

# Distribuição de Chaves Criptográficas em Redes de Sensores Sem Fio

Este exemplar corresponde à redação final da Tese devidamente corrigida e defendida por Leonardo Barbosa e Oliveira e aprovada pela Banca Examinadora.

Campinas, 27 de novembro de 2008.

A handwritten signature in blue ink, appearing to be 'RD', with a stylized flourish at the end.

Prof. Dr. Ricardo Dahab (Orientador)

Tese apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

**FICHA CATALOGRÁFICA ELABORADA PELA  
BIBLIOTECA DO IMECC DA UNICAMP**

Bibliotecária: Miriam Cristina Alves – CRB8a / 5094

Oliveira, Leonardo Barbosa

OL4d                    Distribuição de chaves criptográficas em redes de sensores sem fio/Leonardo Barbosa e Oliveira -- Campinas, [S.P. :s.n.], 2008.

Orientador : Ricardo Dahab

Tese (Doutorado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Criptografia. 2. Redes de computadores – Medidas de segurança.  
3. Sistemas de comunicação sem fio. 4. Segurança. I. Dahab, Ricardo.  
II. Universidade Estadual de Campinas. Instituto de Computação. III.  
Título.

Título em inglês: Cryptographic key distribution in sensor networks.

Palavras-chave em inglês (Keywords): 1. Cryptography. 2. Computer Networks . 3. Sensor Networks. 4. Security.

Área de concentração: Teoria da computação

Titulação: Doutor em Ciência da Computação

Banca examinadora: Prof. Dr. Ricardo Dahab (IC-UNICAMP)  
Prof. Dr. Paulo S. L. M. Barreto (POLI-USP)  
Prof. Dr. Joni da Silva Fraga (DAS-UFSC)  
Prof. Dr. Júlio C. López Hernandez (IC-UNICAMP)  
Prof. Dr. Nelson L. S. Da Fonseca (IC-UNICAMP)

Data da defesa: 29/09/2008

Programa de Pós-Graduação: Doutorado em Ciência da Computação

## TERMO DE APROVAÇÃO

Tese Defendida e Aprovada em 29 de setembro de 2008, pela Banca examinadora composta pelos Professores Doutores:



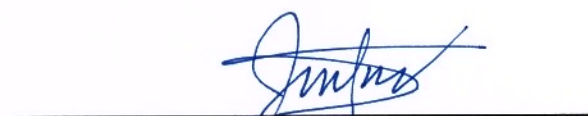
---

Prof. Dr. Joni da Silva Fraga  
Centro Tecnológico - UFSC.



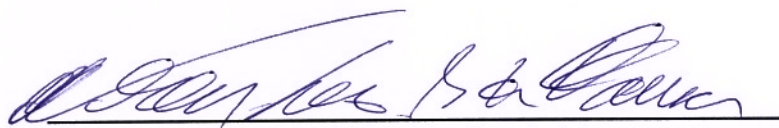
---

Prof. Dr. Paulo Sérgio Licciardi Messeder Barreto  
DEC – Poli-USP.



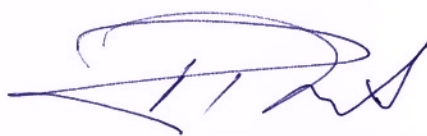
---

Prof. Dr. Julio César López Hernández  
IC – UNICAMP.



---

Prof. Dr. Nelson Luis Saldanha da Fonseca  
IC – UNICAMP.



---

Prof. Dr. Ricardo Dahab  
IC – UNICAMP.

# Distribuição de Chaves Criptográficas em Redes de Sensores Sem Fio

Leonardo Barbosa e Oliveira<sup>1</sup>

Dezembro de 2008

## Banca Examinadora:

- Prof. Dr. Ricardo Dahab (Orientador)
- Prof. Dr. Paulo S. L. M. Barreto  
Escola Politécnica – USP
- Prof. Dr. Joni da Silva Fraga  
Departamento de Automação e Sistemas – UFSC
- Prof. Dr. Julio C. López Hernandez  
Instituto de Computação – Unicamp
- Prof. Dr. Nelson L. S. da Fonseca  
Instituto de Computação – Unicamp
- Prof. Dr. Jeroen Van de Graaf  
Departamento de Computação – UFOP (Suplente)
- Prof. Dr. Marco Aurélio Henriques  
Faculdade de Engenharia Elétrica e de Computação – Unicamp (Suplente)
- Prof. Dr. Jacques Wainer  
Instituto de Computação – Unicamp (Suplente)

---

<sup>1</sup>Suporte financeiro de: Bolsa FAPESP (processo 2005/00557-9), Bolsa CAPES PDEE (processo 4630/06-8) e Bolsa Microsoft (*Microsoft Research Fellowship 2008*).

# Resumo

Redes de Sensores Sem Fio (RSSFs) são compostas em sua maioria por pequenos nós sensores dotados de recursos extremamente limitados. Estes, por sua vez, se comunicam com o mundo externo através de nós poderosos chamados de sorvedouros ou estações rádio base. RSSFs são empregadas com o objetivo de monitorar regiões, oferecendo dados sobre a área monitorada para o resto do sistema. Tais redes podem ser utilizadas para diferentes aplicações, tais como operações de resgate em áreas de conflito/desastre, espionagem industrial e detecção de exploração ilegal de recursos naturais.

Em RSSFs existem aplicações críticas nas quais propriedades de segurança são de vital importância. Segurança, por sua vez, é comumente alavancada através de esquemas de distribuição de chaves. A maioria dos padrões de distribuição de chaves presentes na literatura, todavia, não são apropriados para RSSFs: métodos baseados em esquemas de chave pública convencionais, devido aos seus requisitos de processamento e banda; chaves de grupo, em função das suas vulnerabilidades de segurança; e chaves par-a-par (*pairwise*), por causa da baixa escalabilidade.

Um outro dado é que há uma vasta gama de arquiteturas propostas para RSSFs e que uma mesma técnica de distribuição de chaves pode ser a melhor para uma, mas não para outra, visto que diferentes arquiteturas de rede exibem padrões de comunicação distintos. Em outras palavras, não existe uma panacéia, e mecanismos de distribuição de chaves para RSSFs devem, portanto, levar em consideração as idiossincrasias das arquiteturas para as quais são projetadas.

Tudo isso torna extremamente difícil e desafiadora a tarefa de dotar RSSFs de segurança. O objetivo deste trabalho foi propor soluções de distribuição de chaves que, concomitantemente, (i) fossem compatíveis com os recursos dos sensores e (ii) considerassem as particularidades das arquiteturas para as quais são propostas. Como será mostrado ao longo desta tese, iniciamos nosso trabalho com soluções personalizadas para certas arquiteturas de RSSFs e evoluímos para soluções flexíveis em que a segurança é alavancada de forma não interativa – o que é ideal para este tipo de rede. Até onde sabemos, nosso trabalho é pioneiro em soluções de segurança para RSSFs hierárquicas e em distribuição de chaves de forma autenticada e não interativa, usando Criptografia Baseada em Identidade,

neste tipo de rede.

# Abstract

Wireless sensor networks (WSNs) are ad hoc networks comprised mainly of small sensor nodes with limited resources and one or more base stations, which are much more powerful laptop-class nodes that connect the sensor nodes to the rest of the world. WSNs are used for monitoring purposes, providing information about the area being monitored to the rest of the system. Application areas range from battlefield reconnaissance and emergency rescue operations to surveillance and environmental protection.

There are also critical WSN applications in which security properties are of paramount importance. Security, in turn, is frequently bootstrapped through key distribution schemes. Most of the key distribution techniques, however, are ill-suited to WSNs: public key based distribution, because of its processing and bandwidth requirements; global keying, because of its security vulnerabilities; complete pairwise keying, because of its memory requirements.

It is worth noting, however, that a large number of WSN architectures have been proposed and a key distribution solution that is well suited to one architecture is likely not to be the best for another, as different network architectures exhibit different communication patterns. In other words, there is no panacea and the design of a key distribution scheme must therefore be driven by the peculiarities of the WSN architecture in question.

This all makes extremely hard and challenging the objective of securing WSNs. In this work, we aimed at proposing key distribution schemes that are both (i) lightweight and (ii) able to fulfill architecture-specific needs. As it will be shown throughout this thesis, we began our work with customized solutions for certain types of WSNs and then, subsequently, turned our attention to more flexible solutions, where security is bootstrapped in a non-interactive way through the use of Identity-Based Cryptography.

# Agradecimentos

O “problema” de agradecer a todos que contribuíram para com este trabalho está além das minhas possibilidades e, portanto, enveredar-me-ei nesta tentativa já ciente de seu infortunado desfecho.

A Deus, em primeiro lugar, meu muito obrigado. A minha *Amada*, Paula, o porquê – e como – de aqui estar (obrigado, Ana, pelos colos). Aos meus Pais, os quais sempre me dotaram de doses desmedidas de amor.

Ao meu orientador Doutor, também amigo e eterno credor, Dahab; e a sua família, Cristiane e Francisco, compreensivos sempre que demande a atenção do Professor.

Aos meus mentores: Antonio Alfredo, Julio, Wong, Mike, Zé Marcos, Keith, Aman e Feng.

À minha família; aos membros do LCA/Unicamp, do Networked Embedded Computing Group/MSR, do Information Security Group/RHUL – University of London, e da Dublin City University; aos professores, alunos e funcionários do IC e da Unicamp; e aos amigos de Campinas, do Brasil e do mundo. Aos meus onipresentes amigos das saudosas Minas Gerais. E à, agora também minha, família Zacharias, que majestosamente me acolheu na RMC.

Àqueles com os quais produzi trabalhos científicos, ou seja, R. Dahab., J. López, A. A. F. Loureiro, D. F. Aranha, E. Morais, F. Daguno, H. C. Wong, M. Bern, A. Mota, G. P. Safe, F. Rocha, R. Risério, J. N. Coelho Jr., Adrian C. Ferreira, Marco A. Vilaça, I. G. Siqueira, D. F. Macedo, J. M. Nogueira, E. Habib, P. Szczechowiak, M. Scott, M. Collier, A. Kansal, F. Zhao, B. Priyantha, M. Goraczko e D. Câmara.

Às agências de fomento à pesquisa brasileiras, FAPESP e CAPES, e à Microsoft Research pelo apoio financeiro.

Enfim, a todos que contribuíram para o sucesso desta tese, meu muito obrigado.



# Sumário

<b>Resumo</b>	<b>vii</b>
<b>Abstract</b>	<b>ix</b>
<b>Agradecimentos</b>	<b>xi</b>
<b>1 Introdução</b>	<b>1</b>
1.1 RSSFs . . . . .	3
1.1.1 O sensor . . . . .	4
1.1.2 Arquiteturas . . . . .	4
1.1.3 Modelo de Ataque . . . . .	6
1.1.4 Estratégias de Segurança . . . . .	6
1.2 Objetivos . . . . .	8
1.3 Contribuições . . . . .	9
1.3.1 Publicações Internacionais: Capítulos & Periódicos . . . . .	9
1.3.2 Publicações Internacionais: Anais de Congressos . . . . .	10
1.3.3 Publicações Nacionais . . . . .	12
1.4 Organização . . . . .	12
1.4.1 Capítulo 2 – LHA-SP . . . . .	13
1.4.2 Capítulo 3 – SecLEACH . . . . .	14
1.4.3 Capítulo 4 – TinyPBC . . . . .	15
<b>2 On the Design of Secure Protocols for Hierarchical Sensor Networks</b>	<b>19</b>
2.1 Introduction . . . . .	19
2.2 Hierarchical WSNs . . . . .	20
2.2.1 Organization . . . . .	20
2.2.2 Security . . . . .	21
2.3 Our Model . . . . .	21
2.4 LHA-SP . . . . .	22
2.4.1 Overview . . . . .	22

2.4.2	Protocol Description . . . . .	24
2.4.3	Protocol Implementation . . . . .	29
2.5	Security Analysis . . . . .	31
2.5.1	Network setup . . . . .	31
2.5.2	Network Operation . . . . .	31
2.5.3	Network Maintenance . . . . .	32
2.6	Performance Evaluation . . . . .	33
2.6.1	Communication and Computational Overhead . . . . .	33
2.6.2	Storage Overhead . . . . .	34
2.7	Related Work . . . . .	35
2.8	Conclusion . . . . .	37
<b>3</b>	<b>SecLEACH - On the Security of Clustered Sensor Network</b>	<b>39</b>
3.1	Introduction . . . . .	39
3.2	Adding security to LEACH . . . . .	40
3.2.1	LEACH: protocol description . . . . .	41
3.2.2	Security vulnerabilities . . . . .	42
3.2.3	Why existing key distribution schemes are inadequate . . . . .	43
3.2.4	Key distribution for LEACH: requirements and constraints . . . . .	44
3.3	SecLEACH – Applying random key distribution to LEACH . . . . .	44
3.3.1	$\mu$ TESLA . . . . .	45
3.3.2	Random key predistribution schemes . . . . .	45
3.3.3	SecLEACH: protocol description . . . . .	46
3.3.4	Security analysis . . . . .	47
3.4	Evaluation of our scheme . . . . .	49
3.4.1	Parameter values and their impact on performance . . . . .	50
3.4.2	Resiliency against node capture . . . . .	55
3.5	Related work . . . . .	56
3.6	Conclusion . . . . .	57
3.7	Average distance estimates . . . . .	57
3.7.1	Distance between CH and BS . . . . .	58
3.7.2	Distance between ordinary node and CH . . . . .	58
<b>4</b>	<b>Authenticated ID-based Non-Interactive Key Distribution for WSNs</b>	<b>61</b>
4.1	Introduction . . . . .	61
4.2	Bootstrapping Security in WSNs: Need for New Approaches . . . . .	63
4.3	Synergy between IBC and WSNs . . . . .	64
4.4	Authenticated Identity-Based Non-Interactive Key Distribution in Sensor Networks . . . . .	65

4.4.1	Pairings: Definition . . . . .	65
4.4.2	Setup . . . . .	66
4.4.3	Applying ID-NIKDS in WSNs . . . . .	67
4.5	Evaluation . . . . .	68
4.5.1	Implementation . . . . .	68
4.5.2	Performance . . . . .	70
4.5.3	Storage . . . . .	71
4.6	Related Work . . . . .	72
4.7	Conclusion . . . . .	73
<b>5</b>	<b>Conclusões e Trabalhos Futuros</b>	<b>75</b>

# Lista de Tabelas

2.1	RAM and ROM memory for Motes (in bytes)	35
3.1	Prob. $P_s$ of key sharing as a function of security level $sl$ and key ring size $m$	54
3.2	Energy Overhead	55
4.1	Time costs to evaluate binary field multiplication and the $\eta_T$ pairing on ATmega128L using TinyPBC.	71
4.2	Memory costs to evaluate the $\eta_T$ pairing on ATmega128L using TinyPBC.	71

# Lista de Figuras

1.1	Cálculo do emparelhamento de Tate usando o Algoritmo de Miller. . . . .	17
2.1	The setup protocol. . . . .	26
2.2	Node addition protocol. . . . .	27
2.3	Orphan adoption protocol. . . . .	29
3.1	LEACH protocol . . . . .	42
3.2	SecLEACH protocol . . . . .	48
3.3	Orphan rate, for $sl=0.99$ . . . . .	52
3.4	Average node-to-CH distance . . . . .	53
3.5	Energy consumption, for $m=100$ . . . . .	54
3.6	Energy consumption, for $sl=0.99$ . . . . .	54
3.7	SecLEACH's resiliency against node capture . . . . .	55

# Capítulo 1

## Introdução

Redes de Sensores Sem Fio (RSSFs) [34, 94, 1, 2, 66] são um tipo particular de Redes Móveis Ad hoc (*Mobile Ad hoc Networks* – MANETs). Elas são compostas em sua maioria por pequenos nós (*nodes*) sensores cujos recursos (energia, largura de banda, processamento etc.) são extremamente limitados. Estes sensores, por sua vez, se conectam com o mundo externo por meio de dispositivos poderosos chamados de sorvedouros (*sink*) ou Estações Rádio Base (ERBs). Elas são utilizadas com o intuito de monitorar regiões, oferecendo dados sobre a área monitorada, também chamada de *área de interesse* (*interest area*) para o resto do sistema. Dentre sua vasta gama de aplicações estão operações de resgate em áreas de conflito e/ou desastre, espionagem industrial e detecção de exploração ilegal de recursos naturais. Ainda vale mencionar que, em 2003, ocorreu um *workshop* [79] patrocinado pelo *National Science Foundation* para identificar tópicos de pesquisa fundamentais em redes e a área de RSSFs foi um dos seis selecionados.

Como veremos ao longo desta tese, embutir segurança em RSSFs é uma tarefa complexa e muito desafiadora. Ela é comumente justificada em RSSFs por causa das aplicações militares. Isto é, aplicações executadas durante batalhas e, portanto, na presença de adversários dos quais informações *sensoriadas* precisam ser protegidas. Acreditamos, contudo, que uma vez que as RSSFs comecem a ser empregadas em larga escala o emprego de mecanismos de segurança tornar-se-á mais e mais comum. Isso porque o sigilo deverá ser uma propriedade imperativa também em RSSFs rurais e urbanas. Por exemplo, fazendeiros e indústrias que lançarem mão das redes para monitorar sua cadeia de plantações e suprimento, respectivamente, desejarão manter os dados monitorados secretos, impedindo que os mesmos cheguem ao conhecimento de competidores. Ademais, autenticação – outra propriedade de segurança – poderá ser útil até mesmo em RSSFs domésticas, evitando que sensores de redes vizinhas interajam entre si acidentalmente.

Idealmente, um esquema de segurança para RSSFs tem que prover perfeita conectividade e resiliência. Em outras palavras, sensores devem ser capazes de (i) comunicar-se

com quaisquer outros sensores de forma segura e (ii) os danos do comprometimento de um sensor devem ficar restritos ao mesmo – note-se que essas propriedades têm que ser satisfeitas mesmo para sensores que foram dispostos <sup>1</sup> em diferentes momentos. Ademais, o esquema deve ser de baixo custo tanto em termos de processamento, como de comunicação e armazenamento.

Segurança, por sua vez, é comumente alavancada (*bootstrapped*) através de esquemas de distribuição de chaves. O baixo poder computacional dos sensores, contudo, inviabiliza o emprego de criptossistemas assimétricos, também chamados de sistemas baseados em Criptossistemas de Chave Pública (*Public Key Cryptosystem* – PKC [25]), convencionais (RSA [96], por exemplo) e, até recentemente, propriedades de segurança eram alcançadas por meio de criptossistemas simétricos (RC5, SkipJack etc. [90, 56]) em RSSFs.

Apesar de mais eficientes, criptossistemas simétricos possuem algumas inconveniências [105]. Antes de tudo, as partes que desejam comunicar-se de forma segura precisam, *a priori*, decidir por uma chave em comum e então compartilhar essa chave através de um canal seguro. Logo, o primeiro (e maior) desafio é encontrar tal canal, já que a necessidade do compartilhamento de chaves advém justamente da necessidade de se viabilizar um canal seguro. Este problema é ainda pior em RSSFs, pois ao contrário das redes convencionais, em que existem canais alternativos (telefone, correio, múltiplas rotas etc.) que podem ser usados para a troca de chaves, o único canal existente em RSSFs é o enlace sem fio. A segunda dificuldade é que para se comunicar de forma segura, um indivíduo – idealmente – deveria compartilhar uma chave distinta com cada um dos outros participantes da rede. Isso causa um sério problema de escalabilidade, uma vez que em geral RSSFs são compostas por centenas ou milhares de dispositivos sensores. Uma alternativa, é verdade, seria vários sensores compartilharem uma mesma chave, mas, neste caso, a violação de um único sensor comprometeria todo o grupo. Por fim, o compartilhamento de uma mesma chave por mais de um indivíduo em criptossistemas simétricos possibilita ataques de retratabilidade (*repudiation*) e personificação (*spoofing*).

É bem verdade que existem esquemas de pré-distribuição de chaves baseados em criptossistemas simétricos (e.g., [90, 33, 118, 91, 17, 54, 51, 15, 28, 64, 40, 22, 71, 83]) especialmente projetados para RSSFs. Mesmo eles, contudo, possuem pontos fracos. Isto é, embora sejam apropriados para as aplicações e arquiteturas para os quais foram concebidos, não são adequados a outras. Tais esquemas oferecem um compromisso entre conectividade e robustez, mas não fornecem um nível ideal de ambos. Além disso, a maioria dos esquemas depende de alguma interação entre os sensores para efetuar o acordo de chaves (*key agreement*).

Mais recentemente, descobriu-se que PKCs alternativos são viáveis em RSSFs [110, 41, 67]. Já que nesses sistemas as partes comunicantes possuem apenas um par de chaves,

---

<sup>1</sup>Neste documento, empregaremos o verbete *dispor* como tradução do inglês *deploy*.

PKCs são escaláveis e simples de ser utilizados. Tal conveniência, entretanto, tem custo: um esquema de autenticação das chaves públicas precisa ser fornecido. E autenticação de chaves, por sua vez, seja tradicional (PKI e/ou certificados) ou especialmente voltada para RSSFs (como em [30]), resulta em sobrecarga (*overhead*) – o que vai particularmente de encontro aos quesitos de RSSFs.

Resumindo, dotar RSSFs de segurança é uma tarefa especialmente desafiadora e fundamental para a ampla adoção da tecnologia de RSSFs. No presente trabalho propusemos diferentes soluções de segurança, cujos focos principais são as distribuições de chaves. Como será mostrado, iniciamos nosso trabalho com soluções personalizadas para certas arquiteturas de RSSFs e evoluímos para soluções flexíveis em que a segurança é alavancada de forma não interativa, o que é ideal para este tipo de rede. Até onde sabemos, nosso trabalho é pioneiro em soluções de segurança para RSSFs hierárquicas e o primeiro a realizar distribuição de chaves não interativa usando Criptografia Baseada em Emparelhamentos.

A seguir, discorreremos brevemente sobre RSSFs e seus aspectos de segurança (Seção 1.1), apresentaremos nossos objetivos e contribuições (Seções 1.2 e 1.3, respectivamente), e, por fim, descreveremos a organização do restante deste trabalho (Seção 1.4).

## 1.1 RSSFs

Além das características já mencionadas, RSSFs se diferem das MANETs convencionais nos seguintes aspectos:

1. o tráfego é assimétrico, isto é, parte dos dispositivos sensores em direção a ERB;
2. os integrantes da rede, os sensores, possuem pouca ou nenhuma mobilidade;
3. não existe possibilidade de recarga de energia, o que diminui o *tempo de vida* da rede;
4. escala, isto é, RSSFs são duas ou três ordens de grandeza maior que as MANETs tradicionais;
5. enfim, a escassez de recursos computacionais é ainda mais intensa.

Do ponto de vista de aplicação, RSSFs podem ser empregadas em diversos cenários tais como monitoramento ambiental e agropecuário, rastreamento de eventos, coordenação de ações, mineração e processamento de informação [66]. Dada a biodiversidade brasileira e a vocação do país para a agropecuária, RSSFs podem contribuir – e muito – para o desenvolvimento dessas áreas.



A seguir apresentamos brevemente características de RSSFs. Isto é, primeiro apresentamos o dispositivo sensor em si (Seção 1.1.1) e, subsequentemente, arquiteturas (Seção 1.1.2), modelo de ataque (Seção 1.1.3) e estratégias de segurança (Seção 1.1.4).

### 1.1.1 O sensor

O nó sensor – ou simplesmente sensor – é um dispositivo computacional dotado de processador, memória, rádio, antena e sensor propriamente dito. Este último pode ser de temperatura, umidade, pressão etc. e tem por função coletar dados do ambiente e repassar para a unidade de processamento. Após o dado ser processado ele é então encaminhado para outros nós sensores via rádio e assim sucessivamente até alcançar a ERB.

A família de sensores MICA motes (MICA, MICA2, MICA-DOT e MICAz motes [45, 20]) é praticamente um padrão de pesquisa em RSSFs. O mais novo deles, o MICAz [74], possui as seguintes características:

- processador ATmega128L 7.3828-MHz de 8-*bits*;
- 4-KB de memória de dados (RAM);
- 128-KB de memória programa (*flash*);
- tecnologia de rádio 802.15.4 (*ZigBee*) com até 250 *kbps* de largura de banda.

Além dos MICAs, existem outros sensores populares: TelosB [93], Tmote [108] e o iMote [77]. Empresas como a Crossbow [20] e Sensoria [103] já comercializam e oferecem pacotes de soluções para esses dispositivos.

### 1.1.2 Arquiteturas

Diversas arquiteturas foram propostas para RSSFs. Em *RSSFs planas* [2], todos os dispositivos possuem papéis semelhantes no sensoriamento, processamento de dados e roteamento. Em particular, todos os sensores operam com raio de transmissão limitado para poupar energia e a comunicação dos sensores em direção à ERB (sensor→ERB) é então multi-salto (*multi-hop*), com sensores fazendo o papel de roteadores uns para os outros.

Em *RSSFs hierárquicas* [34], por outro lado, a rede é em geral organizada em grupos (*clusters*), em que líderes (CHs – *cluster-heads*) e membros comuns de grupos exercem papéis diferentes. Enquanto membros comuns são responsáveis pelo sensoriamento, CHs são responsáveis por tarefas adicionais, como coletar e processar o dado *sensoriado* pelos demais membros do grupo, e encaminhar os resultados para a ERB.

RSSFs podem ser, também, *homogêneas*, quando todos os dispositivos – exceto a ERB – possuem capacidades equivalentes [43]; ou *heterogêneas*, quando há sensores mais poderosos que os demais [10, 59].

Além das organizações planas×hierárquicas e homogêneas×heterogêneas, RSSFs podem diferir-se de outras formas. Uma rede hierárquica, por exemplo, pode ter 2 níveis, com CHs no nível superior e seus filhos (sensores membros comuns do grupo) no nível inferior [73], ou pode ter uma hierarquia de vários níveis aninhados, em que sensores de um nível atuam como CHs dos que estão um nível abaixo e como filhos dos que estão um nível acima [6].

Em relação aos grupos, CHs podem ser aleatoriamente escolhidos dentre os sensores comuns de uma rede homogênea (como no protocolo LEACH [43]), ou podem ser os dispositivos mais poderosos que compõem uma rede heterogênea [73]. A forma de agrupamento também pode ser variar. Por exemplo, no protocolo *k-hop* [35], grupos são formados de forma que seus membros estejam todos a *k*-saltos (*hops*) uns dos outros. Alternativamente, alguns sensores podem probabilisticamente se auto-eleger CHs [43], e o restante dos dispositivos se agrupar em torno do CH que estiver mais próximo – note-se que além da distância geográfica, outras métricas podem ser adotadas para a escolha do CH a qual se agrupar, como, por exemplo, a potência do sinal de rádio [43].

O roteamento também pode variar dependendo da RSSF [73]. Em RSSFs planas ele é em geral multi-salto e, em RSSFs hierárquicas, variado. Nessas últimas, ele pode ser *single-hop* de filhos para CHs e de CHs para ERBs; ou, alternativamente, multi-salto dentro dos grupos (isto é, dos filhos para os CHs) e de CH para CH até que se alcance a ERB.

RSSFs também são classificadas quanto à forma de sensoramento. Elas podem ser dirigidas a eventos (*event-driven*), quando os sensores reportam à ERB sempre e imediatamente depois que observam um evento; ou dirigidas a relógio, quando elas reportam todos os eventos ocorridos após intervalos específicos de tempo (por exemplo, de cinco em cinco minutos os sensores reportam todos os eventos ocorridos neste íterim.)

Um dado interessante é que a organização hierárquica [69] em RSSFs aumenta a vazão do sistema e, também, diminui o seu atraso. Tal comportamento deve-se, em grande parte, ao pequeno número de *hops* entre os sensores e a ERB. Além disso, a economia de energia aumenta na medida que o número de níveis hierárquicos são acrescidos à rede. Já foi mostrado [69] que redes heterogêneas apresentam desempenho consideravelmente superior quando comparadas às homogêneas, em termos de degradação da área *sensoriada*, vazão de dados, atraso e consumo de energia.

### 1.1.3 Modelo de Ataque

Como qualquer outra rede *ad hoc* sem fio, RSSFs são vulneráveis a ataques [57, 111]. Porém, além das vulnerabilidades já existentes na comunicação sem fio/ad hoc, RSSFs enfrentam problemas adicionais. Elas são comumente dispostas em ambientes abertos, muitas vezes adversos, o que torna os sensores fisicamente acessíveis a adversários. Não obstante, a forte escassez de recursos inviabiliza a adoção de soluções de segurança convencionais. Por fim, o fato de sensores serem descartáveis e, por conseguinte, requererem baixo custo, torna pouco viável equipá-los com mecanismos contra violação (*tamper*) – isso encareceria muito o custo de produção.

RSSFs são vulneráveis a um grande número de ataques [57, 111], como interferência, personificação (*spoofing*) e retransmissão de mensagens (*replay*). Caso um adversário consiga se passar por um sensor legítimo, ele pode efetuar ataques como o *buraco negro* (*blackhole*) [57] e *reencaminhamento seletivo* (*selective forwarding*) [68] e, potencialmente, causar danos a grandes frações da rede – no ataque de buraco negro o sensor malicioso simplesmente some com toda e qualquer mensagem a ele enviada; já no reencaminhamento seletivo, o adversário não repassa apenas certas mensagens.

Outra opção, obviamente, é não se interferir no roteamento e tentar prejudicar o resultado do sensoriamento injetando dados espúrios na rede. Neste caso, o adversário pode também bisbilhotar (*eavesdrop*) a comunicação entre nós legítimos a fim de obter a leitura de dados efetuada pelos demais sensores.

Note-se que no caso de RSSFs hierárquicas, em particular, ataques envolvendo CHs são os mais devastadores, já que os mesmos possuem uma grande densidade de vizinhos e são responsáveis pelas funções chaves da rede (roteamento, fusão de dados e interface com a ERB, por exemplo).

Por fim, é importante salientar que as ERBs não estão sujeitas aos ataques descritos acima. Por serem dotadas de grandes poderes computacionais e localizarem-se em ambientes onde existe proteção física, as ERBs são consideradas à prova de ataques e entidades de confiança, em RSSFs.

### 1.1.4 Estratégias de Segurança

Segurança é usualmente suscitada através de um sistema de distribuição de chaves (por gentileza, consulte Carman *et al.* [14] para uma interessante introdução à distribuição de chaves em RSSFs). Em alto nível, os principais objetivos de segurança em RSSFs são:

- controle de acesso, isto é, garantia de que apenas sensores legítimos participam da rede;

- fornecer propriedades como autenticação, sigilo e integridade à comunicação de dados *sensoriados*;
- garantir o funcionamento da rede como um todo, isto é, dirimir os efeitos de ataques fazendo com que o impacto dos mesmos seja apenas local.

O baixo poder computacional dos sensores, contudo, inviabiliza o emprego de PKC convencionais (RSA/DSA, por exemplo). Para se ter uma idéia, PKC tradicionais requerem até três ordens de grandeza mais recursos computacionais que criptossistemas simétricos [14] (RC5, SkipJack etc. [90, 56]). Consequentemente, até pouco tempo, primitivas de segurança como sigilo, autenticação e integridade, em RSSFs, eram obtidas através destes últimos.

A fim de se contornar algumas dessas inconveniências, alguns trabalhos propuseram a adoção de criptossistemas simétricos por meio de pré-distribuição de chaves ([33, 90, 118, 64, 28], por exemplo). Basicamente, são três as abordagens:

1. emprego de uma chave global, em que a mesma chave é carregada em todos os sensores antes da disposição. Assim, essa única chave é posteriormente utilizada por todos os dispositivos para proteger a comunicação.
2. compartilhamento de chaves par-a-par entre a ERB e os sensores (e.g., [90]). Ou seja, antes da disposição a ERB carrega cada sensor com um chave distinta e mantém consigo uma cópia de todas. As chaves, posteriormente, são usadas para proteger a comunicação entre os sensores e a ERB;
3. compartilhamento de chaves secretas entre sensores comuns, a qual pode ser total, isto é, cada sensor compartilha uma chave distinta (par-a-par) com cada um dos demais (como em [14], por exemplo); ou aleatória (como em [33], por exemplo), em que se atribui um subconjunto de chaves (“chaveiro”) escolhidas aleatoriamente a cada um dos sensores (a esperança, neste caso, é que um sensor tenha intersecção não vazia do seu chaveiro com o de pelo menos um de seus vizinhos);

A primeira abordagem é a mais simples. É um método conhecido, eficiente e escalável. Há uma chave apenas e os sensores a empregam para proteger todo e qualquer tipo de comunicação. Ela é carregada nos sensores *a priori* e, porque todos a possuem, a conectividade é total. Por outro lado, se um único sensor é capturado e sua chave comprometida, toda a rede torna-se vulnerável. Em outras palavras, esta técnica carece de robustez.

Na segunda abordagem, a ERB atua como um Centro de Distribuição de Chaves (*Key Distribution Center* – KDC). Esse mecanismo, todavia, requer que todos os sensores contatem a ERB caso queiram obter chaves. Por esta causa ele é caro em termos de

comunicação e, portanto, inadequado para nosso contexto. Não obstante, a abordagem é centralizada na ERB, a qual pode tornar-se um gargalo.

A terceira e última é o emprego de chaves secretas distintas entre os sensores. O problema da estratégia é a escalabilidade, já que sensores são também restritos em memória e, portanto, não são capazes de armazenar muitas chaves. Quando a distribuição é aleatória há ainda um outro problema: nem sempre os sensores compartilham uma chave com o interlocutor. Neste caso, eles empregam os enlaces seguros – aqueles entre os sensores com os quais de fato compartilham chaves – para também estabelecerem uma chave com este.

A idéia dos esquemas acima é o compromisso entre conectividade (entre os sensores), resiliência e consumo de recursos. Contudo, na grande maioria dos cenários, não é possível se atingir simultaneamente o melhor dos três. Motivados por isso, a comunidade de criptografia em RSSFs passou a investigar técnicas mais eficientes de PKC. Uma dessas técnicas é a Criptografia de Curvas Elípticas [76, 58] (*Elliptic Curve Cryptography* – ECC). O intuito era encontrar primitivas de PKC “baratas” o suficiente para serem executadas por sensores. E eles obtiveram êxito: utilizando ECC foi mostrado ([41, 67], por exemplo) que PKC não é apenas factível em RSSFs, mas uma promessa para a segurança de dispositivos de baixo poder computacional como um todo, dado que, para um certo nível de segurança, ECC consome muito menos recursos computacionais que métodos convencionais de PKC.

Todavia, para que essas propostas sejam efetivamente aplicadas em RSSFs, é necessário que chaves públicas sejam autenticadas. Caso contrário, ataques do tipo homem-do-meio (*man-in-the-middle*) podem ser disparados contra a rede. Esse tipo de autenticação é geralmente obtido pelo uso de uma Infra-Estrutura de Chaves Públicas (*Public Key Infra-Structure* – PKI), nas quais certificados são emitidos por entidades de confiança e posteriormente verificados por usuários. Essas operações, por sua vez, acarretam sobrecargas (*overhead*) de comunicação, computação e armazenamento, o que torna PKIs inadequadas para RSSFs [30].

Motivados pelo desafio de encontrar um método viável para certificação de chaves publicas em RSSFs, identificamos na Criptografia Baseada em Identidade [104] uma alternativa promissora, que passamos, então, a explorar intensamente. Detalhamos nossos esforços nas seções subseqüentes.

## 1.2 Objetivos

Uma importante questão a ser discutida quando se deseja empregar criptografia para proteger uma rede é a distribuição de chaves, a qual foi e tem sido estudada de maneira intensa em RSSFs ([14, 33, 17, 118, 62, 63, 49, 117, 91, 29, 51, 48, 54, 15, 64, 28, 16, 92],

por exemplo). É importante notar, contudo, que existe uma vasta gama de arquiteturas [107, 1] propostas para RSSFs e que uma mesma técnica de distribuição de chaves pode ser a melhor para uma, mas não para outra, visto que diferentes arquiteturas de rede exibem padrões de comunicação distintos. Em outras palavras, não existe uma panacéia, e mecanismos de distribuição de chaves para RSSFs devem levar em consideração as idiossincrasias das arquiteturas para as quais são projetadas.

Nosso objetivo é, portanto, propor soluções de distribuição de chaves que, concomitantemente, (i) sejam compatíveis com os recursos dos sensores e (ii) considerem as particularidades das arquiteturas para as quais são propostas.

## 1.3 Contribuições

A seguir, listamos o conjunto de trabalhos publicados ao longo do nosso doutorado. Primeiro, listamos os trabalhos publicados em capítulos de livros e periódicos internacionais (Seção 1.3.1). Em seguida, os trabalhos publicados em anais de conferências internacionais (Seção 1.3.2). E, por fim, os trabalhos publicados em meios nacionais (Seção 1.3.3). Vale também lembrar que outra forma de reconhecimento do trabalho foi a obtenção da bolsa de doutorado da Microsoft (*Microsoft Fellowship Award*), a qual é concedida a apenas dois alunos da América Latina por ano.

### 1.3.1 Publicações Internacionais: Capítulos & Periódicos

Felizmente, obtivemos êxito em publicar as contribuições angariadas ao longo do nosso curso em meios de comunicação de qualidade. Prova disso, são os meios listados abaixo: dois capítulos de livros, um deles editado pela *McGraw-Hill International*, um artigo no periódico *Signal Processing, Qualis A*, um artigo no periódico *Journal of Parallel and Distributed Computing* (JPDC), *Qualis A*, e outro no *International Journal of Security and Networks* (IJSN), um fórum especializado em que o índice de aceite foi de 17%.

1. **L. B. OLIVEIRA**, D. Aranha, E. Morais, F. Daguno, J. López and R. Dahab. and R. Dahab. *On the Identity-Based Encryption for Sensor Networks*. Handbook of Wireless Mesh and Sensor Networking. McGraw-Hill International, NY. (Book Chapter: To appear).
2. **L. B. OLIVEIRA**, I. G. Siqueira, D. F. Macedo, A. A. F. Loureiro and J. M. Nogueira. *P2P over MANETs: Application and Network Layers Routing Assessment*. Handbook on Mobile Peer-to-Peer Computing for Next Generation Distributed Environments: Advancing Conceptual and Algorithmic Applications. IGI Global, Pennsylvania (Book Chapter: invited).

3. **L. B. OLIVEIRA**, A. Ferreira, M. Vilaça, H. C. Wong, M. Bern, R. Dahab, and A. A. F. Loureiro *SecLEACH - On the Security of Clustered Sensor Networks, Signal Processing* (Elsevier pub.). Volume 87, issue 12, 2007 (pages 2882–2895). Special Issue on Information Processing and Data Management in Wireless Sensor Networks.
4. **L. B. OLIVEIRA**, H. C. Wong, A. A. F. Loureiro, and R. Dahab. *On the Design of Secure Protocols for Hierarchical Sensor Networks*. International Journal of Security and Networks (IJSN). Special Issue on Cryptography in Networks. Volume 2, issue 3/4, 2007 (p. 216-227) – índice de aceite: 17%.
5. **L. B. OLIVEIRA**, I. G. Siqueira and A. A. F. Loureiro. *On the Performance of Ad hoc Routing Protocols under a Peer-to-Peer Application*. Journal of Parallel and Distributed Computing (JPDC): special issue on the Design and Performance of Networks for Super, Cluster, and Grid-Computing. Volume 65, Issue 11, November 2005 (p. 1337-1347).

### 1.3.2 Publicações Internacionais: Anais de Congressos

Trabalhos publicados em conferências de tradição (isto é, com mais de 4 edições) patrocinadas pelo IEEE são classificados como *Qualis A*. Nove dos onze trabalhos abaixo se enquadram nesta classificação. Dentre os dois restantes, um foi também publicado em uma conferência patrocinada pelo IEEE, porém, em sua quarta edição, e a outra em uma renomada conferência da área, cujo índice de aceite foi de 21% e que será publicada pela *Springer/Lecture Notes in Computer Science*.

1. **L. B. OLIVEIRA**, M. Scott, J. Lopez, and R. Dahab. TinyPBC: Pairings for Authenticated Identity-Based Non-Interactive Key Distribution in Sensor Networks. 5th International Conference on Networked Sensing Systems (INSS'08). Sponsored by IEEE. June 2008, Kanazawa/Japan (p. 173-179) – índice de aceite: 21%.
2. P. Szczechowiak, **L. B. OLIVEIRA**, M. Scott, M. Collier, and R. Dahab. NanoECC: Testing the Limits of Elliptic Curve Cryptography in Sensor Networks, European conference on Wireless Sensor Networks (EWSN'08). Lecture Notes in Computer Science, volume 4913, 2008, Bologna/Italy (p. 305-320) – índice de aceite: 21%.
3. **L. B. OLIVEIRA**, D. Aranha, E. Morais, F. Daguno, J. López, and R. Dahab. *TinyTate: Computing the Tate Pairing in Resource-Constrained Sensor Nodes*. 6th IEEE International Symposium on Network Computing and Applications (NCA'07). July 2007, Cambridge/MA (p. 318-323).

4. **L. B. OLIVEIRA**, R. Dahab, J. Lopez, F. Daguno, and A. A. F. Loureiro. *Identity-Based Cryptography for Sensor Networks*. 5th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'07). March 2007, White Plains/NY (p. 290-294).
5. **L. B. OLIVEIRA**, A. A. F. Loureiro, and R. Dahab. *SOS: Secure Overlay Sensor-nets*. 5th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'07). March 2007, White Plains/NY (p. 548-553).
6. **L. B. OLIVEIRA**, H. C. Wong, M. Bern, R. Dahab, and A. A. F. Loureiro. *SecLEACH - A Random Key Distribution Solution for Securing Clustered Sensor Networks*. 5th IEEE International Symposium on Network Computing and Applications (NCA'06). July 2006, Cambridge/MA (p. 145-154) – índice de aceite: 35%.
7. **L. B. OLIVEIRA** and R. Dahab. *Pairing-Based Cryptography for Sensor Networks*. 5th IEEE International Symposium on Network Computing and Applications (NCA'06). July 2006, Cambridge/MA (fast abstract).
8. A. Mota, **L. B. OLIVEIRA**, G. P. Safe, F. F. Rocha, R. Riserio, A. A. F. Loureiro, J. N. Coelho Jr., and H. C. Wong. *WISENEP: A Network Processor for Wireless Sensor Networks*. 11th IEEE Symposium on Computers and Communications (ISCC'06). June 2006, Pula-Cagliari/Italy (p. 8-14).
9. **L. B. OLIVEIRA**, H. C. Wong, and A. A. F. Loureiro. *LHA-SP: Secure protocols for Hierarchical Wireless Sensor Networks*. 9th IFIP/IEEE International Symposium on Integrated Network Management (IM'05). May 2005/Nice, France (p. 31-44) – Top ranked paper – índice de aceite: 23.5%.
10. Adrian C. Ferreira, Marco A. Vilaca, **L. B. OLIVEIRA**, H. C. Wong and A. A. F. Loureiro. *On the Security of Cluster-Based Communication for Wireless Sensor Networks*. 4th IEEE International Conference on Networking (ICN'05). Lecture Notes in Computer Science, vol. 3420. Springer, April 2005, Reunion Island (p. 449-458).
11. **L. B. OLIVEIRA**, I. G. Siqueira, D. F. Macedo, A. A. F. Loureiro H. C. Wong, and J. M. Nogueira. *Evaluation of Peer-to-Peer Network Content Discovery Techniques over Mobile Ad Hoc Networks*. IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WOWMOM'05). June 2005, Taormina/Italy (p. 51-56).



### 1.3.3 Publicações Nacionais

Abaixo, listamos os trabalhos publicados em conferências nacionais. Note-se que todas elas são patrocinadas pela *Sociedade Brasileira de Computação* (SBC).

1. D. Aranha, D. Camara, J. Lopez, **L. B. OLIVEIRA**, and R. Dahab. *Implementação eficiente de criptografia de curvas elípticas em sensores sem fio*. 8th Brazilian Symposium on Information and Computer System Security (SBSeg'08). September 2008, Gramado/Brazil.
2. **L. B. OLIVEIRA**, F. Daguan, and R. Dahab. *Avaliando Protocolos de Criptografia Baseada em Emparelhamentos em Redes de Sensores Sem Fio*. 7th Brazilian Symposium on Information and Computer System Security (SBSeg'07). August 2007, Rio de Janeiro/Brazil.
3. **L. B. OLIVEIRA**, A. A. F. Loureiro, R. Dahab, and H. C. Wong. *SOS: Sensoriamento Overlay Seguro em Redes de Sensores Sem Fio Hierárquicas*. 6th Brazilian Symposium on Information and Computer System Security (SBSeg'06). August 2006, Santos/Brazil.
4. **L. B. OLIVEIRA**, H. C. Wong, M. Bern, E. Habib, A. A. F. Loureiro, and R. Dahab. *SecLEACH - Uma Solução Segura de Distribuição de Chaves para Redes de Sensores Sem Fio Hierárquicas*. 5th Brazilian Symposium on Information and Computer System Security (SBSeg'05). September 2005, Florianópolis/Brazil.
5. **L. B. OLIVEIRA**, I. G. Siqueira, D. F. Macedo, A. A. F. Loureiro H. C. Wong, and J. M. Nogueira. *Avaliação de Técnicas de Descoberta de Conteúdo em Redes Peer-to-Peer sobre Redes Móveis Ad hoc*. 23rd Brazilian Symposium on Computer Networks (SBRC'05). May 2005, Fortaleza/Brazil (p. 553-564).
6. **L. B. OLIVEIRA**, H. C. Wong, A. A. F. Loureiro, and D. M. Barbosa. *Um Protocolo de Segurança para Redes de Sensores Hierárquicas* 22nd The Brazilian Symposium on Computer Networks (SBRC'04). May 2004, Gramado/Brazil (p. 175-188) – índice de aceite: 25%.

## 1.4 Organização

Nesta seção apresentamos a forma com a qual o restante deste trabalho está organizado. Eis a descrição dos três capítulos subseqüentes, bem como o destaque para suas principais contribuições. Cabe lembrar que o estilo desta tese – “coletânea de artigos” – é uma junção

de artigos já publicados e, segundo normas da Unicamp, tal estilo não permite alterações no texto original dos trabalhos.

### 1.4.1 Capítulo 2 – LHA-SP

O Capítulo 2 discorre sobre uma solução de segurança – incluindo a distribuição de chaves – para RSSFs heterogêneas hierárquicas. No momento da concepção do trabalho, até onde sabemos, não havia propostas de segurança otimizadas para RSSFs hierárquicas especificamente. Por outro lado, já se sabia das vantagens deste tipo de arquitetura organizada em níveis, tanto em termos de latência no envio de dados, como de eficiência energética [69].

Além disso, no momento da concepção do trabalho, as soluções de segurança para RSSFs utilizavam estratégias fim-a-fim ([90], por exemplo), isto é, segurança na camada de rede. Tal estratégia, contudo, impossibilita a fusão de dados, técnica importante de economia de recursos e prolongamento do tempo de vida da rede.

Nossa proposta, batizada de LHA-SP, oferece protocolos para configurar, re-configurar, manter e operar RSSFs hierárquicas e heterogêneas de forma segura. O protocolo também prevê a inserção de outros sensores posteriormente, bem como a re-inclusão daqueles que, seja em função de falhas em sensores intermediários ou devido a obstáculos naturais, se isolaram da rede. Por último, o LHA-SP é adequado para RSSFs hierárquicas com número arbitrário de níveis e capaz de proteger a comunicação na camada de enlace, proporcionando aos sensores então fundir dados.

Mais precisamente, nossa solução impede que intrusos interfiram nas atividades da rede, violem e/ou introduzam mensagens no sistema e que bisbilhotem a comunicação entre sensores legítimos. Ademais, a proposta é altamente distribuída, leva em consideração padrões de interação típicos de RSSFs hierárquicas e tira proveito da heterogeneidade da rede para executar tarefas – por exemplo, atribuindo as atividades que mais demandam a sensores com mais recursos. Enfim, nossa proposta conta apenas com criptossistemas simétricos, o que a torna adequada até mesmo para os sensores mais restritos.

Em nosso modelo, cada sensor interage com um conjunto restrito de sensores durante a fase de configuração. Sensores do nível  $h$  interagem apenas com sensores do nível  $(h - 1)$  e nível  $(h + 1)$ , e uma vez que os grupos são formados, este conjunto é ainda mais reduzido: os sensores passam a interagir apenas com seus respectivos CHs (sensores que os adotaram, do nível  $(h + 1)$ ) e “filhos” (sensores adotados por eles, do nível  $(h - 1)$ ). Ademais, após a fase inicial de configuração, o conjunto de sensores com o qual um outro sensor interage se modificará apenas caso seu CH “morra” e/ou seus filhos sejam adotados por novos CHs.

O conteúdo do capítulo em questão resultou nos seguintes trabalhos publicados [87, 86, 88]. A seguir, resumizamos nossas principais contribuições. São elas:

1. a primeira proposta de segurança para RSSFs Hierárquicas com número arbitrário de níveis;
2. um conjunto de protocolos para proteger a configuração, operação e manutenção da rede;
3. e solução distribuída, que tira proveito de sensores de maior capacidade e adota apenas primitivas simétricas de criptografia.

### 1.4.2 Capítulo 3 – SecLEACH

O Capítulo 3 apresenta uma solução de distribuição de chaves e protocolos de segurança para proteger a comunicação em RSSFs hierárquicas de dois níveis. Para diferenciar de redes hierárquicas com um número arbitrário de níveis, como foca a nossa outra solução LHA-SP (Capítulo 2), referir-nos-emos a tais redes como redes de agrupamentos (*cluster-based networks*), como são também conhecidas.

Organizações baseadas em agrupamentos foram propostas para redes ad hoc em geral, incluindo RSSFs. Em redes de agrupamentos, os sensores são geralmente agrupados em torno de CHs. Esses CHs, por sua vez, atuam como intermediários entre os sensores comuns e as ERBs. Tal arquitetura de rede foi originalmente – e essencialmente – proposta por questões de escalabilidade e eficiência de energia na execução de fusão de dados.

LEACH (*low-energy adaptive clustering hierarchy*) [43] é um exemplo clássico de protocolo para este tipo de rede. Ele considera que todos os sensores podem ajustar seu rádio de forma a alcançar diretamente a ERB. Contudo, para poupar energia, os sensores primeiro enviam os dados coletados para os CHs, os quais os agregam e só então os reencaminham para a ERB. A fim de impedir a drenagem de recursos de um determinado grupo de sensores, o LEACH, de tempos em tempos, renova seu conjunto de CHs. Para isso, ele escolhe aleatoriamente outros sensores da rede para assumir tal função.

Repare que o revezamento de CHs, como ocorre no LEACH, também é interessante do ponto de vista de segurança. Veja, o papel de CH – o qual equivale ao de um roteador e que, portanto, deixa o sensor mais sujeito a ataques – é exercido de maneira interina. Segundo, o papel é revezado dentre todos os demais sensores, isto é, de tempos em tempos uma fração da rede se torna CH. Além disso, a escolha dos sensores que funcionarão como CHs é feita de forma aleatória, de forma que adversários não conseguem prever quais serão os futuros CHs. Tudo isso dificulta o comprometimento dos CHs por parte dos adversários [57] (cabe lembrar, também, que os CHs ao repassarem os dados diretamente à ERB evitam um ataque óbvio: o de impedir que quaisquer dados autênticos cheguem a ERB através do comprometimento dos sensores ao redor da mesma).

Embora a natureza dinâmica seja inerentemente mais segura, ela, paradoxalmente,

dificulta a elaboração de propostas de distribuição de chaves. Os rearranjos periódicos (e aleatórios) dos agrupamentos são incompatíveis com a grande maioria das técnicas de distribuição de chaves. Em outras palavras, as soluções convencionais, quando propostas, não previam relações de confiança efêmeras como as que ocorrem no LEACH. Pelo contrário, elas presumem padrões de comunicação muito pouco flexíveis.

Neste trabalho, oferecemos uma solução para proteger, eficientemente, a comunicação em protocolos iguais ou similares ao LEACH. Para tal, propusemos SecLEACH, uma versão modificada do protocolo que se utiliza da pré-distribuição aleatória de chaves para torná-lo seguro. Ademais, apresentamos uma análise detalhada do nosso esquema, evidenciando como os diversos parâmetros impactam o compromisso entre custo e segurança. Até onde sabemos, SecLEACH foi o primeiro trabalho que investiga a pré-distribuição aleatória de chaves em RSSFs baseadas em agrupamentos. As principais contribuições do trabalho são:

1. a primeira solução de segurança para proteger RSSFs com formação dinâmica de agrupamentos e rotativa de CHs;
2. demonstrar como a pré-distribuição de chaves aleatórias pode ser empregada para proteger RSSFs com formação dinâmica de agrupamentos.

Na verdade, houve muitos estudos sobre a pré-distribuição aleatória de chaves em RSSFs [51], porém sempre no contexto de redes planas. Consequentemente, tais estudos não levam em consideração os padrões de comunicação das redes de agrupamentos e, por sua vez, não lhes são apropriados.

### 1.4.3 Capítulo 4 – TinyPBC

A Criptografia Baseada em Emparelhamentos (*Pairing-Based Cryptography* – PBC) [98, 53, 70] é uma nova tecnologia que vem despertando enorme interesse da comunidade internacional de Criptografia, pois propicia projetos de esquemas criptográficos originais, além de tornar protocolos já conhecidos mais elegantes e eficientes. Não obstante, por meio da PBC, problemas antes em aberto puderam ser resolvidos elegantemente. Talvez a mais fascinante de suas aplicações seja a Cifração Baseada em Identidade (*Identity-Based Encryption* – IBE) [11, 19], a qual por sua vez possibilitou por completo esquemas de Criptografia Baseada em Identidade (*Identity-Based Cryptography* – IBC) <sup>2</sup> [104].

IBC foi originalmente proposta por Shamir [104], mas só se tornou viável com o advento de PBC. Diferentemente das demais propostas de PKC, em que PKIs e verificação de

---

<sup>2</sup>Note-se que atualmente existem também outras formas de se construir um esquema de IBE.

certificados são requeridos, em IBC chaves públicas são derivadas de informações conhecidas (públicas) que univocamente identificam o usuário (seu endereço de correio eletrônico ou o IP da máquina onde trabalha, ou mesmo seu CPF ou RG, por exemplo) e, por conseguinte, dispensam mecanismos de autenticação. Grosso modo, as chaves são “auto-autenticáveis”.

Alguém pode então se perguntar por que IBC ainda não é amplamente utilizada em sistemas de segurança. Bem, além do tempo usual que novas tecnologias levam para ser de fato adotadas, isso se deve a algumas inconveniências da IBC. Particularmente, IBC requer uma Autoridade de Confiança (*Trusted Authority* – TA) a qual é responsável por gerar e manter a custódia das chaves privadas do sistema. Ou seja, ela é capaz de personificar qualquer usuário. Por esta razão, uma TA tem que ser uma entidade de inteira confiança de todos os usuários do sistema. O problema é que na maioria dos sistemas computacionais, infelizmente, não existem elementos com tamanho grau de confiança.

Em RSSFs, entretanto, isso não é um problema. O “dono” (*deployer*) da rede – aquele que carrega o *software* nos sensores, os dispõe em áreas de interesse e analisa os dados coletados – é, obviamente, de confiança. No mundo das RSSFs, esse papel de dono é protagonizado por uma ERB. As ERBs, como já mencionado anteriormente (Seção 1.1.3), são dispositivos dotados tanto de alto poder computacional, como de proteção física. Em outras palavras, elas são ideais para o papel de TAs.

Outra exigência da IBC é que chaves devem ser entregues aos usuários através de canais confidenciais e autenticados. No entanto, se o mecanismo de criptografia estiver sendo usado para alavancar (*bootstrap*) o esquema de segurança – o que usualmente é o caso – tais canais ainda não existem. Mas, novamente, isso não chega a ser um problema em RSSFs. Em seu modelo de segurança, existe claramente um período de tempo – isto é, antes da disposição (*deployment*) – em que há, sim, canais seguros entre sensores e as ERBs. Portanto, além do *software* da aplicação, chaves privadas podem ser carregadas nos sensores antes dos mesmos serem dispostos.

A despeito de todas as suas vantagens, IBC é um sistema assimétrico e, portanto, ordens de magnitude mais complexo computacionalmente que criptossistemas simétricos. Consequentemente, suas operações devem ser otimizadas ao máximo a fim de serem eficientemente calculadas em sensores. A operação mais custosa em IBC é o cálculo de emparelhamentos bilineares (ou emparelhamentos, apenas). Formalmente, emparelhamentos são definidos da seguinte maneira. Seja  $n$  um inteiro positivo. Sejam  $\mathbb{G}_1$  e  $\mathbb{G}_2$  grupos aditivos de ordem  $n$  com identidade  $\mathcal{O}$ , e seja  $\mathbb{G}_T$  um grupo multiplicativo de ordem  $n$  com identidade 1. Um *emparelhamento bilinear* é uma função  $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ , computável, não-degenerativa, cuja propriedade mais importante para criptografia é a bilinearidade, dada por

$$\forall P \in \mathbb{G}_1, \forall Q \in \mathbb{G}_2 \text{ e } \forall a, b \in \mathbb{Z}_n^*, \text{ temos que } e([a]P, [b]Q) = e(P, Q)^{ab}.$$

**Algoritmo 1** Cálculo do emparelhamento de Tate

---

**Entrada:**  $P \in E(\mathbb{F}_{q^k})[\ell]$ ,  $Q \in E(\mathbb{F}_{q^k})[\ell]$   
**Saída:**  $\hat{e}(P, Q)$

1.  $T \leftarrow P$
2.  $f \leftarrow 1$
3. **Para**  $i \leftarrow \lfloor \log(\ell) \rfloor - 1$  **até** 0 **faça**
4.     Calcule as linhas tangente  $l$  e vertical  $v$  para  $[2]T$
5.      $T \leftarrow [2]T$
6.      $f \leftarrow f^2 \cdot \frac{l(Q)}{v(Q)}$
7.     **Se** o  $i$ -ésimo *bit* de  $\ell$  é igual a um, **então:**
8.         Calcule as linhas  $l$  e  $v$  para  $T + P$
9.          $T \leftarrow T + P$
10.         $f \leftarrow f \cdot \frac{l(Q)}{v(Q)}$
11. **Fim**
12. **Retorne**  $f^{(q^k-1)/\ell}$

---

Figura 1.1: Cálculo do emparelhamento de Tate usando o Algoritmo de Miller.

Na prática, os grupos  $\mathbb{G}_1$  e  $\mathbb{G}_2$  são implementados usando subgrupos de pontos de certas curvas elípticas e o grupo  $\mathbb{G}_T$  é implementado usando um grupo finito multiplicativo. Acerca do emparelhamento, atualmente o mais utilizado em criptosistemas é o de Tate. Tal emparelhamento pode ser calculado usando o algoritmo de Miller [75] (Fig. 1.1). No algoritmo,  $E/\mathbb{F}_q$  é uma curva elíptica sobre o corpo finito  $\mathbb{F}_q$  e  $E(\mathbb{F}_q)$  o grupo de pontos desta curva. O algoritmo retorna um *coset* ao invés de um único valor. Para se obter um valor único – como requerido na maioria das aplicações criptográficas –  $f$  é então elevado a  $(q^k - 1)/\ell$  (Fig. 1.1, passo 12).

No Capítulo 4 desta tese, primeiro explicamos como e por que esquemas de IBC poderiam ser empregados para alavancar a segurança em RSSFs. Em seguida, apresentamos a TinyPBC – até onde sabemos a mais eficiente implementação de primitivas de PBC para um processador de 8 bits – e medidas de desempenho da mesma quando executada no microcontrolador ATmega128L (presente nos sensores MICA2 e MICAz). A TinyPBC é baseada na biblioteca de código aberto MIRACL (*Multiprecision Integer and Rational Arithmetic C/C++ Library* [100]), é capaz de computar emparelhamentos (o  $\eta_T$  [3], no caso) a mais cara das operações de PBC, em menos 5.5s e foi disponibilizada na *Web*. Resumindo, nossas principais contribuições neste trabalho foram as seguintes:

1. demonstrar como sensores podem estabelecer chaves par-a-par de maneira eficiente, autenticada e não interativa;

2. demonstrar a viabilidade do cálculo de emparelhamentos em um sensor com extrema escassez de recursos.

Enfim, este trabalho resultou nas seguintes publicações [81, 80, 60, 106, 84].

## Chapter 2

# On the Design of Secure Protocols for Hierarchical Sensor Networks

### 2.1 Introduction

Wireless sensor networks (WSNs) are ad hoc networks comprised mainly of small sensor nodes with limited resources and one or more base stations (BSs), which are much more powerful laptop-class nodes that connect the sensor nodes to the rest of the world [34, 94]. They are used for monitoring purposes, providing information about the area being monitored to the rest of the system. Application areas range from battlefield reconnaissance and emergency rescue operations to surveillance and environmental protection.

Like any wireless ad hoc network, WSNs are vulnerable to attacks [57, 111]. Besides the well-known vulnerabilities due to wireless communication and ad hocness, WSNs face additional problems. For instance, sensor nodes are small, cheap devices that are unlikely to be made tamper-resistant or tamper-proof. Also, they are often deployed in unprotected, or even hostile areas, which makes them more vulnerable to attacks. It is therefore crucial to add security to these networks, specially those that are part of mission-critical applications.

WSNs may be organized in a variety of different ways, and a solution designed for a flat network will unlikely be optimal for a clustered network. (In Section 2.2, we briefly survey different sensor network organizations.) To be effective and efficient, a solution needs to be tailored to the particular network organization at hand.

In this paper we present LHA-SP, a suite of secure protocols (setup, operation, and maintenance) for *heterogeneous hierarchical* sensor networks with *arbitrary number of levels*.

We chose to target this class of networks because it has been shown [69] that, when compared to flat networks, they present a number of advantages including increased sys-



tem throughput and decreased system delay, and increased energy savings as the number of hierarchy levels in the network is increased.

Our solution considers networks consisting largely of highly resource-constrained nodes, and uses exclusively symmetric key mechanisms. Lightweight group key based mechanisms are used whenever possible; more expensive, BS-mediated schemes are used whenever necessary. Our solution prevents intruders from taking part in network activities, tampering with or injecting messages into the network, as well as eavesdropping on communication between legitimate nodes. It is highly distributed and takes into account node interaction patterns specific to clustered WSNs. To our knowledge, LHA-SP is the first work focusing on securing heterogeneous hierarchical WSNs with arbitrary number of levels.

This paper is organized as follows. In Section 2.2, we briefly survey existing organizations for hierarchical WSNs, and discuss their vulnerabilities and needed security. In Section 2.3, we present our network model. In Section 2.4, we present our solution. We evaluate our solution from the security point of view in Section 2.5, and from the performance point of view in Section 2.6. Finally, we discuss related work in Section 2.7, and conclude in Section 2.8.

## 2.2 Hierarchical WSNs

### 2.2.1 Organization

WSNs may be organized in different ways. In *flat* WSNs [2], all nodes play similar roles in sensing, data processing, and routing. In *hierarchical* WSNs [34], on the other hand, the network is typically organized into clusters, with ordinary cluster members and the cluster heads (CHs) playing different roles. While ordinary cluster members are responsible for sensing, the CHs are responsible for additional tasks such as collecting and processing the sensing data from their cluster members, and forwarding the results towards the BS.

Hierarchical networks can differ among themselves in various ways. They can be *homogeneous*, when all nodes except the BSs have comparable capabilities; or *heterogeneous*, when some nodes (typically the CHs) are more powerful than others. In two-level networks, CHs are found in the top level, and their children (those that belong to the cluster headed by a CH) in the lower level. In  $H$ -level networks ( $H > 2$ ), there is a hierarchy of  $H$  nested levels, where CHs in one level are themselves children of nodes that are one level up [6]. CHs can be randomly chosen among the ordinary nodes of a homogeneous network (as in LEACH [43]), or they can be more powerful nodes that compose a heterogeneous network [72]. Clustering can also differ from one network to another. For example, under a  $k$ -hop clustering [35], members of a cluster are all within  $k$ -hops of each other. Alternatively, a subset of nodes can probabilistically self-select CHs, and the remaining nodes

cluster around the CH that is geographically the closest [43].

### 2.2.2 Security

Like any WSN, hierarchical WSNs are vulnerable to a number of attacks [57, 111] including jamming, spoofing, and replay. In these networks, attacks involving CHs are particularly damaging, because CHs are responsible for critical functions such as data aggregation and routing. If an adversary manages to become a CH, it can stage attacks such as sinkhole [57] and selective forwarding [68], thus disrupting potentially large fractions of the network.

Adversaries may leave the routing alone, and try to inject bogus sensor data into the network. Or they may choose to simply eavesdrop on communication between legitimate nodes, obtaining information that is being gathered by the BSs.

At a high level, the main security goals in a hierarchical WSN are: 1) access control, i.e., allow only legitimate nodes to take part in the network (e.g., become CHs and join a cluster); 2) guarantee the authenticity, confidentiality, integrity and freshness of data being passed from one member of the network to another; and 3) guarantee availability (minimize the impact of attempts of DoS attacks). In this work, we design our solution to meet these goals while enabling data aggregation at intermediate points as sensor reports are sent from a sensor node to a BS;

## 2.3 Our Model

We assume heterogeneous networks with two broad classes of nodes. The first class consists of a large number of highly resource-constrained sensing nodes. The second class consists of a smaller number of non-sensing nodes with various levels of resources (e.g., CPU, transmission range, and energy) responsible for data aggregation and routing.

Each node is statically assigned a hierarchy level prior to deployment (based, e.g., on its resource level), with ordinary sensor nodes being assigned level 1. We assume that nodes are deployed with some care, in such a way that level- $h$  nodes always have level- $(h + 1)$  nodes within their communication range. Assuming that level- $(h + 1)$  nodes are always more powerful than level- $h$  nodes, if a level- $(h + 1)$  node  $A$  is within a level- $h$  node  $B$ 's radio range, then  $B$  is within  $A$ 's range.

We use the hierarchy level for clustering. Nodes in one level seek to cluster around the nodes in the next level up in such a way that the network has, at the end of the clustering process, nested clusters where level- $h$  nodes are CHs for level- $(h - 1)$  nodes and children of level- $(h + 1)$  nodes.

Communication can then be single hop within a cluster, with the children of a cluster communicating directly with its CH. Communication with the BS is multi-hop: a message goes from a node to its CH successively until it reaches the BS. The BS can, however, communicate directly with any member of the network.

A node does not move once deployed, but can become unavailable (e.g., by energy exhaustion). When this happens, its children will try to join another cluster.

This network organization is rather static: a node's hierarchy/resource level determines whether it will be a CH, and the clustering structure formed at the initial setup of the network does not change unless a CH becomes unavailable. Nonetheless, we believe it is a reasonable starting point for investigating security in heterogeneous hierarchical WSNs.

We assume clock-driven networks: sensing reports are sent to one's CH at regular intervals. At each CH, the reports are aggregated, and only the result is passed up. Nodes have local clocks to keep track of elapsed time, for the purposes of evaluating freshness of keys and timing out on certain events. Local clocks do not need to be synchronized.

Attacks to WSNs may come from *outsiders* (those that are not legitimate members of the network) or *insiders* (those that are legitimate members of the network). The solution we propose here is meant to protect the network from attacks by outsiders only. In our model, keys can be compromised through cryptanalysis or node tampering. We assume that an attacker using either approach will succeed only after a non-negligible amount of time  $t$ , and the network can be considered secure for  $t$  units of time after deployment. We assume that BSs are trusted.

## 2.4 LHA-SP

In this section we present LHA-SP, a suite of secure protocols for hierarchical ad hoc WSNs as modeled in Section 2.3. Our goal is to address the problems discussed in Section 2.2 (access control; authenticity, confidentiality, integrity and freshness of communications; and availability). We first give an overview of our solution (Section 2.4.1), then the protocol details (Section 2.4.2), and finally the protocol implementation (Section 2.4.3).

### 2.4.1 Overview

One of the first concerns in setting up a WSN is to allow only legitimate nodes to participate in the network. To implement this access control, various cryptographic solutions (e.g., [90, 118, 10]) have been proposed. None of them is optimized for the type of networks we consider, mainly because of their key distribution schemes. In our model, each node interacts with a restricted set of nodes during the initial setup. Level- $h$  nodes inter-

act only with level- $(h - 1)$  and level- $(h + 1)$  nodes, and once the clusters are formed, this set is further reduced: a node interacts only with its CH and children. In addition, after the initial configuration, the set of nodes that a given node interacts with will change only when a CH dies and its children seek new CHs. Thus we need keys that allow legitimate nodes to recognize those that are one level up and one level down, as well as keys to protect their communications with their CH and children. Next, we first show why existing key distribution schemes do not adequately solve our problem, and then sketch our solution.

Given that public key mechanisms are inapplicable to WSNs (because of sensor nodes' resource constraints), most existing solutions rely on mechanisms that predistribute symmetric keys. There are basically three general approaches to predistributing the keys: 1) pairwise key sharing between the BS and each of the remaining nodes (e.g., [90]); 2) pairwise key sharing between ordinary nodes, which can be complete (e.g., [14]) or random (e.g., [33]); and 3) a global group keying (e.g., [5]).

In the first approach, the BS works as the key distribution center (KDC). This is rather costly in terms of communication, given that all nodes need to contact the BS to obtain keys they need to share with their CHs and children. In addition, the BS is a bottleneck.

In the second approach, two nodes that share a key at deployment have a secure link between them; those that do not, can use these links to set up their own secure links. This approach is completely distributed, and does not suffer from high communication costs or from having a bottleneck. However, to give a key to each CH-child link, each node would need to be preloaded with a large number of keys (most of them unnecessary), which is quite wasteful in the type of networks we assume.

In the third approach, everyone in the network or in the vicinity shares the same key. This is the best in terms of cost. Each node only stores one or just a few keys, and no additional keys need to be generated or exchanged. However, when a node is compromised, all links secured by the key stored on it are compromised as well.

In this work, we use a hybrid approach. Prior to deployment, each node is preloaded with: an adoption key, a ring of clustering keys, and a pairwise key it shares with the BS only.

The adoption and clustering keys are both group keys used for setting up the network. They are so named because the former is used to adopt nodes, while the latter to cluster around CHs. By using them, nodes in the network organize themselves into clusters and exchange pairwise keys for securing the links between a node and its CH, needed for later network operation. Once the network is set up, the adoption and clustering keys become invalid and are erased from node memory. The other key – shared between the node and the BS – will be used for orphan adoption, which we explain later.

The pairwise CH-Children keys enable hop-by-hop authentication, allowing data aggregation at the CHs. They also increase the network's resilience against attacks, (avoiding a wholesale compromise of the network if a node ever gets compromised).

Sometimes a network needs additional nodes. We handle addition of nodes the way we handle the initial setup, but using new adoption and clustering keys, which are preloaded to the new nodes, as well as propagated to all level- $(h + 1)$  nodes ( $h$  is the hierarchy level of the new nodes).

When a node becomes orphan, the only trust association between it and the network is the key it shares with the BS. We use this key to get an orphan node back in the network. This key will not only allow the orphan to join a new cluster, but also obtain a shared key between it and its new CH. Note that even though the rejoining process depends on the BS, we assume that only few nodes will become orphans each time, and there will not be resource contention at the BS.

### Notation

In the protocol specifications below, we use single capital letters (e.g.,  $A$ ,  $B$ ) to denote network nodes; calligraphic capital letters (e.g.,  $\mathcal{G}$ ) to denote sets in general;  $|$  to denote concatenation;  $\{m\}_k$  to denote “encryption of  $m$  using key  $k$ ”; and  $\text{MAC}(k, m)$  to denote “message authentication code (MAC) of  $m$  using key  $k$ ”.  $A \rightarrow B : m$  denotes “ $A$  sends message  $m$  to  $B$  in single hop”;  $A \rightarrow\rightarrow B : m$  denotes “ $A$  sends message  $m$  to  $B$  in multiple hops”; and  $A \Rightarrow \mathcal{G} : m$  denotes “ $A$  broadcasts message  $m$  to group  $\mathcal{G}$  in a single hop”.

## 2.4.2 Protocol Description

### Key Predistribution

In our scheme, nodes are preloaded with the following information prior to deployment: the node's id, the node's hierarchy level, an adoption key, a ring of clustering keys, a key it shares with the BS, and the current time. CHs also have information about how many keys will be used within a cluster. We explain the need for this parameter later, when we discuss security levels vs. key scopes.

The distribution of the adoption and clustering keys is carried in such a way that, at the end of the protocol, level- $h$  nodes' ring is the set of all level- $(h + 1)$  adoption keys. Below, we describe this procedure.

1. For each level- $h$  a distinct ring  $r_h$  of distinct clustering keys is computed.
2. The same ring  $r_h$  is then assigned to all level- $h$  nodes.

3. For each level- $(h+1)$  node, the adoption key is chosen by picking at random a single key from  $r_h$ .

This allows nodes to employ the key they share to authenticate themselves during the adoption procedure. It is worth noting that level- $h$  node memory may not be enough to store a ring as large as the number of level- $(h+1)$  nodes. In this case, the same adoption key will be used for more than one CH. Actually, as we will see later in Section 2.5, the size of the rings will dictate the security of the network setup.

### Network Setup

The setup phase in LHA-SP consists of clustering and key distribution. They take place in multiple stages, in a top-down fashion. First, level- $(H-1)$  nodes cluster around level- $H$  nodes ( $H$  is the highest hierarchy level of any node in the network), and keys that will be pairwise shared between a level- $(H-1)$  node and its level- $H$  CH are generated and distributed. Then the same protocol is carried out between level- $(H-2)$  and level- $(H-1)$  nodes, and successively, until level-1 nodes are clustered around level-2 nodes and the keys for communication between them are set. We describe the protocol executed at each of these stages (Fig. 2.1) below.

At each stage, level- $h$  nodes broadcast a **adoption-ad** message looking for level- $(h-1)$  nodes within their radio range (Step 1).

Besides the identity of the broadcasting node, this message includes the hierarchy level, and a MAC along with the identity of the adoption key used to produce the MAC. Therefore, those nodes in one level down know 1) who broadcast the message, 2) that they are the intended recipients, and 3) the key to be used to check the MAC.

Level- $(h-1)$  nodes collect multiple advertisements, and use some criterion to choose their CHs. For example, they could choose the source of the strongest signal in a period of time. Once they choose a CH, they send an **adoption-req** message to the chosen node (Step 2). This message includes both the ids of the requesting node and of the chosen CH, and is protected with the adoption key of the chosen CH.

Upon receiving an adoption request from a node, the CH generates a symmetric key, and sends it back to the node in a **send-key** message (Step 3).

Note that all these message includes a MAC produced using adoption/clustering keys. At each step, a node checks the MAC of the received message. The nodes proceed with the protocol only when the check is successful.

The adoption and clustering keys have a preset validity period (as determined by each sensor's local clock) after which they expire and are discarded by each of the nodes. Thus, the setup protocol should be completed before the key expire.

Adoption being broadcast by level- $h$  nodes (e.g.  $A_h, B_h, C_h$ ):

1.  $A_h \Rightarrow \mathcal{G}_{h-1} : \text{adoption-ad}, h, id_A, id_{k_i}, \text{MAC}(k_i, id_{k_i} \mid h \mid id_A)$
- $B_h \Rightarrow \mathcal{G}_{h-1} : \text{adoption-ad}, h, id_B, id_{k_j}, \text{MAC}(k_j, id_{k_j} \mid h \mid id_B)$
- $C_h \Rightarrow \mathcal{G}_{h-1} : \text{adoption-ad}, h, id_C, id_{k_j}, \text{MAC}(k_j, id_{k_j} \mid h \mid id_C)$
- ...

Nodes from  $\mathcal{G}_{h-1}$  (e.g.,  $M_{h-1}, N_{h-1}, O_{h-1}, P_{h-1}$ ) choose their CHs (e.g.,  $B_h, A_h, C_h$ ) and respond:

2.  $M_{h-1} \rightarrow B_h : \text{adoption-req}, id_M, id_B, \text{MAC}(k_j, id_M \mid id_B)$
- $N_{h-1} \rightarrow A_h : \text{adoption-req}, id_N, id_A, \text{MAC}(k_i, id_N \mid id_A)$
- $O_{h-1} \rightarrow C_h : \text{adoption-req}, id_O, id_C, \text{MAC}(k_j, id_O \mid id_C)$
- $P_{h-1} \rightarrow A_h : \text{adoption-req}, id_P, id_A, \text{MAC}(k_i, id_P \mid id_A)$
- ...

Level- $h$  nodes (e.g.,  $A_h$ ) generate and distribute pairwise keys to be shared with each of their children (e.g.  $N_{h-1}, P_{h-1}$ ):

3.  $A_h \rightarrow N_{h-1} : \text{send-key}, id_A, id_N, \{k_{A,N}\}_{k_i}, \text{MAC}(k_i, id_A \mid id_N \mid \{k_{A,N}\}_{k_i})$
- $A_h \rightarrow P_{h-1} : \text{send-key}, id_A, id_P, \{k_{A,P}\}_{k_i}, \text{MAC}(k_i, id_A \mid id_P \mid \{k_{A,P}\}_{k_i})$
- ...

The various symbols denote:

- $X_h$  : a node  $X$  from level  $h$
- $\mathcal{G}_h$  : the group of all nodes from level  $h$
- $h$  : hierarchy level
- $id_X$  : id of node/key  $X$
- $k_i, k_j$  : adoption keys
- $k_{X,Y}$  : pairwise key shared between nodes  $X$  and  $Y$

Figure 2.1: The setup protocol.

At the end of this protocol (after all  $H - 1$  stages have been executed), each node will have acquired  $c + 1$  pairwise keys: one shared with its CH, and the remaining  $c$  shared between it and each of its children.

## Network Operation

Once the network is set up and the normal operation begins, there will be two types of communications: child-CH communications, which consist mainly of sensing reports and CH-children communications, which consist mainly of network management messages.

In child-CH communications, a child  $A_h$  simply produces a MAC and encrypts the message  $m_A$  with the key it shares with the CH  $D_{h+1}$ . For freshness, a nonce  $n_A$  can be

added before encryption.

$$A_h \rightarrow D_{h+1} : n_A, \{m_A\}_{k_{A,D}}, \text{MAC}(k_{A,D}, n_A \mid \{m_A\}_{k_{A,D}})$$

At each hop, the CH can check MACs and decrypt messages it received. Thus, the CH examines their content and performs data aggregation before sending the aggregate result forward.

Information that flows the opposite direction, i.e., from the BS to the rest of the WSN, can be destined to a particular node or a subset of the nodes. If the information is destined to a single node, our pairwise keying scheme is completely adequate. In cases where the information is destined to a larger number of nodes, it can be distributed, multi-hop, through the intermediate CHs. Note that whenever a CH needs to send the same information to several of its children, the best mechanism would be an authenticated broadcast, which cannot be done with our CH-children pairwise keying. However, we can use the pairwise keys to bootstrap the scheme proposed by LEAP [118], in which broadcasts are authenticated using keys in a hash key chain. Alternately, we can group all the children in a cluster in a few groups, and have each group share a key. E.g., given a cluster with 10 children, there will be 10 keys if we use pairwise keys between the CH and each of its child. There will be 5 keys if each key is shared between the CH and 2 of its children. And one key if all members of the cluster share the same key. The idea is that, when a CH needs to broadcast a message to multiple nodes in the cluster, it can make fewer transmissions: one for each group that shares a key (instead of one for each child). This scheme thus trades security (the scope of a key) with efficiency. In any case, we expect each CH to have a reasonable small number of children (according to [69], between 4% and 10% of a network must be composed of CHs for maximum energy efficiency). And given that there are typically few network management messages, it is not impractical for the CHs to deliver these messages to each child separately, encrypted with the key they share.

1.  $A_h \Rightarrow \mathcal{G}_{h+1} : \text{new-node-ad}, h$
2.  $B_{h+1} \Rightarrow \mathcal{G}_h : \text{adoption-ad}, (h+1), id_B, id_{k_j}, \text{MAC}(k_j, id_{k_j} \mid (h+1) \mid id_B)$
3.  $A_h \rightarrow B_{h+1} : \text{adoption-req}, id_A, id_B, \text{MAC}(k_j, id_A \mid id_B)$
4.  $B_{h+1} \rightarrow A_h : \text{send-key}, id_B, id_A, \{k_{B,A}\}_{k_j}, \text{MAC}(k_j, id_B \mid id_A \mid \{k_{B,A}\}_{k_j})$

All symbols as previously defined;

Figure 2.2: Node addition protocol.



## Network Maintenance

During the lifetime of a network, nodes come and go: existing nodes may depart from the network (e.g., by energy exhaustion) and new nodes may be added. We handle these changes as follows.

**Adding New Nodes** To securely add new nodes to the network, we follow the general scheme used in the initial deployment. Nodes about to be added are preloaded with the same set of data as in the initial deployment; however, the ring of clustering keys and the clock time will have new values. The ring now is composed of a newly generated set of key (the initial ones have expired), and the time is the current time given by the operator preloading these values. These keys are intended to be the trust association between the nodes being added and those already in the network.

To allow nodes being added to be adopted, a new adoption key and the current time need to be known by all pre-existing level- $(h + 1)$  nodes, where  $h$  the level of nodes being added. Again, for each level- $(h + 1)$  node an adoption key is chosen by picking at random a key from the new ring and the BS can transmit these values using single hop communication to the intended recipients.

Fig. 2.2 shows the node addition protocol. Unlike the initial setup protocol, here new nodes seeking to join the network advertise their intention through a **new-node-ad** message (Step 1), which includes the hierarchy level  $h$  of the node broadcasting the message. Those at level  $h + 1$  that hear this broadcast reply with **adoption-ad**, signaling their intention to adopt. The rest of the protocol is identical to the initial setup protocol (Fig. 2.1).

Just like before, the new group key expires after a predefined period of time, before which all new nodes should have joined the network.

**Orphan adoption** We assume that the network provides means for children of a cluster to learn the unavailability of its CH. This can be achieved, e.g., by periodically pinging the CH, or by using mechanisms such as watchdog [68].

Whenever a CH becomes unavailable, it is desirable for the orphans to join another cluster. Given that the pairwise key shared between an orphan and the BS is the only trust association shared between the orphan and the network, we use the BS as an authentication authority and KDC. The protocol (Fig. 2.3) works as follows.

First, the orphan nodes broadcast the **orphan-ad** message searching for a new CH (Step 1). This message includes the level  $h$  of the orphan. Upon receiving an **orphan-ad** message, candidate CHs (i.e., those one level up) reply with **adoption-ad** (Step 2). Neither message is protected, given that the communicating parties do not share any keys.

Each orphan then chooses one among all those that sent a reply, and responds with **adoption-req** (Step 3). This message is authenticated by a MAC, produced with the key

the orphan shares with the BS. It is not destined to the chosen CH, but will be included in the following (**key-req**) message the CH sends to the BS (step 4).

For the **key-req** message (Step 4), the CH adds its own MAC (produced using the key it shares with the BS) to the MAC from the orphan. It then adds another MAC using the key it shares with its own CH. The former MACs are intended for the BS to verify the originators of the request, whereas the latter offers link level security (and will be replaced at each hop).

After checking the authenticity of both the orphan and the adopting CH, the BS generates a symmetric key and sends it, single hop, to both. The orphan node is now back on the network, and there is a secure communication channel between it and its CH.

Node  $A_h$  being adopted by node  $B_{h+1}$

1.  $A_h \Rightarrow \mathcal{G}_{h+1} :$  orphan-ad,  $h$
2.  $B_{h+1} \rightarrow \mathcal{G}_h :$  adoption-ad,  $(h+1), id_B$
3.  $A_h \rightarrow B_{h+1} :$  adoption-req,  $m, \text{MAC}(k_{A,S}, m)$
4.  $B_{h+1} \rightarrow C_{h+2} :$  key-req,  $m', \text{MAC}(k_{B,S}, m'), \text{MAC}(k_{B,C}, m' \mid \text{MAC}(k_{B,S}, m'))$
5.  $C_{h+2} \rightarrow S :$  key-req,  $m', \text{MAC}(k_{B,S}, m'), \text{MAC}(k_{C,D}, m' \mid \text{MAC}(k_{B,S}, m'))$

BS  $S$  authenticates  $A$  and  $B$ , and generates  $k_{A,B}$

6.  $S \rightarrow A_h :$  key-del,  $id_A, n_A, \{k_{A,B}\}_{k_{A,S}}, \text{MAC}(k_{A,S}, id_B \mid n_A \mid \{k_{A,B}\}_{k_{A,S}})$
7.  $S \rightarrow B_{h+1} :$  key-del,  $id_B, n_B, \{k_{A,B}\}_{k_{B,S}}, \text{MAC}(k_{B,S}, id_A \mid n_B \mid \{k_{A,B}\}_{k_{B,S}})$

Symbols as previously defined, with the following additions:

$$m = id_A \mid id_B \mid n_A$$

$$m' = m \mid \text{MAC}(k_{A,S}, m) \mid n_B$$

$n_X$  : nonce produced by node X

Figure 2.3: Orphan adoption protocol.

### 2.4.3 Protocol Implementation

Given the resource-constraints, the protocols specified above need to have efficient implementations. Thus, cryptographic algorithms need to be chosen not only by their security strength, but also by the amount of resource they consume. In this work, we take advantage of the building blocks from SPINS [90], a suite of lightweight symmetric key based security protocols for highly resource-constrained WSNs. We briefly describe these building blocks below.

To save memory, SPINS implements all cryptographic primitives using one single block cipher; RC5 [97] was chosen because of its small code size and its efficiency. Encryption and decryption in SPINS are stream ciphers obtained from using RC5 in the counter (CTR) mode. Message authentication code (MAC) is implemented using RC5 under the CBC-MAC [24] mode: the target message is encrypted under CBC mode, and the message authentication code is the output from the last stage. The same MAC function is used to generate pseudo-random numbers (e.g., nonces) needed by the security module.  $\text{MAC}(k, c)$  produces a sequence of pseudo-random numbers if the value of  $c$  is incremented after each generation. Following good security practice, SPINS uses different keys for different cryptographic functions, all of them derived from a master key  $\chi$ . The MAC function is also used for this derivation. Using different values of  $p$  in  $\text{MAC}(\chi, p)$ , different computationally secure keys can be derived from the master key. Thus, one can, e.g., derive different keys for encryption and MAC code. Or even different keys for different communication directions; i.e., one key for communications from  $A$  to  $B$ , and another from  $B$  to  $A$ .

We use the building blocks described above to implement our protocols. In case of encryption and decryption, a counter value is actually needed in each operation, as they are implemented by RC5 under CTR mode. Because the counter value determines the one-time pad produced by RC5, and one-time pads should not be used twice for security reasons, all encryptions produced using a given key should use different counter values.

In our proposal, counters are dealt with differently depending on the type of keys used. When pairwise keys are used, as e.g. in child-CH communications, counters are not sent between the parties. Instead, they are kept at both ends of the link, and incremented after each encryption (this is the approach used by SPINS). This is feasible because when pairwise link keys are used, the two communicating parties can keep track of counter values that have been used in conjunction with their link key. (The parties can actually get de-synchronized. But they can either try successive increments, or execute simple synchronization protocols to re-synchronize.) When group keys are used, as e.g. in the setup protocol, counters can no longer be synchronized implicitly as above, because not all nodes will hear all transmissions encrypted using the group key, and therefore, would not know which counter has or has not been used. In these cases, we append the counter value being used to each ciphertext. To prevent different nodes from using the same counter, we assign different non-overlapping ranges of values to each node in the network. Each node is expected to start with the smallest value in its range, and successively use increasing values in successive encryptions.

## 2.5 Security Analysis

### 2.5.1 Network setup

The security of our setup protocol depends on two assumptions: 1) that an adversary will take a certain amount of time to compromise the group key or tamper with a node, and 2) that this amount of time exceeds that required to set up the network.

Under these two assumptions, our protocol guarantees that only the legitimate nodes of the network can become CHs, join a cluster, distribute keys and receive them. This is because all message exchanges in the setup protocol are encrypted with adoption or clustering keys, which are known only by the members of the network.

The pairwise keys generated by level- $h$  nodes and distributed to each of their children are encrypted by an adoption key before they are transmitted. This adoption key is also included in the key ring of clustering keys shared among level- $h - 1$  nodes and thus they could potentially eavesdrop on communications intended to some other node, and learn the value of a pairwise key it should not know. However, according to our assumptions, 1) legitimate members of the network would not eavesdrop (misbehave, in general), unless they have been tampered with; and 2) node tampering would take longer than the network setup time. Thus, at the end of the protocol, every legitimate node would have assured its place in the network topology, and each link would have associated with it a pairwise key, known only by the CH that generated it and the child that is its intended recipient.

Note that it is possible for an adversary to capture all this encrypted traffic for later evaluation, after an adoption key is compromised. Using this approach, the adversary can obtain pairwise keys that were encrypted with this key before being exchanged, and use them for eavesdropping or impersonation. The scope of the compromise is limited to clusters whose CHs have employed the key to establish pairwise keys.

This, in turn, depends on the size of the clustering key ring. E.g., let  $\|r_h\|$  and  $\|h\|$  be the size of the key ring and the number of level- $(h + 1)$  nodes, respectively. If  $\|r_h\|$  is big enough so that to each level- $(h + 1)$  node it was assigned a distinct key, then only one cluster will be compromised. On the other hand, if the  $\|r_h\| \leq \|h\|$ , the number of compromised clusters will be, on average,  $\frac{\|h\|}{\|r_h\|}$ .

### 2.5.2 Network Operation

During the network operation, communication between any node and its CH is secured by the pairwise key they share. This ensures confidentiality and authentication of communication between the two, prevents bogus nodes from tampering with and injecting messages, and allows data aggregation to take place at the CH. Replay of old messages is prevented by the use of nonces (which is actually dispensable, given that each new

encryption is produced with a different counter value).

The adoption and clustering keys expire right after the network setup and are not used thereafter. Thus compromise of a single node has limited scope, and would compromise only the links protected by the keys found in the compromised node.

### 2.5.3 Network Maintenance

#### Adding New Nodes

The node addition protocol follows quite closely the initial setup protocol. Thus, the discussion in Section 2.5.1 applies here. The new adoption and clustering keys, used to bootstrap the operation, are known only to legitimate and interested parties: the ring of clustering keys are preloaded to the nodes being added, and the adoption keys are delivered securely to the relevant CHs by the BS.

#### Orphan Adoption

The goal of our orphan adoption protocol is to re-insert an orphan (and the subtree rooted at it) securely into the network routing topology, and to provide it with a key to communicate with the rest of the network securely .

Our proposal relies on the BS as an authentication authority and KDC. The BS authenticates both the **adoption-req** message (step 3, Fig 2.3) from the orphan and the **key-req** message (step 4, Fig 2.3) from the new CH, before it generates and delivers the requested key. Both the requests and the key delivery are protected by the pairwise keys shared between the BS and the nodes. This means that 1) only requests from legitimate members of the network will be processed; and 2) only the orphan and its new CH will learn the value of the new key, which will be used to secure the communication between them.

Note that because the orphans do not share any trust associations (keys) with the nodes that can potentially adopt them, the messages sent in steps 1 and 2 (Fig. 2.3) are not protected. This is a source of vulnerability. For instance, a bogus node can send a large number of **orphan-ad** messages to the network, and try to trigger a response to each of its messages, with the intent of consuming the resources of some of the nodes in the network. Another possible attack is for an intruder to impersonate a potential adopter, and send an **adoption-ad** message (step 2) in response to **orphan-ad** messages. The intruder can simply quit the protocol here or try to submit a **key-req** message (step 3). In any case, the orphan will be left waiting for a key that will never come, and the adoption process will never be completed. We can address the first attack by limiting the number of **orphan-ad** messages a potential CH will handle per period of time. This is reasonable

because we assume that only a small number nodes will become orphans at the same time. To handle the second attack, an orphan can set a waiting time, and if it does not hear from the BS before this time expires, it will contact another potential adopter.

## 2.6 Performance Evaluation

In this section, we consider the overhead incurred by our protocols, as compared to a stripped down version of the protocols without the security devices. For example, the stripped down version of the setup protocol would consist of steps 1 and 2 (Fig. 2.1) only, and the messages exchanged in these steps would not be encrypted.

We evaluate the overheads in terms of computation, communication, and storage. Analysis of these metrics will reveal other costs (e.g. energy consumption and delay). We focus on the protocols for setup and network operation. In what follows,  $n_h$  denotes the total number of level- $h$  nodes.

### 2.6.1 Communication and Computational Overhead

For the setup protocol (Fig. 2.1), security incurs the following cost:

- Each **adoption-ad** and **adoption-req** message transmission incurs one MAC generations, and  $c$  additional bytes for counter value and MAC. **adoption-ad** message sends are executed once by all the nodes in the network, except those at level 1; and **adoption-req** sends are executed once by all the nodes, except those at the highest level.
- Each **adoption-ad** and **adoption-req** message reception incurs one MAC check operation, and reception of  $c$  additional bytes for counter value and MAC. Each level- $h$  node receives no more than  $n_{h+1}$  **adoption-ad** messages, and the set of all level- $h$  nodes will receive a total of  $n_{h-1}$  **adoption-req** messages.
- **send-key** messages are exchanged only in the secure version of the protocol. Each transmission incurs one key and MAC generations, one encryption, and the message itself. Each reception incurs one decryption and MAC check operations, and the message reception itself. The set of all level- $h$  nodes will send a total of  $n_{h-1}$  such messages, whereas each node in the network (except those at the highest level) will receive only one such message.

Our setup protocol is quite scalable. The number of interactions between a level- $h$  node  $A$  and level- $(h+1)$  nodes is bounded by  $n_{(h+1)}$ ; and that between  $A$  and level- $(h-1)$  nodes is bounded by the number of children in the cluster headed by  $A$ .

In CH-children communication, the overhead incurred by security will depend on the pattern of these communications with regard to the number of children a CH tries to reach each time. In the best case scenario, the CH has a message destined to a single child. The overhead is then simply one MAC generation at the CH, transmission of this MAC, and one MAC check at the child. (No explicit counter value need to be enclosed here for the same reason as in child-CH communication.) In the worst case scenario, the transmission from the CH is intended to reach all the children in the cluster. In such scenarios, our solution would be expensive. Instead of a broadcast, our solution requires that the CH sends a separate (protected) message to each of the children (respectively, group of children), because of our pairwise (respectively, group) keying scheme. CH-Children communications, however, are used for network management functions, and do not occur frequently. Thus, a high cost is likely to be tolerable.

For Child-CH communication, which occurs the most in a WSN, LHA-SP is quite efficient: it incurs one encryption and MAC generation at the sender, the transmission of the MAC itself, and one decryption and MAC check at the receiver. Note that the cryptographic operations we use have been shown [90] to incur a very small overhead. Note also that unlike in the setup protocol, counters for encryption/decryption do not need to be explicitly enclosed in the messages. Instead, they can be kept at both ends of the communication (and incremented after each operation). Finally, the use of a MAC makes unnecessary the use of a Cyclic Redundancy Check (CRC), required in unsecured protocols to detect errors in messages.

## 2.6.2 Storage Overhead

The overhead in terms of space includes code space and RAM space for cryptographic functions and the keys.

To estimate the storage overhead for our network operation protocols (CH-children and child-CH communications), we modified the source code for Surge [21], an application in the TinyOS distribution that allows a node to periodically send data to the BS. We modified it to send and receive (RC5-based) encrypted data, instead of plaintext. We used cryptographic code from TinySec [56].

In Table 2.1, “noSec” refers to the original Surge application, whereas “sec” refers to our modified Surge with encryption. Note that security incurs only 990 bytes in ROM (of which the motes have a total of 128K bytes) and 164 bytes in RAM (of which the motes have a total of 4K bytes). Storage overhead incurred by the encryption function is therefore negligible.

Regarding the keys, depending on the level of security required in the setup, a significant amount of the node memory may be used to keep adoption and clustering keys in

	MICA2DOT		MICA2	
Mode	noSec	sec	noSec	sec
ROM	15252	16242	15070	16060
RAM	1843	2007	1843	2007

Table 2.1: RAM and ROM memory for Motes (in bytes)

this phase. However, as soon as the setup phase ends, these keys will be erased and each level- $h$  node,  $h > 1$ , only needs to keep two pairwise keys, and  $k$  keys shared with its children. In the worst case, when each child share a unique key with its CH,  $k$  is equal to the size of the cluster. Level-1 nodes have not children, and have to keep just two keys. Therefore the storage cost is most of time  $O(k)$  for level- $h$  nodes ( $h > 1$ ), and  $O(1)$  for level-1 nodes.

As a whole, we conclude that LHA-SP is efficient and scales gracefully in terms of computation, communication, and storage costs.

## 2.7 Related Work

WSNs are a subclass of MANETS, and much work (e.g., [116, 12, 50, 13, 109, 47, 115]) has been proposed for securing MANETS in general. These studies are not applicable to WSNs because they assume laptop- or palmtop-level resources, which are orders of magnitude larger than those available in WSNs. Public key based solutions are such an example.

Among the studies specifically targeted to resource-constrained WSNs, some [57, 111] have focused on attacks and vulnerabilities. Wood and Stankovic [111] surveyed a number of denial of service attacks against WSNs, and discussed some possible countermeasures. Karlof and Wagner [57] focused on routing layer attacks, and showed how some of the existing WSN protocols are vulnerable to these attacks.

Of those offering cryptographic solutions, a reasonable number (e.g., [14, 33, 112, 118, 17, 117, 62, 63, 49, 91, 29, 48, 54, 51, 15, 64, 28, 92]) have focused on efficient key management schemes without tying them to a particular network organization. We discussed the trade-offs of different key distribution schemes previously in Section 2.4.1. And recently, the cryptography community in WSNs has been investigating more efficient techniques of public key cryptography. By using Elliptic Curve Cryptography [76, 58], for example, it has been shown (e.g., [41, 67, 7]) that sensor nodes are indeed able to compute public key operations. However, public key authentication in the context of WSNs is still an open problem, as they cannot afford a conventional public key infrastructure.



Perrig *et al.* [90] offered a solution for flat and homogeneous networks. They proposed SPINS, a symmetric key based protocol suite for providing baseline security (confidentiality, authentication, integrity, freshness) and authenticated broadcast. Their solution uses pairwise key sharing between each of the nodes and the BS. When two ordinary nodes need to communicate securely between them, the BS works as a key distribution center.

Hierarchical WSNs have quite particular organization patterns, and one can take them into account to design tailored solutions. Carman *et al.* [14] have suggested using higher powered nodes for key generation and management functions, but did not offer concrete protocols. Kong *et al.* [59] and Bohge and Trappe [10] devised solutions for concrete hierarchical and heterogeneous networks. However, they both assume more powerful nodes, and use public key cryptography. More specifically, the former relies on RSA certificates to guarantee authentication. The amount of computation and space resources required by RSA certificates makes this solution infeasible in our context. In addition, it proposes *end-to-end* transport layer security, which prevents data aggregation at intermediary hops. The latter proposes an authentication framework for a concrete 2-tier network organization, in which a middle tier of more powerful nodes were introduced between the BS and the ordinary sensors to carry out authentication functions. However, except for the lowest tier nodes, all other nodes perform public key operations. More recently, Ferreira *et al.* [36] and Oliveira *et al.* [85] proposed SLEACH and SecLEACH, respectively. These works rely exclusively on symmetric key schemes, but they are only adequate for LEACH-like hierarchical WSNs protocols.

There has also been some work on detecting misbehaving nodes. E.g., Marti *et al.* [68] proposed a watchdog scheme that enables network nodes to detect selective forwarding attacks staged by their next hop neighbors.

Detecting and dealing with bogus data has also been focus of research. Zhu *et al.* [119] proposed an interleaved hop-by-hop authentication scheme to prevent injection of false data into sensor networks. The proposal makes sure that the BS can detect a false report when no more than a certain number  $t$  of nodes are compromised. Yea *et al.* [112] proposed SEF, a statistical en-route filtering mechanism for detecting and dropping bogus reports while being forwarded. It allows both the BS and the en-route nodes to detect false data with a certain probability. Przydatek *et al.* [95] proposed SIA, a framework for secure information aggregation in WSNs which makes use of random sampling strategies for allowing an user to infer about the legitimacy of a value.

Other efforts have focused on more specific types of attacks. Hu *et al.* [46] studied and offer solutions for wormhole attacks, whereas Newsome *et al.* [78] investigated sybil attacks in the context of WSNs. Finally, Deng [23] *et al.* address secure in-network processing, and propose a collection of mechanisms for delegating trust to aggregators that *a priori* are not trusted by common sensors. The mechanisms address both dissemination and

aggregation of data.

## 2.8 Conclusion

In this paper, we proposed a solution for securing heterogeneous hierarchical WSNs with arbitrary number of levels. Our solution provides security for network setup and reconfiguration, as well as for the normal network operation traffic. Our scheme sets up pairwise keys between a CH and each of its children (or group of children) using lightweight group key based mechanisms whenever possible, falling back on more expensive, BS-mediated mechanisms whenever necessary.

Our solution is highly distributed, takes into account node interaction patterns that are specific to clustered WSNs, and enables data aggregation at CHs.

We also evaluated the overhead incurred by our solution. The results showed that the overhead incurred by our protocols in terms of energy consumption ranges from small to tolerable. We conclude that our solution is practical.

# Chapter 3

## SecLEACH - On the Security of Clustered Sensor Network

### 3.1 Introduction

Wireless sensor networks (WSNs) [34, 94] are rapidly emerging as a technology for monitoring different environments of interest and they find applications ranging from battle-field reconnaissance to environmental protection. When embedded in critical applications, WSNs are likely to be attacked [57, 111]. Aside from the well known vulnerabilities due to wireless communication, WSNs lack physical protection and are usually deployed in open, unattended environments, which makes them more vulnerable to attacks. It is therefore crucial to devise security solutions to these networks.

An important issue one needs to tackle when using cryptographic methods to secure a network is key distribution, which has been intensively studied recently (e.g., [14, 33, 17, 118, 62, 63, 49, 117, 91, 29, 51, 48, 54, 15, 64, 28, 16, 92]) in the context of WSNs. It is worth noting, however, that a large number [1] of WSN architectures have been proposed and a key distribution solution that is well suited to one architecture is likely not to be the best for another, as different network architectures exhibit different communication patterns.

*Cluster-based* organization (e.g., [43, 113]) has been proposed for ad hoc networks in general and WSNs in particular. In cluster-based networks, nodes are typically organized into clusters, with cluster heads (CHs) relaying messages from ordinary nodes in the cluster to the base stations (BSs). Clustered WSNs were first proposed for various reasons including scalability and energy efficiency while performing data aggregation. Those with rotating CHs, like LEACH [43], are also interesting in terms of security, as their routers (the CHs), which are more prominent targets for adversaries because of their role in routing, rotate from one node to another periodically, making it harder for an adversary

to identify the routing elements and compromise them [57].

Adding security to LEACH-like protocols is challenging, as its dynamic (at random) and periodic rearranging of the network's clustering (and changing links) makes key distribution solutions that provide long-lasting node-to-node trust relationships (to be sure, provided by most existing solutions) inadequate.

In this paper, we focus on providing efficient security communications in LEACH-like protocols. To this end, we first propose SecLEACH, a modified version of LEACH that applies random key predistribution and  $\mu$ TESLA to provide baseline security. We then give a detailed analysis and performance evaluation of our scheme, and present concrete numbers on how the various parameters impact the trade-offs between cost and security. To our knowledge, SecLEACH is the first solution to secure hierarchical (cluster-based) WSNs with dynamic cluster formation. Our main contributions in this paper are:

1. to have provided an efficient solution for securing communications in LEACH; and
2. to have shown how random key predistribution and  $\mu$ TESLA can be used to secure hierarchical WSNs with dynamic cluster formation.

To be sure, random key predistribution has been studied profusely [51], but always in the context of flat WSNs. Due to this fact, these studies have not taken into consideration communication patterns of hierarchical (cluster-based) networks and thus cannot be applied, as is, to them. To the best of our knowledge, ours is the first that investigates random key predistribution as applied to hierarchical (cluster-based) WSNs with rotating CHs.

The rest of this paper is organized as follows. In Section 3.2, we discuss what is needed to cryptographically secure LEACH's communications and why existing solutions are inadequate. We present our solution (SecLEACH) in Section 3.3, and analyze its performance in Section 3.4. Finally, we discuss related work and conclude in Sections 3.5 and 3.6, respectively.

## 3.2 Adding security to LEACH

WSNs typically comprise of one or more BSs and a larger number of resource-scarce sensor nodes. Sensor nodes do not typically communicate directly with the BS because: 1) they typically have transmitters with limited transmission range, and are unable to reach the BS directly; and 2) even if the BS is within a node's communication range, direct communication typically demands a much higher energy consumption.

Multi-hop flat networks are a more energy efficient alternative, that has a node take advantage of its neighboring nodes as routers: farther away nodes send their messages to

intermediate nodes, which then forward them towards the BS in a multi-hop fashion. The problem with this approach is that, even though peripheral nodes actually save energy, the intermediate nodes spend additional energy receiving and forwarding messages, and end up having a shortened lifetime.

LEACH (Low Energy Adaptive Clustering Hierarchy) [43] was proposed to address the aforementioned problem. It assumes that every node can directly reach a BS by transmitting with high enough power. However, to save energy, sensor nodes send their messages to their CHs, which then aggregate the messages, and send the aggregate to the BS. To prevent energy drainage of a restricted set of CHs, LEACH randomly rotates CHs among all nodes in the network, from time to time, thus distributing aggregation- and routing-related energy consumption among all nodes in the network.

LEACH thus works in rounds. In each round, it uses a distributed algorithm to elect CHs and dynamically cluster the remaining nodes around the CHs. The resulting clustering structure is used by all sensor-BS communications for the remaining of the round.

Using a set of 100 randomly distributed nodes, (and a BS located at 75m from the closest node) simulation results [43] show that LEACH spends up to 8 times less energy than other protocols. To be fair, the energy saving comes from a number of sources other than just dynamic cluster-based communication: data aggregation (CHs aggregate data before sending them to the BS), node sleeping (given that only CHs need to forward messages, the remaining nodes are activated only when they themselves are transmitting, and remain in sleep mode for a reasonable amount of time), and transmitter calibration (nodes calibrate their transmitters' power in such a way that they are only high enough to reach the CH).

### 3.2.1 LEACH: protocol description

Rounds in LEACH (Fig. 3.1) have predetermined duration, and have a *setup* phase and a *steady-state* phase. Through synchronized clocks, nodes know when each round starts and ends.

The setup consists of three steps. In Step 1 (*advertisement* step), nodes decide probabilistically whether or not to become a CH for the current round (based on its remaining energy and a globally known desired percentage of CHs). Those that decide to do so broadcast a message (*adv*) advertising this fact, at a level that can be heard by everyone in the network. To avoid collision, a carrier sense multiple access protocol is used. In Step 2 (*cluster joining* step), the remaining nodes pick a cluster to join based on the largest received signal strength of an *adv* message, and communicate their intention to join by sending a *join\_req* (join request) message. Once the CHs receive all the join requests, Step

Setup phase

1.  $H \Rightarrow \mathcal{G} : id_H, \text{adv}$
2.  $A_i \rightarrow H : id_{A_i}, id_H, \text{join\_req}$
3.  $H \Rightarrow \mathcal{G} : id_H, (\dots, \langle id_{A_i}, t_{A_i} \rangle, \dots), \text{sched}$

Steady-state phase

4.  $A_i \rightarrow H : id_{A_i}, id_H, d_{A_i}$
5.  $H \rightarrow BS : id_H, id_{BS}, \mathcal{F}(\dots, d_{A_i}, \dots)$

The various symbols denote:

$A_i,$	$\text{adv},$
$H,$	$\text{join\_req},$
$BS :$ An ordinary node, a cluster head, and the base station, respectively	$\text{sched} :$ String identifiers for message types
$\mathcal{G} :$ The set of all nodes in the network	$d_X :$ Sensing report from node $X$
$\Rightarrow, \rightarrow :$ Broadcast and unicast transmissions, respectively	$\langle id_X, t_X \rangle :$ Node $X$ 's id and its time slot $t_X$ in its cluster's transmission schedule
$id_X :$ Node $X$ 's id	$\mathcal{F} :$ Data aggregation function

Figure 3.1: LEACH protocol

3 (*confirmation* step) starts with the CHs broadcasting a confirmation message that includes a time slot schedule to be used by their cluster members for communication during the steady-state phase. Given that all transmitters and receivers are calibrated, balanced and geographically distributed clusters should result.

Once the clusters are set up, the network moves on to the steady-state phase, where actual communication between sensor nodes and the BS takes place. Each node knows when it is its turn to transmit (Step 4), according to the time slot schedule. The CHs collect messages from all their cluster members, aggregate these data, and send the result to the BS (Step 5). The steady-state phase consists of multiple reporting cycles, and lasts much longer compared to the setup phase.

### 3.2.2 Security vulnerabilities

Like most routing protocols for WSNs, LEACH is vulnerable to a number of security attacks [57], including jamming, spoofing, replay, etc. However, because it is a cluster-based protocol, relying fundamentally on the CHs for data aggregation and routing, attacks involving CHs are the most damaging. If an intruder manages to become a CH, it can stage attacks such as sinkhole and selective forwarding, thus disrupting the workings of

the network. Of course, the intruder may leave the routing alone, and try to inject bogus sensor data into the network, one way or another. A third type of attack is (passive) eavesdropping.

It is worth noting that LEACH is more robust against attacks than most other routing protocols [57]. In contrast to more conventional multihop schemes where nodes around the BS are especially attractive for compromise (because they concentrate all network-to-BS communication flows), CHs in LEACH communicate directly with the BS, can be anywhere in the network, and change from round to round. All these characteristics make it harder for an adversary to identify and compromise strategically more important nodes.

### 3.2.3 Why existing key distribution schemes are inadequate

One of the first steps to be taken to secure a WSN is to prevent illegitimate nodes from participating in the network. This access control can preserve much of a network's operations, unless legitimate nodes have been compromised. (Note that access control does not solve all security problems in WSNs. E.g., it is ineffective against DoS attacks based on jamming wireless channels, or manipulating a node's surrounding environment to induce the reporting of fabricated conditions.) Access control in networks has typically been implemented using cryptographic mechanisms, which rely critically on key distribution. Key distribution is thus of paramount importance in securing a network.

There are a number of standard key distribution schemes in the security literature [99], most of which are ill-suited to WSNs: public key based distribution, because of its processing requirements; global keying, because of its security vulnerabilities; complete pairwise keying, because of its memory requirements; and those based on a key distribution center, because of its inefficiency and energy consumption [118].

Some key distribution schemes (e.g., [33, 118, 117, 91, 51, 64, 28]) have been specifically designed for WSNs. While they are well-suited for network organizations they were designed for, they are inadequate for other organizations. These schemes typically assume that a node interacts with a quite static set of neighbors and that most of its neighborhood is discovered right after the deployment. However, clusters in LEACH are formed dynamically (at random) and periodically, which changes interactions among the nodes and requires that any node needs to be ready to join any CH at any time.

For instance, Zhu *et al.* [118] (LEAP) and Eschenauer and Gligor's [33] schemes are rather efficient for flat networks where nodes interact with a rather static set of neighbors, but are inadequate for LEACH's periodic rearranging of the network. For example, if LEAP were used to secure communication in LEACH, a new key distribution could be required per round. This not only would be inefficient, but also infeasible, as LEAP relies on a master key to perform key distribution, which is erased from the nodes' memory

as soon as the first key distribution is completed. Similarly, Eschenauer and Gligor's scheme, based on random keys, does not provide mechanisms to authenticate broadcasts from CHs to the rest of the network (Fig. 3.1, Steps 1 and 3). Such authentication is essential to secure the periodic (re)clustering procedure. Eschenauer and Gligor's scheme will be explained in more detail in Section 3.3.2.

In what follows, we discuss the network model assumed in LEACH, and the requirements it sets for key distribution.

### 3.2.4 Key distribution for LEACH: requirements and constraints

Our discussion in Section 3.2.2 shows the need for the nodes to authenticate each other as legitimate members of the network both in the setup interactions and the sensor data reporting communications. Given the communication patterns in LEACH, two different types of authentication are required: authenticated broadcast, for broadcasts from the CHs to the rest of the network (Fig. 3.1, Steps 1 and 3); and pairwise authentication for the remaining (node-to-CH and CH-to-BS) communications.

Symmetric-key authenticated broadcasts for WSNs, both global ( $\mu$ TESLA [90]) and local (LEAP [118]), share the core idea of using a one-way key chain (a sequence of keys  $k_1, \dots, k_n$ , where  $k_{i+1}$  is generated from  $k_i$  by applying a one-way hash function  $f()$ , i.e.,  $k_{i+1} = f(k_i)$ ) to achieve authentication. These schemes cannot be applied, as is, to LEACH because: 1) the key chain would require significant storage space in the broadcasting CHs; and more importantly, 2) all nodes in the network would need to store one key for each node in the network, which is neither practical nor scalable. (Each node needs to store one key for every other node in the network because an ordinary node needs to be able to authenticate the CHs in each round, which can be arbitrary nodes in the network.)

Pairwise authentication is also challenging to implement in LEACH, because of key distribution issues. Effectively, given that any node needs to be ready to join any CH (which could be any node in the network), it would need to have shared pairwise keys with every other node in the network. Just like in authenticated broadcast, this is neither practical, nor scalable.

## 3.3 SecLEACH – Applying random key distribution to LEACH

In this section, we first briefly describe the main ideas behind  $\mu$ TESLA (Section 3.3.1) and random key predistribution schemes (Section 3.3.2), then we show how they can be used to secure LEACH (Section 3.3.3). We note that we use LEACH to be concrete,



but that our proposal should have a wider applicability, and be easily adaptable to other similar protocols.

### 3.3.1 $\mu$ TESLA

$\mu$ TESLA was proposed by Perrig *et al.* in SPINS[90] and provides authenticated broadcast. The protocol implements the asymmetry required for authenticated broadcast using one-way key chains constructed with cryptographically secure hash functions, and delayed key disclosure. In  $\mu$ TESLA each node  $X$  is assigned a group key  $k^n$  that is shared by all members of the network.  $k^n$  is the last key of a sequence  $\mathcal{J}$  generated by applying successively a one-way hash function  $f$  to an initial key  $k^0$  ( $\mathcal{J} = k^0, k^1, k^2, \dots, k^{n-1}, k^n$ , where  $f(k^j) = k^{j+1}$ ). The BS keeps  $\mathcal{J}$  secret, but shares the last element  $k^n$  with the rest of the network. After the deployment, whenever the BS wants to broadcast a message it identifies the last key  $k^j$  in  $\mathcal{J}$  that has not been disclosed, produces a MAC<sup>1</sup> of the message using  $k^j$ , and sends both the message and the MAC.  $k^j$  is disclosed after a certain time period, after all nodes in the network have received the previous message. After receiving both the broadcast and the corresponding key, nodes in the network can authenticate the broadcast from the BS by checking if the key is an element of the key chain generated by the BS, and immediately precedes the one that was released last. That is, if  $f(k^j) = k^{j+1}$ .  $\mu$ TESLA requires loose time synchronization. See SPINS[90] for further details on  $\mu$ TESLA.

### 3.3.2 Random key predistribution schemes

Random key predistribution for WSNs was first proposed by Eschenauer and Gligor [33], and has since been studied by several research groups [51]. In a random key predistribution scheme, each node is assigned a set of keys drawn from a much larger key pool. Different schemes have different assignment algorithms, but they all result in probabilistic key sharing among the nodes in the network.

To bootstrap security using Eschenauer and Gligor’s original scheme [33], a network goes through three phases. In the first phase (*key predistribution*), which takes place prior to network deployment, a large pool of  $S$  keys and their ids are generated. Each node is then assigned a ring of  $m$  keys, drawn from the pool at random, without replacement. In the second phase (*shared-key discovery*), which takes place during network setup, all nodes broadcast the ids of the keys on their key rings. Through these broadcasts, a node finds out with which of their neighbors (as determined by communication range) they share a key. These keys can then be used for establishing secure links between the two

---

<sup>1</sup>Note that MAC is often used to stand for medium access control in networking papers. In this paper, we use MAC to stand for message authentication code.

neighbors. Finally, during *path-key establishment* phase, pairs of neighboring nodes that do not share a key can set up their own keys, as long as they are connected by two or more secure links at the end of shared key discovery.

Because of the way keys are assigned, a key can be found in more than two nodes, and used in multiple communication links. When a node is compromised, all its keys are compromised, and all the links secured by these keys are also compromised.

The initial assignment of key rings to nodes can also be done pseudorandomly [117, 91]. Pseudorandom schemes make both the key predistribution and the shared-key discovery more efficient.

### 3.3.3 SecLEACH: protocol description

In our solution <sup>2</sup>, we propose to generate a large pool of  $S$  keys and their ids prior to network deployment. Each node is then assigned a ring of  $m$  keys drawn from the pool pseudorandomly [117], without replacement, as follows. For each node  $X$ , we use a pseudorandom function (PRF) to generate its unique id  $id_X$ .  $id_X$  is then used to seed a pseudorandom number generator (PRNG) of a large enough period to produce a sequence of  $m$  numbers.  $\mathcal{R}_X$ , the set of key ids assigned to  $X$ , can then be obtained by mapping each number in the sequence to its correspondent value *modulus*  $s$ . Also prior to deployment, each node is assigned a pairwise key shared with the BS; and a group key (the last key of a one-way key chain held by the BS) that is shared by all members of the network.

The LEACH clustering algorithm can then be run with the following modifications. When a self-elected CH broadcasts its **adv** message, it includes information of the keys in its key ring. The broadcast is authenticated leveraging on the BS, who is trusted and has more resources. The remaining nodes now cluster around the closest CH with whom they share a key. Fig. 3.2 shows the details of our SecLEACH protocol.

In Step 1, a self-elected CH  $H$  broadcasts (Steps 1.1) its id  $id_H$ , a nonce, and a MAC produced using the key the CH shares with the BS (which will be used by the BS for the purpose of authentication). The BS waits to hear and authenticate (modified) **adv** messages from all CHs; compiles the list of legitimate CHs; and sends the list to the network using  $\mu$ TESLA (Steps 1.2 and 1.3). Ordinary nodes now know which of the **adv** messages they received are from legitimate nodes, and can proceed with the rest of the original protocol, choosing the CH from the list broadcast by the BS.

In Step 2, ordinary nodes  $A_i$  compute the set of  $H$ 's key ids (using the pseudorandom scheme described above), choose the closest CH with whom they share a key  $k_{[r]}$ , and send it a **join\_req** message, protected by a MAC. The MAC is generated using  $k_{[r]}$ , and

---

<sup>2</sup>It is worth noting that the BS→node communication authentication using  $\mu$ TESLA in the protocol is due to Ferreira *et al.* [36]

includes the nonce from  $H$ 's broadcast in Step 1 (to prevent replay attacks), as well as the id  $r$  of the key chosen to protect this link (so that the receiving CH knows which key to use to verify the MAC).

In Step 3, to conclude the setup phase, the CHs broadcast the time slot schedule to the nodes that chose to join their clusters. This broadcast is authenticated the same way as the previous one (Step 1.1). For clarity of presentation, we do not reproduce the full-blown authenticated version here.

In the steady-state phase, node-to-CH communications (Step 4) are protected using the same key used to protect the `join_req` message in Step 2. A value computed from the nonce (*nonce*) and the reporting cycle ( $l$ ) is also included to prevent replay. The CHs can now check the authenticity of sensing reports they receive, perform data aggregation, and send the aggregate result to the BS (Step 5). The aggregate result is protected using the symmetric key shared between the CH and the BS. For freshness, a counter (shared between the CH and the BS) is included in the MAC value as well. Fig. 3.2 shows only one reporting cycle in the steady-state phase. In practice, there will be multiple cycles in a round. In each round  $l$ , the value of the “freshness token” (denoted by “*nonce*+ $l$ ” in Step 4, and “ $c_H$ ” in Step 5) needs to be incremented by 1.

At the end of the clustering process, we expect that a fraction of the ordinary nodes will be matched with a CH, though not necessarily the one they would have matched with in the basic LEACH, because of key sharing constraints; the remaining would not have any CH to match with. We call these nodes orphans.

There are different ways to deal with the orphans: we can have them sleep for the round; we can add a small protocol that would allow the “already-adopted children” to bring the orphans into their clusters; or we can have them communicate directly with the BS for the round. In any case, the number of orphans will depend on the size of the key pool, the size of the key ring, and the number of CHs, and will have an impact on the performance of the network.

In Section 3.4, we show the cost, efficiency, and security of SecLEACH, as well as the tradeoffs when we vary the various parameter values.

### 3.3.4 Security analysis

SecLEACH provides authenticity, integrity, confidentiality, and freshness to communications. Our solution allows authentication of `adv` messages (steps 1.1, 1.2, and 1.3, Fig. 3.2), and prevents unauthorized nodes from becoming CHs and in turn adopting nodes. The message in Step 2, Fig. 3.2, is protected with a key in the key pool; and a successful check of this message allows  $H$  to conclude that the message originated from a legitimate node in the network. Because the protected message includes the nonce from Step 1,  $H$  can

Setup phase

- 1.1.  $H \Rightarrow \mathcal{G} : id_{A_i}, id_H, \text{mac}_{k_H}(id_H \mid c_H \mid \text{adv})$   
 $A_i : \text{store}(id_H)$   
 $BS : \text{if } \text{mac}_{k_H}(id_H \mid c_H \mid \text{adv}) \text{ is valid,}$   
 $\text{add}(id_H, \mathcal{V})$
- 1.2.  $BS \Rightarrow \mathcal{G} : \mathcal{V}, \text{mac}_{k^j}(\mathcal{V})$
- 1.3.  $BS \Rightarrow \mathcal{G} : k^j$   
 $A_i : \text{if } (f(k^j) = k^{j+1}) \text{ and } (id_H \in \mathcal{V}),$   
 $H \text{ is authentic}$   
 $A_i : \text{choose } r \text{ such that } r \in (\mathcal{R}_H \cap \mathcal{R}_{A_i})$
2.  $A_i \rightarrow H : id_{A_i}, id_H, r, \text{join\_req}, \text{mac}_{k_{[r]}}(id_{A_i} \mid id_H \mid r \mid \text{nonce})$
3.  $H \Rightarrow \mathcal{G} : id_H, (\dots, \langle id_{A_i}, t_{A_i} \rangle, \dots), \text{sched}$

Steady-state phase

4.  $A_i \rightarrow H : id_{A_i}, id_H, d_{A_i}, \text{mac}_{k_{[r]}}(id_{A_i} \mid id_H \mid d_{A_i} \mid \text{nonce} + l)$
5.  $H \rightarrow BS : id_H, id_{BS}, \mathcal{F}(\dots, d_{A_i}, \dots), \text{mac}_{k_H}(\mathcal{F}(\dots, d_{A_i}, \dots) \mid c_H)$

Symbols as previously defined, with the following additions:

$r$ :	Id of a key in the key ring	$f()$ :	One-way hash function
$\mathcal{R}_X$ :	Set of key ids in node $X$ 's key ring	$\text{mac}_k(\text{msg})$ :	MAC calculated using key $k$
$\mathcal{V}$ :	A set of CH ids	$c_X$ :	Counter shared by $X$ and $BS$
$k_X$ :	Symmetric key shared by node $X$ and $BS$	$l$ :	Reporting cycle within the current round
$k^j$ :	$j$ -th key in a one-way key chain;	$\text{store}(id_H)$ :	Store $id_H$ for future validation
$k_{[r]}$ :	Symmetric key associated with id $r$	$\text{add}(id_H, \mathcal{V})$ :	Add $id_H$ to $\mathcal{V}$

Figure 3.2: SecLEACH protocol

also conclude that it is not a stale message being replayed. The same observations apply to the message in Step 4. The freshness of all subsequent sensor reports from the ordinary nodes to their CHs is guaranteed by nonces values that are incremented each time. For the message in Step 5, the freshness is guaranteed by the counter value shared between the CH and the BS; the counter value also being incremented each time the CH sends a new report to the BS. Remember (3.3.3) that **sched** messages (Step 3) are authenticated the same way as **adv** messages, but for clarity of presentation we do not reproduce the full-blown authenticated version in Fig. 3.2.

Because link keys used for node-to-CH communications are not pairwise in SecLEACH (i.e., a number of other nodes other than the end points of a compromised link may have the key used in the link), the biggest security issue in SecLEACH is likely to be its resiliency against node captures. We discuss this issue in Section 3.4.2.

### 3.4 Evaluation of our scheme

Random key predistribution schemes have all been introduced and studied in the context of flat networks, which come with the following assumptions: 1) the nodes have antennas with limited transmission range; and 2) node-to-BS communications are multi-hop, with the nodes each relying on their neighbors to forward messages towards the BS. In this context, any two nodes within each other's transmission range has a communication link between them, and a forwarding route can be established between any two nodes (including the BS) as long as one can overlay a connected graph on the network using these range-defined links.

In flat networks where security is to be bootstrapped from random key predistribution, there is a (secure) link between two nodes only if they are within each other's communication range and share a key. In this new context, a (secure) forwarding route can be established between any two nodes (including the BS) only if one can overlay a connected graph on the network using secure links. Given that it is possible that a physical (range-defined) link will be (logically) severed by lack of a shared key between the two end nodes, one needs to choose the parameters  $S$  (size of the key pool) and  $m$  (size of the key ring) in such a way that the resulting network is still (securely) connected, with high probability.

In the context of LEACH, the assumptions are slightly different: 1) any node in the network is reachable from any other node in single hop; but 2) node-to-BS communications are typically carried out in two-hops: from ordinary nodes to CHs, and from CHs to the BS. Because of the first assumption, any ordinary nodes can theoretically join any CH; in practice, they choose the closest to save energy. For energy efficiency, however, a network needs to use just the right number of CHs, as different number of CHs leads to different energy consumptions.

In SecLEACH, because of the constraints imposed by key sharing, not all CHs are accessible to all ordinary nodes. In fact, depending on the values of  $S$  and  $m$ , which determine the probability that two nodes will share a key, an ordinary node will have a larger or smaller number of CHs to choose from. To achieve maximum energy efficiency in the context of SecLEACH, therefore, one needs to find right values for  $S$ ,  $m$ , and the number of CHs. In what follows, we show how different parameter values impact a network, in terms of security and energy efficiency.

### 3.4.1 Parameter values and their impact on performance

Given a WSN, the amount of storage reserved for keys in each node is likely to be a preset constraint, which makes the size of the key ring  $m$  a fixed parameter in the system. Once  $m$  is set, the choice of  $S$  will impact the system in two ways:

1. Its security level:

Given a  $(S, m)$ -network, a network where each node is assigned  $m$  keys from a key pool of size  $S$ ,  $\frac{m}{S}$  is the probability that a randomly chosen link will be compromised when a node that is not either end of the link is compromised. The *security level*  $sl$  of a  $(S, m)$ -network can then be defined as:

$$sl = 1 - \frac{m}{S}$$

which gives the probability that a randomly chosen link is not compromised when a node that is not either end of the link is compromised.

Note that given a fixed  $m$ , the larger the  $S$ , the larger the  $sl$  (the higher the security level).

2. The probability that two nodes will share a key:

Given any two nodes in a  $(S, m)$ -network, the probability  $P_s$  that they will share a key is given by <sup>3</sup>:

$$P_s = 1 - P_{\bar{s}}$$

where  $P_{\bar{s}}$ , the probability that they will not share a key, is given by:

$$P_{\bar{s}} = \frac{[(S - m)!]^2}{S!(S - 2m)!}$$

Note that given a fixed  $m$ , the larger the  $S$ , the smaller the  $P_s$ .

---

<sup>3</sup>This derivatioin was first shown in [33].

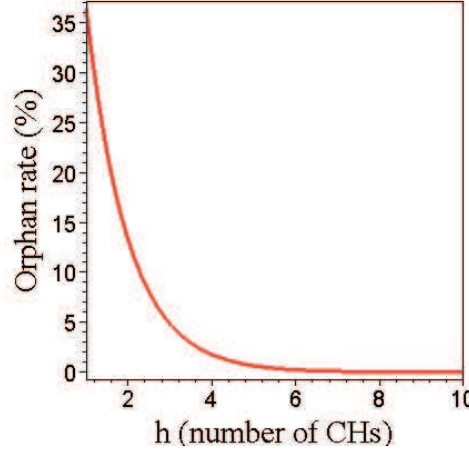
The number  $h$  of CHs in the network is another parameter in the system. In LEACH, the density of CHs in a network determines the average distance between a node and its closest CH. This distance, in turn, determines the amount of energy needed in node-to-CH communications: the denser the CHs, the shorter the average node-to-CH distance, and the smaller the energy consumption for node-to-CH communications. On the other hand, CHs communicate with the BS in single hop. Thus, the larger the number of CHs, the more nodes will be communicating single-hop with the BS, and the more energy will be spent. Taking this reasoning into account, one can find an optimal value for  $h$ , which minimizes the total energy consumption, and maximize the network's lifetime.

In SecLEACH, only a fraction of  $h$  CHs is probabilistically accessible (as determined by key sharing) by an ordinary node. That is,  $h$  is actually a *nominal* value; what ultimately matters is the *effective* value,  $h_e$ , given by  $h_e = h \times P_s$ . Note that, to obtain a given  $h_e$ , one does not need to start with a fixed  $h$ . In fact, one can first fix a value for  $P_s$ , and adjust  $h$  accordingly.

$P_s$  and  $h$  will also determine the expected orphan rate, that is, the probability that an ordinary node will be orphan. Given  $P_s$  (and consequently  $P_{\bar{s}}$ ) and  $h$ , the expected orphan rate  $P_o$  is given by  $P_o = (P_{\bar{s}})^h$ . In a network with  $n$  nodes, it is then expected that  $n \times P_o$  nodes will be orphans, and communicating single-hop with the BS. Fig. 3.3 shows  $P_o$  as function of  $h$  under a  $sl = 0.99$ . Because  $P_o$  depends of the *absolute* number of the CHs, no matter the network size  $n$ , the ratio of orphan nodes will be negligible for  $h \geq 7$ .

To show some concrete numbers and the tradeoffs induced by different parameter values, we provide estimates on energy consumption levels for different scenarios. For our estimates, we assume a network as in LEACH original paper, i.e.,  $n = 100$  nodes, uniformly distributed at random in a  $10^4 m^2$  square area; and a BS located at the center of the square. We consider three key ring sizes for a fixed  $sl$  value ( $m = 50, 100, 150$ , for  $sl = 0.99$ ) and three security levels for a fixed  $m$  value ( $sl = 0.95, 0.98, 0.99$ , for  $m = 100$ ). Table 3.1 exhibits the respective  $P_s$  values for these scenarios. In each case, we take into account only the energy consumed for communication in the steady-state phase since we expected that setup overhead will be amortized among the multiple cycles of the subsequent steady-state phase. In addition, we do not consider the cost incurred by the cryptographic operations, as the operations we use have been shown [90] to incur a very small overhead compared to that incurred by communication.

To estimate the energy consumption, we assume the same radio energy model used in LEACH [43]. In this model, a radio dissipates  $\epsilon_r = 50$  nJ/bit to run the transmitter or receiver circuitry, and  $\epsilon_a = 100$  pJ/bit/m<sup>2</sup> for the transmitter amplifier. Also, the radios expend the minimum required energy to reach the recipients and are turned off to avoid receiving unintended transmissions. An  $\delta^2$  energy loss due to channel transmission

Figure 3.3: Orphan rate, for  $sl=0.99$ 

is assumed as well. Under this model, the costs to transmit ( $\mathcal{E}_T$ ) and receive ( $\mathcal{E}_R$ ) a  $\beta$ -bit message at distance  $\delta$ , and the amount of energy  $\mathcal{E}_{cycle}$  the network consumes to go through one cycle of sensor data reporting are given, respectively, by:

$$\begin{aligned}\mathcal{E}_T(\beta, \delta) &= \beta \epsilon_r + \beta \delta^2 \epsilon_a \\ \mathcal{E}_R(\beta) &= \beta \epsilon_r \\ \mathcal{E}_{cycle} &= (n - h) [\mathcal{E}_T(\beta, \delta_1) + \mathcal{E}_R(\beta)] + h \mathcal{E}_T(\beta, \delta_2)\end{aligned}$$

where  $\delta_1$  is average distance between an ordinary node and its closest CH, and  $\delta_2$  is the average distance between a CH and the BS.

In what follows, we calculated  $\delta_1$  (Fig. 3.4) and  $\delta_2$  by using the derivation in Appendix 3.7. Also, we set SecLEACH messages to be 36 bytes long (the default TinyOS message size [44]) and LEACH messages to be 30 bytes long. The difference is meant to account for the size difference between the MAC (8 bytes [90]) and CRC (2 bytes [56]) – the former present in SecLEACH, but absent in LEACH; and the latter present in LEACH, but absent in SecLEACH.

Using the energy consumption model above, we obtained the energy consumption level for the various scenarios we considered. In Figs. 3.5 and 3.6, the values are for one cycle of sensor data reporting in the steady-state phase. Fig. 3.5 shows the energy consumption in node-CH communication for different security levels. Note that the consumption level is smaller in LEACH than in any instantiations of SecLEACH, and larger values of  $sl$  lead to larger overheads. On the other hand, the higher the  $h$ , the smaller the overhead. For a given security level, larger key rings decrease the energy consumption (Fig. 3.6). Note



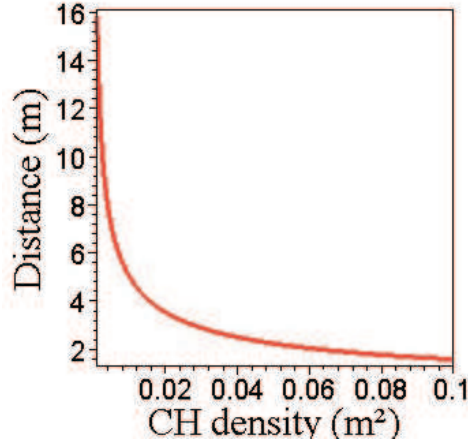


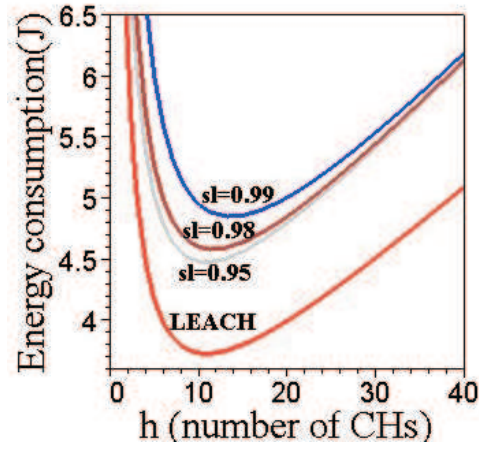
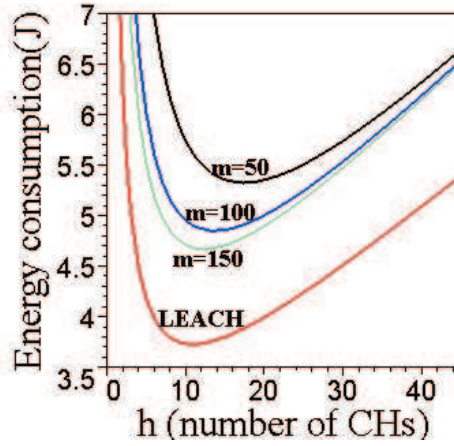
Figure 3.4: Average node-to-CH distance

that, in all cases, there is a value of  $h$  for which the energy consumption is minimum.

We also estimated how scalable SecLEACH is. Table 3.2 shows the overhead incurred by SecLEACH, under the various parameter values and under different network sizes  $n$ , as compared to LEACH. In the estimates, we assume a constant node density (i.e., the larger the  $n$ , the larger the network area, as well) and a single BS. The overheads were computed using the values of  $h$  for which the energy consumption, in each scenario, is minimum. It is worth mentioning that overhead in SecLEACH is due to two factors: the increased message size (20% larger) and the increased node-CH distance – the CH-BS distance in SecLEACH is not increased as compared to LEACH, as every CH shares a key with the BS.

Note that for the maximum security level ( $sl = 0.99$ ), the overhead increases from 30,0% to 46,1%, as the network becomes larger. This increase is due to an increase in the average CH-BS distance, caused by the increase in the size of the network. (Larger distances lead to more expensive communications.) To counterbalance this factor, the network could instantiate a smaller number of CHs. But this would increase the number of nodes performing node-CH communication (the one that incurs overhead) leading to an overall overhead increases as well.

This overhead, however, can be mitigated by using a larger values of  $m$ , as shown in the column  $m = 150$ . Alternatively, one may also choose to live with a lower security level. E.g., the  $P_s$  value for  $sl = 0.95$  is very close to that for LEACH (Table 3.1), and the overhead in this case being due mainly to the increase in the message size. For such  $sl$  value, the solution is very scalable.

Figure 3.5: Energy consumption, for  $m=100$ Figure 3.6: Energy consumption, for  $sl=0.99$ 

sl	m		
	50	100	150
0.95	–	0.995	–
0.98	–	0.870	–
0.99	0.396	0.636	0.780

Table 3.1: Prob.  $P_s$  of key sharing as a function of security level  $sl$  and key ring size  $m$

n	sl (m=100)			m (sl=0.99)		
	0.95	0.98	0.99	50	100	150
100	20,1%	22,9%	30,0%	42,9%	30,1%	25,3%
1000	20,2%	25,6%	39,8%	65,6%	39,8%	30,3%
10000	20,3%	27,4%	46,1%	80,6%	46,1%	33,6%

Table 3.2: Energy Overhead

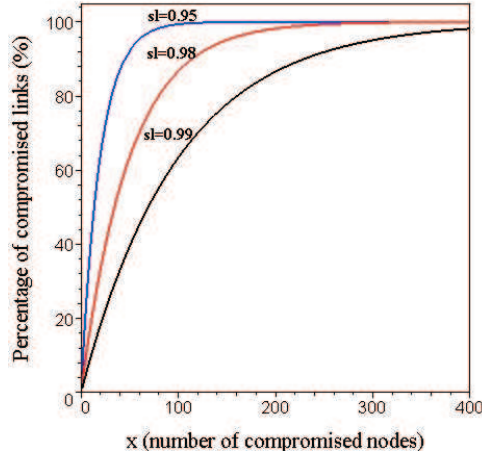


Figure 3.7: SecLEACH's resiliency against node capture

### 3.4.2 Resiliency against node capture

In key distribution schemes, resiliency against node capture measures how much of the network (its communication links) is compromised when a node is compromised. It is a critical performance measure that gauges the robustness of a solution. In SecLEACH, the values of  $m$  and  $S$  determines the probability that a random link will be compromised when a node (that is not either end of the link) is compromised.

The resiliency of random key predistribution has been studied before [17] in the context of flat networks and the same analysis is applicable in our context. Fig. 3.7 shows the percentage  $P_c$  of compromised links as a function of the absolute number of compromised nodes for the considered security levels. Note that  $P_c$  increases as the absolute number of compromised nodes (instead of percentage of nodes in the network) increases. Beyond a certain value of  $x$ ,  $P_c$  reaches the value 1.0, no matter what is the security level. For smaller values of  $x$ , however, there is a significant difference in the  $P_c$  with different values of  $sl$ . This difference decreases as  $x$  increases.

### 3.5 Related work

WSNs are a subclass of MANETS, and much work (e.g., [116, 12, 50, 13, 109, 47, 115]) has been proposed for securing MANETS in general. These studies are not applicable to WSNs because they assume laptop- or palmtop-level resources, which are orders of magnitude larger than those available in WSNs. Conventional public key based solutions are such an example.

Among the studies specifically targeted to resource-constrained WSNs, some [57, 111] have focused on attacks and vulnerabilities. Wood and Stankovic [111] surveyed a number of denial of service attacks against WSNs, and discussed some possible countermeasures. Karlof and Wagner [57] focused on routing layer attacks, and showed how some of the existing WSN protocols are vulnerable to these attacks.

Of those offering cryptographic solutions, a reasonable number (e.g., [14, 33, 112, 118, 17, 117, 62, 63, 49, 91, 29, 48, 54, 51, 15, 64, 28, 92]) have focused on efficient key management of symmetric schemes without tying them to a particular network organization. Others, recently, have been investigating more efficient techniques of public key cryptography. By using Elliptic Curve Cryptography [76, 58], for example, it has been shown (e.g., [41, 67, 7]) that sensor nodes are indeed able to compute public key operations. However, public key authentication in the context of WSNs is still an open problem, as they cannot afford a conventional public key infrastructure and the proposed alternatives (e.g. [30]) are not applicable to all contexts.

Perrig *et al.* [90] proposed SPINS. SPINS includes two efficient symmetric key based security building blocks: SNEP and  $\mu$ TESLA. SNEP provides confidentiality, authentication, and freshness between nodes and the BS, and  $\mu$ TESLA provides authenticated broadcast.  $\mu$ TESLA, which we use in our solution, implements the asymmetry required for authenticated broadcast using one-way key chains constructed with cryptographically secure hash functions, and delayed key disclosure.

Hierarchical WSNs have quite particular organization patterns, and one can take them into account to design tailored solutions. Carman *et al.* [14] have suggested using higher powered nodes for key generation and management functions, but did not offer concrete protocols. Kong *et al.* [59] and Bohge and Trappe [10] devised solutions for concrete hierarchical and heterogeneous networks. They both assume more powerful nodes, and use public key cryptography. More specifically, the former relies on RSA certificates to guarantee authentication. The amount of computation and space resources required by RSA certificates makes this solution infeasible in our context. In addition, it proposes *end-to-end* transport layer security, which prevents data aggregation at intermediary hops. The latter proposes an authentication framework for a concrete 2-tier network organization, in which a middle tier of more powerful nodes were introduced between the BS and the

ordinary sensors to carry out authentication functions. However, except for the lowest tier nodes, all other nodes perform public key operations. Finally, none of these works consider networks where cluster are formed dynamically and periodically.

There has also been some work on detecting misbehaving nodes. E.g., Marti *et al.* [68] proposed a watchdog scheme that enables network nodes to detect selective forwarding attacks staged by their next hop neighbors.

Detecting and dealing with bogus data has also been focus of research. Zhu *et al.* [119] proposed an interleaved hop-by-hop authentication scheme to prevent injection of false data into sensor networks. The proposal makes sure that the BS can detect a false report when no more than a certain number  $t$  of nodes are compromised. Yea *et al.* [112] proposed SEF, a statistical en-route filtering mechanism for detecting and dropping bogus reports while being forwarded. It allows both the BS and the en-route nodes to detect false data with a certain probability. Przydatek *et al.* [95] proposed SIA, a framework for secure information aggregation in WSNs which makes use of random sampling strategies for allowing an user to infer about the legitimacy of a value.

Other efforts have focused on more specific types of attacks. Hu *et al.* [46] studied and offer solutions for wormhole attacks, whereas Newsome *et al.* [78] investigated sybil attacks in the context of WSNs. Finally, Deng [23] *et al.* address secure in-network processing, and propose a collection of mechanisms for delegating trust to aggregators that *a priori* are not trusted by common sensors. The mechanisms address both dissemination and aggregation of data.

## 3.6 Conclusion

In this paper we presented SecLEACH, a protocol for securing LEACH-based networks. SecLEACH achieves baseline security by adapting random key predistribution and  $\mu$ TESLA, and can yield different performance numbers on efficiency and security depending on its various parameter values. Our estimates show that the overhead incurred by SecLEACH is manageable; and memory usage, energy efficiency, and security level can be each traded off for another, depending on what is most critical in a system. Finally, SecLEACH preserves the structure of the original LEACH, including its ability to carry out data aggregation.

## 3.7 Average distance estimates

Next, we will present the derivation for the distance estimates. It is worth noting that both are independent of the network size  $n$ .

### 3.7.1 Distance between CH and BS

Given a square of side length  $2s$  the probability  $P$  that the distance of a randomly chosen point in the square to its center is less or equal to  $x$  is given by:

$$P(d \leq x) = \frac{\pi x^2}{4s^2}, \quad \text{if } 0 \leq x \leq s$$

$$= \frac{\pi x^2 - 4 \left( x^2 \arctan \left( \frac{\sqrt{x^2 - s^2}}{s} \right) - s \sqrt{x^2 - s^2} \right)}{4s^2}, \quad \text{if } s \leq x \leq \sqrt{2}s$$

Hence, this probability density function (pdf) has the form:

$$f(x) = \frac{\pi x}{2s^2}, \quad \text{if } 0 \leq x \leq s$$

$$= \frac{\pi x - 4x \arctan \left( \frac{\sqrt{x^2 - s^2}}{s} \right)}{2s^2}, \quad \text{if } s \leq x \leq \sqrt{2}s$$

Now, we may use the pdf to calculate the expected distance:

$$E(X) = \int_0^{\sqrt{2}s} x f(x) = \frac{(\sqrt{2} + \log(1 + \sqrt{2})) s}{3}$$

### 3.7.2 Distance between ordinary node and CH

In a population of  $i$  individuals distributed at random in an area  $a$ , the expected distance from an individual to its nearest neighbor ( $NN$ ) [18], and the same with edge effect correction [26] ( $NN_c$ ), are, respectively, given by:

$$NN = 0.5 \sqrt{\frac{1}{\rho}}$$

$$NN_c = 0.5 \sqrt{\frac{a}{i}} + \left( 0.0514 + \frac{0.041}{\sqrt{i}} \right) \frac{p}{i}$$

where  $\rho$  stands for neighborhood density, i.e.,  $\rho = \frac{i}{a}$  and  $p$  is the perimeter of the study area <sup>4</sup>.

---

<sup>4</sup>Individuals have been treated as dimensionless points by Clark and Evans, since their dimensions are usually negligible as compared to the total area

To determine the expected distance from an ordinary node to its nearest CH, we may consider only the CHs as neighbors of this node and apply the formula for  $NN$  calculation. Thus, this expected distance for LEACH and SecLEACH are given, respectively, by:

$$NN_{LEACH} = 0.5 \sqrt{\frac{a}{h+1}} + \frac{\left(0.0514 + \frac{0.041}{\sqrt{h+1}}\right) p}{h+1}$$

$$NN_{SecLEACH} = 0.5 \sqrt{\frac{a}{h_e+1}} + \frac{\left(0.0514 + \frac{0.041}{\sqrt{h_e+1}}\right) p}{h_e+1}$$

where  $h$  and  $h_e$ , as defined in Section 3.4.1, stand for the number of accessible CHs by an ordinary node in LEACH and SecLEACH, respectively.

## Chapter 4

# Authenticated ID-based Non-Interactive Key Distribution for WSNs

### 4.1 Introduction

Wireless sensor networks (WSNs) [34] are ad hoc networks composed primarily of perhaps thousands of tiny sensor nodes with limited resources and one or more base stations (BSs). They are used for monitoring purposes, providing information about the *area of interest* to the rest of the system.

On the other hand, Pairing-Based Cryptography (PBC) [98, 52, 70], is an emerging technology that allows a wide range of applications. Pairings have been attracting the interest of the international cryptography community because it enables the design of original cryptographic schemes and makes well-known cryptographic protocols more efficient. Perhaps the main evidence of this is the realization of Identity-Based Encryption (IBE) [11], which in turn, has facilitated complete schemes for Identity-Based Cryptography (IBC) [104].

In the context of a WSNs, the issue of securing and authenticating communications is a difficult one, especially as currently nodes have no capacity for the secure storage of secret keys and are frequently deployed in unprotected areas, which make them more vulnerable to attacks [57]. One simple idea to introduce some kind of security is to fit each sensor node with the same cryptographic key to be used for all communications (e.g. [56]). But this does not authenticate the source of a message, and furthermore if one node is successfully attacked, all communications are compromised.

Assume now that there are  $n$  nodes, and that each has its own unique identifier (ID) from 0 to  $n - 1$ . A better idea would be to fit each pair of nodes with a unique mutual



key for all communications between them. But if that were the case each node would have to store  $n - 1$  secret keys, and furthermore  $n(n - 1)/2$  such keys would need to be generated in all. This is a big requirement in terms of time and storage for large  $n$ , especially considering that after deployment not all nodes will be in a position to talk with all other nodes. Furthermore, if new nodes are to be deployed at a later stage all existing ones must be recalled to be fitted with new keys.

Now consider this scenario: each node is issued with (i) a unique ID; and (ii) a unique secret, not shared with any other entity. Two parties, each knowing only the ID of the other and without communicating, are then able to derive a mutual secret unknown to any other party, and use that secret to derive a cryptographic key to secure their communications. It is also trivial to dynamically add new nodes to the the WSN without any impact on existing nodes

This scheme exists and in the world of Cryptography its known as the *Identity-Based Non-Interactive Key Distribution Scheme* (ID-NIKDS) [98]. It is *Identity-Based*, as only IDs are required – in particular no extra public key data is needed. It is *Non-Interactive*, as only the ID of the “other” is required to determine the key – no interaction is required. And it is a *Key Distribution Scheme*, because each ends up with the same key value. Also, the protocol is authenticated as each party knows that only the other can possibly calculate the same key <sup>1</sup>.

One issue has not been addressed – where does each entity get its unique secret from? It gets it from a *Trusted Authority*. This authority generates the unique secret from nodes IDs and a master secret of its own. Note that this “trusted authority” must be just that, as it is in a position to determine all the keys used within the system. It is our contention that such a set-up is an ideal way to bootstrap a WSN for security. The Trusted Authority is simply the *deployer* of the network, and there will be no issue in assuming their trustworthiness. Indeed it might even be regarded as a “feature” that the deployer is in a position to monitor all wireless traffic.

An alternative idea is to use the well-known Diffie-Hellman interactive key exchange to dynamically derive a mutual key between pairs of nodes. But this is not authenticated, and hence is subject to a deadly *man-in-the-middle* attack. Also interaction involves communication, and wireless communication is expensive in terms of power consumption.

Can the method we suggest be realized using regular public key cryptography? No it cannot. Indeed it is only relatively recently that a viable scheme has been discovered, and its implementation is quite difficult and computationally costly. However we only suggest it as a boot-strapping mechanism. Once the WSN nodes are deployed, they can cache keys, and create their own local keys for use within their own neighborhood. In this way the ID-NIKDS protocol need only be required very occasionally. Note that the ID-NIKDS

---

<sup>1</sup>Actually, as we will see later, an entity that is unconditionally trusted also can.

secret is the only long-term secret that the node possesses, and that possession of such a secret is unavoidable if authentication is a requirement.

We do not claim that a scheme like this is by itself sufficient for securing WSNs. We do not claim that a network bootstrapped in this way will be immune from attack. An attacker could after all in theory compromise every node in the network. We do however claim that it is the best possible way to bootstrap a WSN, given that a node does not have secure storage for its secrets. Built on top of such a system, the network can dynamically evolve and develop routing and communications algorithms with maximum confidence that the damage caused by an attacker will be localized and minimized.

In this work, we firstly discuss why and how ID-NIKDS should be used to bootstrap security in WSNs. After that, we present TinyPBC, to our knowledge the most efficient implementation of PBC primitives for an 8-bit processor, and its performance on an ATmega128L (the MICA2 and MICAZ node microcontroller [45]). TinyPBC is based on Multiprecision Integer and Rational Arithmetic C/C++ Library (MIRACL) [100] – which is a publicly available and open source library – and able to compute pairings, the most expensive operation of PBC, in about 5.5s. To sum up, our key contributions are:

1. demonstrate how sensor nodes can exchange keys in an authenticated and non-interactive way;
2. present the fastest pairing computation on an 8-bit platform; and
3. show the best figures for binary field multiplication on an 8-bit platform.

The remainder of this work is organized as follows. In Section 4.2, we discuss the need for new security solutions in WSNs. We point out the synergy between IBC and WSNs in Section 4.3. In Section 4.4 we show how ID-NIKDS can bootstrap security in WSNs. Implementation and results are presented in Section 4.5. Finally, we discuss related work and conclude in Sections 4.6 and 4.7, respectively.

## 4.2 Bootstrapping Security in WSNs: Need for New Approaches

Security is mainly justified in WSNs because of their battlefield applications. We believe, however, that, once WSNs start to be deployed in large scale, security will become much more common than it is thought today. Apart from the well-known battlefield applications, confidentiality is likely to be a requirement in market and industrial scenarios. For example industries/farmers that employ WSNs to monitor their supply-chains/crops may want to keep their data private from competitors. Additionally, authentication might

be useful even in domestic WSNs, avoiding interaction with nodes from a neighboring network.

Briefly, an ideal security scheme in WSNs should provide perfect connectivity and resilience. In other words, nodes should be able to (i) communicate securely with any other node they wish, and (ii) the compromise of a node should be restricted to itself. (Note that these properties should apply even to nodes deployed at different times.) Also the scheme should be low-cost in terms of both communication and computation.

In WSNs, security is typically bootstrapped using key distribution schemes. Most of standard key distribution schemes in the security literature [99], however, are ill-suited to WSNs: conventional public key based distribution, because of its processing requirements; global keying, because of its security vulnerabilities; complete pairwise keying, because of its memory requirements; and those based on a key distribution center, because of its inefficiency. (See Carman *et al.* [14] for a good introduction to key distribution in WSNs.)

Symmetric key based distribution schemes (e.g., [90, 33, 118, 91, 17, 54, 51, 15, 28, 64, 86, 83]) have been specifically designed for WSNs. While they are well-suited for the applications and organizations they were designed for, they might not be adequate for others. They provide a trade-off between connectivity and resilience, not providing an ideal level of both. Further, most schemes rely on some sort of interaction between nodes so that they can agree on keys.

More recently, it has been shown that alternative methods of Public Key Cryptography (PKC) are feasible in WSNs [110, 41, 67]. Because in those systems communicating parties only have a pair of keys, a private and a public key, PKC schemes are scalable and easy to use. This convenience, though, comes at a price: a way of authenticating public keys must be provided. And key authentication, in turn, whether traditional (PKI and/or certificates) or especially tailored to WSNs (e.g. [30]), often ends up in overhead – which is especially ill-suited to WSNs.

As we will show in Section 4.4, by using ID-NIKDS we are able to resolve these security issues.

### 4.3 Synergy between IBC and WSNs

PBC has paved the way for a new wide range of cryptographic protocols and applications [89]. It has also allowed many long-standing open problems to be solved elegantly. Perhaps the most impressive among those applications is IBE [11], which in turn has allowed complete IBC schemes [104], <sup>2</sup>.

One may thus ask why IBC is still not widely deployed in security systems. Besides

---

<sup>2</sup>Today, however, other ways of providing IBE exist (e.g. [19]).

the usual time it takes for new technologies to be adopted in security systems, this is because IBC faces additional drawbacks. In particular they require a Trusted Authority (TA). A TA is an entity in charge of generating and escrowing users' private keys. That is, it is able to impersonate anybody else in the system. For that reason, a TA must be an entity that is unconditionally trusted by all network users. Such an entity, however, cannot be identified in many systems.

In WSNs, conversely, this is not a problem at all. The deployer – who loads software into nodes, deploys them in areas of interest, and observes collected data – is, obviously, trusted. In the world of WSNs, the deployer's role is represented by BS nodes. These nodes possess both laptop-level resources and physical protection. In other words, they can play the role of TA perfectly.

Another IBC's requirement is that the keys must be delivered over confidential and authenticated channels to users. If the cryptographic scheme is being used to bootstrap security – as very often is the case – there such channels will not exist. But again this is not a great concern to WSNs. In their security model, there is clearly a point in the time (i.e., prior to deployment) where secure channels between the BS and ordinary nodes do exist. Along with application software, nodes' private keys can be preloaded into nodes during the pre-deployment stage.

## 4.4 Authenticated Identity-Based Non-Interactive Key Distribution in Sensor Networks

The notion of Identity Based Cryptography dates back from Shamir's original work [104], but it has only become practical with the advent of PBC [70, 98, 11, 53]. The main idea is that known information that uniquely identifies users (e.g. IP or email address) can be used to derive public keys. As a result, keys are self-authenticated and additional means of public key authentication, e.g. certificates, are thus unnecessary. In this Section we define pairings (4.4.1), and show how to setup IBC schemes in the WSN context (4.4.2), and finally show how ID-NIKDS can be used so that nodes end up agreeing on keys. (4.4.3).

### 4.4.1 Pairings: Definition

Bilinear pairings – or pairings for short – were first used in the context of cryptanalysis [70], but their pioneering use in cryptosystems is due the works of Sakai, Ohgishi, and Kasahara [98] *et al.* and Joux [53]. In what follows, let  $E/\mathbb{F}_q$  be an elliptic curve over a finite field  $\mathbb{F}_q$ ,  $E(\mathbb{F}_q)$  be the group of points of this curve, and  $\#E(\mathbb{F}_q)$  be the group order.

**Bilinear pairing.** Let  $n$  be a positive integer. Let  $\mathbb{G}$  be an additively-written group of order  $n$  with identity  $\mathcal{O}$ , and let  $\mathbb{G}_T$  be a multiplicatively-written group of order  $n$  with identity 1.

A *bilinear pairing* is a computable, non-degenerate function

$$e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$$

The most important property of pairings in cryptographic constructions is the bilinearity, namely:

$$\forall P, Q \in \mathbb{G}, \text{ and } \forall a, b \in \mathbb{Z}^*, \text{ we have}$$

$$e([a]P, [b]Q) = e(P, [b]Q)^a = e([a]P, Q)^b = e(P, Q)^{ab}.$$

In practice, the group  $\mathbb{G}$  is implemented using a group of points on certain elliptic curves and the group  $\mathbb{G}_T$  is implemented using a multiplicative subgroup of a finite extension field. For more on pairing definitions, see for instance Galbraith [37].

Here we are using *Type 1* pairings (in the sense of Galbraith, Paterson and Smart [38]), and so we have the additional property that

$$e(P, Q) = e(Q, P).$$

In addition, pairings of *Type 1* permit strings to be hashed to a specific group. Those two aforementioned properties are required for efficient and simple implementation of the protocol (pairings of *Types 2* and *3* do not provide, at once, both properties).

#### 4.4.2 Setup

To start up an IBC scheme, the TA first needs to generate and distribute private keys and public parameters. Broadly speaking, this procedure can be accomplished as follows in WSNs. Firstly, the BS generates a master secret key  $s$  and then calculates each node's private key. To do this it first maps each node's identity to a point on the elliptic curve, via a hashing-and-mapping function  $\phi$ , so for node  $X$ ,  $P_X = \phi(id_X)$ . It then calculates the node's private key as  $S_X = [s]P_X$ . It next preloads each node  $X$  with the following information: (i) the node's ID  $id_X$ , (ii) the node's private key  $S_X$ . Each node is also equipped with the function  $\phi$  so that it can take any ID (e.g.  $id_Y$ ) as input and outputs the public key corresponding to the ID (e.g.  $P_Y$ ), <sup>3</sup>.

Note that, besides the BS, only node  $X$  knows the key  $S_X$ .

---

<sup>3</sup>To be precise, a small number of non-secret public parameters are also needed to be stored into nodes, but for simplicity's sake, we will omit them.

### 4.4.3 Applying ID-NIKDS in WSNs

WSNs are composed of maybe thousands of tiny resource-constrained sensor nodes for which the scarcest resource is energy. Communication, on the other hand, is the activity that consumes most energy. This, in turn, means that besides meeting the needs described in Section 4.2 (i.e., perfect connectivity and resilience), an ideal key agreement scheme for WSNs should also not require any exchange of messages.

With the advent of PBC, however, a method of accomplishing this has come available. That is, PBC provides means to non-interactively distribute keys between any two network nodes, even if they were deployed at different times. Further, because nodes employ asymmetric primitives, the effect of node compromise is strictly local. In what follows, we show how the protocol due to Sakai, Ohgishi, and Kasahara, ID-NIKDS [98], can be employed to achieve such a goal. We assume that the setup protocol shown in Section 4.4.2 has been already carried out.

Suppose two nodes A and B that know each other's IDs wish to decide on a secret key. Recall from Section 4.4.2 that nodes A's and B's private keys are  $S_A = [s]P_A$  and  $S_B = [s]P_B$ , respectively. Consequently, by bilinearity (Section 4.4.1) we have

$$\begin{aligned}\hat{e}(S_A, P_B) &= \hat{e}([s]P_A, P_B) \\ &= \hat{e}(P_A, P_B)^s \\ &= \hat{e}(P_A, [s]P_B) \\ &= \hat{e}(P_A, S_B) \\ &= \hat{e}(S_B, P_A).\end{aligned}$$

Note that A possesses  $S_A$  and can compute  $P_B = \phi(id_B)$ . Likewise, B possesses  $S_B$  and can compute  $P_A = \phi(id_A)$ . Therefore, both A and B are able to compute the secret key

$$k_{A,B} = \hat{e}(S_A, P_B) = \hat{e}(S_B, P_A).$$

(Formally speaking, a key derivation function must first be applied to  $k_{A,B}$  in order to generate a key appropriate for cryptosystems. For details on this and other PBC protocols refer, e.g., to Paterson [89] and/or Duquesne and Lange [31].) Additionally, A knows that only B – and the BS, a trusted authority – possess  $S_B$  and vice-versa, and consequently the protocol is authenticated.

Observe that due to the non-interactive nature of the communication nodes can agree on keys even if they are not online simultaneously. This is particularly useful in WSNs, where nodes might follow sleeping patterns, may be deployed at different times, and often become temporarily unavailable due to physical obstacles or malfunctions.

Lastly, notice that we assume that nodes already know one another’s IDs. And we understand that it is a reasonable assumption in WSN context. Since routing reports towards the BS already demands nodes to get to know their neighbor’s IDs, exchange of IDs cannot be considered an overhead incurred by the key distribution protocol. Additionally, ID size is negligible when compared to public key and certificate sizes.

## 4.5 Evaluation

The utilization of pairings to implement security in WSNs is quite complex. For an 80-bit security level (RSA-1024 equivalent), as opposed to conventional Elliptic Curve Cryptography (ECC) – which works with 160-bit numbers – PBC works with 1024-bit numbers. In this section, we assess the costs incurred by PBC on an ATmega128L microcontroller, as included in the MICA2 and MICAZ node microcontroller [45]. The platform features an 8-bit/7.3828-MHz processor, 4KB of SRAM, and 128KB of flash memory (ROM).

### 4.5.1 Implementation

By far the most time consuming part when evaluating PBC protocols is the pairing computation itself <sup>4</sup>. In this section we present TinyPBC, an implementation of the  $\eta_T$  [3] pairing (pronounced “eta-t”) for resource-constrained nodes – the source code is available at [www.lca.ic.unicamp.br/~loliveira#tinypbc](http://www.lca.ic.unicamp.br/~loliveira#tinypbc). Due to spaces constraints, we do not describe in depth all implementation details. Instead, we do point the reader out to more descriptive sources. (For instance, we recommend those not familiar with ECC and pairing implementation to look at Hankerson, Menezes, and Vanstone [42] and Scott [101] for a good introduction to the former and the latter, respectively.)

**Security Requirements** To meet their needs for efficiency security requirements in WSNs are often relaxed. For example, some (e.g. [90]) have adopted a 64-bit security level. We, conversely, adopted a more conservative posture and thus used a 80-bit security level, as recommended by the NIST.

**Pairing** The  $\eta_T$  [3] is possibly the fastest known pairing. It was proposed by Barreto, Galbraith, hEigartaigh, and Scott, following the earlier work of Duursma and Lee [32]. Like most pairings, it uses a variant of Miller’s algorithm to evaluate pairings. Its main feature, however, is that the  $\eta_T$  pairing requires only half the number of iterations of the Miller’s loop compared with other pairings (Line 4, Algorithm 3 of [3]).

---

<sup>4</sup>ID-NIKDS also requires hashing, but that can be efficiently computed in the ATmega128L [39].

$\eta_T$  spends most of its time performing extension field multiplications. Our code uses binary fields ( $\mathbb{F}_{2^m}$ ), and the supersingular curve  $y^2 + y = x^3 + x^2$ , which has an *embedding degree* of four. In other words, this means that our implementation spends most of its time carrying out multiplications in  $\mathbb{F}_{2^{4 \times 271}}$ , the quartic extension field. Next we will describe how TinyPBC computes binary field multiplication.

**Finite Field and Big Number Arithmetic** The fastest known algorithms for multiplication on  $\mathbb{F}_{2^m}$  first multiply the field elements as polynomials and then reduce the result modulo an irreducible polynomial  $f(x)$ . For the particular binary field  $\mathbb{F}_{2^{271}}$ , we have selected the pentanomial  $f(x) = x^{271} + x^{207} + x^{175} + x^{111} + 1$ , given in [102], which leads to a faster field square-root algorithm on a computer platform with word length of 8 or 16 bits.

Multiplication in our case is carried out using the *López-Dahab* (LD) method [65]. To compute the product  $ab$ , LD requires a look-up table for storing the product of polynomials of small degree by the operand  $b$ . We have used a look-up table of 16 polynomials; i.e., we process four bits in each iteration. For  $b = (B[t-1], B[t-2], \dots, B[0])$ , the look-up table is defined as the matrix  $T[i][j] = (i \cdot b)[j]$ ,  $i = 0, 1, \dots, 15$  and  $j = 0, \dots, t$ . In our code,  $T$  is computed by columns, i.e., in the following order:  $T[i][t], T[i][t-1], \dots, T[i][0]$  for  $(i = 0, \dots, 15)$ .

Finite field multiplication is the most important operation, but not the only one that needs to be fast for efficiently computing pairings. To carry out other group and finite field operations as well as big number arithmetic, TinyPBC relies on MIRACL, a publicly available library written in C. The library has been used as basis for numerous works on efficient implementation. Also MIRACL already contains a highly efficient code for the  $\eta_T$  pairing. Running the code in resource-constrained nodes, such as those that use the ATmega128L, however, is not straight forward and thus adaptations have been made in order to fit MIRACL into the platform. In particular RAM usage must be minimized.

**Optimization** To speed up the binary field multiplication (polynomial of size 34 bytes – as required to store 271 bits), we have come up with an implementation of LD that uses Karatsuba’s method [55], another multiplication technique, of the depth-1. In other words, we use the LD method to multiply three polynomial of size 17 bytes. In this case LD’s main loop given by

```
for(i=0;i<17;i++){
    u = a[i]>>4;
    for(j=0;j<18;j++)
```



```

        c[i+j]^= T[u][j];
    }
    LSHIFT4(c); /* left shift c 4 bits*/
    for(i=0; i<17; i++){
        u = a[i] & 0xf;
        for(j=0; j<18; j++)
            c[i+j]^= T[u][j];
    }

```

was replaced by a code that uses a software pipeline technique and processes 8 bits in each iteration, namely

```

for(i=0; i<17; i++){
    u0 = a[i]&0xf; u1 = a[i]>>4;
    s0 = T[u0][0]; s1 = T[u1][0];
    c[i]^= s0^(s1<<4);
    s0 = T[u0][1]; s2 = T[u1][1];
    for(j=2; j<18; j++){
        c[i+j-1]^= s0^(s2<<4)^(s1>>4);
        s2 = s1; s0 = T[u0][j];
        s2 = T[u1][j];
    }
    c[i+17]^= s0^(s2<<4)^(s1>>4);
}.

```

Our improved version reduces the number of stores (for  $c$ ) and it was observed that saves cycles (5.6%) when compared to the original LD algorithm. Note that the optimization have worked out for the GCC, but the result could have been different if were we using another compiler. Finally, it is worth noting that we have not used assembly and therefore our code is portable to other platforms.

## 4.5.2 Performance

In this section we summarize performance numbers. Figures are based on the avr-gcc compiler using optimization level -O3 in modules with time critical functions.

TinyPBC takes only 5.45s (Table 4.1) to compute pairings on ATmega128L. That is, it requires less than half the amount of time of the quickest previous result [106], which takes 10.96s to compute pairings. This is mainly due to our faster binary field multiplication. The time required to compute binary field multiplication with our LD implementation,

averaged over 20 trials, is just  $4,019.46\mu\text{s}$  (Table 4.1), with a standard deviation of  $1.82\mu\text{s}$ . This is 44% faster than Karatsuba’s method, which was employed in [106]. The result is particularly interesting because it contrasts sharply with results presented in [4], which claims that Karatsuba’s is the most appropriate method for embedded devices.

Time	
Multiplication	Pairing
$4,019.46\mu\text{s}$	5.45s

Table 4.1: Time costs to evaluate binary field multiplication and the  $\eta_T$  pairing on ATmega128L using TinyPBC.

### 4.5.3 Storage

Table 4.2 summarizes storage requirements of TinyPBC. The requirements for stack and static RAM as well as ROM are 2,687, 368 and, 47,948 bytes, respectively (Table 4.2). Note that our approach allocates virtually all the RAM from the stack, which means that once the pairing is computed the memory is available for other operations. Plus, because the  $\eta_T$  pairing does not benefit from precomputation, ROM is saved.

Storage (bytes)		
Stack	RAM	ROM
2,867	368	47,948

Table 4.2: Memory costs to evaluate the  $\eta_T$  pairing on ATmega128L using TinyPBC.

Besides the cryptographic code, a node needs to store its private key and public parameters in order to run ID-NIKDS. Public parameters are part of the specification of the pairing and they are already taken into consideration in our code. A private key, on the other hand, requires a point on  $E(\mathbb{F}_{2^{271}})$ . That is, an elliptic curve point that in turn is represented by coordinates  $(x, y)$  from the field  $\mathbb{F}_{2^{271}}$ , 271-bits each. Given  $x$  and a single bit of  $y$ , however, a node itself can easily derive  $y$ . So, in addition to the cryptographic code, a node must be loaded with a 272-bit private key, i.e., an overhead of only 34 bytes.

## 4.6 Related Work

The number of studies specifically targeted to secure WSNs has grown significantly. Due to space constraints, we first provide a sample of studies based on symmetric cryptosystems, and then focus on those targeted to efficient implementation of PKC on sensor nodes.

Many security proposals for WSNs (e.g., [90, 33, 118, 91, 17, 54, 51, 15, 64, 28, 86, 83]) have focused on efficient key management of symmetric encryption schemes. Perrig *et al.* [90] proposed SPINS, a suite of efficient symmetric key based security building blocks. Eschenauer *et al.* [33] looked at random key predistribution schemes, which provoked a large number of follow-on studies [51]. In [118] Zhu *et al.* proposed LEAP, a rather efficient scheme based on local distribution of secret keys among neighboring nodes.

The studies specifically targeted to PKC have tried either to adjust conventional algorithms (e.g. RSA) to sensor nodes, or to employ more efficient techniques (e.g. ECC) in this resource-constrained environment. All the seminal papers of Watro *et al.* [110], Gura *et al.* [41], and Malan *et al.* [67] have targeted the ATmega128L. Watro *et al.* [110] proposed TinyPK. To perform key distribution, TinyPK assigns the efficient RSA public operations to nodes and the expensive RSA private operations to better equipped external parties. Gura *et al.* [41] reported results for ECC and RSA primitives on the ATmega128L and demonstrated convincingly that the former outperforms the latter. Their ECC implementation is based upon arithmetic in the prime finite field  $\mathbb{F}_p$ . Malan *et al.* [67] have presented the first ECC implementation over binary fields  $\mathbb{F}_{2^m}$  for sensor nodes. They used a polynomial basis and presented results for the ECDH key exchange protocol.

In the literature there are works that make use of identities to distribute keys in WSNs. Some (e.g Carman *et al.* [14], Du *et al.* [28], and Liu *et al.* [61]) are based on the symmetric cryptosystems due to Blundo *et al.* [9] and Blom *et al.* [8]. These strategies, however, do not provide perfect resilience, as after a certain percentage of nodes have been compromised, the whole network would be compromised as well. Others (e.g [114, 27, 82]) have employed IBC from PBC. The works of Zhang *et al.* [114], Doyle *et al.*, [27] and Oliveira *et al.* [82] employ IBC to distribute keys between nodes. However, they all use interactive protocols and therefore nodes are required to exchange messages to agree on keys.

Software implementation of pairings has also been focus of research. Oliveira *et al.* [80] have come up with an implementation of the Tate pairing. TinyTate, as it is called, uses TinyECC [61] as the underlying library and also targets the ATmega128L. TinyTate, however, takes around 31s to compute pairings and its level of security, equivalent to RSA-512, is not appropriate for all applications. More recently, Szczechowiak *et al.* [106] have shown performance number for ECC operations as well as pairings over binary and primes fields. Their implementation of  $\eta_T$  uses the Karatsuba's multiplication method

and takes 10.96s to be evaluated.

## 4.7 Conclusion

In spite of intense research efforts, the achievement of security in WSNs using cryptography still requires solutions. On the other hand, the advent of PBC has enabled a wide range of new cryptographic solutions. In this work, we first have shown how security in WSNs can be bootstrapped using ID-NIKDS. Subsequently we have presented TinyPBC, to our knowledge, the most efficient implementation of PBC primitives for an 8-bit processor. TinyPBC is able to compute pairings, the most expensive primitive of PBC, in about 5.5s on ATmega128L and it is based on MIRACL, an open source library.

# Capítulo 5

## Conclusões e Trabalhos Futuros

RSSFs são compostas em sua maioria por pequenos sensores cujos recursos (energia, largura de banda, processamento etc.) são extremamente limitados. Estes sensores, por sua vez, se conectam com o mundo externo por meio de dispositivos mais poderosos chamados de sorvedouros ou ERBs. Elas são utilizadas com o intuito de monitorar regiões, oferecendo dados sobre a área monitorada, também chamada de área de interesse para o resto do sistema. Dentre sua vasta gama de aplicações estão operações de resgate em áreas de conflito/desastre, espionagem industrial e detecção de exploração ilegal de recursos naturais.

Como pôde ser observado ao longo desta tese, embutir segurança em RSSFs é uma tarefa complexa e muito desafiadora. Idealmente, um esquema de segurança para RSSFs deveria prover perfeita conectividade e resiliência. Além disso, o esquema deveria ser de baixo custo tanto em termos de processamento, como de comunicação e armazenamento. Tais esquemas por sua vez são usualmente alavancados através de técnicas de distribuição de chaves. O baixo poder computacional dos sensores, contudo, inviabiliza o emprego de criptossistemas tradicionais. Não obstante, há uma vasta gama de arquiteturas propostas para RSSFs e uma técnica de distribuição de chaves que é eficiente para uma pode não ser para outra, visto que diferentes arquiteturas de rede exibem padrões de comunicação diferentes.

O objetivo deste trabalho foi propor soluções de distribuição de chaves que fossem compatíveis com os recursos dos sensores e, simultaneamente, adequadas para as particularidades das arquiteturas para as quais são propostas.

Em nossa primeira solução de distribuição de chaves para RSSFs, LHA-SP (Capítulo 2), apresentamos um mecanismo para proteger redes hierárquicas e heterogêneas com qualquer número de níveis. Nossa proposta oferece segurança para as fases de configuração e re-configuração, bem como para o tráfego normal de operação da rede. O LHA-SP estabelece chaves par-a-par entre CHs e filhos usando técnicas eficientes (*lightweight*), ba-

seadas em chaves de grupo, sempre que possível, e recorrendo a mecanismos mais caros, por meio da ERB, quando necessário. Ademais, nossa solução é altamente distribuída, leva em consideração os padrões de interação específicos de RSSFs hierárquicas e permite a fusão de dados.

Em nossa segunda solução, batizada de SecLEACH (Capítulo 3), outro mecanismo de distribuição de chaves é apresentado, agora para proteger redes de agrupamentos. SecLEACH atinge seu objetivo através de uma adaptação da pré-distribuição aleatória de chaves. Nossas estimativas mostraram que o SecLEACH é eficiente e sua sobrecarga (*overhead*) gerenciável. Isto é, o uso de recursos como memória e energia pode ser balanceado dependendo do que é mais crítico para o sistema. Tal habilidade é especialmente útil em RSSFS, em que, como mencionado anteriormente, existe uma ampla gama de arquiteturas. Além disso, SecLEACH preserva a estrutura original do LEACH, incluindo sua capacidade de realizar fusão de dados.

No Capítulo 4 apresentamos TinyPBC, uma solução de distribuição de chaves utilizando IBC baseado em PBC. IBC, por sua vez, é uma área emergente, muito promissora, e vem possibilitando uma nova gama de aplicações e esquemas criptográficos. Primeiramente, defendemos a idéia de que IBC e RSSFs são muito compatíveis e, em seguida, descrevemos como IBC pode ser usada para solucionar o problema da distribuição de chaves no contexto de RSSFs – ou seja, estabelecendo chaves par-a-par entre quaisquer pares de sensores de forma autenticada e não interativa. Ao final, discutimos questões de implementação e apresentamos resultados do cálculo do emparelhamento  $\eta_T$  em sensores de recursos extremamente limitados. Resultados estes, até onde sabemos, os mais rápidos para uma arquitetura de 8 *bits*.

Em resumo, as principais contribuições desta tese foram:

1. apresentar a primeira proposta de segurança para RSSFs Hierárquicas com número arbitrário de níveis;
2. oferecer um conjunto de protocolos para proteger a configuração, operação e manutenção de RSSFs hierárquicas;
3. apresentar a primeira solução de segurança para proteger RSSFs com formação dinâmica de agrupamentos e rotativa de CHs;
4. mostrar como a pré-distribuição de chaves aleatórias pode ser empregada para proteger RSSFs com formação dinâmica de agrupamentos;
5. demonstrar como sensores podem estabelecer chaves par-a-par de maneira eficiente, autenticada e não interativa;

6. provar que o cálculo de emparelhamentos pode ser computado eficientemente mesmo em sensores com extrema escassez de recursos.

Foram publicados ao todo 22 trabalhos no decorrer desta tese. É importante dizer, ainda, que este trabalho principiou-se pouco depois que a área de segurança em RSSFs começou de fato a ser estudada. Foram anos de intensa pesquisa em que os trabalhos evoluíram de propostas inovadoras, sim, mas as vezes ingênuas, para soluções robustas e mais apropriadas a essa tecnologia. Deste ponto de vista, quando ainda acreditava-se na inexequibilidade de PKC em RSSFs, contribuímos com criptossistemas simétricos. Mais adiante, contribuímos com soluções de PKC as quais permitiram troca de chaves de maneira não autenticada – ideais para RSSFs em que a largura de banda é baixa e o custo de comunicação muito alto.

Para se mensurar a contribuição do nosso trabalho é importante, primeiro, colocar os resultados aqui apresentados sob perspectiva. Nossas soluções de pré-distribuição de chaves foram propostas quando não se havia, ainda, a alternativa do emprego de PKC em RSSFs. Mais que isso, quando principiamos este trabalho sequer existiam soluções de segurança na camada de enlace para RSSFs. Tivéssemos naquela época os “recursos de hoje” – a possibilidade de se utilizar PKC, por exemplo – talvez as direções tomadas tivessem sido diferentes.

Nosso trabalho também abriu vertentes de pesquisa onde ainda há muito o que se explorar. Tanto no campo de segurança em RSSFs hierárquicas/heterogêneas como na área de PBC aplicada a RSSFs – ou mesmo na junção de ambas – as oportunidades de pesquisa são imensas. Na primeira linha, pode-se explorar redes com diferentes níveis de heterogeneidade ou redes em que alguns dispositivos são pobres de alguns recursos, mas ricos em outros e vice-versa.

Em relação à pré-distribuição de chaves simétricas, é possível pensar em métodos híbridos, por exemplo, em que o chaveiro dos sensores são parcialmente constituídos por chaves que foram distribuídas aleatoriamente, e parcialmente constituídos por chaves par-a-par – este último quando uma chave foi atribuída a apenas dois sensores.

Talvez a linha de pesquisa mais promissora daqui por diante será a de PBC em RSSFs. Aqui, mostramos apenas como um dos muitos e úteis protocolos de PBC podem ajudar a proteger RSSFs. Existem protocolos, por exemplo, em que o esforço de computação é assimétrico; e seria interessante avaliá-los no contexto de RSSFs heterogêneas, outorgando os cálculos mais pesados aos sensores mais poderosos. Quanto à implementação, embora nossos números sejam eficientes, temos ciência que foi só o começo. Pode-se explorar computação distribuída, em RSSFs em que há recursos para comunicação; e pode-se investigar o uso de co-processadores ou até, futuramente, sensores *duo* processados.

Enfim, uma área ainda pouco explorada é a distribuição de chaves em RSSFs em que sensores e/ou ERBs são móveis. Devido às escassez de trabalhos nesta frente este é,

sem dúvida, um caminho a trilhar. É verdade que nossa solução não interativa atende a vários destes contextos, mas haverá situações de escassez de recursos extrema em que qualquer computação de PKC será proibitiva, e soluções terão que contar com primitivas exclusivamente simétricas.



# Referências Bibliográficas

- [1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, August 2002.
- [2] Ian F. Akyildiz, W. Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, December 2002.
- [3] Paulo S. L. M. Barreto, Steven Galbraith, Colm O hEigeartaigh, and Michael Scott. Efficient pairing computation on supersingular abelian varieties. In *Designs Codes And Cryptography*, 2006.
- [4] S. Bartolini, I. Branovic, R. Giorgi, and E. Martinelli. Effects of instruction-set extensions on an embedded processor: a case study on elliptic curve cryptography over  $\text{GF}(2^m)$ . *IEEE Transactions on Computers*, 2007. To appear.
- [5] Stefano Basagni, Kris Herrin, Danilo Bruschi, and Emilia Rosti. Secure pebblenets. In *2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 156–163. ACM Press, 2001.
- [6] Elizabeth M. Belding-Royer. Multi-level hierarchies for scalable ad hoc routing. *Wirel. Netw.*, 9(5):461–478, 2003.
- [7] Erik-Oliver Bläß and Martina Zitterbart. Towards Acceptable Public-Key Encryption in Sensor Networks. In *The 2nd Int’l Workshop on Ubiquitous Computing*. ACM SIGMIS, May 2005.
- [8] R. Blom. An optimal class of symmetric key generation systems. In *EUROCRYPT’84*, pages 335–338, 1984.
- [9] Carlo Blundo, Alfredo De Santis, Amir Herzberg, Shay Kutten, Ugo Vaccaro, and Moti Yung. Perfectly-secure key distribution for dynamic conferences. In *CRYPTO ’92*, pages 471–486, 1992.
- [10] M. Bohge and W. Trappe. An authentication framework for hierarchical ad hoc sensor networks. In *2003 ACM workshop on Wireless security*, pages 79–87, 2003.

- [11] Dan Boneh and Matthew Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003. Also appeared in CRYPTO '01.
- [12] S. Capkun, L. Buttyan, and J. P. Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1):17, january-march 2003.
- [13] Srdjan Capkun and Jean-Pierre Hubaux. Biss: building secure routing out of an incomplete set of security associations. In *ACM workshop on Wireless security (WISE'03)*, pages 21–29. ACM Press, 2003.
- [14] David W. Carman, Peter S. Kruus, and Brian J. Matt. Constraints and approaches for distributed sensor network security. Technical Report 00-010, NAI Labs, Network Associates, Inc., 2000.
- [15] Seyit A. Çamtepe and Bülent Yener. Combinatorial design of key distribution mechanisms for wireless sensor networks. In *9th European Symposium on Research Computer Security (ESORICS'04)*, pages 293–308, 2004.
- [16] Haowen Chan and Adrian Perrig. PIKE: Peer intermediaries for key establishment in sensor networks. In *The 24th Conference of the IEEE Communications Society (INFOCOM'05)*, March 2005.
- [17] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *IEEE Symposium on Security and Privacy (S&P'03)*, pages 197–213, may 2003.
- [18] Philip J. Clark and Francis C. Evans. Distance to nearest neighbor as a measure of spatial relationships in populations. *Ecology*, 35(4):445–53, 1954.
- [19] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *8th IMA Int'l Conference on Cryptography and Coding*, pages 360–363. Springer-Verlag, 2001.
- [20] Crossbow Technology, Inc., 41 Daggett Dr., San Jose, CA 95134. *MPR/MIB Mote Hardware Users Manual – Document 7430-0021-05*, December 2003.
- [21] Crossbow Technology, Inc., 41 Daggett Dr., San Jose, CA 95134. *Getting Started Guide Revision A – Document 7430-0022-04*, April 2004.
- [22] Sérgio de Oliveira, Hao Chi Wong, and José Marcos S. Nogueira. Nemap: Estabelecimento de chaves resiliente a intrusos em rssf. In *24º Simpósio Brasileiro de Redes de Computadores (SBRC 2006)*, Maio 2006.

- [23] Jing Deng, Richard Han, and Shivakant Mishra. Security support for in-network processing in wireless sensor networks. In *1st ACM workshop on Security of ad hoc and sensor networks (SASN'03)*, pages 83–93. ACM Press, 2003.
- [24] Data encryption standard modes of operation. Federal Information Processing Standards Publication, 1981.
- [25] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.
- [26] K. P. Donnelly. Simulations to determine the variance and edge effect of total nearest neighbour distance (ed. by i. hodder), 1978. Simulation Studies in Archaeology.
- [27] Barry Doyle, Stuart Bell, Alan F. Smeaton, Kealan McCusker, and Noel O'Connor. Security considerations and key negotiation techniques for power constrained sensor networks. *The Computer Journal*, 49(4):443–453, 2006.
- [28] W. Du, J. Deng, Y. S. Han, P. K. Varshney, J. Katz, and A. Khalili. A pairwise key pre-distribution scheme for wireless sensor networks. *ACM Trans. on Info. and System Security*, 8(2):228–58, 2005. Also in ACM CCS'03.
- [29] Wenliang Du, Jing Deng, Yunghsiang S. Han, Shigang Chen, and Pramod Varshney. A key management scheme for wireless sensor networks using deployment knowledge. In *Conference of the IEEE Communications Society (INFOCOM'04)*, 2004.
- [30] Wenliang Du, Ronghua Wang, and Peng Ning. An efficient scheme for authenticating public keys in sensor networks. In *6th ACM MobiHoc '05*, pages 58–67, New York, 2005.
- [31] S. Duquesne and T. Lange. Pairing-based cryptography. In G. Frey In H. Cohen, editor, *Handbook of Elliptic and Hyperelliptic Curve Cryptography*, chapter 24, pages 573–590. Chapman and Hall/CRC Press, Boca Raton FL, USA, 2006.
- [32] M. Duursma and Hyang-Sook Lee. Tate pairing implementation for hyperelliptic curves  $y^2 = x^p - x + d$ . In *9th Int'l Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'03)*, pages 111–123. Springer, 2003.
- [33] Laurent Eschenauer and Virgil D. Gligor. A key management scheme for distributed sensor networks. In *9th ACM conf. on Computer and communications security (CCS'02)*, pages 41–47, 2002.

- [34] Deborah Estrin, Ramesh Govindan, John S. Heidemann, and Satish Kumar. Next century challenges: Scalable coordination in sensor networks. In *MobiCom'99*, pages 263–270, 1999.
- [35] Yaacov Fernandess and Dahlia Malkhi. K-clustering in wireless ad hoc networks. In *2nd ACM international workshop on Principles of mobile computing*, pages 31–37. ACM Press, 2002.
- [36] A. C. Ferreira, M. A. Vilaça, L. B. Oliveira, E. Habib, H. C. Wong, and A. A. F. Loureiro. On the security of cluster-based communication protocols for wireless sensor networks. In *4th IEEE International Conference on Networking (ICN'05)*, volume 3420 of *Lecture Notes in Computer Science*, pages 449–458, April 2005.
- [37] S. D. Galbraith. Pairings. In Ian F. Blake, Gadiel Seroussi, and Nigel Smart, editors, *Advances in Elliptic Curve Cryptography*, volume 317 of *London Mathematical Society Lecture Notes*, chapter IX, pages 183–213. Cambridge Univ. Press, 2005.
- [38] S.D. Galbraith, K.G. Paterson, and N.P. Smart. Pairings for cryptographers. Cryptology ePrint Archive, Report 2006/165, 2006. <http://eprint.iacr.org/>.
- [39] Prasanth Ganesan, Ramnath Venugopalan, Pushkin Peddabachagari, Alexander Dean, Frank Mueller, and Mihail Sichitiu. Analyzing and modeling encryption overhead for sensor network nodes. In *ACM Int'l conf. on Wireless sensor networks and applications*, pages 151–159, 2003.
- [40] Germano Guimarães, Eduardo Souto, Judith Kelner, and Djamel H. Sadok. Avaliação de mecanismos de segurança em uma plataforma para redes de sensores sem fio. In *V Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg'05)*, Florianopolis, Brazil, September 2005.
- [41] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz. Comparing elliptic curve cryptography and rsa on 8-bit cpus. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES'04)*, pages 119–132, 2004.
- [42] Darrel Hankerson, Alfred Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography*. Springer, 2004.
- [43] Wendi Rabiner Heinzelman, Anantha Chandrakasan, and Hari Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *IEEE Hawaii Int. Conf. on System Sciences*, pages 4–7, january 2000.

- [44] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *9th inter'l conference on Architectural support for programming languages and operating systems*, pages 93–104, 2000.
- [45] Jason L. Hill and David E. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE Micro*, 22(6):12–24, 2002.
- [46] Yih-Chun Hu., A. Perrig, and D. B. Johnson. Packet leashes: A defense against wormhole attacks in wireless sensor networks. In *Conference of the IEEE Communications Society (INFOCOM'03)*, 2003.
- [47] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne:: a secure on-demand routing protocol for ad hoc networks. In *8th annual international conference on Mobile computing and networking*, pages 12–23. ACM Press, 2002.
- [48] Dijiang Huang, Manish Mehta, Deep Medhi, and Lein Harn. Location-aware key management scheme for wireless sensor networks. In *2nd ACM workshop on Security of ad hoc and sensor networks (SASN'04)*, pages 29–42. ACM Press, 2004.
- [49] Qiang Huang, Johnas Cukier, Hisashi Kobayashi, Bede Liu, and Jinyun Zhang. Fast authenticated key establishment protocols for self-organizing sensor networks. In *2nd ACM international conference on Wireless sensor networks and applications (WSNA'03)*, pages 141–150. ACM Press, 2003.
- [50] Jean-Pierre Hubaux, Levente Buttyán, and Srdan Capkun. The quest for security in mobile ad hoc networks. In *2nd ACM int'l symposium on Mobile ad hoc networking & computing*, pages 146–155, 2001.
- [51] Joengmin Hwang and Yongdae Kim. Revisiting random key pre-distribution schemes for wireless sensor networks. In *2nd ACM workshop on Security of ad hoc and sensor networks*, pages 43–52, 2004.
- [52] Antoine Joux. The weil and tate pairings as building blocks for public key cryptosystems. In *ANTS-V: the 5th Int'l Symposium on Algorithmic Number Theory*, pages 20–32, 2002.
- [53] Antoine Joux. A one round protocol for tripartite diffie-hellman. *J. Cryptology*, 17(4):263–276, 2004. Also in ANTS'00.
- [54] R. Kannan, L. Ray, and A. Duresi. Security-performance tradeoffs of inheritance based key predistribution for wireless sensor networks. In *1st European Workshop on Security in Wireless and Ad-Hoc Sensor Networks (ESAS'04)*, 2004.

- [55] A. Karatsuba and Y. Ofman. Multiplication of multidigit numbers on automata. *Soviet Physics-Doklad (Engl. transl.)*, 7(7):595–596, 1963.
- [56] C. Karlof, N. Sastry, and D. Wagner. Tinysec: A link layer security architecture for wireless sensor networks. In *2nd ACM SensSys*, pages 162–175, Nov 2004.
- [57] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Elsevier's AdHoc Networks Journal, Sp. Issue on Sensor Network Applications and Protocols*, 1(2–3):293–315, 2003. Also in 1st IEEE Int'l Workshop on Sensor Networks Protocols and Applications.
- [58] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48:203–209, 1987.
- [59] Jiejun Kong, Haiyun Luo, Kaixin Xu, Daniel Lihui Gu, Mario Gerla, and Songwu Lu. Adaptive Security for Multi-layer Ad-hoc Networks. *Wireless Communications and Mobile Computing, Wiley Interscience Press*, 2(5):533–547, 2002. Special Issue.
- [60] F. Daguan Leonardo B. Oliveira and R. Dahab. Avaliando protocolos de criptografia baseada em emparelhamentos em redes de sensores sem fio. In *7th Brazilian Symposium on Information and Computer System Security (SBSeg'07)*, 2007. short paper.
- [61] A. Liu, P. Kampanakis, and P. Ning. Tinyecc: Elliptic curve cryptography for sensor networks (ver. 0.3), 2005. <http://discovery.csc.ncsu.edu/software/TinyECC/>.
- [62] D. Liu and P. Ning. Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. In *10th Annual Network and Distributed Systems Security Symposium (NDSS'03)*, pages 263–276, 2003.
- [63] Donggang Liu and Peng Ning. Location-based pairwise key establishments for static sensor networks. In *1st ACM workshop on Security of ad hoc and sensor networks (SASN'03)*, pages 72–82. ACM Press, 2003.
- [64] Donggang Liu, Peng Ning, and Rongfang Li. Establishing pairwise keys in distributed sensor networks. *ACM Trans. on Info. and System Security*, 8(1):41–77, 2005. Also in ACM CCS'03.
- [65] Julio López and Ricardo Dahab. High-speed software multiplication in  $GF(2^m)$ . In *Progress in Cryptology - INDOCRYPT'00*, pages 203–212, 2000. Lecture Notes in Computer Science.

- [66] Antonio A. F. Loureiro. Grandes desafios da pesquisa em computação para o período 2006-2016. [www.ic.unicamp.br/~cmbm/desafios\\_SBC/loureiroredesensores.pdf](http://www.ic.unicamp.br/~cmbm/desafios_SBC/loureiroredesensores.pdf).
- [67] David J. Malan, Matt Welsh, and Michael D. Smith. A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography. In *1st IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON'04)*, 2004.
- [68] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Mobile Computing and Networking*, pages 255–265, 2000.
- [69] Enrique J. Duarte Melo and Mingyan Liu. The effect of organization on energy consumption in wireless sensor networks. In *IEEE Globecom 2002*, November 2002.
- [70] A. Menezes, T. Okamoto, and St Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Trans. on Information Theory*, 39(5):1639–1646, 1993.
- [71] Alexandre Menezes and Carlos Westphall. Interconectando clusters com transparência em rede de sensores sem fio segura. In *VI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg'06)*, Santos, Brazil, September 2006.
- [72] V. Mhatre, D. Kofman C. Rosenberg, R. Mazumdar, and N. Shroff. A minimum cost heterogeneous sensor network with a lifetime constraint. *IEEE Transaction on Mobile Computing*, 4(1):4–15, 2005.
- [73] Vivek Mhatre and Catherine Rosenberg. Design guidelines for wireless sensor networks: communication, clustering and aggregation. *Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, 2(1):45–63, January 2004.
- [74] Micaz: Wireless measurement system. [ttp://www.xbow.com/Products/Product\\_pdf\\_files/Wireless\\_pdf/MICAz\\_Dataseet.pdf](http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Dataseet.pdf).
- [75] V. Miller. Short program for functions on curves, 1986. unpublished manuscript.
- [76] V. Miller. Uses of elliptic curves in cryptography, advances in cryptology. In *Crypto'85, Lecture Notes in Computer Science*, volume 218, pages 417–426. Springer-Verlag, 1986.

- [77] Lama Nachman, Ralph Kling, Robert Adler, Jonathan Huang, and Vincent Hummel. The intel mote platform: a bluetooth-based sensor network for industrial monitoring. In *4th IEEE International Symposium on Information Processing in Sensor Networks (IPSN'05)*, pages 437–442, 2005.
- [78] James Newsome, Runtong Shi, Dawn Song, and Adrian Perrig. The sybil attack in sensor networks: Analysis and defenses. In *IEEE International Conference on Information Processing in Sensor Networks (IPSN 2004)*, April 2004.
- [79] Report of the national science foundation workshop on fundamental research in networking, April 2003. <http://www.cs.virginia.edu/~jorg/workshop1>.
- [80] Leonardo B. Oliveira, Diego Aranha, Eduardo Morais, Felipe Daguan, Julio López, and Ricardo Dahab. TinyTate: Computing the tate pairing in resource-constrained nodes. In *6th IEEE International Symposium on Network Computing and Applications (NCA'07)*, pages 318–323, July 2007.
- [81] Leonardo B. Oliveira and Ricardo Dahab. Pairing-based cryptography for sensor networks. In *5th IEEE International Symposium on Network Computing and Applications (NCA'06)*, July 2006. fast abstract.
- [82] Leonardo B. Oliveira, Ricardo Dahab, Julio Lopez, Felipe Daguan, and Antonio A. F. Loureiro. Identity-based encryption for sensor networks. In *5th IEEE Int'l Conference on Pervasive Computing and Communications Workshops (PERCOMW'07)*, pages 290–294, 2007.
- [83] Leonardo B. Oliveira, Adrian Ferreira, Marco A. Vilaça, Hao Chi Wong, Marshall Bern, Ricardo Dahab, and Antonio A. F. Loureiro. SecLEACH— on the security of clustered sensor networks. *Signal Process.*, 87(12):2882–2895, 2007.
- [84] Leonardo B. Oliveira, Michael Scott, Julio López, and Ricardo Dahab. Tinypbc: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. In *5th International Conference on Networked Sensing Systems (INSS'08)*, Kanazawa/Japan, 2008. To appear.
- [85] Leonardo B. Oliveira, Hao Chi Wong, Marshall Bern, Ricardo Dahab, and Antonio A. F. oureiro. SecLEACH – a random key distribution solution for securing clustered sensor networks. In *5th IEEE International Symposium on Network Computing and Applications (NCA'06)*, July 2006. p. 145-154.
- [86] Leonardo B. Oliveira, Hao Chi Wong, Ricardo Dahab, and Antonio A. F. Loureiro. On the design of secure protocols for hierarchical sensor networks. *International*



- Journal of Security and Networks (IJSN)*, 1(2):–, 2006. Special Issue on Cryptography in Networks, to appear.
- [87] Leonardo B. Oliveira, Hao Chi Wong, and Antonio A. F. Loureiro. Lha-sp: Secure protocols for hierarchical wireless sensor networks. In *9th IFIP/IEEE International Symposium on Integrated Network Management (IM'05)*, pages 31–44, Nice, France, May 2005. Top ranked paper.
- [88] Leonardo B. Oliveira, Hao Chi Wong, Antonio A. F. Loureiro, and Daniel M. Barbosa. Um protocolo de segurança para redes de sensores hierárquica. In *22o Simpósio Brasileiro de Redes de Computadores (SBRC) 2004*, pages 175–188, Gramado, Brasil, Maio 2004.
- [89] K. G. Paterson. Cryptography from pairings. In Ian F. Blake, Gadiel Seroussi, and Nigel Smart, editors, *Advances in Elliptic Curve Cryptography*, volume 317 of *London Mathematical Society Lecture Notes*, chapter X, pages 215–251. Cambridge Univ. Press, 2005.
- [90] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J. D. Tygar. SPINS: Security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, 2002. Also in MobiCom'01.
- [91] R. Di Pietro, L. V. Mancini, and A. Mei. Random key-assignment for secure wireless sensor networks. In *1st ACM workshop on Security of ad hoc and sensor networks (SASN'03)*, pages 62–71, 2003.
- [92] Roberto Di Pietro, Luigi V. Mancini, and Alessandro Mei. Efficient and resilient key discovery based on pseudo-random key pre-deployment. *Wireless Networks. special issue of IEEE WMAN '04 Best Papers*, 2005. To appear. Also appeared in 4th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks 2004 (WMAN'04).
- [93] Joseph Polastre, Robert Szewczyk, and David Culler. Telos: enabling ultra-low power wireless research. In *4th international symposium on Information processing in sensor networks IPSN '05*, page 48, Piscataway, NJ, USA, 2005. IEEE Press.
- [94] G. J. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Commun. ACM*, 43(5):51–58, 2000.
- [95] Bartosz Przydatek, Dawn Song, and Adrian Perrig. SIA: Secure information aggregation in sensor networks. In *ACM SenSys 2003*, Nov 2003.

- [96] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.
- [97] Ronald L. Rivest. The RC5 encryption algorithm. In Bart Preneel, editor, *Fast Software Encryption*, pages 86–96. Springer, 1995. (Proceedings Second International Workshop, Dec. 1994, Leuven, Belgium).
- [98] R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *Symposium on Cryptography and Information Security (SCIS'00)*, pages 26–28, Jan 2000.
- [99] B. Schneier. *Applied Cryptography*. Wiley, second edition, 1996.
- [100] Michael Scott. *MIRACL—A Multiprecision Integer and Rational Arithmetic C/C++ Library*. Shamus Software Ltd, Dublin, Ireland, 2003. Available at <http://indigo.ie/~mscott>.
- [101] Michael Scott. Computing the tate pairing. In *Topics in Cryptology - CT-RSA*, volume 3376 of *Lecture Notes in Computer Science*, pages 293–304. Springer, 2005.
- [102] Michael Scott. Optimal irreducible polynomials for  $GF(2^m)$  arithmetic. Cryptology ePrint Archive, Report 2007/192, 2007.
- [103] <http://sensoria.com/>.
- [104] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO'84*, pages 47–53. Springer-Verlag, 1984.
- [105] Douglas R. Stinson. *Cryptography: theory and practice*. CRC Press, Boca Raton, Florida, 1995.
- [106] Piotr Szczechowiak, Leonardo B. Oliveira, Michael Scott, Martin Collier, and Ricardo Dahab. NanoECC: Testing the limits of elliptic curve cryptography in sensor networks. In *European conference on Wireless Sensor Networks (EWSN'08)*, volume 4913, pages 305–320, Bologne/Italy, 2008. Lecture Notes in Computer Science.
- [107] Sameer Tilak, Nael B. Abu-Ghazaleh, and Wendi Heinzelman. A taxonomy of wireless micro-sensor network models. *ACM Mobile Computing and Communications Review*, 6(2):28–36, April 2002.
- [108] Tmote sky - ultra low power ieee 802.15.4 compliant wireless sensor module, 2006. <http://www.moteiv.com/products/docs/tmote-sky-datasheet.pdf>.

- [109] Lakshmi Venkatraman and Dharma P. Agrawal. A novel authentication scheme for ad hoc networks. In *IEEE Wireless Communications and Networking Conference*, pages 1268–1273, 2002.
- [110] Ronald J. Watro, Derrick Kong, Sue fen Cuti, Charles Gardiner, Charles Lynn, and Peter Kruus. TinyPK: securing sensor networks with public key technology. In *2nd ACM Workshop on Security of ad hoc and Sensor Networks (SASN'04)*, pages 59–64, 2004.
- [111] Anthony D. Wood and John A. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62, 2002.
- [112] Fan Yea, Haiyun Luo, Songwu Lu, and Lixia Zhang. Statistical en-route filtering of injected false data in sensor networks. In *Conference of the IEEE Communications Society (INFOCOM'04)*, 2004.
- [113] O. Younis and S. Fahmy. Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach. In *Conference of the IEEE Communications Society (INFOCOM'04)*, pages 629–640, 2004.
- [114] Y. Zhang, W. Liu, W. Lou, and Y. Fang. Securing sensor networks with location-based keys. In *IEEE Wireless Communications and Networking Conference (WCNC'05)*, 2005.
- [115] Yongguang Zhang and Wenke Lee. Intrusion detection in wireless ad-hoc networks. In *6th annual international conference on Mobile computing and networking*, pages 275–283. ACM Press, 2000.
- [116] Lidong Zhou and Zygmunt J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, 1999.
- [117] S. Zhu, S. Xu, S. Setia, and S. Jajodia. Establishing pairwise keys for secure communication in ad hoc networks: A probabilistic approach. In *11th IEEE Inter'l Conference on Network Protocols (ICNP'03)*, pages 326–335, Atlanta, Nov 2003.
- [118] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. LEAP: efficient security mechanisms for large-scale distributed sensor networks. In *10th ACM conference on Computer and communication security (CCS'03)*, pages 62–72. ACM Press, 2003.
- [119] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks. In *IEEE Symposium on Security and Privacy*, pages 259–271, 2004.