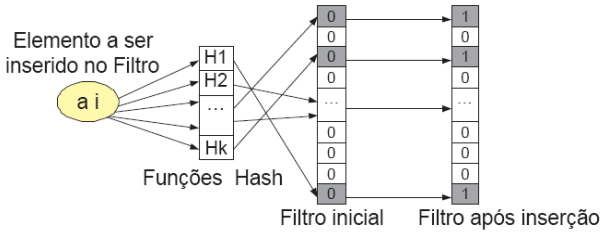


Figura 5.3: Geração de um Filtro de Bloom [25].

Filtros de Bloom no MiniSec-B

O uso de Filtros de Bloom é interessante em ambientes de rede com recursos escassos, já que requer $O(nlog\ n)$ bits para armazenar uma representação de n mensagens. Além disso, as operações de inclusão e verificação de elementos são feitas em tempo constante. A seguir, mostramos como o MiniSec-B utiliza esta estrutura para fornecer proteção contra ataques do tipo replay.

Definição 5. *Dizemos que um pacote está relacionado com a época E_i ou que um pacote é da época E_i se este foi enviado durante E_i, a qual é registrada com o relógio do remetente do pacote.*

Seja M um texto legível, H um header, C_A um contador mantido pelo remetente e $sourceID$ o endereço do nó remetente. Seja ainda E_i a época atual. O NONCE é definido como $sourceID||C_A||E_i$, em que o símbolo $||$ representa concatenação. Logo, o texto cifrado pelo nó remetente no modo OCB com uma chave simétrica K_A é $OCB_{K_A}(NONCE, M, H)$, o qual é enviado pela rede com uma tag de autenticação calculada conforme os padrões do modo OCB. Todos os pacotes recebidos são armazenados em um Filtro de Bloom com o uso da chave $(C_A||sourceID)$, a qual chamaremos de chave do Filtro de Bloom. Todo nó receptor mantém um Filtro de Bloom BF_i para a época E_i e outro para apenas uma das seguintes: E_{i+1} ou E_{i-1}. Por exemplo, um nó mantém os Filtros de Bloom BF_i e BF_{i-1} desde o início da época E_i até $\Delta_T + \Delta_N$ desta época. Após isto todos os pacotes relacionados com a época E_{i-1} serão descartados, enquanto pacotes da época E_{i+1} serão aceitos. Assim, o conteúdo de BF_{i-1} é apagado e este espaço é reutilizado como BF_{i+1}.

Pelo lema 1, sabemos que há somente duas épocas candidatas a NONCE para qualquer pacote recebido. Desta forma, um pacote válido deve corresponder a apenas um dos dois Filtros de Bloom. Logo, um receptor pode decifrar um pacote e em seguida verificar se este foi reenviado, ou seja, se houve um ataque do tipo replay. A seguir detalhamos estes dois passos.

1. **Decifragem do pacote**: para decifrar um pacote é essencial que o receptor conheça todos os componentes do NONCE, ou seja, $sourceID$, C_A e E_i. Os dois primeiros são enviados com a mensagem, restando apenas saber qual época foi utilizada pelo remetente. O receptor utiliza as duas épocas possíveis para obter o NONCE. Se o pacote não for decifrado em duas tentativas então este é descartado, pois o valor da tag que será recalculada com base no pacote decifrado não será igual ao valor anexado no pacote.

2. **Verificação de ataque do tipo replay**: após a autenticação com a tag no modo OCB deve-se verificar se o pacote é um replay. Como o receptor já conhece a época utilizada como parte do NONCE, o que aconteceu no passo de decifragem, é possível saber qual o Filtro de Bloom está relacionado com o pacote recebido. Portanto, é feita uma consulta ao Filtro de Bloom para saber se este pacote foi recebido e armazenado em uma outra ocasião. Se o pacote não está no Filtro de Bloom então este é considerado recente e portanto aceito. Caso contrário, o pacote é descartado. A consulta em questão é feita através de uma função *hash* em que o argumento é a chave do Filtro de Bloom, definida anteriormente.

5.4.5 Confidencialidade, Autenticação e Integridade

A confidencialidade do protocolo é fornecida através da cifragem no modo OCB, o qual também fornece autenticação e integridade através do cálculo de uma tag. Assim como no MiniSec-U, a cifra de bloco utilizada é o Skipjack.

5.4.6 Formatos de Pacote

O miniSec-B possui um formato de pacote definido, o qual é idêntico ao formato da versão para comunicação no modelo *unicast*. Existe a possibilidade de se reutilizar parte do espaço reservado para o endereço de destino do pacote. Isto teria como objetivo aumentar o espaço disponível (de 3 bits para 8 bits) para transmitir o contador. Nada impede de adotar esta opção para a versão *unicast* do protocolo (isto apenas reduz a quantidade de nós que uma rede pode ter). A figura 5.4 ilustra o formato de pacote com o reuso de espaço.

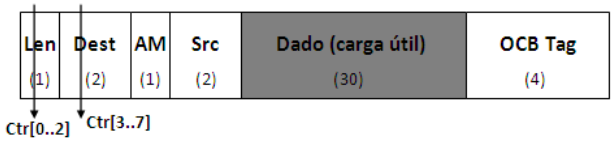


Figura 5.4: Formato de Pacote do MiniSec-B.

Capítulo 6

Análise Comparativa dos Protocolos TinySec e MiniSec-U

Em capítulos anteriores, discutimos alguns mecanismos criptográficos utilizados por protocolos de RSSFs e algumas propriedades de segurança oferecidas em virtude do uso destes mecanismos. Discutimos também algumas outras propriedades que são desejadas em uma RSSF, apesar de não estarem diretamente ligadas com questões de segurança, como por exemplo baixo sobrecusto computacional. Uma forma de comparação entre os protocolos é justamente através das propriedades oferecidas e dos mecanismos criptográficos. Desta forma, poderemos dizer qual protocolo é mais resistente a determinados tipos de ataques. Este capítulo realiza uma análise comparativa dos protocolos para comunicação *unicast* TinySec e MiniSec-U em duas etapas: uma teórica, incluindo as propriedades e mecanismos dos dois protocolos, e outra experimental, incluindo aspectos de implementação e simulação para que se obtenha uma estimativa do consumo de energia dos protocolos.

6.1 Análise Teórica: Metodologia

A tabela 6.1 apresenta as principais propriedades de segurança que serão tratadas nas próximas seções. Demais propriedades, listadas a seguir, serão discutidas posteriormente.

- Autenticação Imediata
- Baixo Sobrecusto Computacional e de Comunicação
- Resistência a perda de pacotes da rede
- Escalabilidade

Propriedade	**TinySec**	**MiniSec-U**
Confidencialidade	CBC-Skipjack	OCB-Skipjack
Integridade	CBC-MAC	OCB
Freshness	Não Possui	Contador Compartilhado
Resistência a Comprometimento de Nós	Não Possui	Não Possui

Tabela 6.1: Principais Propriedades de Segurança dos Protocolos TinySec e MiniSec-U.

Em [12] há uma discussão aprofundada sobre a análise de mecanismos de autenticação, autenticidade e confidencialidade como um todo. Como este capítulo separa a análise de confidencialidade das outras propriedades, o apêndice A traz uma breve apresentação do outro modelo.

6.2 Considerações Iniciais sobre Segurança

Em algumas situações, precisamos analisar o comportamento específico de um mecanismo criptográfico sem se preocupar com a influência de outros mecanismos. Por exemplo, uma cifra de bloco geralmente é utilizada com um modo de operação e quando desejamos estudar o comportamento do modo de operação usado é interessante considerar que a cifra de bloco é segura. Desta forma, podemos tentar entender se um mecanismo estudado pode "enfraquecer" a segurança de um sistema como um todo, apesar deste sistema ter partes consideradas seguras.

Mas o que significa considerar um mecanismo seguro (ou forte)?
Isto depende do mecanismo em questão. Uma cifra de bloco segura, por exemplo, é uma função pseudo-aleatória (FPA) f em que um adversário não é capaz de distinguir, em tempo polinomial e baseado no comportamento da entrada e saída, entre uma caixa preta para $f_a(\cdot)$, para uma chave aleatória a, e uma caixa preta para uma função aleatória $\rho(\cdot)$. Este modelo de segurança foi sugerido para a cifra de bloco DES em [50]. Outro modelo interessante é utilizar uma permutação pseudo-aleatória (PPA) ao invés de uma FPA, sendo que neste caso a cifra de bloco seria segura se um adversário não fosse capaz de distinguir entre a PPA em questão e uma permutação aleatória, o que é utilizado em [11, 75].

O uso de uma caixa preta é um conceito muito utilizado em demonstrações de segurança. Um adversário com posse de uma caixa preta poderia escolher algumas entradas, fazer consultas para a caixa preta e obter saídas sem saber qual foi exatamente o procedimento que gerou o resultado. Este modelo permite esconder propriedades e, ao mesmo tempo, fornecer recursos para o adversário. Além disso, há uma interação entre o adversário, que é um algoritmo, e a caixa preta, que é um sistema criptográfico. Entender o comportamento da caixa preta, mesmo sem saber seu conteúdo exato, é uma forma de mostrar que o sistema utilizado é previsível ou inseguro.

As consultas ou chamadas para a caixa preta e as escolhas de possíveis entradas para obter resultados diversos é uma forma do adversário fazer adaptações e melhorar as chances de sucesso ao realizar ataques ao sistema criptográfico. É comum os ataques possuírem como parte de seus nomes termos como *adaptativos* e *escolhidos*. Por exemplo, um adversário pode escolher quais mensagens serão enviadas para uma caixa preta e alterar (adaptar) as escolhas conforme resultados obtidos em consultas anteriores. Após obter um conjunto de objetos gerados por um sistema criptográfico, o adversário em questão pode finalmente realizar um ataque a este sistema. Um ataque pode ser a geração de *novos* objetos, ou seja, objetos que não foram utilizados nem devolvidos em consultas anteriores. Neste exemplo, dizemos que um adversário escolhe mensagens de forma adaptativa, ou ainda, que realiza *ataques adaptativos de mensagem escolhida*.

Os ataques adaptativos de mensagem escolhida são um modelo de adversário muito utilizado para estudar a segurança de mecanismos criptográficos. Este modelo sempre é acompanhado por um *objetivo*, que é uma forma de especificar as regras do ataque realizado pelo adversário. Neste

capítulo, apresentaremos regras distintas para estudar a segurança dos mecanismos que oferecem confidencialidade e integridade. Intuitivamente, a confidencialidade será relacionada com a capacidade de um adversário em distinguir entre textos legíveis a partir de um texto cifrado, e a integridade está relacionada com a capacidade de um adversário em fraudar um par (mensagem, tag de autenticação). Um objetivo e um modelo de adversário, quando combinados, formam uma *noção de segurança*.

Uma questão é se as noções utilizadas são realmente fortes, ou ainda, se as noções são suficientes para dizer que os mecanismos são seguros. Em [11, 32], um MAC é seguro se resiste a fraudes sob ataques adaptativos de mensagem escolhida. Para estudar a integridade do OCB, esta idéia também é utilizada com algumas adaptações em [75]. Em [9, 75], a confidencialidade de mecanismos como o CBC e o OCB é estudada com a utilização de ataques adaptativos de mensagem escolhida. Os trabalhos citados demonstram que as noções que utilizaremos implicam em algumas propriedades importantes, como as seguintes: a dificuldade computacional de um adversário em recuperar a chave a partir de alguns pares (texto legível, texto cifrado), ou recuperar um texto legível a partir de um texto cifrado. Portanto, se um mecanismo criptográfico é seguro em relação a uma noção de segurança, este mecanismo é resistente a tentativas de se recuperar a chave ou um texto legível.

6.3 Confidencialidade e Segurança Semântica

A comparação dos protocolos *unicast* em relação à confidencialidade se baseia em três etapas de estudo:

a. Modos de operação CBC e OCB sobre uma cifra de bloco segura

b. Condições de segurança semântica dos modos de operação sobre uma cifra de bloco segura

c. Cifra de bloco

O primeiro item considera uma cifra de bloco segura (ou forte) para que se tenha a vantagem de analisar o comportamento dos modos de operação de forma isolada. Isto também acontece no segundo item, pois a segurança semântica de um esquema de cifragem está diretamente relacionada com o modo de operação utilizado. Além disso, são apresentados alguns outros aspectos como variação dos parâmetros de entrada. Utilizaremos noções fortes de segurança nas próximas seções para estudar características de confidencialidade dos protocolos MiniSec-U e TinySec.

6.3.1 Modos de Operação CBC e OCB

Um conceito importante em demonstrações de segurança é a Indistinguibilidade sob um Ataque de Texto Legível Escolhido (IND-CPA). Este conceito é muito utilizado quando se deseja estudar esquemas de cifragem. Iniciaremos esta sessão definindo este conceito para depois apresentar dois teoremas que envolvem a segurança dos modos de operação CBC e OCB. Veremos que um esquema de cifragem é seguro de acordo com o nível de defesa contra ataques do tipo Texto Legível Escolhido (CPA).

ORÁCULO $E_K(LR(M_0, M_1, b))$

Entradas: mensagens (textos legíveis) $M_0, M_1 \in \{0,1\}^*$ e $b \in \{0,1\}$

Saída: mensagem cifrada C

1 **Se** $|M_0| \neq |M_1|$

2 **então devolva** $\perp$

3 $C \leftarrow E_K(M_b)$

4 **devolva** C

Figura 6.1: Oráculo de Cifragem

Indistinguibilidade sob um Ataque de Texto Legível Escolhido (IND-CPA)

O conceito de *indistinguibilidade sob um ataque de texto legível escolhido*[1](IND-CPA), ou simplesmente indistinguibilidade, se baseia na seguinte idéia: um adversário escolhe um par de mensagens de mesmo tamanho. Uma das mensagens é selecionada de forma aleatória e submetida a um algoritmo de cifragem. Após isso, o adversário recebe o texto cifrado produzido. O esquema de cifragem utilizado é considerado seguro se o adversário, sem posse da chave secreta, não é capaz de distinguir em tempo polinomial qual das duas mensagens foi selecionada e posteriormente utilizada para produzir o texto cifrado.

Uma adaptação deste conceito a situações mais práticas é permitir que o adversário tenha mais recursos para tentar distinguir mensagens. Em vista disso, pode-se permitir que o adversário tenha diversos pares de mensagens de mesmo tamanho e um texto cifrado para cada par. Na figura 6.1 apresentamos o algoritmo de um *oráculo aleatório*, o qual é fornecido ao adversário para que este possa obter os textos cifrados a partir de uma mensagem de cada par escolhido. Como o adversário não sabe a priori qual mensagem foi escolhida para cifragem, este realiza chamadas ao algoritmo da figura 6.1 sem conhecimento do bit b, ou seja, escolhe o par (M_0, M_1) e realiza a chamada $E_K(LR(M_0, M_1, b))$, sem saber qual dos dois oráculos foi fornecido: $E_K(LR(M_0, M_1, 0))$ ou $E_K(LR(M_0, M_1, 1))$. Dizemos que uma chamada ao algoritmo em questão é uma consulta ao oráculo.

Uma demonstração de segurança através da interação entre um adversário e um mecanismo criptográfico é chamada de **demonstração por jogos**. Detalharemos a seguir as regras de um jogo que permitirá avaliar se um esquema de cifragem é seguro contra IND-CPA, ou ainda, se a confidencialidade é IND-CPA-segura.

Seja $SE = (K, E, D)$ um esquema de cifragem (definição 1) e $E_K(LR(\cdot, \cdot, b))$ um oráculo de cifragem, como ilustrado na figura 6.1, que recebe duas mensagens distintas e de mesmo tamanho e um bit b, e devolve um texto cifrado que corresponde a uma das duas mensagens, selecionada através do bit de entrada b. Logo, o oráculo, quando consultado, possui apenas uma das seguintes versões (ou instâncias):

[1]Do inglês, indistinguishability under a chosen-plaintext attack. Também definido em [9] como indistinguishability under an adaptative chosen-plaintext attack.

Versão 0: o oráculo é $E_K(LR(\cdot,\cdot,0))$. Assim, sempre que uma consulta ao oráculo é feita, fornecendo como entrada o par de mensagens (M_0, M_1) com $|M_0| = |M_1|$, este calcula $C \leftarrow E_K(M_0)$ e devolve C.

Versão 1: o oráculo é $E_K(LR(\cdot,\cdot,1))$. Assim, sempre que uma consulta ao oráculo é feita, fornecendo como entrada o par de mensagens (M_0, M_1) com $|M_0| = |M_1|$, este calcula $C \leftarrow E_K(M_1)$ e devolve C.

Seja ainda A um algoritmo adversário, ou simplesmente um adversário, que recebe uma sequência de pares de mensagens, $(M_{0,1}, M_{1,1})$, ..., $(M_{0,q}, M_{1,q})$ e faz chamadas ao oráculo de cifragem $E_K(LR(\cdot,\cdot,b))$ - o adversário não tem conhecimento de b. Após q consultas ao oráculo o adversário tem posse da sequência $C_1, ..., C_q$ de textos cifrados e deve responder qual das seguintes sequências foi cifrada: $M_{0,1}, ..., M_{0,q}$ ou $M_{1,1}, ..., M_{1,q}$. Isto equivale a dizer qual versão do oráculo foi utilizada, 0 ou 1.

Observe que quando o adversário recebe a sequência de textos cifrados $C_1, ..., C_q$, apenas uma das seguintes afirmações é verdadeira:

a. C_i é um texto cifrado produzido a partir de $M_{0,i}$ para todo $1 \leq i \leq q$

b. C_i é um texto cifrado produzido a partir de $M_{1,i}$ para todo $1 \leq i \leq q$

Apenas uma versão do oráculo é utilizada para a cifragem de todas as mensagens escolhidas, ou seja, não há alternâncias entre as versões do oráculo. Além disso, somente uma chave é utilizada e o adversário ainda pode escolher a sequência de pares de mensagens da seguinte forma: escolhe o primeiro par de mensagens e recebe C_1, escolhe o segundo par e recebe C_2, e assim até a escolha do último par. Muitas vezes este jogo é chamado de *ataque adaptativo de texto legível*, pois o adversário escolhe um par de mensagens de acordo com resultados obtidos com pares anteriores.

As figuras 6.2 e 6.3 apresentam dois experimentos que nos auxiliam a avaliar se a confidencialidade é IND-CPA-segura. Cada experimento está relacionado com uma instância do oráculo de cifragem, um adversário A e um esquema de cifragem SE. Nestes experimentos, denotamos $A^{E_K LR(\cdot,\cdot,b)}$ como uma consulta ao oráculo $E_K(LR(\cdot,\cdot,b))$ pelo adversário A, o qual devolve 0 se reconhece que a versão 0 do oráculo foi utilizada, ou 1, se reconhece que a versão 1 do oráculo foi utilizada.

Definição 6. *Sejam SE um esquema de cifragem, A um adversário e os dois experimentos definidos nas figuras 6.2 e 6.3. O nível de segurança IND-CPA ou valor IND-CPA de SE em relação a A, ou ainda, a vantagem de A contra SE sob IND-CPA é a diferença das probabilidades dos dois experimentos retornarem o bit 1, definida como a seguir.*

$$\mathrm{Adv}_{SE}^{ind-cpa}(A) = \Pr\left[\mathrm{Exp}_{SE}^{ind-cpa-1}(A) = 1\right] - \Pr\left[\mathrm{Exp}_{SE}^{ind-cpa-0}(A) = 1\right].$$

Quanto menor o valor IND-CPA, melhor a segurança do esquema de cifragem SE contra um ataque de texto legível escolhido. Por exemplo, se o valor for muito próximo de zero significa que o adversário não pode dizer qual das duas versões do oráculo foi utilizada.

Experimento $\text{Exp}_{SE}^{ind-cpa-0}(A)$

Entrada: Adversário A

Saída: Versão do oráculo reconhecida por A

1 $k \xleftarrow{R} K$

2 $d \leftarrow A^{E_K LR(\cdot,\cdot,0)}$

3 **devolva** d

Figura 6.2: Experimento com a versão 0 do oráculo

Experimento $\text{Exp}_{SE}^{ind-cpa-1}(A)$

Entrada: Adversário A

Saída: Versão do oráculo reconhecida por A

1 $k \xleftarrow{R} K$

2 $d \leftarrow A^{E_K LR(\cdot,\cdot,1)}$

3 **devolva** d

Figura 6.3: Experimento com a versão 1 do oráculo

Para comparar a segurança oferecida pelos modos de operação OCB e CBC iremos considerar o uso de uma cifra de bloco segura como subrotina, mais especificamente iremos considerar uma cifra de bloco como uma função pseudo-aleatória ou como uma permutação pseudo-aleatória. Assim, o nível de segurança de um modo de operação poderá ser visto de forma independente da cifra de bloco utilizada. A definição a seguir trata de notações sobre vantagens que um adversário pode ter contra funções.

Definição 7. *Seja* $E : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ *uma cifra de bloco em que* $\mathcal{K}$ *é uma cadeia (finita) de caracteres e* $E_K(\cdot) = E(K,\cdot)$*. Seja* Func*(n) o conjunto de todas as funções de* $\{0,1\}^n$ *em* $\{0,1\}^n$ *e* Perm*(n) o conjunto de todas as permutações de* $\{0,1\}^n$ *em* $\{0,1\}^n$*. Para um adversário A, definimos:*

$$\text{Adv}_E^{prf}(A) = \Pr\left[k \xleftarrow{R} K : A^{E_K(\cdot)} = 1\right] - \Pr\left[\rho \xleftarrow{R} \text{Func}(n) : A^{\rho(\cdot)} = 1\right]$$

$$\text{Adv}_E^{prp}(A) = \Pr\left[k \xleftarrow{R} K : A^{E_K(\cdot)} = 1\right] - \Pr\left[\pi \xleftarrow{R} \text{Perm}(n) : A^{\pi(\cdot)} = 1\right]$$

Uma cifra de bloco E é segura se o valor numérico da função de vantagem for "pequeno" para todos os adversários. Para evitar a escolha de um valor que defina um determinado nível de segurança, iremos considerar as vantagens da definição 7 como funções ínfimas.

Calcular a vantagem IND-CPA de um adversário contra um esquema de cifragem operando em um modo específico (OCB ou CBC) requer a construção de um ou mais jogos que simulam perfeitamente o comportamento do modo de operação. Com isso, um adversário pode utilizar um oráculo como subrotina, enviando pares de mensagens e obtendo textos cifrados. O jogo efetivamente termina quando o algoritmo adversário devolver um bit, o qual é o palpite sobre a versão do oráculo consultado. Esta idéia é utilizada em [9, 10, 75], onde os teoremas a seguir foram apresentados e demonstrados.

Teorema 1. *Sejam* $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ *uma cifra de bloco e* $\mathcal{SE}$-CBC $= (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *o correspondente esquema de cifragem no modo CBC. Seja A um adversário IND-CPA contra* $\mathcal{SE}$ *que executa em tempo máximo t e faz até q consultas ao oráculo com até* σ *blocos de n-bits. Então*

existe um adversário B contra E tal que

$$\mathrm{Adv}^{ind-cpa}_{\mathcal{SE}\text{-CBC}}(A) \leq \mathrm{Adv}^{prf}_{E}(B) + \frac{\sigma^2}{2^n}$$

Além disso, B executa em tempo máximo $t' = t + O(q + n\sigma)$ e faz até $q' = \sigma$ consultas ao oráculo.

Teorema 2. *Sejam $E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$ uma cifra de bloco e $\mathcal{SE}$-OCB $= (\mathcal{K}, \mathcal{E}, \mathcal{D})$ o correspondente esquema de cifragem no modo OCB, com uma tag de autenticação de tamanho fixo. Seja A um adversário IND-CPA contra $\mathcal{SE}$ que executa em tempo máximo t e faz até q consultas ao oráculo com até σ blocos de n-bits. Seja $\bar{\sigma} = \sigma + 2q + 3$. Então existe um adversário B tal que*

$$\mathrm{Adv}^{ind-cpa}_{\mathcal{SE}\text{-OCB}}(A) \leq \mathrm{Adv}^{prp}_{E}(B) + 1,5 \cdot \frac{\bar{\sigma}^2}{2^n}$$

Além disso, B faz até $q' = \sigma + 2q + 1$ consultas ao oráculo e executa em tempo máximo $t' = t + q'\delta + O(n\bar{\sigma})$, em que δ é o tempo para calcular E.

Já que consideramos E como uma cifra de bloco segura, a vantagem IND-CPA de um adversário contra o modo de operação OCB é de até $\frac{1,5\bar{\sigma}^2}{2^n}$. O esquema é seguro apenas se o número total de blocos fornecidos ao adversário for pequeno em relação a $2^{\frac{n}{2}}$, o que também acontece no modo CBC.

Como os modos de operação CBC e OCB são seguros até um número próximo de $2^{\frac{n}{2}}$ blocos, sem grande diferença desta quantidade entre os dois esquemas, não seria correto dizer que um é melhor que o outro no quesito segurança, dentro do contexto IND-CPA. Algumas recomendações interessantes para estes esquemas são a troca de chave após $2^{\frac{n}{2}}$ blocos cifrados e um tamanho de bloco grande para a cifra utilizada (um valor grande para n).

6.3.2 Condições de Segurança Semântica

Na seção anterior, utilizamos o conceito de Ataque de Texto Legível Escolhido para comparar os protocolos MiniSec-U e TinySec. Nesta seção, utilizaremos o mesmo conceito, entretanto, o foco está nas condições oferecidas pelos protocolos para que se tenha segurança semântica (veja a seção 2.3.1).

A idéia geral é entender se os mecanismos utilizados, ou seja, os modos de operação CBC e OCB, permitem o vazamento parcial de informações. Iniciaremos com a apresentação de dois experimentos que auxiliam na definição do conceito de segurança semântica sob um Ataque de Texto Legível Escolhido (SEM-CPA).

Seja $\mathcal{M}$ um espaço de mensagens e f uma função definida em $\mathcal{M}$. Seja ainda SE um esquema de cifragem. A seguir, definimos os experimentos $\mathrm{Exp}^{ss-cpa-0}_{SE}(A)$ e $\mathrm{Exp}^{ss-cpa-1}_{SE}(A)$, que recebem como entrada um adversário A e devolvem um texto cifrado a partir de uma mensagem aleatoriamente escolhida de $\mathcal{M}$.

Experimento $\mathrm{Exp}^{ss-cpa-0}_{SE}(A)$:

1. A escolhe um espaço de mensagens $\mathcal{M}$ e uma função f.
2. $M, M^{'}$ são escolhidos de forma aleatória e independente a partir de $\mathcal{M}$.
3. $M^{'}$ é cifrada, ou seja, $C \leftarrow E_k(M^{'})$.
4. O texto cifrado é devolvido ao adversário A, o qual calcula $f(M)$.

Experimento $\mathrm{Exp}_{SE}^{ss-cpa-1}(A)$:

1. A escolhe espaço de mensagens $\mathcal{M}$ e função f.
2. $M, M^{'}$ são escolhidos de forma aleatória e independente a partir de $\mathcal{M}$.
3. M é cifrada, ou seja, $C \leftarrow E_k(M)$.
4. O texto cifrado é devolvido ao adversário A, o qual calcula $f(M)$.

Informalmente, podemos dizer que $f(M)$ representa uma informação sobre M. Observe que no primeiro experimento o adversário não tem posse de qualquer informação relacionada com M. O esquema é seguro se o adversário obtém sucesso em calcular $f(M)$ com a mesma frequência nos dois experimentos, independente de f e do espaço $\mathcal{M}$, escolhidos pelo adversário. Isto está formalizado na definição 8.

Definição 8. *Sejam os experimentos* $\mathrm{Exp}_{SE}^{ss-cpa-0}(A)$ *e* $\mathrm{Exp}_{SE}^{ss-cpa-1}(A)$*, definidos anteriormente. Seja* $Exp_{SE}^{ss-cpa-i}(A)$ *o evento em que o adversário* A *tem sucesso em calcular* $f(M)$ *para o experimento de índice* i*. Sejam ainda* A *um algoritmo adversário e* SE *um esquema de cifragem. O nível de segurança SEM-CPA ou valor SEM-CPA de* SE *em relação a* A*, ou ainda, a vantagem de* A *contra* SE *sob SEM-CPA é definida como a seguir.*

$$\mathrm{Adv}_{SE}^{sem-cpa}(A) = \Pr\left[\mathrm{Exp}_{SE}^{ss-cpa-1}(A)\right] - \Pr\left[\mathrm{Exp}_{SE}^{ss-cpa-0}(A)\right].$$

Os experimentos anteriores podem ser repetidos para que o adversário torne o ataque adaptativo, ou seja, ele pode escolher espaços de mensagem diferentes a cada experimento, assim como informações parciais que ele tentará obter das mensagens.

É possível criar um algoritmo adversário, digamos B, que ataque a segurança de um esquema de cifragem SE no contexto IND-CPA. B teria acesso a um oráculo de duas versões como o da figura 6.1. Assim, o algoritmo B poderia coincidir com os experimentos $\mathrm{Exp}_{SE}^{ss-cpa-1}(A)$ e $\mathrm{Exp}_{SE}^{ss-cpa-0}(A)$, de acordo com a versão do oráculo instanciada. Isto seria suficiente para entender a proximidade dos valores IND-CPA e SEM-CPA, ou ainda, isto implicaria em $\mathrm{Adv}_{SE}^{ind-cpa}(B) \geq \mathrm{Adv}_{SE}^{sem-cpa}(A)$, o que é apresentado no teorema 3.

Teorema 3. *Sejam* SE *um esquema de cifragem e* A *um algoritmo que ataca a segurança SEM-CPA de* SE *com até* q *chamadas a um oráculo que totalizam até* μ *bits. Além disso,* A *executa*

em tempo máximo t. Então existe um adversário B que ataca a segurança IND-CPA de SE e que obtém a seguinte vantagem:

$$\mathrm{Adv}_{SE}^{ind-cpa}(B) \geq \mathrm{Adv}_{SE}^{sem-cpa}(A)$$

Além disso, B roda em tempo $t + O(\mu)$ e realiza até q chamadas ao oráculo, totalizando μ bits.

Portanto o teorema 3 traz a idéia de que os modos de operação CBC e OCB serão seguros no contexto SEM-CPA se eles também forem seguros no contexto IND-CPA. Informalmente, isto acontece porque a vantagem de um adversário obter informação parcial de textos cifrados não é maior que a vantagem de se saber qual texto claro, dentre dois escolhidos e de posse do texto cifrado com apenas um deles, foi utilizado para cifragem.

Na realidade IND-CPA e SEM-CPA são equivalentes, o que foi introduzido em [58] e apresentado especificamente para criptografia assimétrica em [90]. Referências de demonstrações específicas para criptografica simétrica estão em [9, 13]. Uma apresentação seminal dos conceitos está em [31]. Em geral, essas referências também demonstram implicações para outros tipos de ataque, que também são considerados como noções fortes de segurança, como por exemplo o Ataque de Texto Cifrado Escolhido (CCA).

6.3.3 Valor Inicial Previsível

Os mecanismos OCB e CBC utilizam um VI, o qual também é chamado de nonce no modo OCB (veja seções 3.3.4 e 3.3.4). Há uma importante diferença desse parâmetro entre os mecanismos: o modo CBC não é seguro com um VI previsível [74]. Um contador como VI, por exemplo, torna o CBC vulnerável, o que não acontece no modo OCB. Um exemplo de ataque pode ser encontrado em [13].

O protocolo TinySec cifra um contador antes de utilizá-lo como VI para o modo CBC. Se a cifra de bloco for segura, então temos que o VI não pode ser calculado pelo adversário em tempo polinomial.

6.3.4 Cifra de Bloco

Como os dois protocolos utilizam a mesma cifra de bloco, ou seja, o Skipjack, é interessante apresentar vantagens e desvantagens deste mecanismo. Em [38], são apresentadas algumas desvantagens do Skipjack e algumas alternativas eficientes, sendo uma delas elaborada especialmente para redes de sensores.

As maiores dúvidas em relação ao Skipjack são em relação ao seu tamanho de chave e de bloco, pequenos em relação a novas demandas de padrões dos últimos anos, como foi o caso do AES (Advanced Encryption Standard). Além disso, há uma criptanálise diferencial desta cifra em [14], mas não há trabalhos mostrados como corretos que de fato consigam um ataque em todos os rounds do Skipjack.

As vantagens do Skipjack realmente são em relação ao custo-benefício. A eficiência em consumo

de tempo e energia desta cifra foi mostrada em vários trabalhos [33, 38, 42], o que torna esta cifra atrativa em redes de sensores. Seria interessante a possibilidade de se utilizar essa cifra com maiores tamanhos de chave e de bloco sem apresentar novas deficiências, mas não há trabalhos que obtiveram sucesso com essa atividade.

De qualquer forma, uma cifra de bloco segura não garante um mecanismo geral seguro. O modo de operação ECB com uma cifra de bloco segura qualquer, por exemplo, é vulnerável a ataques, como mostrado em [13]. Deve-se escolher cifras de blocos consideradas seguras, porém isso é pouco relevante se o modo de operação não for forte.

6.4 Autenticidade e Integridade de Mensagem

Durante a análise de confidencialidade e segurança semântica (seção 6.3) apresentamos experimentos em que um adversário pode realizar consultas a um oráculo ou caixa preta, permitindo sua interação com um sistema criptográfico. Uma característica interessante desses experimentos é que um adversário não precisa escolher previamente todos os argumentos que serão enviados ao oráculo. É possível que tal escolha seja feita de forma adaptativa. Como o objetivo do adversário é atacar a segurança do sistema criptográfico em questão, esses experimentos são chamados de ataques adaptativos (veja seção 6.2). Nesta seção também seguiremos este modelo, tendo como base [11, 13, 75].

6.4.1 Considerações Iniciais

Os experimentos desta seção têm como objetivo apresentar aspectos de segurança dos algoritmos CBC-MAC e OCB, no contexto da autenticidade e integridade fornecida por esses mecanismos. Em geral, teremos o seguinte experimento: um adversário escolhe uma mensagem, obtem a tag de autenticação correspondente através de uma consulta ao oráculo, escolhe outra mensagem, obtem a tag correspondente através de nova consulta, e assim por diante. Após várias interações com o sistema, o adversário poderá tentar fraudar um par (mensagem, tag), vencendo somente se obter sucesso em fraudar um par em que a mensagem não pode ter sido passada como parâmetro durante consultas anteriores ao oráculo.

Como os algoritmos CBC-MAC e OCB utilizam uma cifra de bloco para produzir a tag de autenticação, é necessário considerar que esta cifra é segura. Uma discussão sobre este aspecto é apresentada na seção 6.2. Além disso, é importante entender que o OCB utiliza parte do texto cifrado para produzir a tag, o que não acontece com o CBC-MAC, o qual utiliza somente a chave e o texto legível como entrada para o algoritmo de geração de tag. Por isso, o experimento relacionado com o OCB envolve um oráculo de decifragem. Outro detalhe é que o CBC-MAC utiliza como VI uma cadeia de zeros e o OCB utiliza um VI que nunca repete, chamado de nonce. Consideramos que o nonce nunca é reutilizado.

6.4.2 Tentativa de fraudar o algoritmo CBC-MAC

Seja $\mathrm{GAuthCBC}_K(M)$ um oráculo (algoritmo CBC-MAC) que devolve a tag de autenticação correspondente à mensagem M. Seja o oráculo de verificação $\mathrm{VFCBC}_K(\cdot,\cdot)$ da figura 6.4. Apresentamos a seguir o experimento utilizado para definir a vantagem de um adversário A em fraudar

Oráculo $VFCBC_K(M, tag)$

Entradas: mensagem M e tag de autenticação

Saída: 1 se o par (M, tag) for válido. 0 caso contrário.

1 $\text{tag}' \leftarrow \text{GAuthCBC}_K(M)$
2 **Se** $\text{tag} = \text{tag}'$
3 **então devolva** 1
4 **devolva** 0

Figura 6.4: Oráculo de Verificação para o algoritmo CBC. $\text{GAuthCBC}_K(M)$ é um algoritmo que devolve a tag de autenticação correspondente a M e K

Oráculo $VFOCB_K(N, C)$

Entradas: nonce N e texto cifrado C.

Saída: 1 se o par (N, C) for válido. 0 caso contrário.

1 **Se** $\text{D}_K(N, C) \neq \perp$
2 **então devolva** 1
3 **devolva** 0

Figura 6.5: Oráculo de Verificação para o algoritmo OCB.

um par (M,tag), em que tag é o valor correspondente à mensagem M, em relação ao algoritmo CBC-MAC.

Experimento $\text{Exp}^{auth}_{CBC}(A)$:

1. A realiza q consultas $\text{GAuthCBC}_K(\cdot)$ e uma consulta $\text{VFCBC}_K(\cdot, \cdot)$, escolhendo mensagens de forma adaptativa.

2. Se A fez uma consulta de verificação $\text{VFCBC}_K(M, \text{tag})$ e obteve 1 como resposta

   - Então, se A não utilizou M em consultas anteriores a $\text{GAuthCBC}_K(\cdot)$, devolva 1

3. Devolva 0

6.4.3 Tentativa de fraudar o algoritmo OCB

Seja o oráculo de verificação $\text{VFOCB}_K(\cdot, \cdot)$ da figura 6.5. Apresentamos a seguir o experimento utilizado para definir a vantagem de um adversário A em fraudar um par (N,C), em que o nonce N é o VI correspondente à geração do texto cifrado C, em relação ao algoritmo OCB.

Experimento $\text{Exp}^{auth}_{OCB}(A)$:

1. A realiza q consultas $\text{E}_K(N, M)$ e uma consulta $\text{VFOCB}_K(N, C)$, escolhendo de forma adaptativa N e M

2. Se A fez uma consulta de verificação $\text{VFOCB}_K(N, C)$ e obteve 1 como resposta

- Então, se A não utilizou (N, M) que resultou em C em consultas anteriores, devolva 1

3. Devolva 0

6.4.4 Vantagem de adversários contra os algoritmos CBC e OCB

Os teoremas que serão enunciados estão relacionados com a segurança dos mecanismos de autenticação OCB e CBC, os quais estão descritos nas seções 3.3.4 e 3.3.4, respectivamente. Esses teoremas apresentam as vantagens de adversários contra os mecanismos em questão através de consultas ao oráculo de geração de tag (ataque adaptativo) e, após essas consultas, uma única tentativa de fraudar o sistema criptográfico, a qual é a consulta ao oráculo de verificação. Antes de "quantificar a segurança", cabe definir as vantagens de adversários em fraudar os sistemas criptográficos estudados nesta seção.

Definição 9. *Seja $CMAC$ o algoritmo CBC-MAC definido na seção 3.3.4 e A um adversário. Seja o experimento $\mathrm{Exp}^{auth}_{CBC}(A)$ definido anteriormente. A vantagem de A em fraudar $CMAC$ é*

$$\mathrm{Adv}^{auth}_{CMAC}(A) = \Pr\left[\mathrm{Exp}^{auth}_{CBC}(A) = 1\right].$$

Definição 10. *Seja OCB o algoritmo definido na seção 3.3.4 e A um adversário. Seja o experimento $\mathrm{Exp}^{auth}_{OCB}(A)$ definido anteriormente. A vantagem de A em fraudar OCB é*

$$\mathrm{Adv}^{auth}_{OCB}(A) = \Pr\left[\mathrm{Exp}^{auth}_{OCB}(A) = 1\right].$$

Teorema 4. [75] *Seja OCB o algoritmo definido na seção 3.3.4, τ o tamanho da tag de autenticação e A um adversário que realiza até q consultas ao oráculo $\mathrm{E}_K(\cdot,\cdot)$, o que totaliza até σ blocos de n-bits, e apenas uma consulta ao oráculo de verificação da figura 6.5. Seja $\bar{\sigma} = \sigma + 2q + 5c + 11$, em que c é o número máximo de blocos produzidos por A em sua tentativa de fraudar a tag. Com os parâmetros n e τ fixos, temos que*

$$\mathrm{Adv}^{auth}_{OCB}(A) \leq 1,5 \cdot \frac{\bar{\sigma}^2}{2^n}$$

Teorema 5. [11] *Seja $CMAC$ o algoritmo CBC-MAC (definido na seção 3.3.4) sobre o espaço de mensagens $\{0,1\}^{mn}$, com $m \geq 1$. Seja A um adversário que realiza até q consultas ao oráculo de geração de tag $\mathrm{GAuthCBC}_K(\cdot)$ e uma consulta ao oráculo de verificação da figura 6.4. Com os parâmetros m e n fixos, temos que*

$$\mathrm{Adv}^{auth}_{CMAC}(A) \leq 1,5 \cdot \frac{q^2 m^2}{2^n}$$

A seguir, mostramos alguns exemplos numéricos que podem ser encontrados em [11, 75].

Suponha que um adversário utilize o CBC-MAC para autenticar 2^{30} mensagens de 16Kbyte cada. Fixe o tamanho de bloco em 128 bits. Logo, temos que $m = 2^{10}$ blocos. Então nenhum adversário,

após obter tags de $q = 2^{30}$ mensagens de forma adaptativa, tem uma vantagem maior que 10^{-14} em distinguir essas tags de cadeias binárias puramente aleatórias. Assim, nenhum adversário teria probabilidade maior que $10^{-14} + \frac{1}{2^{128}}$ em fraudar um novo valor MAC (tag) de 128 bits (o tamanho do valor MAC é justamente o tamanho de bloco).

Suponha agora que um adversário utilize o OCB para autenticar até 2^{40}byte de textos cifrados escolhidos de forma adaptativa. Fixe $n = 128$bits e uma tag de 64bits. Então nenhum adversário, depois de estudar 1 TB de dados, possui chance maior que $2^{-55} + \frac{1}{2^{-64}}$ em fraudar uma mensagem válida.

As fórmulas mostram que a chave deve ser trocada antes de $2^{n/2}$ blocos processados. Isto para que o mecanismo utilizado não tenha sua segurança comprometida através de ataques adaptativos. Além disso, se a cifra de bloco é segura, então os mecanismos também são seguros, o que significa dizer que um adversário não precisaria concentrar seus esforços de ataque no CBC-MAC, por exemplo, mas na cifra de bloco. Na realidade o CBC-MAC preserva a pseudo-aleatoriedade de uma função, ou seja, o CBC-MAC de uma função pseudo-aleatória produz uma função pseudo-aleatória [11]. No caso do OCB, se um adversário A consegue fraudar textos cifrados, então existe um adversário B que consegue distinguir entre saídas de oráculos de cifragem para o sistema em questão e para uma permutação aleatória [75].

6.4.5 Autenticação de Mensagens de Tamanho Fixo e Variável

Em relação ao CBC-MAC, o teorema 5 fixa o tamanho das mensagens (apenas mensagems com m blocos de n-bits, para m e n fixos), sendo a vantagem de um adversário uma função do número de consultas a um oráculo. Isso é extremamente importante para a segurança deste mecanismo. Um exemplo interessante é apresentado a seguir, sendo ainda consideramos o uso de uma cifra de bloco segura.

Seja o algoritmo CBC-MAC definido na seção 3.3.4. Considere que este algoritmo recebe como entrada apenas cadeias binárias que são múltiplas de n-bits[2], ou seja, $\{0,1\}^{mn}$ com n fixo e $m \geq 1$. Com cadeias de tamanho variável é possível que um adversário A comprometa a segurança do mecanismos através do seguinte jogo:

1. A escolhe M e realiza uma consulta a $f^{(1)}(M)$, recebendo t_m.
2. A fornece t_m e realiza um consulta a $f^{(1)}(t_m)$, recebendo t_{t_m}
3. A calcula o valor MAC de $M||0$, $f^{(2)}(M||0) = t_{t_m}$. A cadeia $M||0$ não foi utilizada em consultas anteriores à função $f^{(m)}$ e mesmo assim A conseguiu calcular o seu valor MAC.

Este jogo pode ser convertido para o experimento $\mathrm{Exp}^{auth}_{CBC}(A)$ (veja seção 6.4.2). Basta trocar a função $f^{(m)}$ pelo oráculo $\mathrm{GAuthCBC}_K(\cdot)$ e, por fim, realizar uma consulta $\mathrm{VFCBC}_K(M||0, t_{t_m})$. Desta forma, o experimento em questão devolve 1, pois a cadeia $M||0$ não foi utilizada em consultas anteriores ao oráculo de geração de tag. Portanto esta construção mostra que o algoritmo CBC-MAC para tamanhos de cadeias variáveis não é seguro.

[2]Isto é viável na prática, já que poderiam ser concatenadas à entrada cadeias de 0s ou 1s

Como o CBC-MAC em sua implementação original apenas preserva a aleatoriedade se a entrada for uma cadeia de tamanho fixo, resta saber se há implementações deste algoritmo que preservam a segurança mesmo com o uso de mensagens de tamanhos variáveis. Em [11] há três métodos demonstrados seguros, sendo que o TinySec utiliza o seguinte método:

- Cifra o tamanho da mensagem, em blocos, e utiliza o resultado como VI.

Assim, a única diferença entre esta variante e a versão original está na primeira iteração do algoritmo, a qual é a aplicação de uma função ao resultado do XOR do VI alterado com o primeiro bloco do texto legível.

O mecanismo OCB não apresenta vulnerabilidades com o uso de cadeias de tamanho variável [75].

6.5 *Freshness*: Proteção Contra Ataques do Tipo Replay

Para fornecer a propriedade de confidencialidade, o protocolo TinySec utiliza um contador como parte do VI, o qual é fornecido como entrada para o algoritmo CBC. Este contador também poderia ser utilizado para evitar a replicação de pacotes por um adversário. Entretanto, o TinySec não fornece este tipo de proteção, já o MiniSec-U, utiliza justamente um contador como proteção contra esses ataques.

6.6 Resistência a Comprometimento de Nós da Rede

Evitar o comprometimento de nós de uma RSSFs é essencialmente tratar da questão do gerenciamento de chaves, pois uma chave global compartilhada entre os nós de uma rede, se obtida através de ataques físicos, comprometeria a segurança de toda a rede.

Os protocolos TinySec e MiniSec-U não especificam mecanismos relacionados com o gerenciamento de chaves criptográficas. Há apenas a recomendação do MiniSec-U de se utilizar diferentes chaves para nós remetentes. Esta é uma grave omissão dos protocolos. Com os resultados obtidos em seções anteriores é possível observar que a troca das chaves é fundamental para manter o nível de segurança dos mecanismos criptográficos. Ataques adaptativos podem ocorrer e em determinados momentos há maiores chances de sucesso, mas com a existência de um protocolo específico de mudança das chaves dificilmente um advesário teria chances relevantes em quebrar os mecanismos utilizados pelos protocolos TinySec e MiniSec-U.

6.7 Demais Propriedades

6.7.1 Autenticação Imediata de Mensagens

Os protocolos MiniSec-U e TinySec não apresentam atrasos relacionados com a autenticação de mensagens, exceto aqueles ocasionados pelo consumo de tempo dos algoritmos de verificação de tags. Além disso, as mensagens processadas por um receptor não são armazenadas para futuras consultas.

6.7.2 Baixo Sobrecusto Computacional e de Comunicação

Um sobrecusto computacional de grande importância é o número de chamadas para a cifra de bloco, o que é realizado pelo modo de operação adotado pelo protocolo. O modo CBC possui $|M|/n$ chamadas, sendo $|M|$ o tamanho da mensagem que se deseja cifrar e n o tamanho de bloco. Já o modo OCB, possui $\lceil |M|/n \rceil + 2$ chamadas. Utilizando-se apenas o mecanismo de cifragem, temos que o CBC-Skipjack é mais eficiente que o OCB-Skipjack neste quesito. Entretanto, o protocolo TinySec utiliza o CBC-MAC em conjunto com o CBC para que se tenha autenticação e cifragem em dois passos distintos. Isto resulta em $2\lceil |M|/n \rceil + 1$ chamadas para a cifra de bloco. Portanto o modo OCB-Skipjack, utilizado pelo MiniSec-U, é mais eficiente em chamadas para a cifra de bloco que o "CBC-MAC + CBC-Skipjack" adotado pelo TinySec.

Resultados experimentais contidos em [75] mostram que o OCB realmente apresenta menor sobrecusto computacional. Os experimentos deste trabalho são apresentados na seção 6.8.4 e envolvem resultados do consumo de energia de dispositivos sensores que implementam os protocolos estudados.

Em relação a sobrecustos de comunicação, temos que o MiniSec-U possui um tamanho de pacote menor que o TinySec (2 bytes de diferença). Isto acontece porque o MiniSec-U utiliza um contador compartilhado entre um remetente e um destinatário, ou seja, um contador não é enviado de forma integral através de um pacote de rede. A tabela 6.2 apresenta o aumento do tamanho de pacote de cada protocolo.

	Sobrecusto de Autenticação	**Sobrecusto de Confidencialidade**	**Sobrecusto Total de Pacote**
MiniSec-U	2	1	3
TinySec	2	3	5

Tabela 6.2: Aumento do tamanho de pacote, em bytes, dos protocolos de segurança MiniSec-U e TinySec em relação à versão original de pacote do sistema TinyOS.

6.7.3 Resistência a perda de pacotes da rede

Como o protocolo MiniSec-U utiliza um contador compartilhado entre um remetente e um destinatário, uma alta taxa de perda de pacotes da rede poderia ocasionar maior consumo de energia dos dispositivos sensores. Isto porque um contador é transmitido parcialmente e, em caso de perdas, pode haver necessidade de uma sincronização ou até a perda da informação, já que contadores são utilizados como VI para cifragem e portanto não é possível efetuar o processo inverso (decifragem) sem a posse do VI.

O protocolo MiniSec-U estabelece um número de tentativas de decifragem antes de efetuar a sincronização de um contador compartilhado. Essas tentativas são uma forma de "prever" o VI[3], o que é chamado de *sincronização implícita*. Em caso de falha na sincronização implícita, pode-se

[3]O mecanismo utilizado pelo MiniSec-U permite que o VI seja previsível, ou seja, não há necessidade de utilizar uma função pseudo-aleatória para gerar um novo VI.

optar pelo uso de um protocolo de sincronização de contadores ou pelo descarte da informação.

O protocolo TinySec não apresenta aumento significativo do consumo de energia dos dispositivos em caso de perda de pacotes da rede. Isto porque este protocolo envia o contador de forma integral a cada troca de mensagens e portanto não necessita de processamento adicional ou sincronização de informações no momento da decifragem ou verificação de autenticidade de mensagens. Alguns experimentos destes protocolos em relação à taxa de perda da rede estão em [51].

6.7.4 Escalabilidade

Os protocolos TinySec e MiniSec-U não apresentam restrições em relação ao aumento do número de nós da rede. A escalabilidade geralmente é crítica em protocolos para comunicação *broadcast*, pois para prover comunicação segura, todos os nós que recebem mensagens devem possuir um contador para cada remetente. Portanto a comunicação *broadcast* que envolve muitos nós que enviam mensagens requer dispositivos sensores com maior capacidade de armazenamento, além de implicações para o consumo de energia. O simples abandono de um contador para permitir a escalabilidade implicaria em menor segurança, como por exemplo a perda da proteção contra ataques do tipo replay.

6.8 Análise Experimental: Implementação e Simulação

A implementação e simulação tem como objetivo comparar o comportamento dos protocolos TinySec e MiniSec-U em relação ao consumo de energia de cada dispositivo da rede simulada. O principal ingrediente para esta atividade é o sistema TinyOS, o qual é bastante utilizado para gerenciar os recursos de hardware de dispositivos sensores e fornecer uma interface com o usuário do dispositivo.

Inicialmente apresentaremos a estrutura do sistema TinyOS e o funcionamento básico dos módulos TOSSIM, que simula eventos da rede, e PowerTOSSIM, que estima o consumo de energia. Depois trataremos da metodologia de implementação para finalmente mostrarmos os resultados obtidos.

6.8.1 TinyOS

O TinyOS [47] é um sistema operacional baseado em eventos que foi projetado para uso em dispositivos de redes de sensores. Os principais desafios que motivaram o desenvolvimento do TinyOS são fornecer suporte a operações concorrentes e permitir baixo consumo de energia.

Inicialmente desenvolvido na Universidade da Califórnia, em Berkley, o TinyOS atraiu uma comunidade de respeitadas instituições acadêmicas e comerciais, as quais formaram a TinyOS Open Technology Alliance [2], que fornece manutenção e desenvolvimento para o sistema. Além disso, protocolos [34, 66, 87, 93, 95], sistemas [27, 45] e aplicações [45, 53, 54, 94] foram construídas com o uso do TinyOS.

A composição básica deste sistema é um conjunto de componentes reutilizáveis e um escalonador de tarefas.

Características do TinyOS

O TinyOS possui um modelo de programação baseado em componentes, o qual é fornecido através da linguagem nesC [30]. Isto o torna também um arcabouço de programação para sistemas de plataformas específicas. Logo, os programas do TinyOS são construídos a partir de componentes, os quais possuem duas importantes abstrações computacionais: eventos e tarefas. As tarefas fornecem concorrência interna para um componente, já os eventos, fornecem comunicação interna e externa. Os comandos formam um outro tipo de abstração e consistem de requisições para outros componentes, uma espécie de chamada de função para realização de serviços, como por exemplo iniciar uma leitura a partir de um sensor, sendo que o fim desta execução é sinalizado por um evento. Exemplos de tarefas são computar a média de um vetor e mostrar dados em um display de um dispositivo. Exemplos de eventos são confirmações de envio e recebimento de mensagens e interrupções de hardware. Nota-se que eventos geralmente possuem pequena quantidade de processamento, enquanto tarefas podem possuir justamente o contrário, ou seja, uma grande quantidade de processamento. Além disso, tarefas normalmente não são críticas em relação ao tempo e podem sofrer preempção a partir de eventos, ou seja, eventos podem interromper o processamento de tarefas por um intervalo de tempo, mas a recíproca não é verdadeira. A tabela 6.3 apresenta alguns exemplos de comandos, tarefas e eventos.

comando	getData()	Aciona leitura a partir do sensor
evento	dataReady()	Sinal de fim de leitura do sensor
comando	send()	Envio de uma mensagem
evento	sendDone()	Sinal indicando que uma mensagem foi enviada
evento	received()	Sinal indicando que uma mensagem foi recebida
evento	fired()	Indicação de fim de um período de tempo
tarefa	encrypt()	Cifragem de uma mensagem
tarefa	sendit()	Preparação de uma mensagem antes do envio
tarefa	update_routing()	Atualiza árvore de roteamento (Multihop)

Tabela 6.3: Exemplos de Comandos, Eventos e Tarefas Implementadas em linguagem nesC para o sistema TinyOS.

Escrever aplicações em TinyOS significa desenvolver componentes para plataformas específicas e conectá-los aos componentes do sistema. Por isso a denominação programação baseada em componentes. O TinyOS fornece um grande número de componentes como por exemplo abstrações para sensores, gerenciamento de energia e armazenamento.

Como estabelecer a comunicação entre componentes através dos comandos e eventos?

As interfaces são responsáveis pela comunicação entre os componentes, pois declaram os comandos e eventos fornecidos ou utilizados pelos componentes. Isto quer dizer que um componente pode implementar comandos e eventos ou apenas utilizar implementações de outros componentes. Diversas interfaces também são fornecidas pelo TinyOS, como por exemplo envio e recebimento de mensagens.

Uma aplicação desenvolvida em TinyOS é compilada em apenas um executável, composto de componentes e um escalonador. Não há separação entre espaço de usuário e kernel como nos sistemas operacionais tradicionais. É como se o sistema operacional estivesse integrado com a aplicação em apenas um executável, sendo que não existe um kernel, mas uma manipulação direta do hardware. Outros conceitos ausentes em relação aos sistemas operacionais tradicionais são gerenciamento de processos e memória virtual, já que há apenas um processo executando por vez e um único espaço de endereçamento físico. Não há alocação dinâmica de memória, pois alocações são feitas apenas em tempo de compilação.

As características descritas são a solução do TinyOS para suportar um ambiente com recursos extremamente escassos de memória, processamento e energia, além de considerar um número mínimo de sobrecustos no sistema e permitir operações intensamente concorrentes.

6.8.2 TOSSIM: Um Simulador de Eventos para TinyOS

O TOSSIM [44, 46] é um simulador escalável de eventos de redes de sensores que utiliza a estrutura de componentes do TinyOS. A arquitetura deste simulador é composta de cinco partes:

- Suporte para compilação de componentes do TinyOS em uma infra-estrutura de simulação
- Uma fila de eventos
- Re-implementação de alguns componentes do TinyOS que abstraem o hardware
- Serviços de comunicação para que programas externos possam interagir com uma simulação
- Mecanismos para gerência de Convervão Analógica-Digital (ADC) e rádio

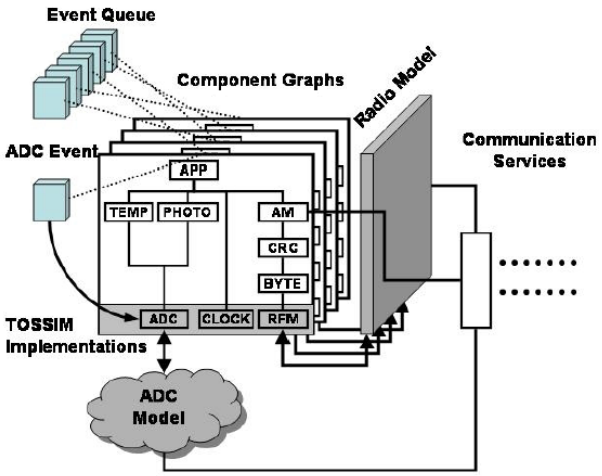


Figura 6.6: Arquitetura do TOSSIM [46].

A figura 6.6 mostra a arquitetura geral do TOSSIM. A substituição de alguns componentes de hardware do TinyOS (cor cinza na figura) permite a simulação do comportamento de um dispositivo.

O compilador nesC também é alterado para permitir a compilação de um código TinyOS para uma simulação.

A re-implementação de componentes de baixo nível do TinyOS torna o TOSSIM um simulador interessante para protocolos como TinySec e MiniSec-U. Isto porque estes protocolos estão mais próximos do hardware do que da aplicação, já que alteram a estrutura das mensagens enviadas pela rede. Outra característica favorável é a compilação de códigos nativos do TinyOS para geração de simulação através do TOSSIM, já que o simulador utiliza componentes do TinyOS. Isso permite uma rápida transição entre simulação e teste real em hardware. Apesar disso, é possível que um código executado em uma simulação apresente problemas em uma execução através de hardware sensores reais, isto porque o TOSSIM considera que interrupções não podem gerar preempção, ou seja, diferente do conceito do TinyOS, eventos não poderiam por exemplo interromper a execução de tarefas, o que iria gerar efeitos colaterais no sistema.

O TOSSIM pode ajudar a entender o comportamento de algoritmos e compará-los através de diversos tipos de ambientes programados. Por exemplo, seria possível simular uma rede de sensores considerando a possibilidade da perda de sinal entre alguns nós, ou ainda gerar uma simulação com transmissão perfeita de pacotes, sendo esta última interessante para observar e comparar algoritmos em um ambiente isolado e sem perdas. Apesar desta complexidade, é importante observar se o ambiente criado está relacionado de alguma forma com um cenário real. Em [46] são apresentadas algumas falhas em componentes do TinyOS que foram descobertas através de simulações com o TOSSIM.

PowerTOSSIM

A utilização do simulador TOSSIM permite observar o comportamento de uma rede de sensores, porém não é possível obter uma estimativa do consumo de energia dos dispositivos sem o uso do PowerTOSSIM [80]. Este módulo do TinyOS opera juntamente com o TOSSIM e fornece uma medida com precisão entre 0,45% e 13% da energia real de um dispositivo que executa uma aplicação idêntica.

O PowerTOSSIM foi projetado para simular o consumo de energia de um nó particular que executa uma aplicação específica. A figura 6.7 mostra a arquitetura do PowerTOSSIM, em que componentes que simulam o hardware (radio CC1000, sensores ADC, LEDs, etc) realizam chamadas para o módulo PowerState, o qual emite mensagens sobre o consumo de cada componente. As mensagens podem ser combinadas com um modelo de energia específico para cada plataforma, o que permite um detalhamento do consumo de energia de um dispositivo.

Características interessantes do PowerTOSSIM são a flexibilidade a modelos de energia de diferentes plataformas e a eficiência em simular uma rede com centenas de nós. Em contrapartida, este simulador possui limitações, sendo a principal delas a dificuldade em se detectar variações de energia que podem ocorrer devido a pequenas mudanças na diferença de potencial (voltagem) da bateria de um dispositivo, já que a voltagem tem efeito quadrático em energia. Como as flutuações de voltagem e a interação com o ambiente podem dificultar a precisão da medição, é interessante evitar ambientes de simulação com muitos eventos (leitura de sensores, envio e recebimento de

mensagens, manipulação de LEDS, etc). Comparações com experimentos reais mostram que poucos eventos simulados podem ter uma precisão bastante significativa. Um exemplo de sucesso é o uso dos seguintes eventos: leitura de sensores de temperatura e envio/recebimento de mensagens.

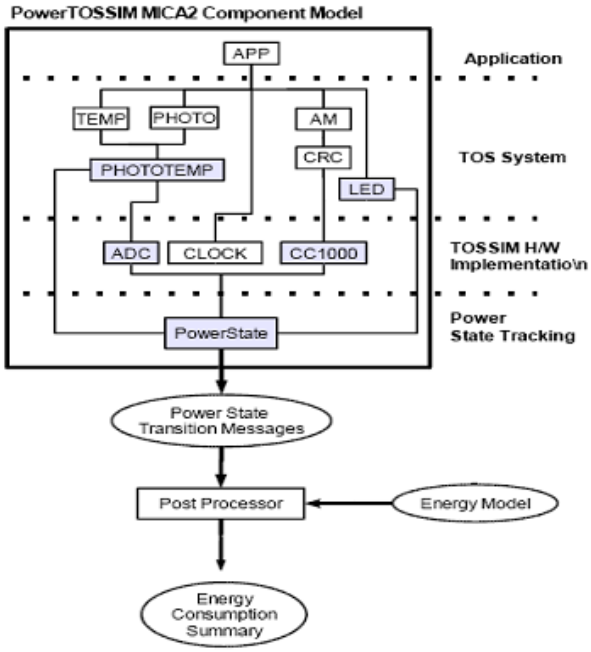


Figura 6.7: Arquitetura do PowerTOSSIM [80].

6.8.3 Metodologia de Implementação

Plataforma Utilizada, Software e Versões

Todos os resultados de simulações são baseados na plataforma Mica2 [4]. Este dispositivo sensor foi desenvolvido pela Universidade de Berkley e comercializado pela Crossbow Technology. O Mica2 utiliza o microcontrolador ATmega128L e possui 128 KB de memória de programa (FLASH), 4 KB de memória de dados (SRAM) e 512 KB de memória de armazenamento.

Os sistemas utilizados estão listados na tabela 6.4.

Software	**Versões**
XUbuntuOS	kernel 2.6.20-15
TinyOS	versão 1.1
TOSSIM	incluído com TinyOS 1.1
PowerTOSSIM	incluído com TinyOS 1.1
TinySec	adaptado da versão 1.12, incluída com TinyOS 1.1
MiniSec-U	adaptado da versão 2.0 (plataforma Telos)
SkipJack	implementação de Naveen Sastry
OCB	implementação de Ted Krovitz

Tabela 6.4: Versões de Software Utilizados em Simulações.

As implementações da cifra de bloco Skipjack e do modo de operação OCB estão de acordo com as especificações originais do NIST [60] e de Rogaway [75], respectivamente. Informações sobre a implementação original do MiniSec-U podem ser obtidas em [51].

Aplicação e Parâmetros Específicos de Simulação

Além dos sistemas TinyOS, TOSSIM e PowerTOSSIM e dos protocolos MiniSec-U e TinySec, é necessário o uso de uma aplicação que gerencie a simulação de uma rede de sensores. Esta aplicação é responsável por utilizar componentes dos sistemas envolvidos de forma que os dispositivos da rede possam trocar mensagens com o uso de um dos protocolos de segurança. A aplicação Surge, incluída com o TinyOS 1.1, foi adaptada e utilizada por dois principais motivos: adota o modelo multihop de comunicação e possui registro de apenas 3% de erro entre energia real e simulada (a energia real é um pouco menor) [80].

O modelo multihop é muito utilizado em redes de sensores. Dois nós da rede que necessitam trocar mensagens podem estar distantes, o que dificulta a comunicação direta. Neste caso um ou mais nós intermediários recebem e reenviam as mensagens até que elas cheguem ao destinatário. A implementação do modelo multihop é feita através do uso de componentes do TinyOS que criam e gerenciam uma árvore de roteamento, sendo a estação base a raiz desta árvore. Nesta estrutura, cada nó possui um ponteiro para o respectivo nó pai, não sendo possível o envio de mensagens de um nó pai para um nó filho. Desta forma, os nós folhas, por exemplo, enviam mensagens para a estação base através dos nós internos.

A aplicação Surge funciona da seguinte forma: os nós da rede efetuam leituras com um sensor e as enviam para a estação base através da árvore de roteamento. É enviada apenas uma mensagem a cada dois segundos, pois uma alta taxa de envio poderia congestionar a fila de mensagens mantida pela aplicação e isto comprometeria os resultados. Esta aplicação foi adaptada para que fosse possível o uso dos protocolos MiniSec-U e TinySec.

Alguns parâmetros utilizados durante a simulação estão listados na tabela 6.5. A escolha desses parâmetros estão alinhadas com as especificações dos protocolos. Por exemplo, a Otimização LB, existente apenas no protocolo MiniSec-U, é definida com o padrão de 3 bits. O tamanho de chave e de bloco é fixo pela cifra Skipjack, a qual foi adotada pelos protocolos estudados.

	MiniSec-U	**TinySec**
Confidencialidade	OCB-Skipjack	CBC-Skipjack
Autenticação	OCB	CBC-MAC
Tamanho de Bloco	64 bits	64 bits
Tamanho de Chave	80 bits	80 bits
Otimização LB	3 bits	N/A

Tabela 6.5: Parâmetros de Implementação dos Mecanismos dos Protocolos TinySec e MiniSec-U

Compilação, Execução e Pósprocessamento

A compilação da aplicação Surge gera um arquivo para simulação que pode ser executado diretamente do computador. A diferença entre compilar para o sensor e para um computador pessoal é geralmente uma simples questão de sintaxe, já que o simulador utiliza um código próximo ao nativo do sistema TinyOS.

A simulação pode ser executada em uma console (modo texto) do sistema operacional ou através de um plugin Java (modo gráfico). Neste trabalho, as simulações foram executadas apenas em modo texto. Todos os comandos e eventos foram programados através da aplicação, sem necessidade de interação em tempo real com a simulação.

A execução da simulação produz um arquivo que contém os eventos gerados pelos nós sensores. Este arquivo deve ser processado pelo PowerTOSSIM para que se tenha o resultado final, ou seja, o consumo de energia dos nós sensores. Logo, o processo que permite obter a estimativa de consumo de energia de um protocolo envolve a compilação da aplicação, seguida pela execução da simulação e de um pósprocessamento. Este processo foi repetido pelo menos três vezes para cada protocolo, sendo uma vez para cada quantidade de nós simulados.

6.8.4 Resultados dos Experimentos

Cada simulação de uma RSSF foi realizada por 1200 segundos e com o uso de apenas um protocolo de segurança. Simulações sem uma camada de segurança são apresentadas com a legenda TinyOS. Inicialmente, as redes simuladas possuem 10, 50 e 100 nós, sendo um nó de cada configuração reservado como estação base, a qual é a raiz da árvore dinâmica de roteamento. Além disso, não consideramos perda de pacotes, apesar de haver possibilidade de erros (pacotes corrompidos). Cenários com perdas e mais simulações são apresentados posteriormente.

Os gráficos da figura 6.8 representam o consumo total de energia dos nós em joules, incluindo a estação base. Nas figuras 6.8(a) e 6.8(b) observamos que o consumo de energia com o uso do protocolo MiniSec-U foi maior que aquele apresentado com o uso do TinySec. Porém, na figura 6.8(c), observamos o contrário, ou seja, o uso do protocolo MiniSec-U apresentou menor consumo se comparado com o TinySec. Observando os dados experimentais foi possível entender que o uso do MiniSec-U implica em um consumo de energia alto para apenas poucos nós da rede. Isto significa que apesar da maioria dos nós apresentar baixo consumo de energia, "picos" de consumo de alguns nós elevam o valor total de energia consumido. Logo, quanto maior o número de nós, maior a chance de se economizar mais energia com o uso do MiniSec-U, já que poucos nós apresentam um consumo elevado.

Os gráficos da figura 6.9 apresentam a média do consumo de energia dos nós em milijoules, sendo que o gráfico da figura 6.9(b) descarta os nós que apresentaram consumo elevado com o uso de cada protocolo. Definimos como consumo de energia elevado um valor pelo menos a média mais duas vezes o desvio padrão, calculados para cada simulação. É interessante observar nestes gráficos que evitar o consumo elevado de alguns nós implicaria em maior eficiência em consumo de energia do MiniSec-U, se comparado com o protocolo TinySec. Neste cenário, duas questões são

importantes dentro do contexto do protocolo MiniSec-U:

1. O motivo da existência de picos de consumo
2. Como evitar os picos de consumo

O consumo elevado de alguns nós pode acontecer porque o protocolo MiniSec-U utiliza um contador sincronizado entre o remetente e o destinatário. Apesar deste conceito reduzir o tamanho do cabeçalho do pacote e portanto economizar energia durante a maioria das trocas de mensagens, pode haver situações em que o contador compartilhado não se mantenha atualizado. Isto requer maior atividade de CPU para que seja possível decifrar uma mensagem. Algumas vezes isto requer até maior atividade de rede ou o pior, um pacote pode não ser decifrado.

Evitar o consumo elevado de alguns nós da rede significa evitar a perda de sincronia do contador. Isto pode ser feito através do aumento do tamanho do contador, o que implica em aumentar o tamanho do pacote. Portanto esta solução também aumenta o consumo de energia de todos os nós.

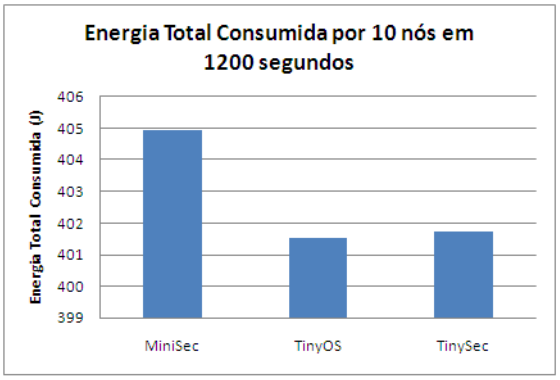
Algo interessante nas simulações é que não há discrepâncias significativas de consumo de energia entre nós de diferentes profundidades da árvore de roteamento. Por exemplo, a figura 6.10 , baseada em [80], mostra que os nós 3, 4 e 5 dependem do nó 1 para enviarem mensagens para a estação base (nó 0). Neste exemplo, o nó 1 deveria consumir maior quantidade de energia que os nós folhas. Dois fatores, já relatados em [80], permitem que os nós intermediários não apresentem consumo de energia relativamente alto após as simulações: a ausência de modos de economia de energia entre o envio de mensagens e a baixa taxa de envio de mensagens da aplicação utilizada (apenas uma a cada dois segundos). Outro fator importante é que a árvore de roteamento é dinâmica, ou seja, pode se alterar durante a simulação, o que não é incomum.

Uma comparação direta entre os protocolos TinySec e MiniSec-U com mais simulações é apresentada na figura 6.11. Essa comparação reforça a hipótese de que a economia de energia com o protocolo MiniSec-U é progressiva em um ambiente livre de perdas de pacotes e com comunicação multihop.

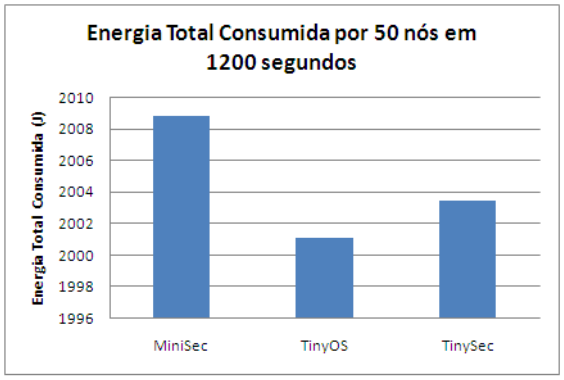
Cenário em uma Rede com Perda de Pacotes

Como as simulações anteriores não consideraram um cenário de perdas, é interessante um novo conjunto de simulações. Neste ambiente, consideramos apenas redes com dispositivos sensores distribuídos em uma área geográfica de 100x100m. Isso foi possível com a utilização do programa LossyBuilder, o qual gera um arquivo de entrada para a simulação. Maiores informações sobre o ambiente de perdas do TOSSIM podem ser encontradas em [44, 46]. Outro detalhe importante é que o alcance de rádio de cada dispositivo é de 15 metros.

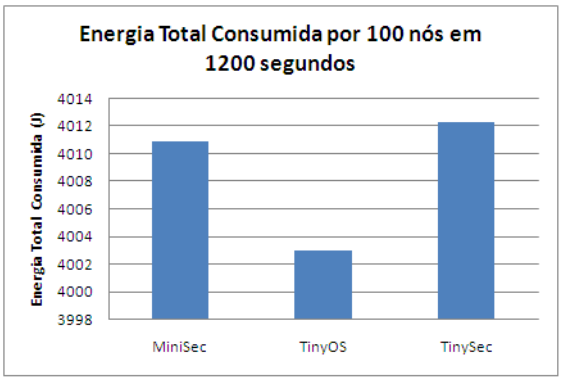
Na simulação anterior, ou seja, sem perda de pacotes, quanto maior o número de nós, maior a chance de se economizar energia com o uso do MiniSec-U, pois geralmente poucos nós apresentam um consumo elevado. Isso não acontece em um cenário com perda de pacotes, como mostrado na figura 6.12. Neste caso, o MiniSec-U foi o que mais consumiu energia em uma rede com 100 nós, apesar de ter sido levemente mais econômico em configurações com menor número de nós. Como



(a)

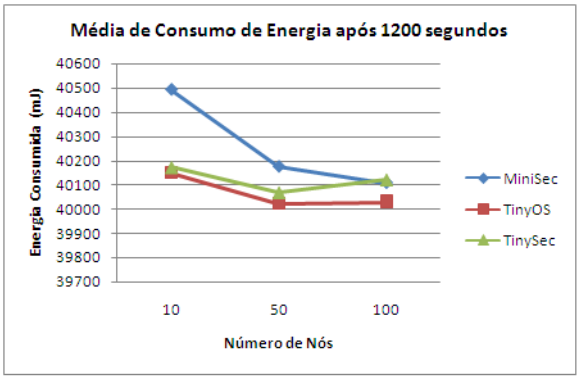


(b)

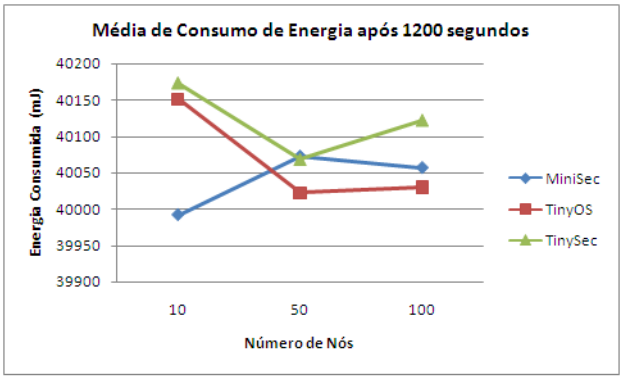


(c)

Figura 6.8: PowerTOSSIM: Consumo Total de Energia em uma RSSF com 10 (a), 50 (b) e 100 (c) Nós após 1200 segundos.



(a)



(b)

Figura 6.9: PowerTOSSIM: Média do Consumo de Energia dos Dispositivos em RSSFs com 10, 50 e 100 Nós após 1200 segundos com (a) e sem (b) Picos de Consumo de cada Protocolo.

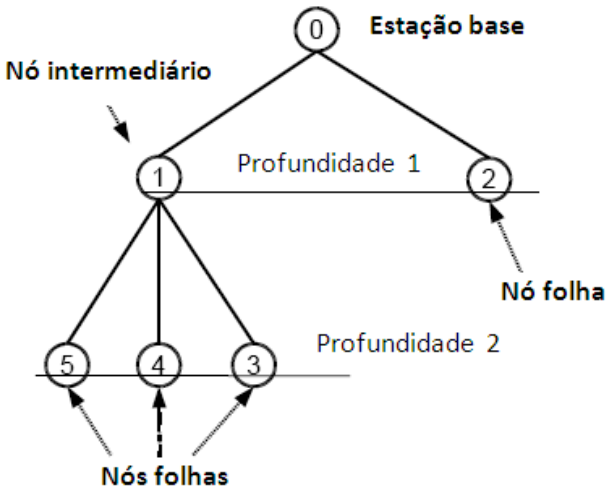


Figura 6.10: Exemplo de Árvore de Roteamento para uma Rede com Seis Nós Sensores.

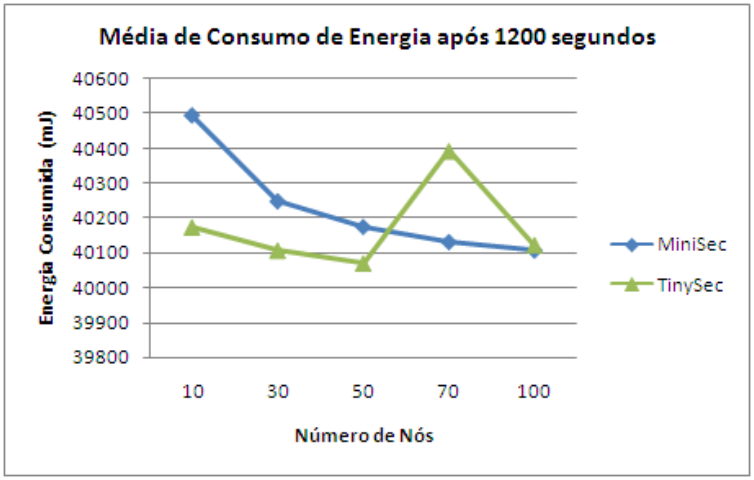


Figura 6.11: PowerTOSSIM: Média do Consumo de Energia dos Dispositivos em RSSFs com 10, 30, 50, 70 e 100 Nós após 1200 segundos.

as perdas dificultam o recebimento dos pacotes, os protocolos se comportam de forma semelhante, ou seja, poucos milijoules de diferença em consumo, já que o envio de mensagens cifradas é menos complexo que a decifragem. De qualquer forma, a expectativa é que o MiniSec-U apresente maior consumo de energia em uma rede com perdas, pois após o recebimento de mensagens (e durante a decifragem) haveria problemas, como a questão da falta de sincronia de um contador compartilhado entre um remetente e um destinatário.

Uma dificuldade com este tipo de simulação no TOSSIM é que o alcance do rádio está limitado em 15 metros, um valor pequeno se considerarmos poucos dispositivos sensores distribuídos uniformemente em uma região geográfica. Tentativas de se alterar o código do simulador em busca de maiores valores não resultaram em sucesso, pois o sistema apresentou o mesmo comportamento

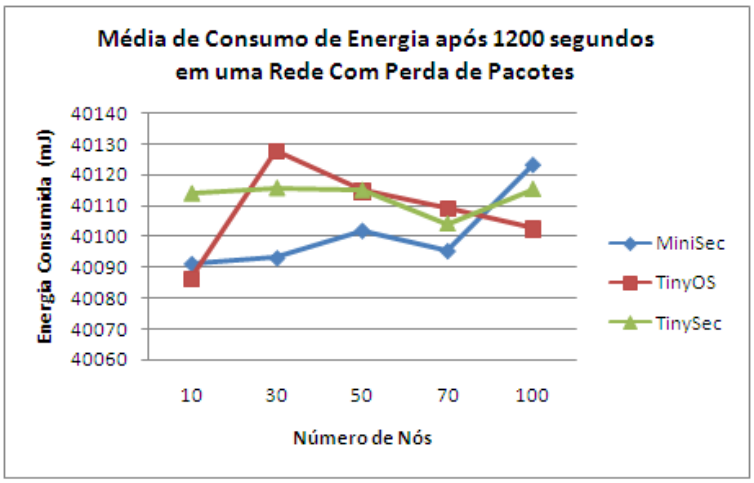


Figura 6.12: PowerTOSSIM: Média do Consumo de Energia dos Dispositivos em RSSFs com 10, 30, 50, 70 e 100 Nós após 1200 segundos. As Simulações Consideram Possibilidades de Perda de Pacotes.

quando foram feitas novas simulações[4]. É comum ver trabalhos que omitem o alcance do rádio, provavelmente devido a essa dificuldade. Isso acontece por exemplo em [33], onde também há uma forma diferente de construção da árvore de roteamento. Desta forma, essa simulação com perda de pacotes geralmente diminui o consumo de energia de todos os protocolos porque com altas taxas de perda não há recebimento dos pacotes enviados.

Espaço de Memória

Em relação ao espaço utilizado por cada protocolo, a figura 6.13 apresenta os valores de memória RAM e ROM que devem ser alocados em um sensor da plataforma mica2. Os valores consideram também o espaço utilizado pela aplicação.

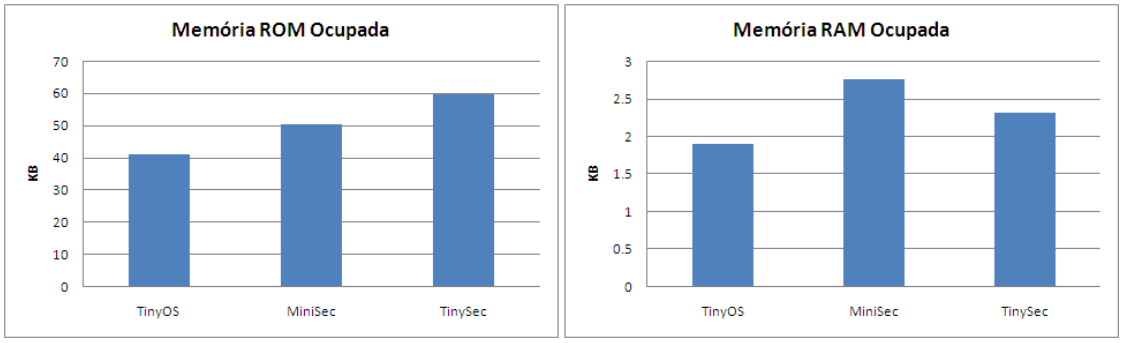


Figura 6.13: Espaço de Memória ROM e RAM, em KBytes, ocupados pela aplicação compilada com cada protocolo de segurança (plataforma Mica2)

[4]A dificuldade de se alterar o alcance do rádio nesta versão do simulador também é relatada na lista de discussão oficial do sistema, localizada no endereço `http://mail.millennium.berkeley.edu/pipermail/tinyos-help`. A nova versão do simulador, agora já em versão estável, provavelmente resolve esse problema.

Capítulo 7

Análise Comparativa dos Protocolos μTESLA e MiniSec-B

Em capítulos anteriores, consideramos que o envio de mensagens dos nós de uma RSSFs para a estação base é essencial. Isto geralmente acontece porque dispositivos sensores estão distribuídos em uma região geográfica para obtenção de dados, como por exemplo alterações de temperatura e detecção de movimento, cabendo à estação base receber e armazenar esses dados para posterior processamento. Esse tipo de comunicação, em que um nó sensor envia mensagens diretamente para a base, caracteriza o modelo *unicast* e por isso a importânca dos protocolos de segurança projetados especialmente para comunicação *unicast*.

Este capítulo compara as propriedades de segurança dos protocolos μTESLA e MiniSec-B, que foram projetados para um tipo diferente de comunicação. No modelo *broadcast*, uma mensagem possui vários destinatários e normalmente tem como único remetente a estação base. Exemplos de aplicações neste modelo de comunicação são o envio de comandos administrativos e distribuição de chaves criptográficas, sempre da estação base para os nós sensores. Pode haver algumas situações em que um nó sensor é o remetente de uma mensagem *broadcast*, como por exemplo a descoberta de nós vizinhos para construção da árvore de roteamento[1], mas consideramos, por propósito de comparação, que este modelo de comunicação tem como único remetente a estação base. Além disso, o objetivo principal dos protocolos de comunicação *broadcast* é prover autenticação e integridade, já que mensagens oriundas da estação base geralmente não são confidenciais.

7.1 Metodologia

A comparação teórica dos protocolos será feita através das seguintes propriedades:

- Confidencialidade
- Autenticidade e Integridade de Mensagem
- *Freshness*
- Resistência a comprometimento de nós

[1] A construção da árvore de roteamento não é tarefa de um protocolo de segurança. Protocolos de roteamento realizam essa atividade.

- Autenticação Imediata
- Baixo sobrecusto computacional e de comunicação
- Resistência a perda de pacotes da rede
- Escalabilidade

Em [52] há o desafio de se projetar um protocolo de comunicação *broadcast* que oferece todas essas propriedades, já que os protocolos projetados até então sempre desconsideram pelo menos uma. O simples fato de não oferecer uma propriedade pode viabilizar o uso de alguns mecanismos criptográficos. Por exemplo, na seção 2.4.1 relatamos que assinaturas digitais representam alto sobrecusto computacional, se comparados com MACs. Assim, se esta propriedade for prescindível, é possível que se adote o uso de assinaturas digitais.

Uma questão que surge naturalmente é se faz sentido uma propriedade ser prescindível. Por que um protocolo seria interessante mesmo sem oferecer uma ou outra propriedade? Na realidade cada aplicação de uma RSSF pode apresentar uma particularidade. Em alguns casos, pode não haver necessidade de se ter alto desempenho computacional devido à baixa quantidade de dados que trafegam na rede, ou ainda, autenticação imediata pode não ser necessário em aplicações que não são críticas em relação ao tempo. Outras situações podem tornar uma propriedade desnecessária para alguma aplicação, apesar da autenticação e integridade de mensagens ser uma propriedade importante na maioria das situações.

7.2 Confidencialidade

A propriedade de confidencialidade não é oferecida pelo μTESLA, mas um protocolo da mesma família[2], o SNEP [66], sugere o uso do modo de operação CTR (veja seção 3.3.4) com uma cifra de bloco segura. Em [9], há uma interessante análise deste modo de operação, destacando seu excelente nível de segurança com o uso de contadores como um VI sem repetição, ou seja, como um nonce. Há também uma versão do modo CTR que não utiliza um algoritmo determinístico. Esta variante probabilística também é IND-CPA segura, apesar de não possuir maior vantagem computacional contra ataques de adversários. Infelizmente, sem uma especificação do protocolo, pode ser necessário reutilizar o contador em algumas situações, o que significa repetir o VI. A existência do nonce é uma premissa na análise de segurança do modo CTR, pois, em caso contrário, este mecanismo apresenta as mesmas deficiências que as cifras de fluxo (veja também a seção 4.1).

O uso do μTESLA com suporte à propriedade de confidencialidade significa apenas utilizar uma recomendação do protocolo SNEP, o qual não foi projetado para o modelo de comunicação *broadcast* [66]. Já o protocolo MiniSec-B, utiliza o OCB-Skipjack (veja seções 3.3.4 e 3.3.5) para cifrar as mensagens que trafegam na rede e portanto já é especificado com um suporte à confidencialidade.

Como já foi discutido anteriormente, consideramos que a confidencialidade não é uma propriedade essencial para o modelo de comunicação *broadcast*, mas se desejarmos comparar a se-

[2] O μTESLA é o único dos protocolos da família SPINS (Security Protocol for Sensor Networks) que foi projetado especificamente para comunicação *broadcast*.

gurança do modo CTR com o modo OCB, teremos resultados próximos àqueles apresentados no capítulo anterior, ou seja, os modos são IND-CPA seguros e oferecem segurança semântica sob ataques adaptativos de mensagem (SEM-CPA).

7.3 Autenticidade e Integridade de Mensagem

No contexto da autenticidade e integridade de mensagens, o protocolo μTESLA utiliza o CBC-MAC, e o MiniSec-B, o OCB. No capítulo anterior, comparamos justamente os mecanismos CBC-MAC e OCB, considerando o uso de uma cifra de bloco segura[3]. Veja a seção 6.4 para uma análise detalhada dos mecanismos que oferecem esta propriedade.

7.4 Freshness

Os protocolos de segurança projetados para suportar o modelo de comunicação *broadcast* normalmente têm como principal objetivo de projeto a autenticidade de mensagens e a proteção contra ataques do tipo replay, ou seja, o fornecimento da propriedade de *Freshness*. Isto geralmente é feito através de um algoritmo criptográfico que "observa" a ordem parcial de mensagens recebidas por um nó da rede. Uma mensagem reenviada integralmente seria descartada por não ser um novo elemento de uma sequência de mensagens. As principais técnicas de registro parcial da ordem de mensagens envolvem o uso de contadores e a sincronia de tempo entre os nós sensores, o que requer uma estrutura de armazenamento localizada no nó receptor. De fato, um desafio em aberto para o modelo de comunicação *broadcast* é a criação de um protocolo que forneça a propriedade de *Freshness* sem exigir qualquer estrutura adicional. Neste contexto, os protocolos estudados neste capítulo apresentam as seguintes estruturas para fornecer a propriedade de *Freshness*:

- O MiniSec-B requer um contador para cada nó remetente, e para cada nó receptor de mensagens, apenas dois filtros de Bloom. Deve haver uma sincronia de tempo entre todos os participantes da rede.
- O μTESLA requer que um nó receptor de mensagens tenha uma cadeia de chaves criptográficas exclusiva para cada nó remetente. Uma sincronia de tempo é necessária entre todos os participantes da rede.

Observando o caso do MiniSec-B, temos uma estrutura de armazenamento constante em relação à quantidade de nós remetentes da rede (dois filtros de Bloom). Isto não acontece com o μTESLA, que aumenta a quantidade de armazenamento de acordo com o número de nós remetentes. De qualquer forma, os dois protocolos necessitam de uma estrutura adicional para fornecer proteção contra ataques do tipo replay. A diferença das estruturas que permitem esta propriedade está principalmente na questão da escalabilidade (veja seção 7.9).

É possível que o MiniSec-B seja utilizado sem o uso dos filtros de Bloom. Porém, como descrito na seção 5.4.3, há uma janela de vulnerabilidade, ou seja, um período de tempo em que a proteção

[3] A cifra de bloco normalmente é especificada pelo protocolo, porém um usuário interessado pode adotar outras opções. Além disso, os protocolos podem fornecer várias opções de implementação, apesar de haver apenas uma especificação, e delegar ao usuário a customização da cifra de bloco.

contra ataques do tipo replay não é fornecida. Resta portanto o uso dos filtros de Bloom para uma proteção completa. Mas há um inconveniente com o uso dos filtros: pode haver falso-positivos. Um filtro de Bloom armazena informações sobre mensagens recebidas. Uma consulta a um filtro é feita para saber se uma mensagem já foi recebida em algum momento anterior, sendo que o algoritmo de consulta devolve *Sim*, se a mensagem já foi recebida, e *Não*, caso contrário. O falso-positivo acontece justamente no momento em que o algoritmo de consulta devolve *Sim* para uma mensagem que na realidade ainda **não** foi recebida, descartando uma informação que deveria ser considerada como autêntica. Um motivo de conforto é que não há possibilidade de falso-negativo, ou seja, um ataque do tipo replay não teria sucesso.

Uma importante questão relacionada com o protocolo MiniSec-B é se a quantidade de falso-positivos é grande. Em [16], onde a estrutura dos filtros de Bloom foi inicialmente formalizada, há uma análise da probabilidade em se obter falso-positivos. Seja m o tamanho do filtro em bits e k o número de funções *hash*. Após inserir n elementos no filtro, temos que a probabilidade de se ter falso-positivos é a seguinte:

$$\left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k}$$

Desta fórmula é imediato que quanto menor o número de elementos inseridos no filtro e maior o tamanho do filtro, menor a probabilidade de se ter falso-positivos. Consideramos o uso de pelo menos uma função *hash*. Por exemplo, com um filtro de 900 bits, 8 funções *hash* e 92 elementos inseridos, a probabilidade de ocorrer falso-positivo é de aproximadamente 1%. Observe que o filtro ocupa pouco espaço (menos de 120 bytes).

7.5 Resistência a Comprometimento de Nós

Os protocolos MiniSec-B e μTESLA permitem o uso de uma chave exclusiva para cada nó da rede. Esta é uma forma de evitar que o comprometimento de apenas um nó implique no comprometimento de todos os dispositivos da rede. Apesar disso, esses protocolos não especificam uma forma de tratar o problema do comprometimento de nós, já que se a chave principal de um nó for obtida então pelo menos um nó está comprometido por todo o período de funcionamento da rede. Mesmo para o protocolo μTESLA isso é verdade, já que uma cadeia de chaves possui toda a segurança baseada na chave principal, compartilhada inicialmente com uma estação base.

Como foi relatado na seção 6.6, um mecanismo de gerenciamento das chaves criptográficas é o ingrediente principal da proteção contra ataques físicos a dispositivos da rede. Isso porque a obtenção de chaves é uma falha de segurança apenas durante o período em que a chave obtida está em uso.

7.6 Autenticação Imediata

A propriedade de autenticação imediata normalmente tem forte influência da propriedade de *Freshness*. Esta última requer que um nó descarte pacotes replicados, ou seja, pacotes interceptados e reenviados por um nó mal intencionado. Oferecer a propriedade de *Freshness* no modelo *broadcast*

implica em armazenar os pacotes recebidos ou informações sobre eles, cabendo a comparação do que foi armazenado com o que foi recentemente recebido pela rede. Como armazenar grande quantidade de pacotes ainda é algo inviável em dispositivos sensores, os protocolos MiniSec-B e μTESLA utilizam mecanismos para reter informações dos pacotes que já foram recebidos. O μTESLA possui autenticação com atraso, pois armazena pacotes recebidos até que chaves secundárias também sejam recebidas, permitindo a verificação da autenticidade dos pacotes armazenados com as novas chaves. O MiniSec-B estabelece que um nó remetente deve verificar a autenticidade de uma mensagem assim que esta é recebida. Não há necessidade de armazenar mensagens para qualquer processamento posterior. Esta facilidade é viabilizada pelo uso de dois filtros de Bloom, os quais também fornecem a propriedade de *Freshness*.

Apesar da desvantagem caracterizada por não autenticar mensagens de forma imediata, o protocolo μTESLA não descarta pacotes que são realmente autênticos. Já o MiniSec-B, pode descartar mensagens que deveriam ser consideradas válidas, como observado na seção 7.4. Isso é o que chamamos de falso-positivo, sendo que esse tipo de ocorrência aumenta proporcionalmente à quantidade de pacotes recebidos, já que eles são armazenados parcialmente nos filtros de Bloom.

Uma aplicação crítica em relação ao tempo, ou seja, que necessita de autenticação imediata, deve evitar o uso do μTESLA. Já uma aplicação que necessita de alta confiabilidade na troca de mensagens deve evitar o protocolo MiniSec-B ou ainda utilizar uma implementação que evite a ocorrência de falso-positivo (veja seção 7.4).

7.7 Baixo Sobrecusto Computacional e de Comunicação

O protocolo μTESLA pode apresentar uma alta taxa de sobrecusto computacional em caso de perda de pacotes da rede. Isto acontece porque há uma cadeia de chaves, a qual é inicialmente calculada por um remetente e recalculada por um destinatário. Se algumas chaves enviadas através da rede não são recebidas devido a perda de pacotes, então um remetente deve aguardar até que uma próxima chave seja recebida com sucesso. Após isso, uma série de funções devem ser aplicadas ao valor da última chave recebida para que se possa finalmente efetuar a verificação. Além da perda de pacotes, outro fator pode influenciar no processamento das chaves. O envio de chaves inválidas ou falsas por um adversário pode elevar o número de cálculos de um nó remetente, criando grande sobrecusto computacional para simplesmente tentar autenticar uma chave que nunca será considerada válida.

Apesar de não possuir sobrecustos com cadeias de chaves, o protocolo MiniSec-B requer cálculos relacionados com a cifragem de dados e processamento dos filtros de Bloom. Como o MiniSec-B apresenta melhor desempenho que o protocolo TinySec [51], temos que tais cálculos não geram sobrecustos computacionais significativos se comparados com aqueles gerados pelo protocolo μTESLA, o qual realiza cálculos de funções MAC semelhantes ao TinySec.

Em relação ao sobrecusto de comunicação por pacote, temos que o μTESLA, em sua versão original, requer apenas um espaço adicional de 2 bytes para a tag de autenticação (valor MAC). Já o MiniSec-B requer 3 bytes adicionais, sendo 1 byte relacionado com o nonce (confidencialidade) e 2 bytes relacionados com a tag de autenticação no modo OCB. A tabela 7.1 apresenta os sobrecustos

de comunicação para o modelo original dos protocolos, entretanto é importante observar que o protocolo μTESLA não possui sobrecustos relacionados com a confidencialidade justamente por não oferecer esta propriedade em sua versão original.

	Sobrecusto de Autenticação	**Sobrecusto de Confidencialidade**	**Sobrecusto Total de Pacote**
MiniSec-B	2	1	3
μTESLA	2	0	2

Tabela 7.1: Aumento do tamanho de pacote, em bytes, dos protocolos de segurança μTESLA e MiniSec-B em relação à versão original de pacote do sistema TinyOS.

Pela tabela 7.1, o sobrecusto de autenticação por pacote dos protocolos estudados são iguais (2 bytes). Apesar disso, o funcionamento do protocolo μTESLA requer o tráfego de pacotes adicionais relacionados com chaves secundárias, isso devido ao que chamamos de cadeia de chaves. Como cada mensagem recebida por um nó requer pelo menos um pacote adicional para a verificação de autenticidade, temos que o sobrecusto real apresentado pelo μTESLA é muito maior que aquele relatado em [66] e descrito na tabela 7.1. Em vista disso e considerando que o consumo de energia é proporcional ao sobrecusto de pacote, podemos concluir que o protocolo MiniSec-B, mesmo com maior sobrecusto total por pacote, tem grande potencial de economizar mais energia que o μTESLA.

7.8 Resistência a perda de pacotes da rede

O protocolo μTESLA não apresenta impactos de segurança em uma RSSFs com alta taxa de perda de pacotes. Se considerarmos que há uma chave simétrica principal estabelecida entre a estação base e um nó arbitrário, temos que chaves secundárias não recebidas devido a perda de pacotes apenas poderão atrasar o momento da autenticação. Desta forma, um nó pode aguardar até que uma nova chave secundária seja recebida e aplicar sucessivamente uma função até que o cálculo do valor MAC seja obtido, o que gera impactos apenas para o sobrecusto computacional e de comunicação, como descrito na seção 7.7. Portanto estabelecer uma cadeia de chaves é interessante para a propriedade de resistência a perda de pacotes da rede.

Em relação ao protocolo MiniSec-B, não há uso de um contador compartilhado como na versão *unicast* (MiniSec-U). Assim, um contador é enviado integralmente através da rede, sendo incrementado apenas pelo próprio remetente da mensagem. As seguintes propriedades viabilizam essa implementação:

1. O tamanho de um contador é relativamente pequeno, já que a cada intervalo de tempo pré-definido é possível que se reinicie o valor do contador utilizado.

2. Um contador é enviado (integralmente) através da rede sobrescrevendo espaços vagos do pacote (veja figura 5.4), o que aumenta o espaço disponível para o contador.

Normalmente há espaço suficiente para armazenamento completo do contador, o que explica a experiência, relatada em [51], de que o protocolo MiniSec-B não apresenta aumento significativo

do consumo de energia devido a uma alta taxa de perda de pacotes da rede, além de não apresentar problemas de confiabilidade do protocolo. É interessante observar que se o contador fosse compartilhado e houvesse perda de pacotes da rede teríamos grande impacto no funcionamento do protocolo. Isto porque poderia haver perda de informações, já que o contador é parte do nonce de cifragem e uma perda de sincronia poderia impedir a decifragem.

7.9 Escalabilidade

No modelo de comunicação *broadcast*, é comum haver restrição ao crescimento de uma RSSFs. A inclusão de novos dispositivos sensores pode implicar no aumento do número de espaço em disco dos participantes da rede. Por exemplo, o protocolo μTESLA requer que um nó receptor de mensagens tenha uma cadeia de chaves criptográficas exclusiva para cada nó remetente. Portanto, se este protocolo for utilizado para que muitos nós sensores enviem mensagens *broadcast*, então cada dispositivo receptor deve possuir diversas cadeias de chaves, o que é um empecilho para o crescimento da rede. Isso poderia também gerar um grande sobrecusto computacional e de comunicação. É por isso que o protocolo μTESLA retringe o número de nós remetentes da rede. Em contrapartida, é comum que apenas a estação base seja o único remetente *broadcast* de participação efetiva, restando aos demais participantes da rede enviar mensagens *broadcast* de forma esporádica[4].

O protocolo MiniSec-B possui uma estrutura inovadora para fornecer a propriedade de escalabilidade no modelo de comunicação *broadcast*. Há o uso de apenas dois filtros de Bloom por nó, independente do número de remetentes da rede. Desta forma, é possível que se tenha muitos participantes ativos na rede sem grande impacto na estrutura de armazenamento de um dispositivo.

[4]Um dispositivo sensor poderia por exemplo enviar mensagens no modelo *broadcast* para descobrir nós vizinhos.

Capítulo 8

Conclusões

Neste trabalho comparamos dois pares de protocolos de segurança projetados especificamente para RSSFs. A principal parte do trabalho trata dos protocolos *unicast* MiniSec-U e TinySec, já que esta forma de comunicação é a mais utilizada em RSSFs. Os protocolos MiniSec-B e μTESLA foram estudados dentro do modelo de comunicação *broadcast*. Foi possível entender que há grande dificuldade em se desenvolver um protocolo que apresente todas as propriedades desejadas em RSSFs. Neste contexto, fornecer alto nível de segurança e baixo consumo de energia é um dos principais desafios.

A preocupação em fornecer as propriedades estudadas é importante, pois existem ataques em sensores que podem exercer influência no comportamento de uma aplicação. Neste sentido, algumas recomendações de segurança foram feitas, como o estabelecimento de diferentes chaves criptográficas para os nós sensores e a frequência ideal de troca destas chaves. O estudo dos mecanismos utilizados pelos protocolos permitiu expor as vantagens e limitações existentes. A seguir, repassamos sobre os principais resultados deste trabalho em relação a cada modelo de comunicação.

8.1 Comparação dos Protocolos *Unicast* MiniSec-U e TinySec

A comparação dos protocolos de comunicação *unicast* mostrou que o TinySec possui vantagem quando a rede em questão tem uma alta taxa de perda de pacotes. Por outro lado, o MiniSec-U oferece uma propriedade omitida pelo TinySec: proteção contra ataques do tipo replay (propriedade *Freshness*).

Os dois protocolos também possuem características semelhantes em relação a algumas propriedades. Ambos não oferecem resistência a comprometimento de dispositivos da rede. Além disso, o Minisec-U e o TinySec não apresentam restrições ao crescimento de nós da rede em larga escala e possuem mecanismos demonstrados seguros em relação às propriedades de integridade, autenticidade e confidencialidade de mensagens.

As simulações de consumo de energia auxiliaram no estudo dos sobrecustos gerados pelos protocolos. Teoricamente, o TinySec apresenta grande potencial de consumo de energia por possuir maior tamanho de pacote, mas os experimentos mostraram que, em uma rede com poucos nós e livre de perdas, o MiniSec-U é que apresenta maior média de consumo por dispositivo sensor. Quando a rede possui pelo menos 50 nós, esta situação pode se inverter, pois os sobrecustos do MiniSec-U se tornam menos significativos.

Em simulações com perda de pacotes, vimos que houve pouca diferença entre os consumos de energia dos protocolos. Isto aconteceu porque o ambiente de perdas que foi simulado não permitiu grande quantidade de recebimento de pacotes. Portanto houve, apenas neste novo cenário, uma medição baseada no envio de mensagens. Apesar disso, a prática acabou se alinhando com a teoria, já que o TinySec se mostrou mais eficiente em um ambiente com 100 nós sensores.

8.2 Comparação dos Protocolos *Broadcast* MiniSec-B e μTESLA

Os protocolos *broadcast* estudados apresentam grandes diferenças em relação a duas propriedades: autenticação imediata e escalabilidade. O μTESLA realiza verificações de autenticidade com atrasos relevantes e não permite o crescimento de nós da rede em grande escala. Essas desvantagens não são apresentadas pelo protocolo MiniSec-B.

A proteção contra ataques do tipo replay é fornecida pelos dois protocolos, apesar do MiniSec-B possuir pequena desvantagem, pois pode descartar dados que deveriam ser considerados como legítimos.

Um detalhe omitido é que o MiniSec-B também poderia ser utilizado em comunicações *unicast*, com grande potencial em ser mais eficiente em consumo de energia que os dois protocolos *unicast* estudados. Em contrapartida, seria necessário sincronia de tempo entre os nós da rede para manter a estabilidade do MiniSec-B.

8.3 Contribuições

Esta dissertação gerou a publicação do seguinte artigo (trabalho completo):

- SANTOS, M. A. S. ; TERADA, ROUTO . **Análise Comparativa de Protocolos de Segurança para Redes de Sensores Sem Fio**. In: *Simpósio de Informática da Região Centro do RS (SIRC/RS), 2009, Santa Maria.* Anais do Simpósio de Informática da Região Centro do RS, 2009.

O seguinte artigo também foi produzido para futura submissão: TERADA, ROUTO ; SANTOS, M. A. S. . **A Comparative Analysis of Security Protocols in Wireless Sensor Networks**.

8.4 Trabalhos Futuros

Assim como os protocolos *unicast* estudados, o MiniSec-B e o μTESLA não possuem proteção específica para a questão do comprometimento de nós. Esta proteção poderia ser fornecida através do uso de mecanismos de gerenciamento de chaves criptográficas. Outra oportunidade de melhoria está relacionada com a análise comparativa dos sobrecustos dos protocolos *broadcast*, já que o μTESLA, por exemplo, nunca foi totalmente implementado.

Este trabalho apresentou resultados de protocolos de segurança baseados em criptografia simétrica. Este modelo foi escolhido principalmente porque permite a construção de algoritmos eficientes em consumo de energia. Com a evolução dos estudos de mecanismos de criptografia de chave pública, é possível que se desenvolva novos tipos de protocolos que sejam atrativos em redes de sensores. Por isso, seria interessante um projeto de pesquisa que não só trate esta questão, mas também compare os modelos simétricos e assimétricos.

Em relacão ao número de simulações, trabalhos anteriores apresentam informações de consumo de energia com o uso de apenas um dispositivo, em geral através da medição do envio de apenas um pacote [39, 51]. Este trabalho utilizou simulações com até 100 nós sensores. Seria interessante uma abordagem com mais simulações e que observe resultados com diferentes parâmetros, apesar dos protocolos adotarem um padrão. Além disso, tempos maiores de simulação e testes com um cenário de perda de pacotes através da nova versão do simulador TOSSIM seriam interessantes.

Apêndice A

Esquema de Autenticação e Cifragem

O termo *Esquema de Autenticação e Cifragem* é utilizado para se referir a mecanismos de criptografia simétrica que têm como objetivo fornecer integridade, autenticidade e confidencialidade[1]. O modo de operação OCB-Skipjack, utilizado pelo protocolo MiniSec, é claramente um esquema de autenticação e cifragem, pois utiliza o mesmo algoritmo para cifragem de mensagens e cálculo de tags. O protocolo TinySec utiliza uma abordagem diferente. A cifragem de mensagens é feita através do algoritmo CBC-Skipjack e o cálculo de tags é feito através do CBC-MAC. Estes mecanismos, se analisados de forma independente, não são esquemas de cifragem e autenticação, pois não fornecem as duas propriedades. Apesar disso, o protocolo TinySec utiliza esses dois mecanismos em conjunto, pois um receptor deve primeiro decifrar uma mensagem antes de fazer o cálculo do valor MAC (tag) correspondente. Isto acontece porque este valor é calculado a partir do texto legível.

Em criptografia simétrica, esquemas de cifragem e autenticação geralmente consistem de três estratégias:

1. *Cifra-e-Autentica texto legível*: um remetente concatena o texto cifrado com a tag de autenticação calculada a partir do texto legível. A autenticação do lado de um destinatário é feita da seguinte forma: decifragem da mensagem para obtenção do texto legível e posteriormente verificação da tag.

2. *Autentica-depois-cifra*: um remetente concatena a tag com o texto legível e então cifra este resultado. Um receptor deve decifrar a mensagem antes de verificar a tag.

3. *Cifra-depois-autentica*: um remetente cifra o texto legível. O texto cifrado resultante é utilizado como entrada para o algoritmo de cálculo da tag, a qual é concatenada com o texto cifrado para envio através da rede. Um destinatário deve inicialmente verificar a tag para depois decifrar a mensagem.

O TinySec utiliza a primeira estratégia, enquanto o MiniSec tem propriedades da primeira e da última estratégias, já que o último bloco do texto cifrado também é parte da tag de autenticação. Em [12] há uma comparação entre as estratégias que traz a seguinte conclusão: o esquema *cifra-*

[1]Em geral, integridade e autenticidade são utilizados de forma intercambiável, pois, apesar de não serem equivalentes, apresentam alguns resultados em comum (veja seção 2.3.2)

depois-autentica é o mais indicado, pois é demonstrado seguro para um maior número de noções de segurança.

Apêndice B

Dados Experimentais

B.1 Consumo de Energia

Os resultados dos principais experimentos realizados são valores numéricos que correspondem ao consumo de energia (em miliJoules) de cada dispositivo sensor da rede. A saída que será apresentada foi gerada a partir dos seguintes passos:

1. Compilação da aplicação com o **target** pc (TOSSIM)

   ```
   make pc
   ```

2. Inicialização da variável DBG

   ```
   export DBG=power
   ```

3. Execução do arquivo compilado indicando número de nós, tempo de simulação e flag para utilizar eventos de consumo de energia

   ```
   ./build/pc/main.exe -t=1200 -p 10 > myapp.trace
   ```

   , em que myapp.trace é o arquivo de saída com os eventos simulados

4. Pós-processamento dos eventos com o uso do modelo de energia do dispositivo mica2 (PowerTOSSIM)

   ```
   postprocess.py --sb=0 --em mica2_energy_model.txt myapp.trace
   ```

Após o último passo, obtivemos a seguinte saída:

```
Mote 0, cpu total: 14824.717056
Mote 0, radio total: 25168.232400
Mote 0, adc total: 0.000000
Mote 0, leds total: 0.000000
Mote 0, sensor total: 0.000000
Mote 0, eeprom total: 0.000000
Mote 0, cpu_cycle total: 0.000000
```

```
Mote 0, Total energy: 39992.949456

Mote 1, cpu total: 14824.717056
Mote 1, radio total: 25128.205191
Mote 1, adc total: 0.000000
Mote 1, leds total: 0.000000
Mote 1, sensor total: 0.000000
Mote 1, eeprom total: 0.000000
Mote 1, cpu_cycle total: 0.000000
Mote 1, Total energy: 39952.922247

Mote 2, cpu total: 14824.717056
Mote 2, radio total: 25141.971023
Mote 2, adc total: 0.000000
Mote 2, leds total: 0.000000
Mote 2, sensor total: 0.000000
Mote 2, eeprom total: 0.000000
Mote 2, cpu_cycle total: 0.000000
Mote 2, Total energy: 39966.688079

Mote 3, cpu total: 14824.717056
Mote 3, radio total: 25154.551762
Mote 3, adc total: 0.000000
Mote 3, leds total: 0.000000
Mote 3, sensor total: 0.000000
Mote 3, eeprom total: 0.000000
Mote 3, cpu_cycle total: 0.000000
Mote 3, Total energy: 39979.268818

Mote 4, cpu total: 14824.717056
Mote 4, radio total: 25209.795273
Mote 4, adc total: 0.000000
Mote 4, leds total: 0.000000
Mote 4, sensor total: 0.000000
Mote 4, eeprom total: 0.000000
Mote 4, cpu_cycle total: 0.000000
Mote 4, Total energy: 40034.512329

Mote 5, cpu total: 14824.717056
```

```
Mote 5, radio total: 25105.662317
Mote 5, adc total: 0.000000
Mote 5, leds total: 0.000000
Mote 5, sensor total: 0.000000
Mote 5, eeprom total: 0.000000
Mote 5, cpu_cycle total: 0.000000
Mote 5, Total energy: 39930.379373

Mote 6, cpu total: 14824.717056
Mote 6, radio total: 25139.506944
Mote 6, adc total: 0.000000
Mote 6, leds total: 0.000000
Mote 6, sensor total: 0.000000
Mote 6, eeprom total: 0.000000
Mote 6, cpu_cycle total: 0.000000
Mote 6, Total energy: 39964.224000

Mote 7, cpu total: 14824.717056
Mote 7, radio total: 25221.051418
Mote 7, adc total: 0.000000
Mote 7, leds total: 0.000000
Mote 7, sensor total: 0.000000
Mote 7, eeprom total: 0.000000
Mote 7, cpu_cycle total: 0.000000
Mote 7, Total energy: 40045.768474

Mote 8, cpu total: 14824.717056
Mote 8, radio total: 25236.800898
Mote 8, adc total: 0.000000
Mote 8, leds total: 0.000000
Mote 8, sensor total: 0.000000
Mote 8, eeprom total: 0.000000
Mote 8, cpu_cycle total: 0.000000
Mote 8, Total energy: 40061.517954

Mote 9, cpu total: 14768.930858
Mote 9, radio total: 30239.514192
Mote 9, adc total: 0.000000
Mote 9, leds total: 0.000000
```

```
Mote 9, sensor total: 0.000000
Mote 9, eeprom total: 0.000000
Mote 9, cpu_cycle total: 0.000000
Mote 9, Total energy: 45008.445050
```

Observamos o consumo de energia devido às atividades da cpu e do rádio. Neste experimento com 10 nós foi utilizado o protocolo MiniSec-U.

A execução correta do experimento depende da instalação e configuração dos software (inclusive componentes internos), além da criação da aplicação.

Os dados experimentais completos podem ser obtidos em http://www.ime.usp.br/~mateusss.

B.2 Espaço Utilizado

Para se obter o espaço utilizado por uma aplicação deve-se compilar o código com o **target** do sensor, neste caso o mica2 [4]. O seguinte passo realiza esta tarefa:

```
make mica2
```

As saídas serão diferentes para cada protocolo, como a seguir.

TinySec

```
make mica2

compiled Surge to build/mica2/main.exe
           61188 bytes in ROM
            2359 bytes in RAM
avr-objcopy --output-target=srec build/mica2/main.exe build/mica2/main.srec
avr-objcopy --output-target=ihex build/mica2/main.exe build/mica2/main.ihex
    writing TOS image
```

MiniSec

```
make mica2

compiled Surge to build/mica2/main.exe
           51574 bytes in ROM
            2815 bytes in RAM
avr-objcopy --output-target=srec build/mica2/main.exe build/mica2/main.srec
avr-objcopy --output-target=ihex build/mica2/main.exe build/mica2/main.ihex
    writing TOS image
```

TinyOS (sem segurança)

```
make mica2
```

```
compiled Surge to build/mica2/main.exe
            42080 bytes in ROM
             1944 bytes in RAM
avr-objcopy --output-target=srec build/mica2/main.exe build/mica2/main.srec
avr-objcopy --output-target=ihex build/mica2/main.exe build/mica2/main.ihex
    writing TOS image
```

Apêndice C

Listagem de Códigos

Os códigos que serão listados dependem de diversos componentes que são nativos do sistema TinyOS. Alguns exemplos são os componentes de roteamento MultiHopEngineM, MultiHopLEPSM e MultiHopRouter, responsáveis pela lógica de roteamento de pacotes no modelo multi-saltos, criação e atualização da árvore de roteamento, e conexão dos componentes, respectivamente, além dos componentes básicos para envio e recebimento de mensagem AMStandard.nc e GenericComm. Todos os códigos devem ser compilados especificamente com as bibliotecas do TOSSIM (**target** pc) para que seja possível gerar o executável da aplicação. Para uma listagem completa consulte http://www.ime.usp.br/~mateusss e http://www.tinyos.net.

C.1 Funções da Aplicação Surge

Esta aplicação envia mensagens para a estação base através do modelo de comunicação multi-saltos.

***Task* de Envio de Mensagens**

```
task void SendData() {
    SurgeMsg *pReading;
    uint16_t Len;
    dbg(DBG_USR1, "SurgeM: Enviando leitura do sensor\n");

    if (pReading = (SurgeMsg *)call Send.getBuffer(&gMsgBuffer,&Len)) {
      pReading->type = SURGE_TYPE_SENSORREADING;
      pReading->parentaddr = call RouteControl.getParent();
      pReading->reading = gSensorData;

      if ((call Send.send(&gMsgBuffer,sizeof(SurgeMsg))) != SUCCESS)
        atomic gfSendBusy = FALSE;
    }

  }
```

***Event* Leitura do Sensor**

```
event result_t ADC.dataReady(uint16_t data) {

    dbg(DBG_USR1, "SurgeM: leitura ADC: 0x%x\n", data);  //apenas para debug
    atomic {
      if (!gfSendBusy) {
        gfSendBusy = TRUE;
        gSensorData = data;
        post SendData(); //executa tarefa para envio de dado pela rede
      }
    }
    return SUCCESS;
  }
```

***Event* Confirmação de Envio**

```
 event result_t Send.sendDone(TOS_MsgPtr pMsg, result_t success) {
    dbg(DBG_USR2, "SurgeM: mensagem enviada 0x%x\n", success);
    //call Leds.greenToggle();   //desativado por não precisar de Leds
    atomic gfSendBusy = FALSE;
    return SUCCESS;
  }
```

***Event* Novo Período de Tempo**

```
 event result_t Timer.fired() {
    dbg(DBG_USR1, "SurgeM: Inicio de novo ciclo\n");
    timer_ticks++;
    if (timer_ticks % TIMER_GETADC_COUNT == 0)
      call ADC.getData();

    return SUCCESS;
  }
```

C.2 Protocolos MiniSec e TinySec

Em relação ao TinySec, as adaptações foram feitas a partir da versão 1.1 do TinyOS [47]. Todos os componentes e interfaces de segurança já acompanham esta versão, apesar de ser necessário fazer alterações em flags de compilação e inicialização de variáveis como por exemplo a chave criptográfica.

O protocolo MiniSec foi adaptado da implementação para a plataforma Telosb [51], ou seja, portado para o sensor mica2 [4]. Alguns componentes não funcionam sem as devidas alterações, pois o MiniSec reescreve a camada de rede do sistema TinyOS, que inclui componentes como o AMStandard.nc. As flags de compilação também devem ser alteradas.

Referências

[1] Ian F. Akyildiz, Welljan Su, Yogesh Sankarasubramaniam, and Erdal Cayirci, *A survey on sensor networks*, Communications Magazine, IEEE **40** (2002), 102–114.

[2] TinyOS Alliance, *Open technology alliance*, Disponível em http://sing.stanford.edu/doc/tinyos-alliance.pdf.

[3] ZigBee Alliance, *Zigbee specification*, Tech. Report 053474r06, ZigBee Alliance, June 2005.

[4] _____, *Mica2 wireless measurement system*, Tech. Report Tech. Rep. 6020-0042-08, Crossbow Technology, 2008.

[5] ANSI and IEEE, *Information technology, telecommunications and information exchange between systems, local and metropolitan area networks, specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications*, 1999.

[6] Steve Babbage, Dario Catalano, Carlos Cid, Orr Dunkelman, Christian Gehrmann, Louis Granboulan, Tanja Lange, Arjen Lenstra, Phong Nguyen, Christof Paar, Jan Pelzl, Nigel Smart, and Michael Ward, *Ecrypt yearly report on algorithms and keysizes*, 2008, Disponível em http://www.ecrypt.eu.org/documents/D.SPA.28-1.1.pdf.

[7] Steve Babbage, Carlos Cid, Henri Gilbert, Thomas Johansson, Matthew Parker, Bart Preneel, Vincent Rijmen, Anne Canteaut, and Matthew Robshaw, *The estream portfolio*, 2008.

[8] Haowei Bai, Mohammed Atiquzzaman, and David Lilja, *Wireless sensor network for aircraft health monitoring*, BROADNETS '04: Proceedings of the First International Conference on Broadband Networks (Washington, DC, USA), IEEE Computer Society, 2004, pp. 748–750.

[9] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, *A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation*, In Proceedings of 38th Annual Symposium on Foundations of Computer Science (FOCS 97), 1997.

[10] Mihir Bellare and Shafi Goldwasser, *Lecture notes on cryptography*, 2008, Disponível em https://cseweb.ucsd.edu/ mihir/papers/gb.html.

[11] Mihir Bellare, Joe Kilian, and Phillip Rogaway, *The security of the cipher block chaining message authentication code*, Journal of Computer and System Sciences **61** (2000), no. 3, 362–399.

[12] Mihir Bellare and Chanathip Namprempre, *Authenticated encryption: Relations among notions and analysis of the generic composition paradigm*, Journal of Cryptology **21** (2008), no. 4, 469–491.

[13] Mihir Bellare and Phillip Rogaway, *Introduction to modern cryptography*, 2003, Disponível em http://cseweb.ucsd.edu/ mihir/cse207/classnotes.html.

[14] Eli Biham, Alex Biryukov, and Adi Shamir, *Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials*, Journal of Cryptology **18** (2005), no. 4, 291–311.

[15] Erik-Oliver Blas and Martina Zitterbart, *Towards acceptable public-key encryption in sensor networks*, ACM 2nd International Workshop on Ubiquitous Computing (Miami, USA), INSTICC Press, May 2005, ISBN 972-8865-24-4, pp. 88–93.

[16] Burton H. Bloom, *Space/time trade-offs in hash coding with allowable errors*, Commun. ACM **13** (1970), no. 7, 422–426.

[17] Tatiana Bokareva, Wen Hu, Salil Kanhere, Branko Ristic, Neil Gordon, Travis Bessell, Mark Rutten, and Sanjay Jha, *Wireless sensor networks for battlefield surveillance. proceedings of the land warfare conference*, 2006.

[18] Nikita Borisov, Ian Goldberg, and David Wagner, *Intercepting mobile communications: the insecurity of 802.11*, MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking (New York, NY, USA), ACM, 2001, pp. 180–189.

[19] Miguel Elias Campista and Otto Carlos Duarte, *Segurança em redes de sensores*, Disponível em http://www.gta.ufrj.br/seminarios/CPE825/, 2008.

[20] Shang-Ming Chang, Shiuhpyng Shieh, Warren W. Lin, and Chih-Ming Hsieh, *An efficient broadcast authentication scheme in wireless sensor networks*, ASIACCS '06: Proceedings of the 2006 ACM Symposium on Information, computer and communications security (New York, NY, USA), ACM, 2006, pp. 311–320.

[21] Joan Daemen and Vincent Rijmen, *Specification for the advanced encryption standard (AES)*, Federal Information Processing Standards Publication 197, 2001.

[22] Jing Deng, Carl Hartung, Richard Han, and Shivakant Mishra, *A practical study of transitory master key establishment forwireless sensor networks*, SECURECOMM '05: Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks (Washington, DC, USA), IEEE Computer Society, 2005, pp. 289–302.

[23] Whitfield Diffie and Martin E. Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory **IT-22** (1976), no. 6, 644–654.

[24] Morris Dworkin, *Nist special publication 800-38a. recommendation for block cipher modes of operation, methods and techniques*, 2001, Disponível em http://www.itl.nist.gov/publications/publications.cgi.

[25] Natalia Castro Fernandes e Otto Carlos Muniz Bandeira Duarte, *Autoconfiguração de endereços baseada em filtros de bloom para redes ad hoc*, XXVI Simpósio Brasileiro de Redes de Computadores - SBRC'08, 2008, Rio de Janeiro, RJ., 2008, pp. 273–286.

[26] Thomas Eisenbarth, Sandeep Kumar, Christof Paar, Axel Poschmann, and Leif Uhsadel, *A survey of lightweight-cryptography implementations*, IEEE Des. Test **24** (2007), no. 6, 522–533.

[27] Jeremy Elson, Lewis Girod, and Deborah Estrin, *Fine-grained network time synchronization using reference broadcasts*, SIGOPS Oper. Syst. Rev. **36** (2002), no. SI, 147–163.

[28] Amin F., Jahangir A. H., and Rasifard H., *Analysis of public-key cryptography for wireless sensor networks security*, Proceedings of World Academy of Science, Engineering and Technology, 2008.

[29] International Organization for Standards, *ISO/IEC 9797-1 data integrity mechanism using a cryptographic check function employing a block cipher algorithm*, 1999.

[30] David Gay, Philip Levis, Robert von Behren, Matt Welsh, Eric Brewer, and David Culler, *The nesc language: A holistic approach to networked embedded systems*, PLDI '03: Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation (New York, NY, USA), ACM, 2003, pp. 1–11.

[31] Shafi Goldwasser and S. Micali, *Probabilistic encryption*, Special issue of Journal of Computer and System Sciences **28** (1984), 270–299.

[32] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest, *A digital signature scheme secure against adaptive chosen-message attacks*, SIAM Journal on Computing **17** (1988), no. 2, 281–308.

[33] Germano Guimarães, Eduardo Souto, Judith Kelner, and Djamel Sadok, *Impacto da utilização de mecanismos de segurança em redes de sensores sem fio*, Proceedings of the XXIII Simpósio Brasileiro de Redes de Computadores, 2005.

[34] John Heidemann, Fabio Silva, Chalermek Intanagonwiwat, Ramesh Govindan, Deborah Estrin, and Deepak Ganesan, *Building efficient wireless sensor networks with low-level naming*, SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles (New York, NY, USA), ACM, 2001, pp. 146–159.

[35] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister, *System architecture directions for networked sensors*, ASPLOS-IX: Proceedings of the ninth international conference on Architectural support for programming languages and operating systems (New York, NY, USA), ACM, 2000, pp. 93–104.

[36] Zhihua Hu and Baochun Li, *Fundamental performance limits of wireless sensor networks*, in Ad Hoc and Sensor Networks, Yang Xiao and Yi Pan, Editors, Nova Science Publishers, 2005, pp. 81–101.

[37] Jakob Jonsson, *On the security of CTR + CBC-MAC*, SAC '02: Revised Papers from the 9th Annual International Workshop on Selected Areas in Cryptography (London, UK), Springer-Verlag, 2003, pp. 76–93.

[38] Marcos Antonio Simplício Junior, *Algoritmos criptográficos para redes de sensores*, 2008, Dissertação de Mestrado. Escola Politécnica da Universidade de São Paulo.

[39] Chris Karlof, Naveen Sastry, and David Wagner, *Tinysec: a link layer security architecture for wireless sensor networks*, SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems (New York, NY, USA), ACM, 2004, pp. 162–175.

[40] 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, 2003, pp. 113–127.

[41] Yee Wei Law, Jeroen Doumen, and Pieter Hartel, *Survey and benchmark of block ciphers for wireless sensor networks*, ACM Trans. Sen. Netw. **2** (2006), no. 1, 65–93.

[42] _____, *Survey and benchmark of block ciphers for wireless sensor networks*, ACM Trans. Sen. Netw. **2** (2006), no. 1, 65–93.

[43] Arjen K. Lenstra and Eric R. Verheul, *Selecting cryptographic key sizes*, PKC '00: Proceedings of the Third International Workshop on Practice and Theory in Public Key Cryptography (London, UK), Springer-Verlag, 2000, pp. 446–465.

[44] P. Levis, *TOSSIM: A simulator for tinyos networks*, Tech. report, University of California, Berkeley, 2003, Disponível em http://www.cs.berkeley.edu/ pal/pubs/nido.pdf.

[45] Philip Levis and David Culler, *Maté: a tiny virtual machine for sensor networks*, ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems (New York, NY, USA), ACM, 2002, pp. 85–95.

[46] Philip Levis, Nelson Lee, Matt Welsh, and David Culler, *TOSSIM: accurate and scalable simulation of entire tinyos applications*, SenSys '03: Proceedings of the 1st international conference on Embedded networked sensor systems (New York, NY, USA), ACM, 2003, pp. 126–137.

[47] Philip Levis, Sam Madden, Joseph Polastre, Robert Szewczyk, Alec Woo, David Gay, Jason Hill, Matt Welsh, Eric Brewer, and David Culler, *Tinyos: An operating system for sensor networks*, in Ambient Intelligence, Springer Verlag, 2004, pp. 115–148.

[48] Helger Lipmaa, Phillip Rogaway, and David Wagner, *Comments to nist concerning AES modes of operation: CTR-mode encryption*, 2000.

[49] An Liu and Peng Ning, *Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks*, IPSN '08: Proceedings of the 2008 International Conference on Information Processing in Sensor Networks (ipsn 2008) (Washington, DC, USA), IEEE Computer Society, 2008, pp. 245–256.

[50] Michael Luby and Charles Rackoff, *How to construct pseudorandom permutations from pseudorandom functions*, SIAM Journal on Computing **17** (1988), no. 2, 373–386.

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

[52] Mark Luk, Adrian Perrig, and Bram Whillock, *Seven cardinal properties of sensor network broadcast authentication*, SASN '06: Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks (New York, NY, USA), ACM, 2006, pp. 147–156.

[53] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong, *Tag: a tiny aggregation service for ad-hoc sensor networks*, SIGOPS Oper. Syst. Rev. **36** (2002), no. SI, 131–146.

[54] Alan Mainwaring, David Culler, Joseph Polastre, Robert Szewczyk, and John Anderson, *Wireless sensor networks for habitat monitoring*, WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications (New York, NY, USA), ACM, 2002, pp. 88–97.

[55] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, *Handbook of applied cryptography*, CRC Press, 2001.

[56] Ralph C. Merkle, *A digital signature based on a conventional encryption function*, CRYPTO '87: A Conference on the Theory and Applications of Cryptographic Techniques on Advances in Cryptology (London, UK), Springer-Verlag, 1988, pp. 369–378.

[57] _____, *A certified digital signature*, CRYPTO '89: Proceedings on Advances in cryptology (New York, NY, USA), Springer-Verlag New York, Inc., 1989, pp. 218–238.

[58] Silvio Micali, Charles Rackoff, and Bob Sloan, *The notion of security for probabilistic cryptosystems*, SIAM Journal on Computing **17** (1988), no. 2, 412–426.

[59] James Newsome, Carnegie Mellon, and Elaie Shi, *The sybil attack in sensor networks: analysis & defenses*, In Proceedings of IEEE International Conference on Information Processing in Sensor Networks (ISPN), ACM Press, 2004, pp. 259–268.

[60] NIST, *Skipjack and kea algorithms specifications*, 1998, Disponível em http://csrc.nist.gov/groups/ST/toolkit/documents/skipjack/skipjack.pdf.

[61] _____, *Proposal to extend CBC mode by ciphertext stealing*, 2007, Disponível em http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/ciphertext0proposal.pdf.

[62] U.S. Department of Commerce/National Bureau of Standards, *Federal information processing standards publication 113. computer data authentication*, 1994, Disponível em http://www.itl.nist.gov/fipspubs/fip113.htm.

[63] Leonardo B. Oliveira, Diego Aranha, Eduardo Morais, Felipe Daguano, Julio Lopez, and Ricardo Dahab, *Tinytate: Computing the tate pairing in resource-constrained nodes*, 6th IEEE International Symposium on Network Computing and Applications (Cambridge,MA), July 2007, pp. 318–323.

[64] Adrian Perrig, *The biba one-time signature and broadcast authentication protocol*, CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security (New York, NY, USA), ACM, 2001, pp. 28–37.

[65] Adrian Perrig, Ran Canetti, Dawn Song, and J. D. Tygar, *Efficient and secure source authentication for multicast*, In Network and Distributed System Security Symposium, NDSS 01, 2001, pp. 35–46.

[66] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar, *Spins: security protocols for sensor networks*, MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking (New York, NY, USA), ACM, 2001, pp. 189–199.

[67] Adrian Perrig, J. D. Tygar, Dawn Song, and Ran Canetti, *Efficient authentication and signing of multicast streams over lossy channels*, Proceedings of the 2000 IEEE Symposium on Security and Privacy (Washington, DC, USA), IEEE Computer Society, 2000, p. 56.

[68] Kui Ren, Wenjing Lou, and Yanchao Zhang, *Multi-user broadcast authentication in wireless sensor networks*, Sensor, Mesh and Ad Hoc Communications and Networks, 2007. SECON '07. 4th Annual IEEE Communications Society Conference on, 2007, pp. 223–232.

[69] Leonid Reyzin and Natan Reyzin, *Better than biba: Short one-time signatures with fast signing and verifying*, ACISP '02: Proceedings of the 7th Australian Conference on Information Security and Privacy (London, UK), Springer-Verlag, 2002, pp. 144–153.

[70] R. L. Rivest, A. Shamir, and L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Commun. ACM **21** (1978), no. 2, 120–126.

[71] Ronald L. Rivest, *The rc5 encryption algorithm*, In Proceedings of the Leuven Workshop on Fast Software Encryption, Springer-Verlag, 1995, pp. 86–96.

[72] M. J. B. Robshaw, *Stream ciphers*, Tech. Report TR-701, RSA Laboratories, July 1995.

[73] Phillip Rogaway, *OCB: Background*, Disponível em http://www.cs.ucdavis.edu/ rogaway/ocb/ocb-faq.htm.

[74] ———, *Nonce-based symmetric encryption*, Proc. FSE 2004, volume 3017 of LNCS, Springer, 2004, pp. 348–359.

[75] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz, *OCB: a block-cipher mode of operation for efficient authenticated encryption*, CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security (New York, NY, USA), ACM, 2001, pp. 196–205.

[76] Linnyer Beatrys Ruiz, Luiz Henrique A. Correia, Luiz Filipe Vieira, Daniel Macedo, Eduardo Nakamura, Carlos Figueiredo, Marcos Augusto Vieira, Eduardo Habib Mechelane, Daniel Camara, Antonio Loureiro, José Marcos Nogueira, and Diógenes da Silva Jr, *Arquiteturas para redes de sensores sem fio, sbrc*, 2004.

[77] Mohit Saxena, *Security in wireless sensor networks - a layer based classification*, Tech. Report Tech Report 2007-04, Center for Education and Research in Information Assurance and Security, Purdue University, West Lafayette, 2007.

[78] Bruce Schneier, *Applied cryptography, second edition: Protocols, algorthms, and source code in c*, John Wiley Sons, Inc., 1996.

[79] Claude Shannon, *Communication theory of secrecy systems*, BELL-SYS-TECH **28** (1949), no. 4, 656–715.

[80] Victor Shnayder, Mark Hempstead, Bor rong Chen, Geoff Werner Allen, and Matt Welsh, *Simulating the power consumption of large-scale sensor network applications*, SenSys '04: Proceedings of the 2nd international conference on Embedded networked sensor systems (New York, NY, USA), ACM, 2004, pp. 188–200.

[81] Piotr Szczechowiak, Leonardo Oliveira, Michael Scott, Martin Collier, and Ricardo Dahab, *Nanoecc: Testing the limits of elliptic curve cryptography in sensor networks.*, 2008.

[82] Erik Tews, Ralf-Philipp Weinmann, and Andrei Pyshkin, *Breaking 104 bit wep in less than 60 seconds*, Cryptology ePrint Archive, Report 2007/120, 2007.

[83] Salil Vadhan and Alon Rosen, *Introduction to cryptography*, 2006, Disponível em http://eecs.harvard.edu/ salil/cs120/docs/lec3.pdf.

[84] Ramnath Venugopalan, Prasanth Ganesan, Pushkin Peddabachagari, Alexander Dean, Frank Mueller, and Mihail Sichitiu, *Encryption overhead in embedded systems and sensor network nodes: modeling and analysis*, CASES '03: Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems (New York, NY, USA), ACM, 2003, pp. 188–197.

[85] G. S. Vernam, *Cipher printing telegraph systems for secret wire and radio telegraphic communications*, Journal American Institute of Electrical Engineers **XLV** (1926), 109–115.

[86] David Wagner, *Lecture 1. cryptography*, 2006, Disponível em http://www.cs.berkeley.edu/ daw/teaching/cs276-s04/01.ps.

[87] Chieh-Yih Wan, Andrew T. Campbell, and Lakshman Krishnamurthy, *Psfq: a reliable transport protocol for wireless sensor networks*, WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications (New York, NY, USA), ACM Press, 2002, pp. 1–11.

[88] Arvinderpal S. Wander, Nils Gura, Hans Eberle, Vipul Gupta, and Sheueling Chang Shantz, *Energy analysis of public-key cryptography for wireless sensor networks*, PERCOM '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications (Washington, DC, USA), IEEE Computer Society, 2005, pp. 324–328.

[89] Haodong Wang and Qun Li, *Efficient implementation of public key cryptosystems on mote sensors (short paper)*, ICICS, 2006, pp. 519–528.

[90] Yodai Watanabe, Junji Shikata, and Hideki Imai, *Equivalence between semantic security and indistinguishability against chosen ciphertext attacks*, PKC '03: Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography (London, UK), Springer-Verlag, 2003, pp. 71–84.

[91] Ronald Watro, Derrick Kong, Sue fen Cuti, Charles Gardiner, Charles Lynn, and Peter Kruus, *Tinypk: securing sensor networks with public key technology*, SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks (New York, NY, USA), ACM, 2004, pp. 59–64.

[92] Matt Welsh, Dan Myung, Mark Gaynor, and Steve Moulton, *Resuscitation monitoring with a wireless sensor network. american heart association, resuscitation science symposium. in supplement to circulation: Journal of the american heart association*, 2003.

[93] Alec Woo and David E. Culler, *A transmission control scheme for media access in sensor networks*, MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking (New York, NY, USA), ACM, 2001, pp. 221–235.

[94] Yong Yao and Johannes Gehrke, *The cougar approach to in-network query processing in sensor networks*, SIGMOD Rec. **31** (2002), no. 3, 9–18.

[95] Wei Ye, John Heidemann, and Deborah Estrin, *An energy-efficient mac protocol for wireless sensor networks*, Proceedings of the IEEE Infocom, 2002, pp. 1567–1576.