UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

MÁRIO TEIXEIRA LEMES

Um Esquema de Acordo de Chaves Baseado em Identidade para o *Framework* de Segurança TinySec

Goiânia
2014

UNIVERSIDADE FEDERAL DE GOIÁS

INSTITUTO DE INFORMÁTICA

AUTORIZAÇÃO PARA PUBLICAÇÃO DE DISSERTAÇÃO EM FORMATO ELETRÔNICO

Na qualidade de titular dos direitos de autor, **AUTORIZO** o Instituto de Informática da Universidade Federal de Goiás – UFG a reproduzir, inclusive em outro formato ou mídia e através de armazenamento permanente ou temporário, bem como a publicar na rede mundial de computadores (*Internet*) e na biblioteca virtual da UFG, entendendo-se os termos "reproduzir" e "publicar" conforme definições dos incisos VI e I, respectivamente, do artigo 5º da Lei nº 9610/98 de 10/02/1998, a obra abaixo especificada, sem que me seja devido pagamento a título de direitos autorais, desde que a reprodução e/ou publicação tenham a finalidade exclusiva de uso por quem a consulta, e a título de divulgação da produção acadêmica gerada pela Universidade, a partir desta data.

Título: Um Esquema de Acordo de Chaves Baseado em Identidade para o *Framework* de Segurança TinySec

Autor(a): Mário Teixeira Lemes

Goiânia, 21 de Fevereiro de 2014.

Mário Teixeira Lemes – Autor

Dr. Iwens Gervasio Sene Junior – Orientador

Dr. Renato de Freitas Bulcão Neto – Co-Orientador

MÁRIO TEIXEIRA LEMES

Um Esquema de Acordo de Chaves Baseado em Identidade para o *Framework* de Segurança TinySec

Dissertação apresentada ao Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Redes e Sistemas Distribuídos.

Orientador: Prof. Dr. Iwens Gervasio Sene Junior

Co-Orientador: Prof. Dr. Renato de Freitas Bulcão Neto

Goiânia
2014

MÁRIO TEIXEIRA LEMES

Um Esquema de Acordo de Chaves Baseado em Identidade para o *Framework* de Segurança TinySec

Dissertação defendida no Programa de Pós–Graduação do Instituto de Informática da Universidade Federal de Goiás como requisito parcial para obtenção do título de Mestre em Ciência da Computação, aprovada em 21 de Fevereiro de 2014, pela Banca Examinadora constituída pelos professores:

Prof. Dr. Iwens Gervasio Sene Junior
Instituto de Informática – UFG
Presidente da Banca

Prof. Dr. Renato de Freitas Bulcão Neto
Instituto de Informática – UFG

Prof. Dr. Vinicius da Cunha Martins Borges
Instituto de Informática - UFG

Profa. Dra. Michele Nogueira Lima
Departamento de Informática - UFPR

Mário Teixeira Lemes

Graduou-se em Engenharia de Computação pela Pontifícia Universidade Católica de Goiás (PUC-GO). Durante o mestrado em Ciência da Computação pela Universidade Federal de Goiás (UFG), atuou no desenvolvimento de pesquisas e soluções direcionadas ao problema de distribuição de chaves criptográficas em Redes de Sensores Sem Fio.

Dedico este trabalho ás pessoas mais importantes da minha vida: aos meus pais, á minha irmã e aos meus amigos.

Agradecimentos

"... nossas lembranças permanecem coletivas, e elas nos são lembradas pelos outros, mesmo que se trate de acontecimentos nos quais só nós estivemos envolvidos, e com objetivos que só nós vimos. É porque, em realidade, nunca estamos sós"(HALBWACHS, 1990, p.26). A conquista de um objetivo, assim como as lembranças, também nos é possibilitada pelas pessoas no qual compartilhamos nossas vidas. Sendo assim, reservo este espaço para agradecer as pessoas que me ajudaram, de uma forma ou outra, a conquistar meus sonhos e objetivos que almejo para minha vida.

Primeiramente agradeço em especial a Deus pelo dom da vida, por me dar saúde e iluminar meu caminho, sem sua divina intercessão nada disso seria possível.

Agradeço ao professor Dr. Iwens Gervasio Sene Junior, meu orientador de mestrado, por todo apoio, pela orientação prestada e pelo conhecimento transmitido ao longo da realização desta dissertação.

Ao professor Dr. Renato de Freitas Bulcão Neto, meu co-orientador, pela ajuda na correção dos artigos científicos, por suas dicas valiosas em relação ao trabalho que estava sendo desenvolvido.

Agradeço aos professores Dra. Solange da Silva e a ao meu orientador de graduação, o professor Dr. José Luiz de Freitas Júnior, por incentivarem o meu ingresso no programa de mestrado. Vocês foram muito importantes na minha formação pessoal e profissional. Sou grato pelos seus conselhos e pela amizade.

Aos meus amigos da turma do mestrado Roberto Vito Rodrigues Filho e Elio Ribeiro de Brito Filho, pelo tempo dedicado, pela paciência nas discussões nas reuniões e por suas considerações.

Agradeço de forma especial aos meus pais, Mário José Lemes Rodrigues e Terezinha de Fátima Teixeira Lemes, à minha irmã Thaís Teixeira Lemes, e a minha namorada Débora Martins por toda a paciência que tiveram comigo durante essa jornada, pelas orações nos momentos de dificuldades, por todo o carinho e compreensão. Eu amo muito vocês.

Por fim, agradeço também a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior e à Fundação de Amparo à Pesquisa do Estado de Goiás, pelo apoio financeiro.

"O futuro pertence àqueles que acreditam na beleza de seus sonhos"

Elleanor Roosevelt,

.

Resumo

Teixeira Lemes, Mário. **Um Esquema de Acordo de Chaves Baseado em Identidade para o *Framework* de Segurança TinySec**. Goiânia, 2014. 83p. Dissertação de Mestrado. Instituto de Informática, Universidade Federal de Goiás.

Os esquemas de distribuição de chaves criptográficas comumente são utilizados para alavancar propriedades de segurança em Redes de Sensores Sem Fio (RSSF). Nenhum mecanismo de distribuição de chaves é atrelado a arquitetura da camada de enlace TinySec, o que compromete consideravelmente o seu nível de segurança. O objetivo deste trabalho é propor uma abordagem de distribuição de chaves baseada em identidade para ser utilizada em conjunto com o *framework* TinySec, solucionando a fragilidade desta arquitetura de segurança por se basear em um esquema de estabelecimento de chaves muito simples: o compartilhamento de uma mesma chave criptográfica antes da fase de implantação. Este esquema de distribuição de chaves baseado em identidade utilizado em conjunto com o *framework* TinySec faz com que os danos ocasionados por ataques se tornem estritamente locais e permite que um nó sensor envie informações encriptadas para outro nó que ainda não tenha calculado o segredo criptográfico. A junção resulta em um protocolo com um maior nível de segurança sendo indicado para aplicações críticas que fazem uso das RSSF, tais como na área militar ou na área da saúde.

Palavras–chave

TinySec, Distribuição, Identidade

Abstract

Teixeira Lemes, Mário. **<A Identity-Based Key Agreeement for the Security Framework TinySec>**. Goiânia, 2014. 83p. MSc. Dissertation. Instituto de Informática, Universidade Federal de Goiás.

Key distribution schemes are commonly used to leverage security properties in Wireless Sensor Networks (WSN). No key distribution scheme is coupled to the link layer architecture TinySec, which significantly compromises their security level. The goal of this work is propose a new approach of identity-based key distribution to be used in conjunction with the framework TinySec, solving the weakness of this architecture that is based on a key establishment scheme very simple: the sharing of a same key before the deployment. The identity based key agreement to be use together with TinySec causes the damage from attacks become local and allows that a sensor node send encrypted information to another node the not yet calculated the secret key. The junction results in a protocol with high level of security and is suitable to critical applications of WSN, such as the military or in health.

Keywords

<TinySec, Key Distribution, Identity-Based.>

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Códigos de Programas

Lista de Abreviaturas e Siglas

RSSF	Redes de Sensores Sem Fio	17
PKC	*Public Key Cryptography*	22
RSA	*Rivest, Shamir and Adleman*	22
ECC	*Elliptic Curve Cryptography*	22
PKI	*Public Key Infrasctruture*	22
PBC	*Pairing Based Cryptography*	22
IBE	*Identity Based Encryption*	22
IBKA	*Identity Based Key Agreement*	22
GF	*Galois Field*	28
DLP	*Discret Logarithm Problem*	30
ECDLP	*Elliptic Curve Discrete Logarithm Problem*	30
NAF	*Non Adjacent Form*	32
BREW	*Binary Environment for Wireless*	36
EB	Estação Base	40
SPIN	*Sensor Protocol for Information via Negotiation*	40
LEAP	*Localized Encryption and Authentication Protocol*	40
KDC	*Key Distribution Center*	40
IV	*Initialization Vector*	51
NESC	*Network Embedded Systems C*	59

CAPÍTULO 1

Introdução

As Redes de Sensores Sem Fio (RSSF) podem ser definidas como uma classe especial das redes *ad hoc* de múltiplos saltos e são compostas por pequenos dispositivos autônomos que processam dados [14]. Estes dispositivos que compõem as RSSF possuem recursos limitados de processamento, armazenamento, comunicação e energia, além dos sensores propriamente ditos [13]. O baixo custo e a rapidez na implantação fazem com que as RSSF sejam atrativas para diversas aplicações, tais como monitoramento da saúde, proteção de construções, operações de vigilância e proteção ambiental [28].

Os avanços na eletrônica sem fio e nas tecnologias de comunicação permitiram o desenvolvimento em larga escala de RSSF que emergiram como um novo tipo de solução de controle e monitoramento para várias aplicações ubíquas. A computação ubíqua é um paradigma de interação usuário-computador que tem como objetivo o suporte ao usuário em suas tarefas habituais fora do ambiente usual de trabalho [66, 23]. A presença de dispositivos computacionais em todo o ambiente é a característica marcante da computação ubíqua. Os objetos do mundo real são equipados com processadores e sensores embutidos proporcionando assim novas formas de acesso á informação [29].

As aplicações desenvolvidas para as RSSF, que podem ser caracterizadas como uma parte da computação ubíqua, possuem diferentes requisitos de processamento, armazenamento, energia e segurança (integridade, confidencialidade e autenticidade). Independente de qual seja a aplicação , os dados coletados pela RSSF devem ser transmitidos de forma exata, a integridade dos dados deve ser mantida entre o emissor e o receptor da mensagem.

A confidencialidade da informação, ou seja, a garantia que apenas partes autorizadas têm acesso a uma determinada informação, é um requisito de segurança que também deve ser garantido pelas aplicações de RSSF. Os dados referentes às aplicações de monitoramento de aspectos climáticos de uma floresta ou características de animais em uma região não são sigilosos, porém a divulgação destes dados poderia levar a prejuízos ou vantagem a concorrência.

Tanto para aplicações de automação residencial quanto para aplicações ligadas à área da saúde, além da confidencialidade, é importante garantir a autenticidade dos dados

e dos dispositivos. Outro requisito de segurança importante é a garantia da disponibilidade das aplicações, ou seja, os serviços oferecidos devem estar disponíveis para as partes autorizadas no momento em que for necessário.

Além dos requisitos já citados (integridade, confidencialidade, autenticidade dos dados e dispositivos e disponibilidade), também é importante assegurar que os dados enviados pelos nós sensores são recentes (característica de *freshness*), deve-se certificar que nenhum intruso está replicando dados antigos. O mesmo conceito pode ser aplicado às chaves criptográficas (*key freshness*) em uso na RSSF, fornecidas pelos mecanismos de distribuição e gerenciamento de chaves criptográficas [42].

O termo chave criptográfica se refere ao segredo utilizado pelos algoritmos de criptografia para encriptação e decriptação dos dados. Os mecanismos responsáveis pelo gerenciamento de chaves criptográficas devem garantir que as chaves em uso nas RSSF são recentes, impedindo o uso de chaves antigas que poderiam ter sido obtidas após o ataque a um nó [26].

Como em qualquer rede sem fio *ad hoc*, as RSSF são vulneráveis a diversos ataques [62]. Os ataques às redes de sensores podem ser classificados em duas categorias: passivos ou ativos [2]. Os ataques passivos são aqueles em que o atacante não interfere no funcionamento da RSSF, apenas monitora e captura os dados trafegados. Já nos ataques ativos, um adversário pode realizar a captura física de um nó, alterá-lo e logo depois repô-lo na rede. Pode interceptar, retransmitir e injetar mensagens, fazendo-se passar ou não, por um nó legítimo ou ainda copiar a identidade de um.

Para tratar os problemas relacionados aos ataques, diversos mecanismos de segurança têm sido implementados, como o uso de criptografia. Algumas arquiteturas de segurança foram desenvolvidas, dentre elas destaca-se o TinySec [30].

O *framework* da camada de enlace TinySec é uma das arquiteturas mais populares e uma das soluções de segurança mais utilizadas em RSSF por ter uma implementação real, incorporada em uma versão oficial do TinyOS [49], enquanto outras propostas não foram totalmente validadas ou implementadas na prática [5], tais como o *Security Protocols for Sensor Networks* (SPINS) [48] e o *Link Layer Security Protocol* (LLSP) [50].

Um aspecto crucial no uso de algoritmos de criptografia em RSSF é como as chaves criptográficas são distribuídas e gerenciadas. Os esquemas de distribuição de chaves criptográficas são comumente utilizados para alavancar (*bootstrapped*) propriedades de segurança em RSSF e devem fornecer um bom balanceamento entre conectividade da rede, resiliência contra captura de nós e poder de escalabilidade.

A expressão conectividade da rede se refere à probabilidade dos sensores estabelecerem chaves compartilhadas. A resiliência indica qual a possibilidade de toda RSSF ser comprometida caso um adversário ataque um nó fisicamente e recupere informações secretas a partir de sua memória. A escalabilidade se refere à flexibilidade da adição

de novos nós. Dentre os métodos existentes de distribuição de chaves criptográficas em RSSF, a pré-distribuição [68] é um dos mais indicado devido á inserção das chaves nos nós sensores em um momento seguro (antes da fase de implantação (*deployment*)) [51].

1.1 Segurança em RSSF

A segurança é um dos temas mais importantes e desafiadores em RSSF [64]. Para que uma RSSF seja considerada segura, é importante que a mesma tenha soluções de segurança implementadas nos seguintes aspectos:

- Gerenciamento e Distribuição de Chaves Criptográficas;
- Privacidade e Confidencialidade da Informação;
- Roteamento Seguro;
- Sistema de Detecção de Intrusos;
- Integridade dos Dados e Dispositivos;
- Autenticação das Entidades;
- Agregação de Dados Segura;

Os requisitos de segurança confidencialidade da informação e integridade dos dados são alcançados com a incorporação de simples mecanismos de segurança na camada de enlace dos dados. Os mecanismos de detecção de intrusos são resposáveis por detectar se alguém está tentando entrar no sistema ou se algum usuário legítimo está fazendo mau uso do mesmo, tal medida de segurança vai além do escopo de solução de segurança apresentada nesta dissertação.

A distribuição de chaves criptográficas entre nós sensores é um dos serviços de segurança mais importantes em RSSF e é a principal motivação para este trabalho. Um mecanismo adequado de distribuição de chaves é uma premissa essencial que permite que outras primitivas de segurança sejam construídas corretamente.

Outras características de segurança, tais como: o roteamento seguro, a confidencialidade das mensagens e a autenticação de entidades só podem ser construídas adequadamente se houverem mecanismos seguros de distribuição de chaves criptográficas. Estes mecanismos vão garantir que o *bootstrapping* da RSSF seja feito de forma segura e correta. A partir da inicialização segura, outros mecanismos de segurança devem ser implementados para que a RSSF seja considerada totalmente segura. Note na Figura 1.1 algumas características de segurança desejáveis.

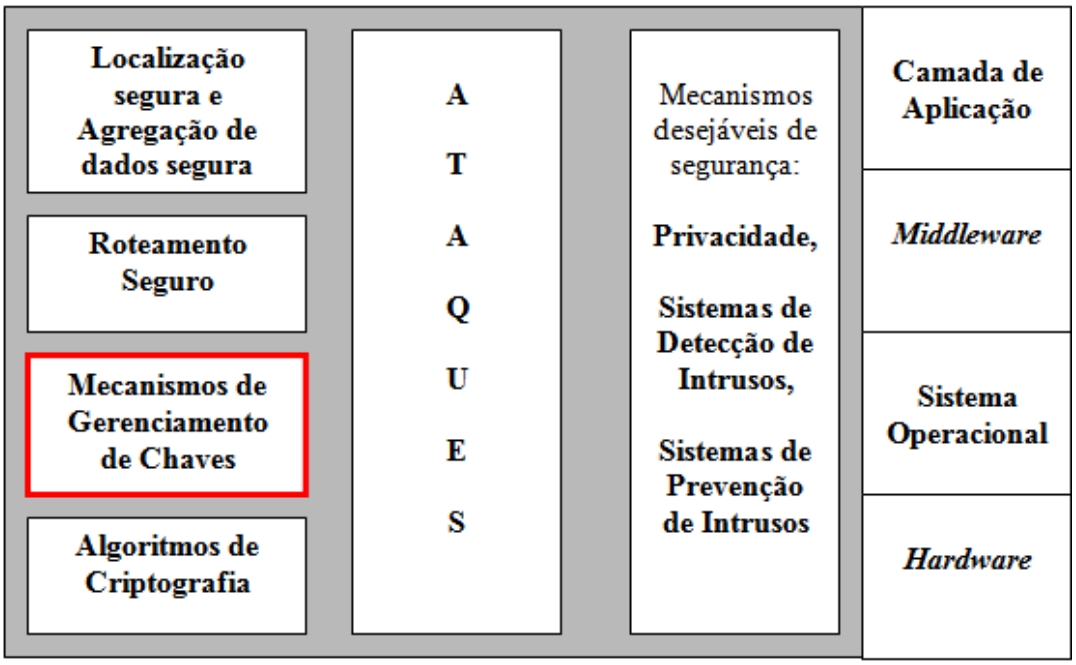

Figura 1.1: *Mapa de Segurança para RSSF*

Na ordem do desenvolvimento de soluções de segurança para RSSF, alguns desafios inerentes a estes ambientes devem ser superados. Dentre alguns desses desafios, podem-se citar:

- Limitação no *Hardware*;
- Comunicação Sem Fio;
- Escalabilidade;
- Versatilidade;
- Ataques Físicos;

A limitação na arquitetura de *hardware* dos sensores é uma característica que deve ser contornada pelos desenvolvedores de aplicações para RSSF. As RSSF são compostas tipicamente por equipamentos que possuem recursos limitados, com baixo poder de processamento, pouca memória para armazenamento dos dados, baixa largura de banda e fonte de alimentação com capacidade limitada. Os nós sensores utilizam baterias como fonte de energia possuindo assim um fornecimento limitado de energia.

A própria natureza da comunicação sem fio das RSSF é outro desafio. A comunicação sem fio é um tipo de comunicação passiva a diversos tipos de ataques, tais como a espionagem e a injeção de pacotes maliciosos. Um outro problema adicional é a largura de banda limitada presente nesses cenários.

As soluções de segurança devem ser flexíveis, considerando o poder de armazenamento da maioria dos sensores um recurso limitado, as soluções desenvolvidas devem ser escaláveis, isto é, devem permitir a inclusão de novos nós sem comprometer substancialmente a eficiência da RSSF.

A versatilidade é um desafio em um ambiente que possui diferentes plataformas de *hardware*. Além da variedade de sensores, as soluções de segurança projetadas para um cenário de rede específico normalmente não são adequados a um cenário diferente.

Além dos problemas já citados, as RSSF não possuem mecanismos seguros de armazenamento dos dados. Isso significa que um acatante pode capturar um determinado nó e recuperar dados sigilosos a partir de sua memória. O baixo custo dos sensores excluem a possiblidade do uso de *hardware* a prova de ataques.

Estas limitações ocasionam um profundo impacto na adoção dos protocolos e algoritmos de comunicação e de segurança. Todas as escolhas relacionadas com a segurança devem ser cuidadosamente analisadas levando em conta as restrições destes ambientes. Todos estes desafios fazem com que a aplicação de segurança em RSSF seja mais desafiadora do que em redes tradicionais, já que em ambientes de rede tradicionais não há a escassez de recursos computacionais.

1.2 O Problema da Distribuição de Chaves em RSSF

As soluções de distribuição de chaves que se baseiam na criptografia simétrica, no qual uma mesma chave é utilizada para encriptação e decriptação dos dados, são simples e de fácil gerenciamento. O uso desta abordagem não oferece um nível ideal de resiliência, já que o comprometimento de apenas um único nó pode comprometer toda a RSSF. Para aumentar a segurança nesses esquemas existe a possibilidade do compartilhamento de chaves entre cada par de nós. O problema desta abordagem alternativa é que, para RSSF de larga escala, o armazenamento de muitas chaves criptográficas não é um problema trivial.

Apesar de serem mais eficientes, o uso de criptossistemas simétricos para distribuição de chaves criptográficas em RSSF não oferecem níveis ideais de resiliência contra captura de nós e poder de escalabilidade. A maioria dos esquemas simétricos são dependentes de alguma interação entre os sensores para efetuar o acordo de chaves (*key agreement*) [62], o que pode representar uma vulnerabilidade de segurança, uma vez que a necessidade de interação entre os sensores para o acordo de chaves pode ocasionar diversos tipos de ataques.

Os esquemas de distribuição de chave randômica são também métodos simples e de baixa complexidade computacional. Nestes esquemas, um conjunto de chaves é carregado na memória de cada sensor antes da fase de implantação. Durante o funcionamento da rede, através de um processo de descoberta, inicia-se a identificação da intersecção não vazia do conjunto de chaves de um sensor com todos os outros sensores. Devido à natureza randômica do esquema de distribuição, pode não haver uma chave secreta compartilhada

por dois nós sensores quaisquer. A principal falha dessa abordagem é a possiblidade de vizinhos não possuírem chaves em comum, impossibilitando a comunicação.

Os esquemas arbitrados de distribuição de chaves dependem de nós especiais, com maior responsabilidade do que simples nós sensores, para estabelecer e gerenciar as chaves da rede. Esta abordagem mostra-se bastante atrativa caso entidades com maior poder de processamento já estejam disponíveis, pois elas podem concentrar o ônus de processamento de operações complexas. Este costuma ser o caso de RSSF hierárquicas, nas quais as entidades em questão são as estações-base ou os *cluster heads*. O maior risco neste caso é que tais entidades podem se tornar um alvo preferencial de ataques, os quais podem afetar parcelas consideráveis da RSSF caso sejam bem sucedidos [28].

Os esquemas mais seguros de distribuição de chaves criptográficas em RSSF são baseados em criptossistemas assimétricos ou de chave pública (*Public Key Cryptography* - PKC). Os algoritmos mais estudados são o RSA (*Rivest, Shamir and Adleman*) e o ECC (*Elliptic Curve Cryptography*), sendo que o RSA é classificado como um algoritmo convencional/tradicional e o ECC um algoritmo alternativo/moderno [47].

Com a utilização do ECC obtém-se o mesmo nível de segurança que o RSA, porém consumindo poucos recursos computacionais [62]. O ECC, comparado com o RSA, oferece rápida computação, economia nos recursos de memória, energia e largura de banda. Entretanto, o nível de segurança alcançado por meio da utilização destes algoritmos de chave pública têm um custo alto: a necessidade de certificação das chaves públicas dos sensores através de uma Infraestrutura de Chaves Públicas (*Public Key Infrasctruture* - PKI) , o que é inviável ser realizado dentro da RSSF [17, 62, 12, 32].

A maioria dos esquemas de distribuição de chaves padrões presentes na literatura [54], são inadequados para RSSF: esquemas baseados em PKC convencionais pela alta demanda de processamento e necessidade de uma PKI para certificação das chaves públicas; chaves globais simétricas, carregados na memória de cada nó antes da fase de implantação, pela alta vulnerabilidade de segurança; esquemas par-a-par, aqueles que cada nó mantém uma lista das $n-1$ chaves dos outros n sensores da rede, pelo alto consumo de memória.

1.3 Motivação

A Criptografia Baseada em Emparelhamentos (*Pairing Based Cryptography* - PBC) [53] possibilita o projeto e a utilização de esquemas criptográficos originais de uma maneira mais eficiente. Dentre as aplicações de PBC para resolver o problema de distribuição de chaves criptográficas em RSSF, está a Encriptação Baseada em Identidade (*Identity Based Encryption* - IBE) [11], mais especificamente os esquemas de Acordo de Chaves Baseados em Identidade (*Identity Based Key Agreement* - IBKA). IBE foi

originalmente proposto por Shamir em [56], mas apenas se tornou viável com o advento de PBC [33].

No IBE as informações que identificam unicamente os usuários (IP, endereço de email, serial do sensor) são utilizadas para troca de chaves e encriptação dos dados. Devido á uma importante propriedade matemática denominada emparelhamento bilinear ou simplesmente emparelhamento, os nós sensores são capazes de calcular, de forma não interativa, uma mesma chave criptográfica de sessão, e esta pode ser utilizada para encriptação/decriptação da comunicação.

Para dois nós quaisquer A e B realizarem o acordo de chaves através do uso de IBE, o nó A necessita apenas de sua chave privada e a chave pública de B (que pode ser obtida através da identidade pública de B) para calcular o emparelhamento. Outra característica interessante é que esse tipo de abordagem dispensa a necessidade de certificados para a autenticação dos nós através de uma PKI [46, **?**].

Os esquemas de IBKA são baseados em PBC e no ECC. O uso de esquemas IBKA para o acordo de chaves em RSSF é muito interessante, pois através deles dois ou mais nós têm a capacidade de calcular uma mesma chave de sessão sem a necessidade de interação entre os mesmos, o que é bastante útil, visto que os nós sensores podem ter padrões de desligamento, ser empregados em momentos diferentes ou se tornarem temporariamente indisponíveis devido a obstáculos ou mau funcionamento [47].

O *framework* de segurança da camada de enlace mais referenciada da literatura, o TinySec, não contempla mecanismos para troca de chaves,o que sacrifica consideravelmente o nível de seguraça fornecido por essa camada. Os esquemas de distribuição IBKA são a solução ideal para o estabelecimento de chaves criptográficas em RSSF, simplificando o processo de distribuição e resolvendo a forma inadequada que o *framework* simétrico TinySec utiliza para estabelecimento destas chaves.

1.4 Objetivos

O objetivo deste trabalho é propor uma abordagem para o problema de distribuição de chaves criptográficas apresentado pelo TinySec, solucionando a fragilidade desta arquitetura por se basear em um esquema de estabelecimento de chaves muito simples: o compartilhamento de uma única chave antes da fase de implantação.

A nova abordagem de distribuição de chaves proposta para ser utilizada em conjunto com o TinySec baseia-se na utilização de um esquema de IBKA para solucionar a maneira inadequada como esse estabelecimento de chave é realizado pelo TinySec. A proposta do uso conjunto de um mecanismo de distribuição de chaves e a arquitetura de segurança TinySec têm como objetivo fornecer aos desenvolvedores de aplicações de RSSF uma maneira prática para que suas aplicações: i) primeiramente utilizem o

mecanismo de estabelecimento e distribuição de chaves proposto e ii) posteriormente recorram ao *framework* Tinysec para autenticação e/ou encriptação dos dados.

A proposta é que o esquema de distribuição de chaves seja utilizado unicamente no *bootstrapping* da rede. Após os nós concordarem uma chave de sessão em comum, métodos de encriptação mais eficientes (baseados na criptografia simétrica), mais especificamente o *framework* simétrico TinySec deve ser utilizado para proteção da comunicação.

1.5 Metodologia

Para alcançar o objetivo principal deste trabalho, diversas tarefas secundárias foram desenvolvidas. Tendo em vista todo o processo de conduta da pesquisa, as tarefas completadas foram:

1. Investigação da aplicação de PBC em RSSF.
2. Investigação do uso de IBE, mais especificamente de esquemas IBKA, como solução para o problema de distribuição de chaves criptográficas em RSSF.
3. Estudo dos princípios matemáticos de PBC e de ECC, como por exemplo funções de emparelhamento bilinear, tipos de emparelhamentos, algoritmos de emparelhamentos, curvas elípticas, tipos de curvas, e etc.
4. Estudo das bibliotecas criptográficas de PBC disponíveis (TinyPairing [67], RELIC-*toolkit* [3], NanoPBC [4]) para que uma delas fosse utilizada como base da implementação da junção proposta;
5. Investigação do funcionamento do *framework* de segurança da camada de enlace TinySec, dos modos de autenticação e/ou encriptação, dos componentes e das interfaces.
6. Investigação da linguagem nesC [20], do sistema operacional TinyOS [49] em sua versão 1.x e do simulador Avrora [63] para testes e validação das implementações.

1.6 Contribuições

As contribuições deste trabalho de dissertação são:

1. Proposta de um novo esquema de distribuição de chaves criptográficas baseado em identidade para ser utilizado em conjunto com o *framework* de segurança TinySec.
2. Implementação/Adequação de um esquema de acordo de chaves baseado em identidade baseado no trabalho conduzido em [44], utilizando a biblioteca criptográfica RELIC-*toolkit* [3], a fim de ser utilizada como mecanismo de endereçamento de chaves do TinySec.

3. Implementação de uma aplicação que testa a junção entre o esquema de acordo de chaves e o *framework* de segurança TinySec.
4. Verificação dos requisitos de hardware requeridos devido a junção. Dentre as variáveis analisadas, podem-se citar: quantidade necessária de memória *RAM* e *ROM* e tempo total de execução da aplicação.
5. Verificação dos requisitos de segurança adicionados com a utilização do esquema proposto juntamente com o *framework* TinySec;

1.7 Estrutura da Dissertação

Este trabalho está divido em seis capítulos. O Capítulo 1 é constituído basicamente por este capítulo, no qual contêm a introdução sobre o problema de distribuição de chaves criptográficas em RSSF, os motivos que ocasionaram o estudo deste assunto, os objetivos que foram traçados e as contribuições alcançadas.

O Capítulo 2 é composto por princípios matemáticos básicos para o entendimento de PBC e sua aplicação em RSSF. Fundamentos matemáticos de ECC também são necessários, uma vez que os grupos cíclicos usados no emparelhamento bilinear são grupos de pontos sobre curvas elípticas especiais [1].

As técnicas de distribuição de chaves criptográficas em RSSF e as métricas utilizadas para avaliação da viabilidade da aplicação destes esquemas são apresentadas no Capítulo 3. Também é apresentado o estado da arte dos trabalhos que abordam o problema de distribuição de chaves em RSSF. O estado da arte contempla apenas soluções de distribuição de chaves criptográficas baseadas em criptossistemas simétricos, assimétricos tradicionais/alternativos, e baseados em emparelhamentos, o foco de pesquisa dessa dissertação.

O *Framework* de segurança da camada de enlace TinySec é apresentado no Capítulo 4. As peculiaridades desta arquitetura são mostradas, tais como os modos de funcionamento, o formato dos pacotes, os componentes e as interfaces disponíveis.

O Capítulo 5 é responsável pela apresentação dos resultados. Mostra-se também como foi realizada a junção do protocolo de estabelecimento de chave baseado em identidade e o TinySec. São dadas duas visões diferentes da junção, uma geral, com o intuito de orientação dos desenvolvedores de aplicações para RSSF para habilitação do uso do processo de estabelecimento de chaves antes da utilização do TinySec, e outra, mais específica, mostrando as características do protocolo de estabelecimento de chave. São apresentados também os custos adicionados ao TinySec com a utilização do esquema de troca de chaves e os requisitos de segurança adicionados com a junção.

[1] Veja mais detalhes no Capítulo 2

Finalmente no Capítulo 6 encontram-se as considerações finais, as dificuldades encontradas, a definição de possíveis trabalhos futuros e as aprentações/publicações alcançadas durante a realização deste trabalho.

CAPÍTULO 2

Fundamentos Matemáticos

Neste capítulo são apresentados fundamentos matemáticos básicos necessários para a compreensão de criptossistemas baseados em emparelhamento. Como primitivas de ECC são requisitos para a construção de criptossistemas completos de PBC, também são dadas algumas noções sobre curvas elípticas, operações aritméticas sobre essas curvas e formas de representações de pontos.

2.1 Grupo

Grupo é um conjunto não vazio dotado de uma operação $\circ$. Um grupo possui as seguintes propriedades:

- Possui um elemento identidade.
- Possui o elemento inverso.
- Associativo.
- Fechado.

Quando um grupo é definido para operações de adição pode-se dizer que é um grupo aditivo, representado aqui por $\mathbb{G}_1$; quando um grupo é definido para operações de multiplicação pode-se dizer que é um grupo multiplicativo, analogamente representado por $\mathbb{G}_2$.

A aplicação da operação $\circ$ sobre o mesmo elemento Q do grupo n vezes, com n um número natural, resulta em: $Q \circ Q \circ Q \circ \ldots \circ Q$. Para $Q \in G_1$ esta operação é representada por nQ, no caso de $Q \in G_2$ representa-se por Q^n .

Se através de um elemento Q pode-se representar todos os elementos de um grupo (Q operado sobre ele mesmo), o elemento Q é chamado elemento gerador deste grupo. Se um grupo possui um elemento gerador, então ele é chamado de grupo cíclico [22].

Tabela 2.1: *Operações aritméticas no conjunto* $\mathbb{F}_{15}$

Operação	Sem o módulo	Com o Módulo	Resultado
Adição	9 + 8	$9 + 8 \mod 15 = 17 \mod 15$	2
Subtração	9 - 8	$9 - 8 \mod 15 = 1 \mod 15$	1
Multiplicação	9×8	$9 \times 8 \mod 15 = 72 \mod 15$	12
Inverso	8^{-1}	$8 \times 2 \mod 15 = 16 \mod 15$	1

2.2 Corpos Finitos

Corpos são abstrações dos sistemas numéricos e suas propriedades básicas. Um corpo é dito finito se sua ordem, ou seja, se seu número de elementos é um número finito e consiste de um conjunto $\mathbb{F}$ definido sob duas operações binárias: a adição $(+)$ e a multiplicação $(\cdot)$. Essas operações satisfazem as seguintes propriedades:

- O grupo $\mathbb{F}$ sob a operação de $+$ é um grupo abeliano com elemento identidade denotado por 0.
- O grupo $\mathbb{F}$ sob a operação de $\cdot$ é um grupo abeliano com elemento identidade denotado por 1.
- Regra distributiva, ou seja, $(a+b) \times c = a \times c + b \times c$.

2.2.1 Corpos Primos

Existe um corpo finito ou um *Galois Field* (GF) de ordem q se e unicamente se q é o único primo. Tal corpo é denotado como $GF(q)$ ou $\mathbb{F}_q$ com $q = p^m$ onde p é um número primo chamado característica de $\mathbb{F}$ e m é um número inteiro positivo chamado extensão do corpo.

Se $m = 1$, então $\mathbb{F}$ é chamado de corpo primo; caso contrário, com $m \succeq 2$, então $\mathbb{F}$ é definido sobre a extensão de um corpo. Para determinado primo q, o corpo finito de ordem q, $GF(q)$, é definido pelo conjunto Z_q de inteiros $\{0, 1, \ldots, q-1\}$ juntamente com as operações aritméticas no módulo q. Veja na Tabela 2.1 um exemplo de operações aritméticas no conjunto $\mathbb{F}_{15}$.

2.2.2 Corpos Binários

Corpos finitos de ordem 2^m são conhecidos como corpos binários ou corpos finitos de característica dois, e são denotados por $\mathbb{F}_{2^m}$. A escolha de corpos binários é vantajosa, já que a representação interna dos dados nos computadores também é binária.

Os corpos binários podem ser construídos utilizando representações polinomiais. Nesse caso, os elementos do corpo $\mathbb{F}_{2^m}$ são polinômios binários, ou seja, os coeficientes dos polinômios estão no corpo $\mathbb{F}_2 = \{0, 1\}$ com grau máximo igual a m - 1

$$\mathbb{F}_{2^m} = a_{m-1}z^{m-1} + a_{m-2}z^{m-2} + \ldots + a_2z^2 + a_1z + a_0 : a_i \in \{0,1\} \quad (2\text{-}1)$$

Um elemento importante na aritmética de corpo finito é o polinômio irredutível de grau m. O polinômio é dito irredutível porque não pode ser fatorado como o produto de polinômios binários cujo grau seja menor que m.

Considere o corpo $\mathbb{F}_{2^3}$, o polinômio irredutível:

$$f(z) = z^3 + z + 1 \quad (2\text{-}2)$$

e o conjunto de elementos do corpo $\{0,1,z,z+1,z^2,z^2+1,z^2+z,z^2+z+1\}$. A adição e a subtração de elementos do corpo $\mathbb{F}_{2^m}$ são realizadas como na adição e na subtração usuais de polinômios, com coeficientes reduzidos no módulo 2. Veja na Equação 2-3 como é realizada a adição de dois elementos:

$$(z^2+z+1)+(z^2+1) \bmod 2 = 2z^2+z+2 \bmod 2 = z \quad (2\text{-}3)$$

O resultado da subtração é o mesmo da adição, desde que $a = -a$ para todo $a \in \mathbb{F}_{2^m}$. O mesmo ocorre com a multiplicação de elementos do corpo, mas o método de redução é feito com o polinômio irredutível da Equação 2-2.

2.2.3 Corpos de Extensão

Um corpo de extensão pode ser obtido a partir da representação da base de polinômios sobre corpos binários. Dessa forma, corpos de extensão podem ser representados por $\mathbb{F}_{2^m}$, onde m representa a extensão do corpo.

A adição de dois polinômios é feita pela adição de seus coeficientes no módulo p. A multiplicação de um corpo de extensão é computada pela multiplicação completa de dois polinômios e da subsequente redução modular do polinômio irredutível $f(z)$ de grau de incorporação k.

Corpos de extensão são utilizados para construir sistemas de Curvas Elípticas e frequentemente postos na forma de Corpos de Extensão Ótimos. A idéia atrás de Corpos de Extensão Ótimos (F_p^k) é selecionar p,k, e um polinômio de redução $f(z)$ mais semelhantes possíveis com as características do *hardware*.

2.3 Curvas Elípticas

Curvas elípticas são definidas sobre corpos binários $\mathbb{F}_{2^m}$, com $m \succeq 1$ ou sobre corpos primos $\mathbb{F}_q$, com $q \succeq 3$. Seja q um número primo grande, uma curva elíptica $\mathbb{E}_q$

definida sobre o corpo finito $\mathbb{F}_q$ é dada por:

$$y^2 = x^3 + ax + b | a, b \in \mathbb{F}_q \tag{2-4}$$

A curva elíptica $\mathbb{E}_q$ é então um conjunto de pontos $P(x,y)$ no qual $x, y \in \mathbb{F}_q$, satisfaz a equação da curva $y^2 = x^3 + ax + b \mid a, b \in \mathbb{F}_q$, mais um ponto extra, chamado ponto no infinito, denotado por θ e $\Delta \neq 0$, onde Δ é o discriminante da curva.

A segurança de criptossistemas baseados em curvas elípticas baseia-se na dificuldade de resolução do Problema de Logaritmo Discreto (*Discret Logarithm Problem* - DLP) sobre um grupo de curva elíptica. O Problema de Logaritmo Discreto Sobre Curva Elíptica (*Elliptic Curve Discrete Logarithm Problem* - ECDLP) definido sobre $\mathbb{F}_q$ pode ser caracterizado como: dado P e $kP \in E$, é muito difícil determinar o valor de k.

2.3.1 Curvas Supersingulares e Não-Supersingulares

Se uma curva elíptca E é definida sobre $\mathbb{F}_q$ então o número de pontos da curva, nomeadamente o símbolo $\#E(\mathbb{F}_q)$, é de aproximadamente:

$$q + 1 - 2\sqrt{q} \preceq \#E(\mathbb{F}_q) \preceq q + 1 + 2\sqrt{q} \tag{2-5}$$

sendo que q é a ordem do corpo finito.

Se o valor: $|t| \preceq 2\sqrt{q}$ for divisível pela característica de $\mathbb{F}_q$, então a curva elíptica é conhecida como supersingular. Uma curva elíptica supersingular têm ordem de grupo $q+1$.

Já se o valor $|t| \preceq 2\sqrt{q}$ não for divisível pela característica de $\mathbb{F}_q$, a curva elíptica é não-supersingular. As curvas elípticas supersingulares têm relação direta com criptossistemas baseados em PBC.

2.3.2 Operações sobre Curvas Elípticas

As principais operações sobre curvas elípticas são:

- Identidade
- Negativo
- Adição de ponto
- Duplicação de ponto
- Multiplicação de ponto escalar

Considere a seguinte curva supersingular:

$$(E/\mathbb{F}_{2^m}) : y^2 + y = x^3 + ax + b \tag{2-6}$$

tem-se as seguintes operações:

Identidade

$A + \theta = \theta + A$, para todo $A \in (E/\mathbb{F}_{2^m})$.

Negativo

Se um ponto $A = (x,y) \in (E/\mathbb{F}_{2^m})$ e se $(x,y) + (x,y+c) = 0$, então o ponto $(x,y+c)$ é denotado por $-A$ e é chamado de negativo de A.

Adição de Ponto

Seja os pontos $P = (x_1,y_1) \in (E/\mathbb{F}_{2^m})$ e $Q = (x_2,y_2) \in (E/\mathbb{F}_{2^m})$ e $P = \pm Q$. Então, a adição de $P+Q$, representado pelo ponto (x_3,y_3), vale:

$$x_3 = \left(\frac{y_1+y_2}{x_1+x_2}\right)^2 + x_1 + x_2 \tag{2-7}$$

$$y_3 = \left(\frac{y_1+y_2}{x_1+x_2}\right)(x_1+x_3) + y_1 + c \tag{2-8}$$

Duplicação de ponto

Seja o ponto $A = (x_1,y_1) \in (E/\mathbb{F}_{2^m})$, com $A = -A$. Então $2A = (x_3,y_3)$, onde:

$$x_3 = \left(\frac{{x_1}^2 + a}{c}\right)^2 \tag{2-9}$$

$$y_3 = \left(\frac{{x_1}^2 + a}{c}\right)(x_1+x_3) + y_1 + c \tag{2-10}$$

Um outro conceito importante a respeito de curvas elípticas é o grau de imersão. Seja $(E/\mathbb{F}_q)$ uma curva elíptica definida sobre o corpo $\mathbb{F}_q$. Seja também r um inteiro positivo, coprimo a q, tal que $(E/\mathbb{F}_q)$ contém um ponto de ordem r. Em implementações de criptografia, r é geralmente um número primo grande que divida a ordem da curva, ou seja, $r|\#(E/\mathbb{F}_q)$. Seja k o menor inteiro que satisfaz $r|q^k - 1$. O valor de k consiste no grau de imersão da curva [38].

Multiplicação de ponto escalar

A operação de multiplicação de ponto escalar é uma das principais operações sobre curvas elípticas, e é a operação que demanda mais tempo em criptossistemas

baseados em ECC. Se P é um ponto sobre uma curva elíptica $E(\mathbb{F}_q)$, o resultado da multiplicação escalar kP é um outro ponto sobre a mesma curva.

Uma das formas de se calcular o valor kP é através do método binário direita-para-esquerda. O Algoritmo 2.1 [15], processa os bits de k um-por-um da direita para a esquerda. Como pode ser visto no algoritmo, a eficiência da multiplicação de ponto escalar depende do quantidade do número 1 na representação binária de k.

O número de 1's na representação binária de k pode ser reduzido usando diferentes técnicas de representação. O Número $255P$ em binário pode ser representado por:

$$(1111111)_2 P \tag{2-11}$$

Da mesma forma, o número $255P$ também pode ser representado por:

$$(10000000-1)P = 256P - P \tag{2-12}$$

Na Equação 2-11 é necessário 8 adições de ponto. Já na Equação 2-12 é necessário apenas 1 adição de ponto e 1 subtração.

Algoritmo 2.1: Método direita-para-esquerda para Multiplicação de Ponto

Entrada: $k = (k_{t-1}, \cdots, k_1, k_0)_2$, $P \in E(\mathbb{F}_q)$
Saída: $Q = kP$

for $i \leftarrow 0$ **to** $t-1$ **do**
    if $k_i = 1$ **then**
        $Q \leftarrow Q + P$
    $P = 2P$
return Q

Outra forma para representação de k na operação de ponto de multplicação é o uso da Forma Não-Adjacente (*Non-Adjacent Form* - NAF) . A NAF de um inteiro k pode ser representada como:

$$\sum_{i=0}^{l-1} k_i 2^i \tag{2-13}$$

sendo $k_i \in \{0, \pm 1\}$ e $k_{l-1} = 0$. A representação NAF de k têm poucos dígitos não-zeros de qualquer representação de k. Os dígitos de $NAF(k)$ podem ser calculados pela divisão repetitiva de k por 2 com os remanescentes de 0 e ± 1. O uso da representação NAF de k no Algoritmo 2.1 reduz o número de pontos de adição necessários em 33%.

2.3.3 Representações de Pontos de Curvas Elípticas

O sistema de representação de pontos de curvas elípticas têm grande influência na performance das operações aritméticas sobre a curva. A escolha do sistema de coordenadas no projeto de protocolos de distribuição de chaves criptográficas é uma escolha crítica, pois o número de *bits* transmitidos devem ser minimizados ao máximo. A coordenada padrão (x,y) é uma das escolhas mais adequadas porque inclui apenas dois valores para transmissão, diferentemente dos outros sistemas de coordenadas. Essa Seção é responsável por mostrar os principais sistemas de coordenadas usados em criptossistemas baseados em ECC.

Coordenadas Afinitivas

Representa o sistema de coordenada padrão (x,y).

Coordenadas Projetivas

O ponto (X,Y,Z) sobre uma curva E corresponde a coordenada padrão $(X/Z,Y/Z)$ quando $Z \neq 0$ e o ponto no infinito $\theta = \{0,1,0\}$.

Coordenadas Jacobianas

O ponto (X,Y,Z) sobre uma curva E em coordenadas jacobianas correspondem a coordenada padrão $(X/Z^2,Y/Z^3)$ quando $Z \neq 0$ e o ponto no infinito $\theta = \{1,1,0\}$. Operações de duplicação de ponto são mais eficientes nas coordenadas jacobianas em relação as coordenadas projetivas. Já em relação as operações de ponto de adição a situação se inverte.

Coordenadas López-Dahab

São projetadas para curvas de característica dois, ou seja, para curvas definidas sobre $\mathbb{F}_{2^m}$. Um ponto (X,Y,Z) sobre uma curva E em coordenadas Lópes-Dahab corresponde a coordenada padrão $(X/Z,Y/Z^2)$ quando $Z \neq 0$ e e o ponto no infinito $\theta = \{1,0,0\}$. Segundo [36], coordenadas López-Dahab são a escolha ótima para o cálculo da operação de pontos de multiplicação sobre curvas definidas sobre corpos binários.

2.4 Emparelhamento Bilinear

Um emparelhamento bilinear é definido como um isomorfismo que mapeia elementos de um conjunto de pontos de uma curva $(E/\mathbb{F}_q)$ para um subgrupo de um corpo de extensão de $\mathbb{F}_q$.

Sejam $\mathbb{G}_1$ e $\mathbb{G}_2$ grupos cíclicos de ordem prima q. Um emparelhamento bilinear admissível, de acordo com a definição em [11], é um mapeamento $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$, que satisfaz as seguintes condições:

1. Bilinear: para qualquer $P,Q \in \mathbb{G}_1$ e a, $b \in Z_q$, temos: $\hat{e}(aP,bQ) = \hat{e}(P,Q)ab = \hat{e}(abP,Q) = \hat{e}(P,abQ)$.
2. Não Degenerado: não leva todos os pares $\mathbb{G}_1 \times \mathbb{G}_1$ à identidade em $\mathbb{G}_2$.
3. Computável: existe algoritmo eficiente que calcula $\hat{e}(P,Q)$ para todos $P,Q \in \mathbb{G}_1$.

Pode-se definir que o emparelhamento é um mapa bilinear entre dois grupos cíclicos. O Emparelhamento de Tate e o Emparelhamento ηT sobre curvas elípticas são exemplos de tais mapas. Os emparelhamentos que são realizados sobre curvas elípticas supersingulares tem a propriedade adicional de simetria: $\hat{e}(P,Q) = \hat{e}(Q,P)$.

Na prática, o grupo $\mathbb{G}_1$ é implementado usando grupos de pontos sobre curvas elípticas especiais e $\mathbb{G}_2$ é implementado usando um subgrupo multiplicativo de uma extensão de um corpo finito.

2.4.1 Tipos de Emparelhamentos Bilineares

A classificação do tipo de emparelhamento depende do uso dos grupos cíclicos. Basicamente existem três tipos:

- Tipo 1: $\mathbb{G}_1 = \mathbb{G}_2$.
- Tipo 2: $\mathbb{G}_1 = \mathbb{G}_2$ e existe um isomorfismo eficientemente computável φ: $\mathbb{G}_2 \rightarrow \mathbb{G}_1$.
- Tipo 3: $\mathbb{G}_1 = \mathbb{G}_2$ e não existe um isomorfismo eficientemente computável φ: $\mathbb{G}_2 \rightarrow \mathbb{G}_1$.

O emparelhamento tipo 1 existe apenas sobre curvas elípticas supersingulares. Para todas as curvas elípticas supersingulares, o grau de imersão é um dos valores $k = \{1,2,3,4,5,6\}$. Se uma curva elíptica é definida sobre um corpo primo $\mathbb{F}_q$, com $q > 3$, então o valor de k pode ser um dos valores: $\{1,2\}$. Todas as curvas elípticas supersingulares com $k = 4$ são definidas sobre corpos de característica dois, já aquelas com $k = 6$ são definidas sobre corpos de característica três.

Um mapa de distorção Ψ existe em curvas supersingulares, no qual mapeia um ponto de uma curva definida sobre um corpo finito $(E/\mathbb{F}_q)$ para a extensão do corpo, $(E/\mathbb{F}_{q^k})$. O emparelhamento tipo 1, como visto na Seção 2.4, têm a propriedade adicional de simetria: $\hat{e}(P,Q) = \hat{e}(Q,P)$.

Quando o ponto $Q = xP$, então:

$$\hat{e}(P,Q) = \hat{e}(P,xP) = \hat{e}(P,P)^x = \hat{e}(xP,P) = \hat{e}(Q,P) \tag{2-14}$$

A propriedade de simetria contida na Equação 2-14 é útil para simplificar protocolos de segurança, inclusive protocolos de distribuição de chaves baseados em PBC.

2.4.2 Algoritmos de Emparelhamento

Emparelhamento de Weil e Tate

Os Emparelhamentos mais utilizados, o de Weil e o de Tate, possuem a seguinte forma:

$$\hat{e} : E(\mathbb{F}_q)[r] \times E(\mathbb{F}_{q^k}) \rightarrow \mathbb{F}^*_{q^k} \tag{2-15}$$

ou seja, associam pares de pontos em curvas elípticas a um elemento de um corpo finito. O Emparelhamento de Tate é mais eficiente que o de Weil, e por isso é mais preferido na prática [8].

O Algoritmo 2.2 [43] calcula os emparelhamentos de Weil ou Tate em tempo polinomial. O algoritmo tem $O(\log r)$ iterações, cada uma requerendo um número constante de operações aritméticas no corpo de extensão $\mathbb{F}_{q^k}$.

Algoritmo 2.2: Algoritmo de Miller para computar $\hat{e}(P,Q)$

Entrada: $Q \in \mathbb{F}_{q^k}, Q \in \mathbb{F}_{q^k}$, P tem ordem $r = (r_t, \cdots, r_0)_2$ e $t = \lfloor \log r \rfloor - 1$
Saída: $\hat{e}(P,Q)$

for $i \leftarrow t$ **to** 0 **do**
  $f \leftarrow f^2 \cdot \frac{l_{T,T}(Q)}{v_{2T}(Q)}$
  $T \leftarrow 2T$
  if $r_i = 1$ **then**
    $f \leftarrow f^2 \cdot \frac{l_{T,P}(Q)}{v_{T+P}(Q)}$
    $T \leftarrow T + P$
$f \leftarrow f^{(q^{k-1})/r)}$
return f

O algoritmo de Miller precisa ser otimizado para ser aplicado de forma eficiente em dispositivos embarcados [59].

Emparelhamento Eta-t (η_T)

O trabalho conduzido em [7] propôs um novo algoritmo chamado Emparelhamento η_T. A implementação do Emparelhamento η_T em quatro diferentes nós sensores para RSSF (MICA2, TmoteSky, Imote2 (13MHz), Imote2 (104MHz)) é mais rápida e consome menos energia do que o Emparelhamento Tate [61]. Veja na Tabela 2.2 a comparação entre os dois emparelhamentos em quatro nós sensores:

Tabela 2.2: *Custo do emparelhamento em diferentes sensores*

	MICA2		Tmote Sky		Imote2 (13MHz)		Imote2 (104MHz)	
Emparelhamento	Tate	Eta-t	Tate	Eta-t	Tate	Eta-t	Tate	Eta-t
Tempo	7.43	2.66	4.61	1.71	0.62	0.46	0.08	0.06
ROM (KB)	60.51	47.41	34.9	23.66	44.4	25.55	44.4	29.55
Pilha (KB)	3.39	3.17	3.39	4.17	3.75	4.12	4.12	3.75

Emparelhamento Ate

O Emparelhamento Ate [24] é relativamente parecido com o Emparelhamento Eta-t. A comparação entre os dois emparelhamentos foi realizada em [27] utilizando um Ambiente de Execução Binário (*Binary Environment for Wireless* - BREW) . A Tabela 2.3 mostra a comparação entre o Emparelhamento Eta-t e o Ate, nos níveis de 80-*bits* e 128-*bits*, no mote ARM9 225MHz.

Tabela 2.3: *Emparelhamento Tate x Emparelhamento Eta-t*

Emparelhamento	Eta-t		Ate	
Nível de Segurança (*bits*)	80	128	80	128
Tempo (segundos)	0.26	3.76	0.7	1.6

No nível de 80-*bits*, o Emparelhamento Eta-t é mais rápido que o Emparelhamento Ate. Contudo, para um nível maior de segurança (128-*bits*), o Emparelhamento Ate é cerca de duas vezes mais eficiente que o Emparelhaento Eta-t.

O Emparelhamento Ate é o emparelhamento mais rápido para algumas curvas elípticas e provê um maior nível de segurança, porém não existem trabalhos do emprego deste emparelhamento em protocolos baseados em PBC para uso em RSSF [52].

CAPÍTULO 3

Distribuição de Chaves Criptográficas em RSSF

Neste capítulo são apresentadas as técnicas de distribuição de chaves criptográficas existentes em RSSF. Para cada tipo de abordagem é dada a descrição do funcionamento do esquema e as vantagens e desvantagens do uso como solução para o estabelecimento de chaves criptográficas.

3.1 Divisão das Técnicas de Distribuição

Os esquemas de distribuição de chaves criptográficas comumente são utilizados para alavancar (*bootstrapped*) propriedades de segurança em RSSF. Em geral, as técnicas básicas de distribuição de chaves criptográficas são divididas nas seguintes categorias:

1. Esquemas de Pré-Distribuição
   - Soluções baseadas em criptossistemas simétricos ou de chave única.
   - Esquemas de pré-distribuição de chave randômica.
2. Esquemas Arbitrados
3. Esquemas Auto Regulados
   - Soluções baseadas em criptossistemas assimétricos ou de chave pública convencionais.
   - Soluções baseadas em criptossistemas assimétricos alternativos/modernos.
   - Soluções baseadas em variantes de criptossistemas assimétricos.

3.2 Formas de Avaliação

Para avaliar a aplicabilidade das técnicas de distribuição de chaves citadas na Seção 3.1 são utilizadas algumas métricas para avaliar os requisitos relacionados com a segurança, a eficiência e a flexibilidade. As métricas de avaliação mais comuns, de acordo com [28], estão listadas a seguir:

1. Métricas de Segurança:

- Autenticação dos Nós - Idealmente as técnicas de gerenciamento de chave devem possibilitar que os nós verifiquem a identidade dos outros nós participantes da RSSF de uma maneira segura.
- Resiliência - Refere-se a resistência do esquema contra a captura de nós, no qual um adversário ataca fisicamente um sensor e recupera informações secretas de sua memória.
- Revocação de Nós - Soluções de gerenciamento de chaves devem fornecer métodos dinâmicos de revogação de nós compromissados.

2. Métricas de Eficiência:
   - Memória - O uso de memória total para armazenamento dos dados (chaves criptográficas, identificadores, parâmetros públicos, e etc).
   - Processamento - A quantidade de ciclos do processador para realizar o estabelecimento das chaves.
   - Largura de Banda - A quantidade de dados trocados entre os nós durante o processo de geração das chaves.
   - Energia - O consumo de energia envolvido no processo de acordo de chaves.
   - Conectividade da Rede - Probabilidade dos nós sensores estabelecerem chaves compartilhadas.
3. Métricas de Flexibilidade:
   - Conhecimento prévio após fase de implantação - Refere-se a necessidade do conhecimento prévio da posição final dos nós após a fase de implantação. Técnicas de distribuição de chaves mais flexíveis não devem depender do conhecimento prévio da posição dos nós para inicialização do acordo de chaves.
   - Escalabilidade - Os esquemas de distribuição de chaves ideais devem suportar RSSF de larga escala, e ao mesmo tempo, permitir a introdução de novos nós sem ocasionar problemas de segurança.

3.3 Abordagem Clássica de Distribuição de Chaves

A abordagem clássica de distribuição de chaves criptográficas envolve os esquemas de pré-distribuição baseados em criptossistemas simétricos e baseados em esquemas randômicos. A abordagem clássica também envolve as tentativas de adequação do uso de algoritmos assimétricos convencionais para solucionar o problema de estabelecimento e gerenciamento de chaves criptográficas em RSSF.

3.3.1 Criptografia Simétrica

Segundo [57], a criptografia simétrica é uma forma de criptossistema em que a chave usada para realizar a criptografia e a decriptografia é a mesma. Em outras palavras, a criptografia simétrica realiza a criptografia de um texto claro com uma chave secreta e um algoritmo de criptografia. Usando a mesma chave secreta e um algoritmo de decriptografia, consegue-se obter o texto claro a partir do texto cifrado. Na criptografia simétrica, têm-se as seguintes características:

- Texto claro - Refere-se ao texto original sem nenhuma alteração. Este texto serve como entrada para o algoritmo de criptografia.
- Chave secreta - Refere-se a um valor independente do texto claro e do algoritmo. A chave secreta mais o algoritmo de criptografia, aplicado ao texto claro, gera um texto cifrado. Um texto cifrado, junto com a mesma chave secreta e um algoritmo de decriptografia, gera o texto claro.
- Texto cifrado - É o texto obtido da criptografia de um texto claro.
- Algoritmo de Criptografia - Refere-se ao algoritmo utilizado em conjunto com a chave secreta para gerar o texto cifrado.
- Algoritmo de Decriptografia - Refere-se ao algoritmo utilizado em conjunto com a chave secreta para gerar o texto claro.

Apresenta-se na Figura 3.1 o funcionamento da criptografia simétrica:

Figura 3.1: *Modo de Funcionamento da Criptografia Simétrica*

No contexto de RSSF, para a criptografia simétrica funcionar, dois nós devem basicamente obter a mesma chave secreta e esta chave então é utilizada para proteção de toda a comunicação. Esta chave deve ser protegida e mantida em sigilo a fim de evitar o acesso não autorizado por outros nós.

O grande problema da aplicação da criptografia simétrica se resume na forma em que as chaves secretas são implantadas dentro da RSSF, o que é de fato um problema não-trivial. Basicamente o uso de criptografia simétrica como solução para o estabelecimento de chaves criptográficas em RSSF possui três abordagens distintas:

1. Emprego de uma chave global.
2. Compartilhamento de uma chave par-a-par entre a Estação Base (EB) e o sensores.
3. Compartilhamento de uma chave secreta entre sensores comuns.

Em relação ao problema de distribuição de chaves criptográficas em RSSF utilizando criptografia simétrica, destacam-se o protocolo (*Sensor Protocol for Information via Negotiation* - SPIN) [48], que é constituído por dois blocos de segurança: o SNEP e o μTESLA. O SNEP provê confidencialidade e autenticação dos dados e a garantia que os dados são recentes. Já o μTESLA fornece autenticação *broadcast* para os sensores. O protocolo SPIN lida com três tipos de padrões de comunicação:

- Nó sensor para a EB.
- EB para um determinado nó.
- EB para todos os nós.

No protocolo SPIN, cada sensor compartilha uma chave secreta pré-distribuída com a EB. Todas as outras chaves são inializadas a partir desta chave secreta inicial. Um dos inconvenientes deste protocolo é que o mesmo não considera diferentes requisitos de segurança para diferentes mensagens dentro da rede.

Outro protocolo baseado em criptossistemas simétricos é o (*Localized Encryption and Authentication Protocol* - LEAP) [71]. O protocolo LEAP resolve o problema da diversidade dos requisitos de segurança das mensagens dentro da RSSF estabelecendo quatro chaves para cada sensor:

- Uma chave individual compartilhada com a EB.
- Uma chave compartilhada entre cada par de sensores.
- Uma chave de *cluster* compartilhada por sensores dentro de um mesmo *cluster*.
- Uma chave de grupo compartilhada por todos os nós sensores.

Em [30], o *framework* TinySec foi desenvolvido para possibilitar a integração de diversas aplicações em RSSF. Ao contrário do SPIN [48] e do LEAP [71], o TinySec não é ligado a nenhum mecanismo de distribuição de chaves. O TinySec é a primeira implementação de um protocolo de criptografia na camada de enlace para RSSF.

As soluções de distribuição de chaves baseadas em criptografia simétrica são soluções simples e de fácil gerenciamento. A adoção de uma única chave global para todos os sensores faz com que o gerenciamento das chaves seja bastante simples, porém o uso desta abordagem oferece baixo nivel de resiliência, uma vez que o comprometimento de apenas um único nó pode comprometer toda a RSSF.

Para contornar o problema, tem-se a abordagem alternativa de incorporação de um Centro de Distribuição de Chaves (*Key Distribution Center* - KDC). O KDC é

responsável pela distribuição e gerenciamento de chaves para cada par de nós caso os mesmos queiram se comunicar. A exigência de uma infraestrutura fixa com servidores confiáveis limita a adoção desta abordagem,

Outra abordagem alternativa para aumentar a segurança em criptossistemas simétricos é o compartilhamento de chaves entre cada par de nós. Neste caso, cada sensor teria uma lista de $n-1$ chaves, uma para cada um dos outros $n-1$ nós de uma RSSF de n nós. O problema desta abordagem alternativa é que, para RSSF de larga escala, o armazenamento de muitas chaves criptográficas se torna algo complicado devido as restrições de armazenamento.

Apesar do ganho em relação a eficiência, o uso de criptossistemas simétricos para distribuição de chaves criptográficas em RSSF não oferecem niveis ideais de resiliência e escalabilidade. A adição de novos nós é problemática e requer novas chaves para cada novo nó dentro da RSSF.

As cifras simétricas são viáveis para a encriptação dos dados, contudo a forma que as chaves são estabelecidas e gerenciadas é realizada de forma inadequada. A maioria dos esquemas simétricos são dependentes de alguma interação entre os sensores para efetuar o acordo de chaves.

A forma utilizada por criptossistemas simétricos para distribuição de chaves mostra como essas técnicas são limitadas. Essas técnicas sempre requerem que dois nós já compartilhem uma chave (através da pré-distribuição) ou pelo recebimento de chaves por meio de um KDC.

3.3.2 Criptografia Assimétrica

Diferentemente do método que utiliza chave simétrica, esse tipo utiliza duas chaves, uma pública e um privada. O sistema funciona a partir da geração de um par de chaves, onde uma dessas, a chave pública, é enviada ao emissor. Com ela é feita a criptografia da mensagem. Para o receptor decriptografar a mensagem é necessário utilizar a outra chave, a privada, que é secreta.

Apresenta-se na Figura 3.2 um exemplo em que a chave pública do receptor é compartilhada entre emissor e o receptor da mensagem. Note que a chave secreta do receptor é mantida em sigilo.

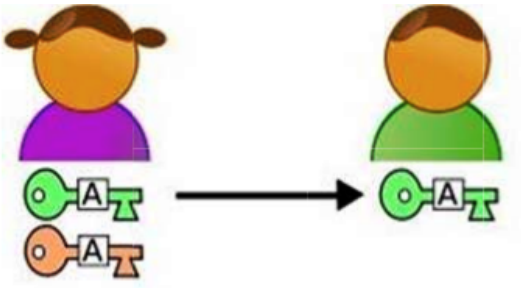

Figura 3.2: *Compartilhamento da Chave Pública*

Em criptossistemas assimétricos, ás vezes a criptografia da mensagem é realizada com a chave privada do emissor e decriptada com a chave pública do mesmo. Esse procedimento é utilizado na criação de assinaturas digitais.

Em RSSF, os esquemas mais seguros de distribuição de chaves criptográficas são baseados em criptossistemas assimétricos ou de chave pública. O algoritmo RSA é um dos mais clássicos e diversas foram as tentativas de adequá-lo para o uso em nós sensores com recursos limitados. Um outro algoritmo que se destaca é o ECC. Este último é considerado um algoritmo de chave pública alternativo/moderno.

No contexto de RSSF, o implantador da rede é obviamente uma entidade confiável, o mesmo pode implantar o par de chaves pública e privada na memória dos nós. Assim, um KDC não é mais necessário. Um nó pode enviar sua chave pública para os outros nós via *broadcast*, os nós que a recebem a utilizam para encriptar as mensagens.

O uso do ECC obtém o mesmo nível de segurança do RSA, porém consumindo menos recursos computacionais [62]. Veja na Tabela 3.1 o tamanho das chaves utilizadas em criptossistemas baseados em ECC quando comparados com o RSA dado um nível de segurança. Os níveis especificados correspondem a diferentes tipos de cifras de bloco e o tamanho de chave correspondente para cada cifra. A diferença é ainda maior quando os níveis de segurança aumentam.

Porém, o uso do RSA e do ECC exigem a existência de uma PKI[1], o que como foi visto anteriormente, não é possível devido as restrições dos nós sensores [17, 62, 12, 32].

A maioria das deficiências de protocolos baseados em criptografia simétrica podem ser superadas utilizando criptossistemas baseados em PKC. De acordo com [1], o RSA não é adequado para RSSF por causa da complexidade de tempo e pelo alto demanda de energia.

Ainda na tentativa da aplicação do algoritmo RSA em RSSF, os autores em [65] propuseram o *framework* TinyPK. Neste esquema, as operaçẽs tradicionais de chave privada são atribuídas a uma entidade externa que administrava a RSSF.

[1] Veja a Seção 3.4 para mais detalhes

Tabela 3.1: *Tamanho de Chave do ECC x RSA*

Cifra	Skipjack-80	3-DES-112	AES-128	AES-256	AES-512
ECC	160	224	256	384	512
RSA	1024	2048	3072	8192	15360

As operações tradicionais de chave privada do RSA são inviáveis de serem realizadas pelos nós devido suas restrições de *hardware*. A fatoração do número inteiro *N*, que no RSA é o produto de dois números primos grandes, exige muito poder de processamento e memória, o que torna essa operação inviável.

Estudos sobre PKC também mostraram tentativas do uso de técnicas mais eficientes, como é o caso do ECC. Os pesquisadores em [41] implementaram o ECC utilizando corpos binários e base polinomial. Finalmente em [35], os autores introduziram uma biblioteca configurável do ECC em TinyOS [49] para RSSF, denominada TinyECC.

3.4 Infraestrutura de Chave Pública

Uma PKI é um conjunto de *hardware*, *software*, pessoas, políticas e procedimentos necessários para criar, gerenciar, armazenar, distribuir e revogar certificados digitais com base na criptografia assimétrica [57].

Uma forma simples de PKI é fornecer um modo de estrutura baseado em hierarquias de Autoridades Certificadoras (ACs), como pode ser observado na Figura 3.3. Nesse exemplo são mostrados três níveis. A AC de nível superior, chamada de raiz, certifica ACs do segundo nível, denominadas de Autoridades de Regiostro (ARs), que por sua vez, certificam as ACs de nível mais baixo, que emitem os certificados para organizações e indivíduos.

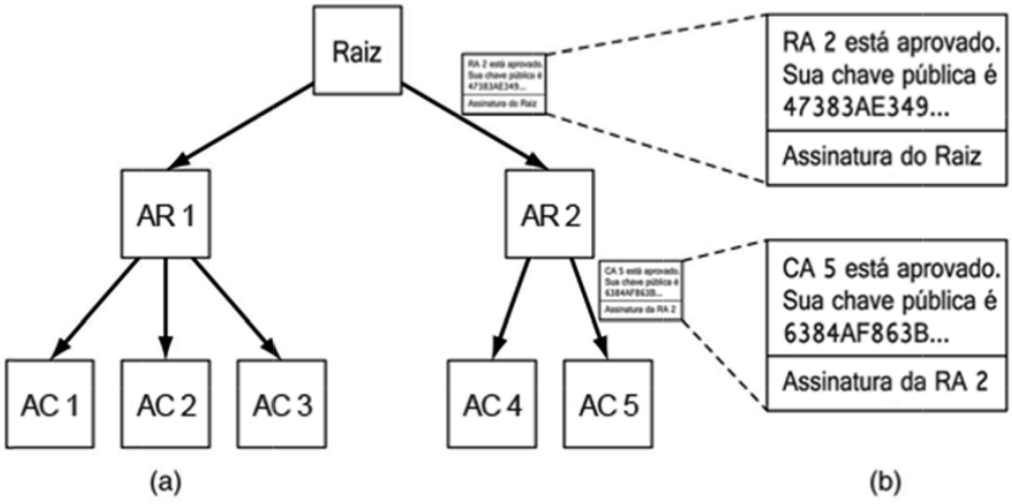

Figura 3.3: *a) Uma PKI Hierárquica e b) Uma Cadeia de Certificados*

Os principas elementos de uma PKI são:

- Entidade final: Um termo genérico usado para indicar os usuários finais, dispositivos ou qualquer outra entidade que possa ser identificada através de um certificado de chave pública.
- AC: O emissor dos certificados e, normalmente, das Listas de Revogação de Certificados (CRLs). Pode fazer o papel de uma AR.
- AR: Um componente opcional que pode assum diversas funções administrativas da AC, geralmente está associada ao processo de resigro da Entidade Final.
- Emissor da CRL: Um componente opcional que uma AC pode delegar para publicar as CRLs.
- Repositório: Indica qualquer método para armazenamento dos certificados e das CRLs de modo que possa ser recuperado pelas Entidades Finais.

O processo de autenticação de chaves públicas geradas por uma AC é bastante dispendioso para ser feito por uma RSSF. Note na Figura 3.4 que quando um usuário *B* deseja verificar o certificado digital de um usuário *A*, o mesmo, após receber o certificado digital de *A*, deve abrir este certificado com a chave pública da AC, já que o mesmo foi assinado com a chave privada da AC. Este processo de verificação de chave pública é inviável de ser realizado dentro da RSSF devido as limitações computacionais dos nós sensores.

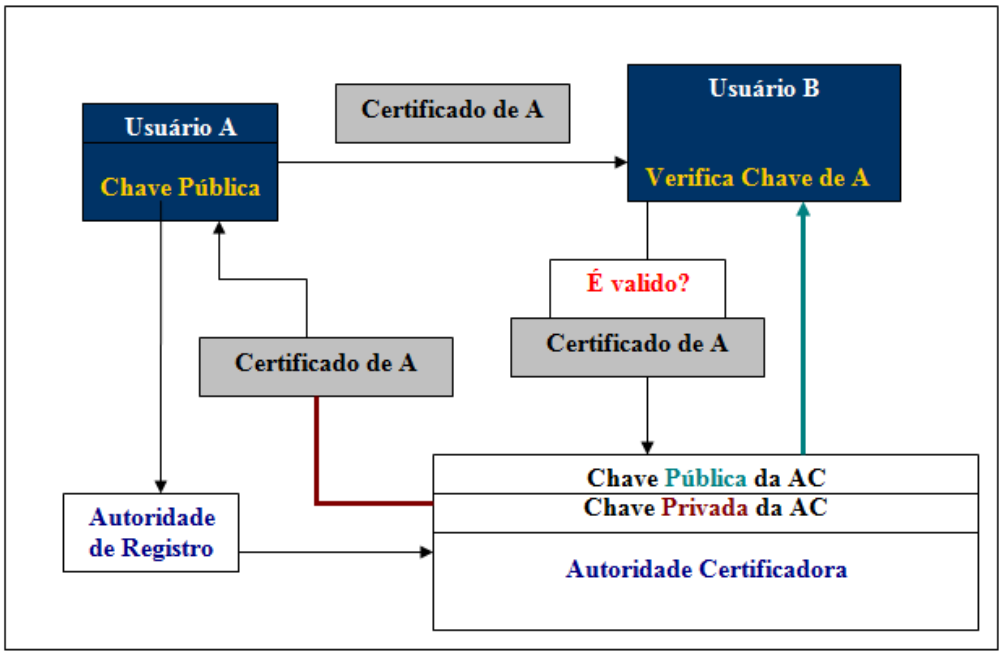

Figura 3.4: *Processo de Verificação de um Certificado Digital Emitido por uma PKI*

3.5 Abordagens Recentes de Distribuição de Chaves

Os criptossistemas baseados em ECC são especialmente interessantes para RSSF pois são mais eficientes na utilização de recursos do que outros criptossistemas de chave pública [62]. O emparelhamento bilinear sobre curvas elípticas é uma tecnologia de chave pública que permite que protocolos clássicos de criptografia sejam aplicados de uma forma muito mais eficiente.

Dentre as aplicações de PBC para resolver o problema de distribuição de chaves criptográficas em RSSF, está a Encriptação Baseada em Identidade (*Identity-Based Encryption* - IBE) [11], mais especificamente os esquemas de Acordo de Chaves Baseados em Identidade (*Identity Based Key Agreement* - IBKA). IBE foi originalmente proposto por Shamir em [56], mas apenas se tornou viável com o advento de PBC [33].

3.5.1 Criptografia Baseada em Emparelhamento

Os primeiro uso da Criptografia Baseada em Emparelhamento foi o trabalho proposto em [53]. Uma função de emparelhamento é um mapa entre dois grupos de pontos de curvas elípticas. A principal característica de criptossistemas baseados em PBC remete-se a propriedade de bilinearidade:

$$\hat{e}(aP,bQ) = \hat{e}(P,Q)ab = \hat{e}(abP,Q) = \hat{e}(P,abQ) \tag{3-1}$$

O uso de emparelhamentos pode ser utilizado para implementar protocolos de segurança já existentes de uma forma mais eficiente e com funcionalidades adicionais, tais como esquemas de acordo de chaves. Em RSSF, o emparelhamento habilita a construção de diferentes esquemas criptográficos baseados em identidade que simplificam o estabelecimento e a distribuição de chaves.

Os protocolos baeados em PBC são mais complicados que esquemas PKC tradicionais. Note na Figura 3.5 os parâmetros que devem ser implementados afim do funcionamento de protocolos baseados em PBC:

Figura 3.5: *Frameworks necessários para a construção de protocolos baseados em emparelhamento*

- Números Grandes e Aritmética Modular: As operações de criptografia são realizadas sobre números grandes, com centenas de *bits* de tamanho. Estes números não são suportados por muitas linguagens computacionais e devem ser implementados externamente.
- Aritmética de Corpo Finito: Curvas elípticas são definidas sobre corpos finitos. Portanto as operações sobre os corpos finitos devem maximizar a eficiência afim da construção de protocolos criptográficos eficientes.
- Aritmética de Curva Elíptica: A eficiência de operações sobre curvas elípticas dependem do tipo da curva, do tamanho do corpo de extensão, do sistema de representação de pontos e da implemetanção destes algoritmos.
- Primitivas ECC: A principal primitiva de ECC é a multiplicação de ponto escalar. Esta operação deve ser eficiente para que protocolos baseados em ECC sejam também eficientes.
- Emparelhamento Bilinear: O custo do cálculo da função de emparelhamento bilinear é a operação que consome mais recursos computacionais em criptossistemas baseados em emparelhamento. Essas funções também devem ser eficientes para que criptossistemas baseados em PBC também sejam.

3.5.2 Esquemas de Acordo de Chaves Baseados em Identidade

Um esquema IBKA genérico pode ser dividido em três fases:

- Inicialização.
- Extração.
- Acordo.

Na fase de inicialização, são obtidos dois grupos cíclicos $\mathbb{G}_1$ e $\mathbb{G}_2$ de ordem prima q e uma função de emparelhamento bilinear $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$. Escolhe-se uma

função *hash* unidirecional $H_1 : \{0,1\}^* \mapsto Z_q$. O dono da rede, ou seja, o responsável pela implantação dos sensores, se torna a Autoridade de Confiança (AC) de todo o sistema.

A AC seleciona um gerador P de $\mathbb{G}_1$, escolhe uma chave secreta $s \in Z_q$ e calcula a chave pública como $Q_{chavePublica} = sP$. A chave secreta s da AC deve ser mantida em sigilo, já a chave secreta de cada nó e o restante dos parâmetros públicos devem ser carregados na memória dos sensores.

Na fase de extração, a AC calcula a chave privada de todos os nós sensores participantes da RSSF. Considere que a AC calculará a chave secreta de um usuário hipotético chamada Alice. A chave privada de Alice é calculada por $s_{Alice} = (a+s)^{-1}P$, onde $a = H_1(ID_{Alice})$.

O valor da chave pública de Alice é dado por $Q_{chavePublicaAlice} = (a+s)P$ e pode ser calculada por qualquer usuário por $aP + P_{Publico}$. De forma semelhante, um usuário qualquer Bob possui uma chave pública $Q_{chavePublicaBob} = (b+s)P$ e uma chave privada $s_{Bob} = (b+s)^{-1}P$.

Na fase de acordo, Alice escolhe um valor aleatório $x_1 \in Z_q^*$, calcula o seguinte valor $T_1 = x_1 Q_{chavePublicaBob} = x_1(b+s)P$ e envia T_1 para Bob. Da mesma maneira, Bob escolhe um valor aleatório $x_2 \in Z_q^*$, calcula $T_2 = x_2 Q_{chavePublicaAlice} = x_2(a+s)P$ e envia T_2 para Alice. Para calcular o mesmo segredo em comum, Alice calcula $K_{AB} = e(T_2, s_{Alice})^{x_1}$ e Bob calcula $K_{BA} = e(T_1, s_{Bob})^{x_2}$. Note que $K_{AB} = K_{BA}$.

Os autores em [16, 21, 46] envisionaram o uso de Encriptação Baseada em Identidade (IBE) como solução de segurança para RSSF. Todos os três trabalhos propõem o esquema de encriptação baseado em identidade de Boney e Franklin [11] para distribuição de chaves criptográficas em RSSF. Contudo a viabilidade das soluções em aplicações práticas não foi demonstrado.

Em [47], os autores implementaram um protocolo de troca de chaves criptográficas baseado em IBE indicando o uso do mesmo apenas no *bootstrapping* da rede. Além disso, foi apresentado uma aplicação denominada TinyPBC, baseada na biblioteca de Aritmética de Multi-precisão Inteira e Racional C/C++ (MIRACL), capaz de computar funções de emparelhamento em aproximadamente 1.90 segundos no ATMega128L, o microcontrolador do sensor MICAz.

Já em [60], os autores propõem um protocolo de acordo de chaves baseado em identidade não tão complexo quanto o proposto por Boney e Franklin [11] e que como em [11] não requer interação entre as partes para concordar uma chave criptográfica em comum. O esquema utiliza o emparelhamento ηT e leva em torno de 4.3 segundos para completar o estabelecimento de chave sobre a plataforma de sensores MICAz. Este trabalho é a primeira implementação prática de um esquema completo de IBC em nós sensores com recursos limitados.

Em [67], os autores apresentaram uma biblioteca criptográfica TinyPairing que possui as seguintes funções implementadas: o Emparelhamento ηT [7], operações de grupo de curva elíptica, aritmética sobre campo subjacente e campo de extensão, geração de número randômico, inclusive a função *hash* mapa-para-ponto e outras diversas operações aritméticas.

Os esquemas criptográficos baseados em emparelhamentos suportados pelo TinyPairing são: o BLL assinatura curta [9], o BB assinatura curta [10] e o BF encriptação baseada em identidade [11]. De acordo com os autores, o TinyPairing é uma biblioteca portátil, projetada para o ambiente TinyOS 2.x [49] e desenvolvida na linguagem nesC.

No sentido de projeto de um esquema de troca de chaves para uma RSSF heterogênea hierárquica, os autores em [58] implementaram um protocolo denominado TinyIBE. TinyIBE é um protocolo para ser utilizado no *bootstrapping* da RSSF e explora as capacidades de *hardware* aprimoradas dos nós líderes dos *clusters*. Contudo, o TinyIBE é extremamente vulnerável a ataques sobre estes nós lideres.

CAPÍTULO 4

Segurança na Camada de Enlace para RSSF

Este capítulo é responsável pela apresentação do porque da necessidade de segurança na camada de enlace para RSSF. Em seguida é apresentado uma das soluções de segurança mais utilizadas na camada de enlace para RSSF: o *framework* TinySec, posteriormente são detalhados os modos de funcionamento, o formato dos pacotes e os componentes e interfaces presentes nesta arquitetura de segurança.

4.1 Porque proteger a Camada de Enlace

O padrão de comunicação fim-a-fim predominante em redes convencionais faz com que alguns requisitos de segurança tais como a autenticidade de mensagens, a integridade e a confidencialidade sejam obtidos através destes mecanismos fim-a-fim. Dentre eles, podem-se citar o SSH [69], o SSL [70] ou o IPSec [31].

Em RSSF, o padrão dominante de comunicação é muitos-para-um, ou seja, diversos nós realizam leituras intermediárias, em uma topologia de múltiplos saltos, e enviam estes dados para uma EB. Para reduzir o tráfego de mensagens e economizar energia, os nós sensores utilizam técnicas de processamento dentro da RSSF tais como agregação de dados e a eliminação de dados duplicados [40, 39].

O processamento dentro da RSSF requer que nós sensores acessem e modifiquem o conteúdo das mensagens. Dessa forma, mecanismos de segurança fim-a-fim não são soluções ideais para garantir os requisitos de autenticidade, integridade e confidencialidade dos dados.

Os mecanismos de segurança que operam na camada de enlace dos dados conseguem detectar pacotes não autorizados na primeira vez que os mesmos são injetados na RSSF. Mecanismos de segurança fim-a-fim detectam pacotes maliciosos apenas no destino final, neste caso a RSSF pode rotear pacotes não autorizados por diversos saltos antes que os mesmos sejam detectados, o que gastaria desnecessariamente energia e largura de banda.

4.2 *Framework* TinySec

4.2.1 Descrição

O *framework* Tinysec é uma das principais soluções de segurança na camada de enlace para RSSF. É um mecanismo para prover os requisitos de confidencialidade, integridade e autenticidade na camada de enlace dos dados.

O *TinySec* pode ser facilmente integrado em aplicações de RSSF tendo como principais objetivos a facilidade no uso e o mínimo impacto no desempenho de aplicações que o utilizam. A arquitetura não endereça alguns problemas tais como a distribuição de chaves criptográficas, ataques de depleção de energia e ataques físicos. O presente trabalho desta dissertação propõe uma forma adequada de distribuição de chaves criptográficas, aumentando o nível de segurança das aplicações que usam o TinySec como primitiva de segurança.

O *framework* TinySec é baseado apenas em algoritmos simétricos. Cifras de Bloco são utilizadas para cifragem e geração dos MACs para autenticação. A cifra de bloco padrão é o Skipjack [25], porém toda a arquitetura é independente de cifra, o que possibilita a utilização de outros algoritmos simétricos, tais como o RC5 ou o AES.

Segundo [30], o tempo para encriptar uma mensagem através do Skipjack em um sensor MICA2 é de 0.38 ms (cifragem de um bloco de 64 *bits*). A implementação da arquitetura detém mais de 3000 linhas de código nesC e requer 728 *bytes* de *RAM* e 7146 *bytes* de *ROM* de um nó sensor MICA2. A adição do *framework* TinySec representa um acréscimo de menos de 10% em termos de latência na entrega do pacote e no consumo de energia.

4.2.2 Modos de Funcionamento

É possível operar de duas maneiras diferentes no TinySec:

- Modo TinySec-Auth: A mensagem é enviada usando apenas o Código de Autenticação de Mensagem (MAC).
- Modo TinySec-AE: A mensagem é enviada de forma cifrada e autenticada através do MAC.

No modo TinySec-AE, a carga útil é de até 29 *bytes*, que é cifrada, e o cabeçalho de pacote é de 8 *bytes*, sendo que o pacote inteiro (dados e cabeçalho) é autenticado. Veja na Figura 4.1 a estrutura do pacote no modo mais completo do TinySec, o modo TinySec-AE.

Tabela 4.1: *Quantidade de bits adicionados pelo uso do TinySec*

Modo de funcionamento	Quantidade de *bits* adicionados	Quantidade total de *bits*
Sem o TinySec	0 *bytes*	36 *bytes*
Modo TinySec-Auth	1 *bytes*	37 *bytes*
Modo TinySec-AE	5 *bytes*	41 *bytes*

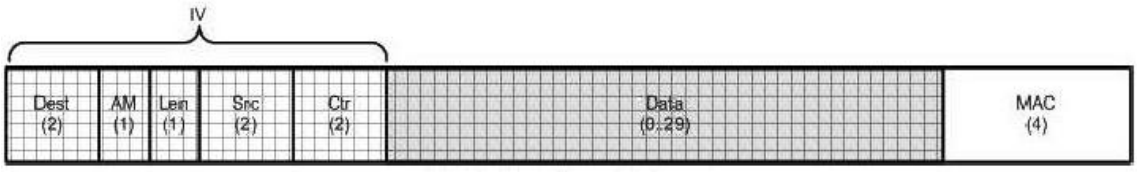

Figura 4.1: *Formato do Pacote no Modo TinySec-AE.*

No modo TinySec-Auth, o pacote inteiro (dados e cabeçalho) é autenticado, sendo a carga útil também de até 29 *bytes*, mas o cabeçalho de pacote é de 4 *bytes* [42]. Veja na Figura 4.2 a estrutura do pacote do TinySec no modo TinySec-Auth.

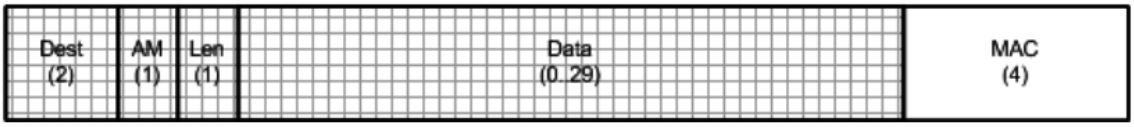

Figura 4.2: *Formato do Pacote no Modo TinySec-Auth.*

Na Figura 4.3 nota-se que o formato do pacote no TinySec é baseado no pacote padrão do TinyOS. Os 2 *bytes* padrões para o CRC (*Cyclic Redundancy Check*) são substituídos por 4 *bytes* do MAC. O TinySec utiliza um Vetor de Inicialização (*Iniatilization Vector* - IV) especial com endereços de origem e destino (2 *bytes* cada), tamanho do pacote (1 *byte*), tipo da mensagem ativa (1 *byte*) e um contador (2 *bytes*).

Figura 4.3: *Formato do Pacote Padrão do TinyOS.*

Os dois modos de funcionamento do TinySec adicionam uma quantidade de *bits* diferente ao pacote final de dados que será enviado, e isso faz com que o consumo de energia em um nó sensor aumente, pois o sensor terá que transmitir uma maior quantidade de *bits*. Esse acréscimo de *bits* ao pacote final se dá, principalmente, pelo MAC utilizado para verificação da autenticidade e da integridade das mensagens.

Veja na Tabela 4.1 a quantidade de *bits* que são adicionadas ao pacote normal, sem a utilização do *framework* TinySec, e em cada modo de funcionamento do TinySec.

Tabela 4.2: *Compatibilidade entre os modos do TinySec*

Modos de Recebimento	Modos de Tranmissão
Modo 1	Modos 1 **ou** 2
Modo 2	Modo 3
Modo 3	Modos 1, 2 **ou** 3

4.2.3 Componentes do *Framework* TinySec

Esta seção é responsável pela definição das Interfaces do *framework* TinySec. As principais interfaces são relacionadas com o modo de transmissão e recebimento das mensagens (Interface TinySecMode) e com a atualização das chaves em uso pela RSSF (Interface TinySecControl).

Interface TinySecMode

Esta interface é responsável pela definição dos modos de transmissão e recebimento de uma mensagem no TinySec. Os comandos disponíveis neste componente são:

- command result_t setTransmitMode (uint8_t mode);
- command result_t setReceiveMode (uint8_t mode);

O modo setTransmitMode apenas aceita três argumentos:

1. TINYSEC_AUTH_ONLY (Modo padrão de envio)
2. TINYSEC_ENCRYPT_AND_AUTH
3. TINYSEC_DISABLED

Analogamente, o modo setReceiveMode aceita os três possíveis argumentos:

1. TINYSEC_RECEIVE_AUTHENTICATED (Modo padrão de recebimento)
2. TINYSEC_RECEIVE_CRC
3. TINYSEC_RECEIVE_ANY

Um receptor que esteja configurado no modo de recebimento TINYSEC_RECEIVE_AUTHENTICATED só pode receber mensagens oriundas de um emissor configurado em um dos seguintes modos: TINYSEC_AUTH_ONLY ou TINYSEC_ENCRYPT_AND_AUTH.

Um receptor no modo TINYSEC_RECEIVE_CRC aceita apenas mensagens de emissores no modo TINYSEC_DISABLED. Da mesma forma, um receptor no modo TINYSEC_RECEIVE_ANY recebe mensagens de emissores configurados em qualquer modo de transmissão. Veja na Tabela 4.2 as possíveis combinações dos modos de envio e recebimento:

Interface *TinySecControl*

A interface TinySecControl habilita a atu8alização das chaves utilizadas pelo TinySec. Esta interface possui os seguintes comandos:

1. command result_t updateMACKey(uint8_t * MACKey);
2. command result_t getMACKey(uint8_t * result);
3. command result_t updateEncryptionKey(uint8_t * encryptionKey);
4. command result_t getEncryptionKey(uint8_t * result);
5. command result_t resetIV();
6. command result_t getIV(uint8_t * result)

Os comandos updateMACKey(uint8_t * MACKey) e result_t updateEncryption-Key(uint8_t * encryptionKey) são responsáveis por atualizarem as chaves utlizadas para autenticação e encriptação dos dados, respectivamente. O tipo de retorno desses comandos é SUCESSO se a chave for atualizada corretamente e FALHA caso contrário.

Os comandos getMACKey(uint8_t * result) e getEncryptionKey(uint8_t * result) retornam os valores correspondentes das chaves de autenticação e encriptação. O parâmetro result_t referencia um tamanho de chave de no mínimo TINYSEC_KEYSIZE *bytes*.

O comando resetIV() reseta a porção do contador do IV especial para 0. O comando resetIV() retorna SUCESSO caso o IV seja resetado corretamente e FALHA, caso contrário. Normalmente este comando é utilizado em conjunto com os comandos de atualização das chaves do TinySec. O parâmetro de retorno result_t referencia uma mensagem de tamanho no mínimo de TINYSEC_IV_LENGHT *bytes*.

CAPÍTULO 5

Um Esquema de Distribuição de Chaves Baseado em Identidade para o *Framework* de Segurança TinySec

Esta capítulo é responsável pela apresentação de como foi realizada a junção de um esquema de acordo de chaves baseado em identidade e o *framework* de segurança da camada de enlace TinySec. São dadas duas visões diferentes da junção, uma geral, com o intuito de orientação dos desenvolvedores de aplicações para RSSF de como habilitar o uso do protocolo de distribução de chaves antes da utilização do TinySec, e outra, mais específica, mostrando as características deste esquema de distribuição de chaves. São apresentados também os custos adicionados a uma aplicação que realiza primeiramente o acordo de chaves para posteriormente recorrer ao *framework* TinySec para autenticação e/ou encriptação dos dados. Os requisitos de segurança adicionados com a abordagem também são demonstrados.

5.1 Visão Geral

O *framework* de segurança TinySec não é atrelado a nenhum mecanismo de distribuição de chaves criptográficas. A arquitetura utiliza chaves criptográficas armazenadas em um arquivo padrão (arquivo tinyos_keyfile). Portanto, o desenvolvedor deve estar ciente das chaves que serão utilizadas pelo algoritmo de criptografia. As chaves utilizadas pelo TinySec, um par de chaves da cifra de bloco Skipjack [25], são setadas em tempo de compilação. Se nenhum argumento extra é passado no processo normal de compilação, o arquivo de chave padrão é utilizado.

A abordagem para distribuição de chaves baseia-se na utilização de um esquema IBKA distribuição de chaves no *framework* TinySec. A Figura 5.1 descreve qual a sequência de passos que os desenvolvedores devem seguir com o objetivo de adequar suas aplicações para que:

1. Utilizem o mecanismo de estabelecimento e distribuição de chaves proposto;

2. Recorram ao *framework* Tinysec para autenticação e/ou encriptação dos dados;

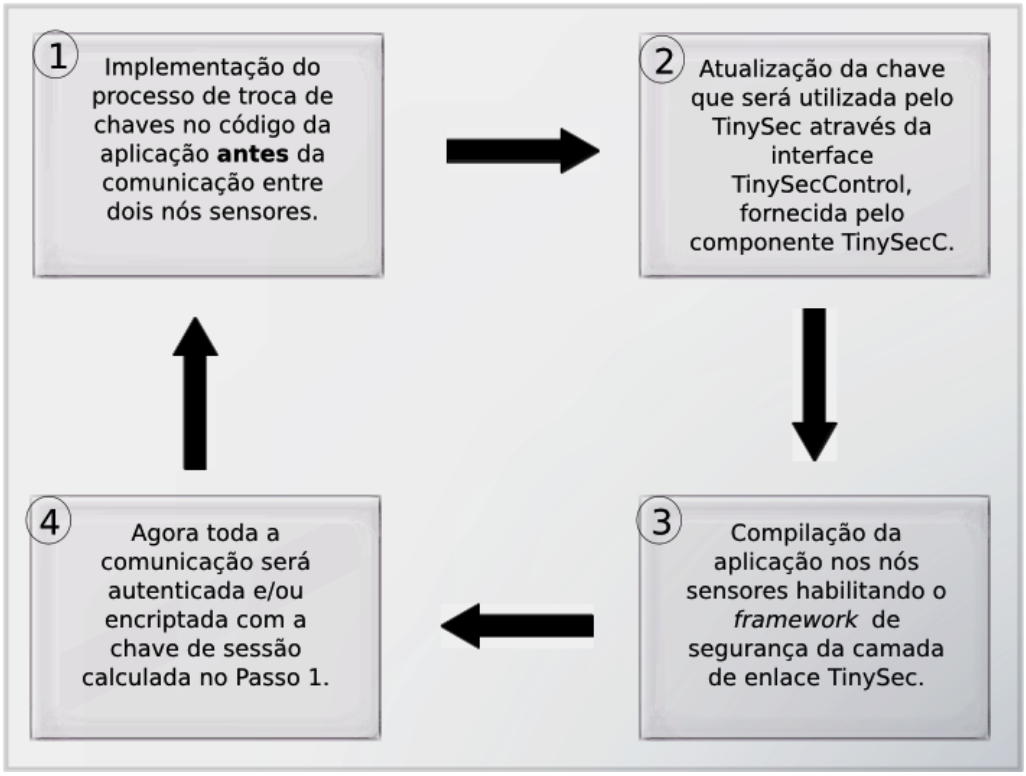

Figura 5.1: *Passos para utilização da nova abordagem de distribuição de chaves e o TinySec*

O desenvolvedor da aplicação deverá realizar o Passo 1 que consiste em adicionar ao código da aplicação o novo mecanismo de distribuição de chaves proposto na Seção 5.2. Esta inclusão deve ser realizada antes da comunicação entre dois nós quaisquer, pois ao habilitar o TinySec nas aplicações de RSSF, o mesmo já é configurado para enviar os dados somente autenticados ou autenticados e encriptados.

Após o algoritmo de troca de chaves ser executado, a chave de sessão resultante deverá então ser utilizada para substituir a chave padrão do TinySec. Após o acordo de chaves, o desenvolvedor deverá invocar a interface TinySecControl, que é exportada pelo componente TinySecC, para atualizar a chave que será utilizada pelo *framework* de segurança da camada de enlace TinySec (Passo 2).

Após realizar os Passos 1 e 2 da Figura 5.1, a aplicação está pronta para ser compilada nos nós sensores. Após a adequação da aplicação, a mesma pode ser compilada com a adição do comando “TINYSEC=TRUE” no momento da compilação. Dessa forma, primeiramente será criada uma chave no arquivo padrão de chaves do TinySec, porém essa chave não será utilizada. Após o acordo de chaves da nova abordagem de distribuição, a interface responsável por realizar o *update* da chave do TinySec irá atualizá-la e esta será utilizada para proteção da comunicação.

O *framework* de segurança TinySec provê os requisitos de confidencialidade, integridade e autenticidade dos dados. A proposta de junção do TinySec juntamente com um mecanismo de distribuição de chaves baseado em primitivas assimétricas, o IBKA-Sec, faz com que os danos provenientes de um ataque a um determinado nó se tornem estritamente locais, ou seja, o comprometimento de um nó não afetará a comunicação entre os nós restantes da RSSF. Já a utilização apenas do TinySec como solução de segurança pode representar o comprometimento de toda a RSSF caso apenas um único nó seja comprometido.

5.2 Visão Detalhada

A abordagem para distribuição de chaves criptográficas para o *framework* TinySec é baseado em [44]. O esquema provê uma forma simples, prática e segura para distribuição de chaves criptográficas. A proposta de distribuição de chaves criptográficas possui duas fases: uma fase antes da implantação para estabelecimento de todos os parâmetros necessários e outra fase, após a implantação, para o acordo de chaves entre dois nós sensores.

5.2.1 Fase Antes da Implantação : Estabelecimento de Parâmetros

O desenvolvedor da aplicação é responsável por carregar as chaves secretas dentro da memória de cada nó juntamente com todos os parâmetros públicos, ou seja, o mesmo se torna a Autoridade de Confiança (AC) da RSSF.

Primeiramente a AC gera uma chave secreta s que deve ser mantida em sigilo. A AC também é responsável por assinalar as identidades de todos os nós e calcular a chave secreta de cada um deles.

Para calcular a chave secreta de um determinado nó, a AC precisa utilizar uma função de *Hash*-Mapeamento (ϕ) para assinalar a identidade de cada nó i a um ponto de uma curva elíptica:

$$R_i = \phi(ID_i) \tag{5-1}$$

A chave secreta de cada nó i é então calculada através da expressão:

$$S_i = [s]R_i \tag{5-2}$$

Uma Função de Derivação da Chave (FDC) deve ser utilizada para adequar o tamanho da chave de sessão calculada para o tamanho da cifra de bloco Skipjack, que é a cifra padrão do *framework* TinySec.

Tabela 5.1: *Notação e descrição das variáveis*

Notação	Descrição
s	Chave secreta do desenvolvedor da aplicação
ID_i	Identidade pública do nó i
S_i	Chave privada do nó i
$\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$	Grupos cíclicos
$\hat{e}$	Mapa bilinear $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$
ϕ	Função de *Hash*-Mapeamento
FDC	Função de Derivação da Chave

Cada nó i é então carregado com os seguintes parâmetros:

- Identidade Pública: ID_i.
- Chave Secreta: S_i.
- Função de *Hash*-Mapeamento: ϕ.
- Função de Derivação da Chave: FDC.

Veja na Tabela 5.1 a notação das variáveis utilizadas no protocolo de estabelecimento de chave.

5.2.2 Fase Após a Implantação: Acordo de Chaves

As chaves privadas dos nós X e Y são respectivamente S_X e S_Y. A chave pública de X pode ser calculada a partir da identidade pública de X, que é conhecida por Y. O valor:

$$ChavePublica_X = \phi(ID_X) \tag{5-3}$$

representa a chave pública de X. Da mesma forma, a chave pública de Y é dada pela expressão:

$$ChavePublica_Y = \phi(ID_Y) \tag{5-4}$$

No momento que o nó X quer estabelecer uma chave de sessão com Y, ele calcula a função de emparelhamento bilinear:

$$\hat{e}(S_X, ChavePublica_Y = \phi(ID_Y)) \tag{5-5}$$

A FDC é utilizada sobre o cálculo de emparelhamento para adequação do tamanho da chave. Dessa forma, o valor:

$$ChaveSessao(X,Y) = FDC((\hat{e}(S_X, ChavePublica_Y))) \tag{5-6}$$

deverá resultar em uma chave de sessão de 80-*bits* que será utilizada pelo algoritmo de criptografia Skipjack para autenticação ou encriptação dos dados.

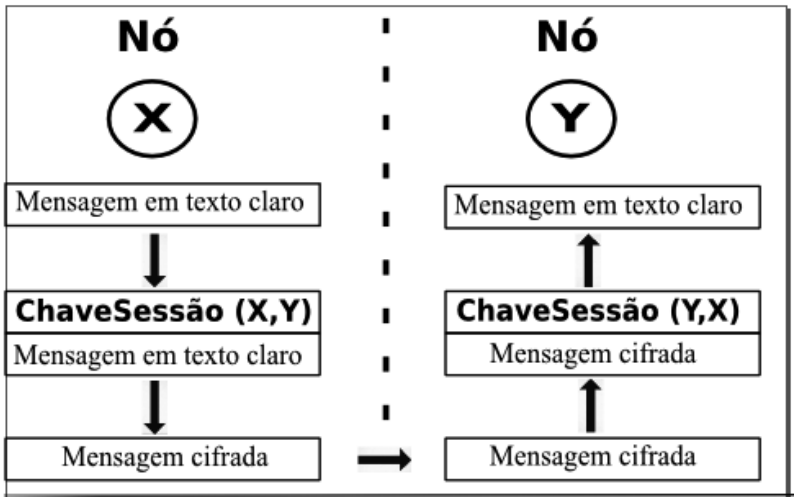

Figura 5.2: *Envio da mensagem do nó X para o nó Y*

Note na Figura 5.2 que quando a mensagem chega no nó Y, o mesmo consegue decriptar a mensagem utilizando a mesma chave de sessão utilizada por X através da expressão:

$$ChaveSessao(Y,X) = FDC((\hat{e}(S_Y, ChavePublica_X))) \quad (5\text{-}7)$$

Devido a propriedade de simetria do emparelhamento bilinear, tem-se que:

$$ChaveSessao(X,Y) = ChaveSessao(Y,X) \quad (5\text{-}8)$$

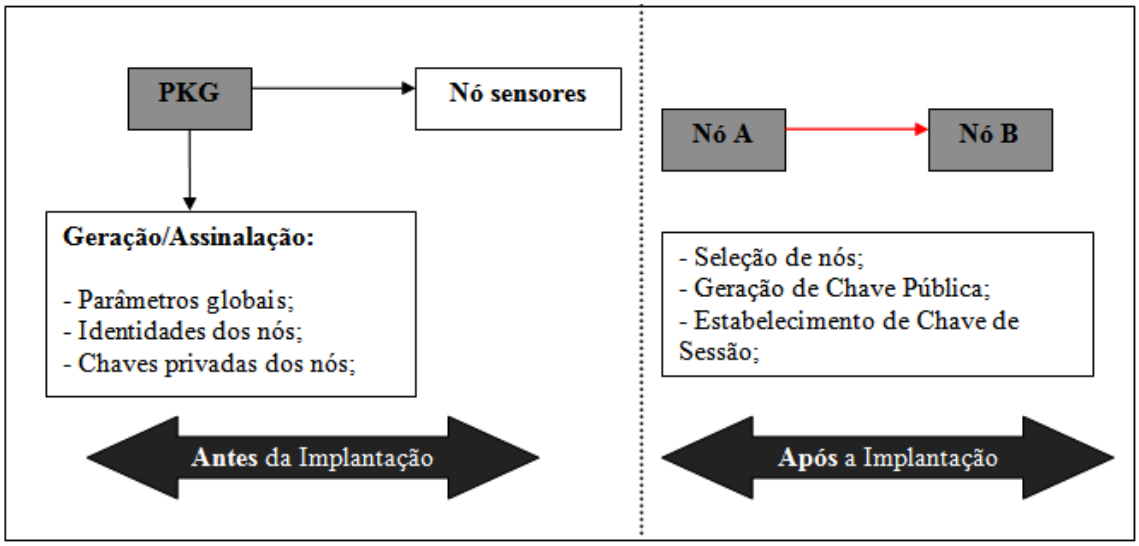

Figura 5.3: *Passos realizados Antes e Após a Implantação dos Nós Sensores*

Os dois nós sensores conseguem calcular o mesmo segredo compartilhado sem nenhuma interação entre os mesmos e de uma forma mais segura que o pré-compartilhamento de uma mesma chave criptográfica antes da fase de implantação, que é a forma inapropriada que o *framework* de segurança da camada de enlace TinySec utiliza para estabelecimento de chaves na RSSF. Veja na Figura 5.3 os parâmetros que são estabelecidos antes e após a implantação.

5.3 Bibliotecas Criptográficas Baseadas em PBC

Dada a complexidade da construção e implementação de protocolos baseados em PBC, esta seção é destinada à análise das bibliotecas *open source* baseadas em PBC disponíveis na literatura. Esta análise servirá como base para seleção de qual ferramenta criptográfica será utilizada no auxílio da solução da proposta dessa dissertação, a junção de um protocolo IBKA com o *framework* de segurança da camada de enlace TinySec.

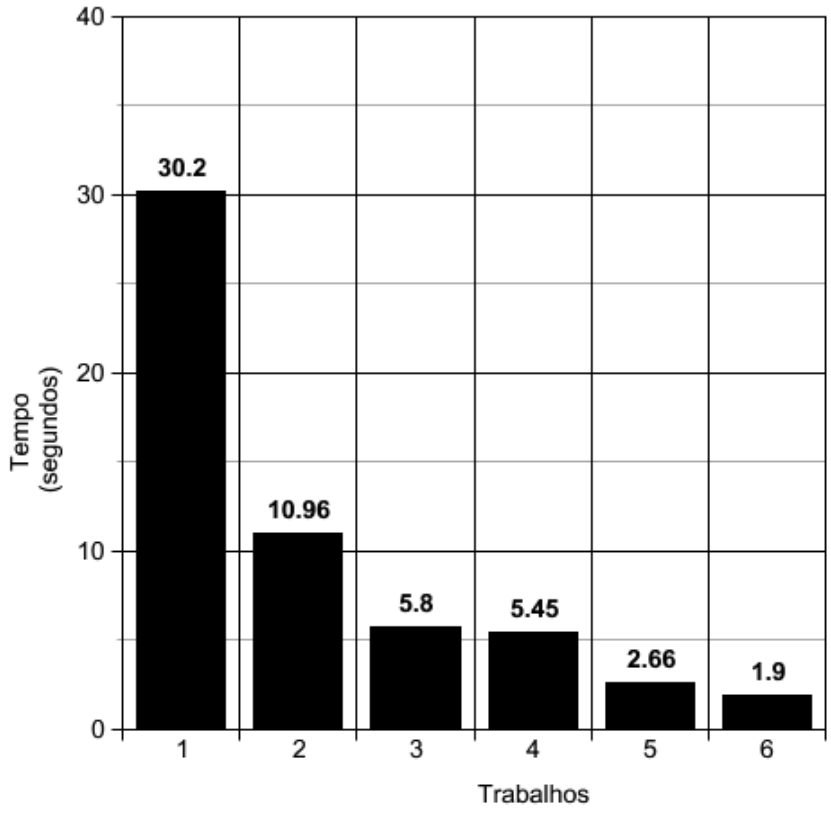

Figura 5.4: *Tempo para cálculo do emparelhamento bilinear na plataforma ATMega128L por diferentes trabalhos*

Dentre as bibliotecas criptográficas disponíveis, destaca-se a aplicação TinyPBC [44], baseada na RELIC-*toolkit* [3]. TinyPBC é escrita na linguagem *Network Embedded Systems C* (*nesC*), é a que possui o menor tempo para cálculo da função de emparelha-

mento bilinear, a função mais complexa em termos computacionais em criptossistemas baseados em emparelhamentos.

Veja na Figura 5.4 a comparação do tempo gasto para cálculo do emparelhamento na família de sensores mica (processador ATMega128L) pelos seguintes trabalhos:

1. TinyTate [45];
2. NanoECC [62];
3. Ishiguro et. al [19];
4. TinyPBC (2007) [47];
5. Szczechowiak et. al [60];
6. TinyPBC (2011) [44];

5.3.1 Aplicação TinyPBC baseada na RELIC-*toolkit*

Nível de Segurança

A aplicação TinyPBC, baseada na RELIC-*toolkit*, adota um nível de segurança de 80-*bits*, o que é equivalente a um RSA de 1024-*bits*.

Emparelhamento Bilinear

A função de emparelhamento é definida sobre corpos binários. A implementação do emparelhamento é baseada no Emparelhamento η_T [7]. Segundo [61], o emparelhamento η_T é um dos métodos mais rápidos no nível de segurança considerado.

Curva Elíptica

A implementação é realizada sobre o corpo binário $\mathbb{F}_{2^{271}}$. A curva supersingular $y^2 + y = x^3 + x$ com grau de imersão $k = 4$ é utilizada para mapeamento da identidade dos nós. A execução do emparelhamento gasta a maioria do tempo realizando multiplicações na quarta extensão do corpo ($\mathbb{F}_{2^{4x271}}$).

Representação do Corpo de Extensão

Os elementos do corpo $\mathbb{F}_{2^m}$ são representados usando polinômios. Para o corpo de extensão $\mathbb{F}_{2^{271}}$, TinyPBC utiliza o pentanomial $f(x) = x^{271} + x^{207} + x^{175} + x^{111} + 1$ dado em [55].

A escolha de $f(x)$ têm duas características principais:

- A redução modular para $f(z)$ requer mudanças de 1 ou 7 *bits*.
- A extração da raiz quadrada não requer mudanças nos registradores para palavras com tamanho 8 ou 16 *bits*.

Quarta Extensão

A representação binária de um elemento pertencente a $\mathbb{F}_{2^{271}}$ pode ser computado pela inserção do *bit* 0 entre cada par sucessivo de *bits* da representação binária do elemento. Isso pode ser reduzido com o uso de uma tabela de consulta de 16 *bytes* no qual armazena em sua memória o quadrado de todos os polinomiais de 4 *bits*. Com o intuito de evitar o desperdício de memória, ou seja, diminuir os valores carregados nos registradores, a extração de raiz é realizada em duas etapas:

1. Calcula-se a raiz da metade inferior utilizando uma tabela de expansão convencional de 4 *bits*.
2. Calcula-se a raiz da metade superior combinado com a redução modular.

Raiz Quadrada

A extração de raiz quadrada na aplicação TinyPBC foi implementada de acordo com [18], no qual requer uma divisão de passos para concatenar coeficientes com índices pares ou ímpares com elementos do corpo. Tabelas de consulta podem ser utilizadas para implementação porém o microcontrolador permite uma rápida implementação dessa operação através da manipulação de *bits*.

Operação de Multiplicação de Ponto Escalar

A operação de multiplicação de ponto escalar é realizada pelo Algoritmo de López-Dahab [37]. Com o intuito de evitar o acesso á memória de forma desnecessária, TinyPBC realiza algumas otimizações no algoritmo original: um vetor de dígito intermediário é armazenado dentro de um registrador como no Algoritmo 5.1. De acordo com os autores, essa otimização diminui o número de operações de leitura pela metade e de escrita por um fator quadrático. Veja o Algoritmo 5.1 com a proposta de otimização para o Algoritmo original [37].

Algoritmo 5.1: Algoritmo otimizado para multiplicação no corpo $\mathbb{F}_{2^m}$

Entrada: $a(z) = a[0 \cdots n-1], b(z) = b[0 \cdots n-1]$
Saída: $c(z) = c[0 \cdots 2n-1]$

v_i representa um vetor de $n+1$ registradores $(r_{i-1}, \cdots, r_0, r_n, \cdots, r_i)$
Computar $T(u) = u(z)b(z)$ para todos os polinônimos de $u(z)$ com grau menor que 4.
Seja u_i os 4 *bits* mais significativos de $a[i]$
$v_0 \leftarrow T(u_0), c[0] \leftarrow r_0$
$v_1 \leftarrow v_1 \oplus T(u_1), c[1] \leftarrow r_1$
$\cdots$
$v_{n-1} \leftarrow v_{n-1} \oplus T(u_{n-1}), c[n-1] \leftarrow r_{n-1}$
$c \leftarrow ((r_{n-2}, \cdots, r_0, r_n)||(c[n-1], \cdots, c[0])) \ll 4$
Seja u_i os 4 *bits* menos significantes de $a[i]$
$v_0 \leftarrow T(u_0), c[0] \leftarrow c[0] \oplus r_0 \cdots$
$v_{n-1} \leftarrow v_{n-1} \oplus T(u_{n-1}), c[n-1] \leftarrow c[n-1] \oplus r_{n-1}$
$c[n \cdots 2n-1] \leftarrow c[n \cdots 2n-1] \oplus (r_{n-2}, \cdots, r_0, r_n)$
return c

5.4 Plataformas Utilizadas

Os experimentos foram realizados na plataforma de *hardware* mica2, uma vez que o *framework* de segurança da camada de enlace TinySec foi projetado para as plataformas mica, mica2 e mica2dot. A plataforma detêm as seguintes características:

- Consumo do processador - Ativo: 8 mA; Dormindo: 15 mA
- Microcontrolador - Atmega128L.
- Clock - 7.3728 MHz.
- Memória FLASH - 128 KB.
- Memória SRAM - 4 KB.
- Taxa de transmissão do rádio - 38.4 kbps.
- Alcance do rádio - 500 ft.
- Consumo do rádio - Ativo: 27 mA (TX Máximo) e 10 mA (RX); Dormindo: 1 mA.

O sistema operacional utilizado foi o TinyOS 1.1.15, a última versão disponível do TinyOS 1.x, já que o *framework* de segurança TinySec não foi transportado para o TinyOS 2.x. O TinyOS apresenta uma arquitetura baseada em componentes, o que minimiza o tamanho do código através do desenvolvimento de novas aplicações a partir apenas da ligação de componentes existentes.

O modelo de execução no TinyOS é orientado a eventos, o que permite um melhor gerenciamento de energia, além de permitir a flexibilidade de programação necessária considerando a imprevisibilidade da comunicação sem fio.

A linguagem de programação utilizada foi o *nesC* [20], linguagem que possui sintaxe semelhante ao *C*, mas com recursos para suportar a estrutura e a execução das aplicações em TinyOS [34]. Também foi utilizada a biblioteca criptográfica RELIC -*toolkit* [3] (versão 0.3.3) com suporte para operações sobre curvas elípticas e emparelhamentos bilineares, desenvolvida em C e assembly, e com prioridade em eficiência. Essa biblioteca permite a configuração flexível para níveis de segurança específicos, corpos e curvas elípticas particulares, dentre outras características.

O ambiente de desenvolvimento foi a distribuição Linux Ubuntu 12.04.3 LTS, sendo utilizado o simulador Avrora [63] para realização de testes e para validar a implementação. O Avrora é um simulador de alta acurácia que provê ferramentas para simulação e análise (monitores) para programas escritos para microcontroladores AVR e motes mica2. O comportamento dos programas é analisado em nível de instrução de máquina, além disso o Avrora oferece uma API, escrita em Java, que permite extensão das funcionalidades providas pelo simulador.

5.5 Adequação da Aplicação TinyPBC

TinyPBC é uma aplicação baseada na RELIC-*toolkit* [3], compatível com TinyOS, e que tem com alvo os processadores da família dos motes mica (processador ATmega128L). A aplicação mostra como dois nós sensores são capazes de processar uma mesma chave de sessão comum sem nenhuma interação entre os mesmos, através do uso de emparelhamentos bilineares.

A aplicação TinyPBC é baseada em identidade, ou seja, apenas a identidade dos nós é requerida. Porém, essa aplicação está originalmente escrita na linguagem *nesC* para o TinyOS 2.x. O *framework* de segurança da camada de enlace TinySec foi originalmente projetado e integrado em um versão anterior do TinyOS, mais especificamente para a versão 1.x.

Dada as incompatibilidades e diferenças inerentes das versões 1.x e 2.x do TinyOS, algumas mudanças na aplicação TinyPBC precisaram ser realizadas, a fim de possibilitar a junção entre o esquema de acordo de chaves baseado em identidade e a arquitetura de segurança da camada de enlace TinySec.

Dentre as mudanças realizadas no arquivo de configuração da aplicação TinyPBC, podem-se citar:

1. Substituição do Componente MainC por Main;

2. Substituição da ligação MainC.Boot <- TinyPBCApp por Main.StdControl <- TinyPBC.StdControl;
3. Inclusão do arquivo relic.h, através do comando: "includes relic.h", no cabeçalho do arquivo antes do início do arquivo de configuração;

A interface Boot, usada no Tinyos 2.x, para inicialização dos componentes , deve ser substituída pela interface StdControl. A interface StdControl provê três comandos:

- init().
- start().
- stop().

Todos componentes que necessitam de inicialização devem prover a interface StdControl. Após o boot, todos os componentes devem ser inicializados com o comando init(). Após o comando init() ser invocado, os comandos start() e stop() podem ser invocados múltiplas vezes.

Em relação as mudanças realizas no módulo de TinyPBC, ou seja, no arquivo que mostra como é realizado a lógica interna do programa, podem-se citar:

1. Remoção do comando: "#include <relic.h>";
2. Substituição do uso da Interface Boot pelo provimento da interface StdControl;
3. Substituição da variável TOS_NODE_ID pela TOS_LOCAL_ADDRESS;
4. Substibuição do evento booted() da Interface Boot pela implementação dos três comandos da interface StdControl: init(), start() e stop();

5.6 IBKA-Sec: Aplicação de Junção entre o TinySec e o TinyPBC

Para testar a junção entre o *framework* TinySec e a implementação do acordo de chaves TinyPBC contida na RELIC-*toolkit* [3], foi desenvolvida uma aplicação, a IBKA-Sec, baseada na aplicação TestTinySec (incorporada no tinyos), que envia um contador de um nó para outro utilizando métodos de autenticação e encriptação oferecidos pelo TinySec, e na aplicação TinyPBC.

A implementação da aplicação IBKA-Sec em dois nós sensores faz com que os LEDs vermelho e verde de ambos os motes fiquem piscando constantemente, a tarefa de envio de uma mensagem é definida para o endereço de *BROADCAST* da RSSF. Logo, todos os nós enviam e recebem uma determinada mensagem. O LED verde indica o sucesso da tarefa de envio da mensagem e o LED vermelho sinaliza o sucesso no recebimento.

5.6.1 Componente de Configuração da Aplicação IBKA-Sec

As configurações correspondem a uma maneira alternativa de definir componentes, que resultam da interligação entre módulos ou entre outras configurações. O arquivo de configuração pode incluir uma seção inicial onde se especifica se usa (*uses*) ou se oferece (*provides*) interfaces.

O componente do tipo configuração pode ser interligado como qualquer outro componente. No corpo principal da declaração de um componente de configuração, são declarados os componentes que são interligados, após a palavra-chave *components*, e as ligações entre eles (quem usa e quem fornece a interface).

Os componentes que são interligados, que podem ser verificados no Código 5.1, já as ligações entre os componentes pode ser verificado no Código 5.2.

Código 5.1 Componentes

```
1  includes relic;
2  includes IntMsg;
3  includes AvroraPrint;
4  configuration TinyPBCAndTinySecApp {
5  }
6  implementation {
7          components Main,
8          LedsC,
9          GenericComm as Comm,
10         TinyPBCAndTinySec,
11         Counter,
12         TimerC,
13         TinySecC,
14         SysTimeC;
15
16 //Codigo Omitido
17 }
```

Código 5.2 Ligações entre os Componentes

```
1  Main.StdControl-> TinyPBCAndTinySec.StdControl;
2  Main.StdControl -> Comm.Control;
3  Main.StdControl -> Counter.StdControl;
4  Main.StdControl -> TimerC.StdControl;
5  TinyPBCAndTinySec.Send -> Comm.SendMsg[AM_INTMSG];
6  TinyPBCAndTinySec.ReceiveIntMsg -> Comm.ReceiveMsg[AM_INTMSG];
7  Counter.Timer -> TimerC.Timer[unique("Timer")];
8  Counter.IntOutput -> TinyPBCAndTinySec.IntOutput;
9  TinyPBCAndTinySec.Leds -> LedsC;
10 TinyPBCAndTinySec.TinySecMode -> TinySecC.TinySecMode;
11 TinyPBCAndTinySec.TinySecControl -> TinySecC.TinySecControl;
12 TinyPBCAndTinySec.SysTime -> SysTimeC;
```

5.6.2 Componente de Módulo da Aplicação IBKA-Sec

A especificação do módulo está dividida em dois blocos: a primeira parte, precedida de *module* especifica quais são as interfaces providas pelo componente (com a palavra-chave *provides*) e/ou usadas pelo componente (com a palavra-chave *uses*); a segunda parte, dentro de parênteses após *implementation*, contem a realização das funções que tratam os eventos ou realizam os comandos. Os componentes que são usados e providos pelo módulo da aplicação podem ser verificados no Código 5.3.

Código 5.3 Componentes do Módulo

```
1  module TinyPBCAndTinySec {
2
3  provides {
4          interface StdControl;
5          interface IntOutput;
6          }
7  uses {
8          interface Leds;
9          interface SendMsg as Send;
10         interface ReceiveMsg as ReceiveIntMsg;
11         interface TinySecMode;
12         interface TinySecControl;
13         interface SysTime;
14       }
15 }
```

Após o *boot* de dois nós, a função incializar(), presente no Código 5.4, é a primeira a ser executada. Dentro dessa função são realizados os seguintes procedimentos:

1. A inicialização da biblioteca.
2. A geração da chave privada da Autoridade de Confiança (AC).
3. A geração da chave privada dos nós a partir da chave privada da AC.

Código 5.4 `inicializar()`

```
1  static int inicializar() {
2      snprintf(thisID, 10, "%d", TOS_LOCAL_ADDRESS);
3      core_init();
4      if(pc_param_set_any() != STS_OK) {return 1;}
5      bn_init(&masterKey, BN_DIGS);
6      bn_read_str(&masterKey,
7      "123456789ABCDEF123456789ABCDEF123456789ABCDEF123456789ABCDEF", 64, 16);
8      cp_sokaka_gen_prv(privateKey, thisID, strlen(thisID), &masterKey);
9      return 0;
10 }
```

A biblioteca RELIC-*toolkit* é inicializada através da chamada da função core_init(). Se a mesma for inicializada corretamente, o comando pc_param_set_any() retorna STS_OK, caso contrário retorna o código de falha STS_ERR.

A geração da chave privada da AC é realizada pela função bn_read_str() e a geração da chave privada dos nós é realizada através da função cp_sokaka_gen_prv(). Os processos citados anteriormente devem ser realizados, em implementações reais, antes da fase de implantação, como pode ser verificado na Seção 5.2.1. Estes procedimentos são presentes na aplicação apenas para ilustrar o funcionamento do protocolo de estabelecimento de chave.

Se a inicialização da biblioteca, a geração das chaves secretas da AC e dos nós sensores forem executadas com sucesso, a aplicação verifica o ID do nó que está sendo executado pela variável global TOS_LOCAL_ADDRESS e realiza a chamada da função agreeKey(). Dentro da função agreekey(), note no Código 5.5, existe a chamada da função cp_sokaka_key() responsável pelo cálculo da chave compartilhada entre duas entidades.

Código 5.5 `agreeKey()`

```
1 void agreeKey(int node) {
2   char nodeID[10];
3   snprintf(nodeID, 10, "%d", node);
4   cp_sokaka_key(&sharedKey, 10, thisID, strlen(thisID),
5   privateKey, nodeID, strlen(nodeID));
6 }
```

Após o cálculo da chave compartilhada, define-se o modo de transmissão do TinySec, altera-se a chave que será utilizada e reinicializa-se o VI especial da arquitetura de segurança TinySec. Estes procedimentos são realizados com as chamadas das seguintes funções:

1. *call* TinySecMode.setTransmitMode(TINYSEC_ENCRYPT_AND_AUTH).
2. *call* TinySecControl.updateEncriptionKey(&sharedKey).
3. *call* TinySecControl.resetIV().

O modo de transmissão na aplicação foi definido como autenticação e encriptação, porém decidiu-se atualizar apenas a chave de encriptação. A função updateEncriptionKey() é então utilizada para atualizar a chave que será utilizada pelo TinySec para fornecer encriptação dos dados. É necessário também resetar o vetor de inicialização da arquitetura de segurança através da função resetIV() fornecida pela interface TinySecControl. Os procedimentos descritos acima podem ser analisados no Código 5.6.

Código 5.6 `start()`

```
1  command result_t StdControl.start() {
2    if(inicializar() != 0) {
3    }
4    else {
5    inicial = call SysTime.getTime32();
6    if (TOS_LOCAL_ADDRESS == 0) {
7    agreeKey(1);
8    } else {
9    agreeKey(0);
10          }
11   }
12   call TinySecMode.setTransmitMode(TINYSEC_ENCRYPT_AND_AUTH);
13   call TinySecControl.updateEncryptionKey (&sharedKey);
14   call TinySecControl.resetIV();
15   final = call SysTime.getTime32();
16   result = final - inicial;
17   printInt32(result);
18   return SUCCESS;
19 }
```

A aplicação TinyPBC, contida dentro da RELIC-*toolkit*, foi utilizada em sua versão original com as adequações indicadas na Seção 5.5 . Um outro detalhe é que foi utilizada a versão 0.3.3 da RELIC-*toolkit* (quando a mais nova era a 0.3.5), pois as outras versões ou não compilavam com as configurações setadas [1], ou realizavam cálculos errados. Segundo o próprio autor da RELIC-*toolkit*, a diferença de desempenho entre as versões é mínima para emparelhamentos.

5.7 Requisitos de *Hardware* Exigidos dos Nós Sensores

Esta seção verifica os custos de *hardware* exigidos dos nós sensores ao utilizar um esquema de acordo de chaves baseado em identidade para o estabelecimento adequado de chaves criptográficas ao invés do pré-compartilhamento de um par de chaves antes da implantação. Assim, os custos aqui apresentados são relativos ao uso do esquema de acordo de chaves juntamente com o framework de segurança da camada de enlace TinySec. Dentre os parâmetros analisados podem-se citar:

[1] Os detalhes de implementação setados estão presentes na Seção 5.3.1

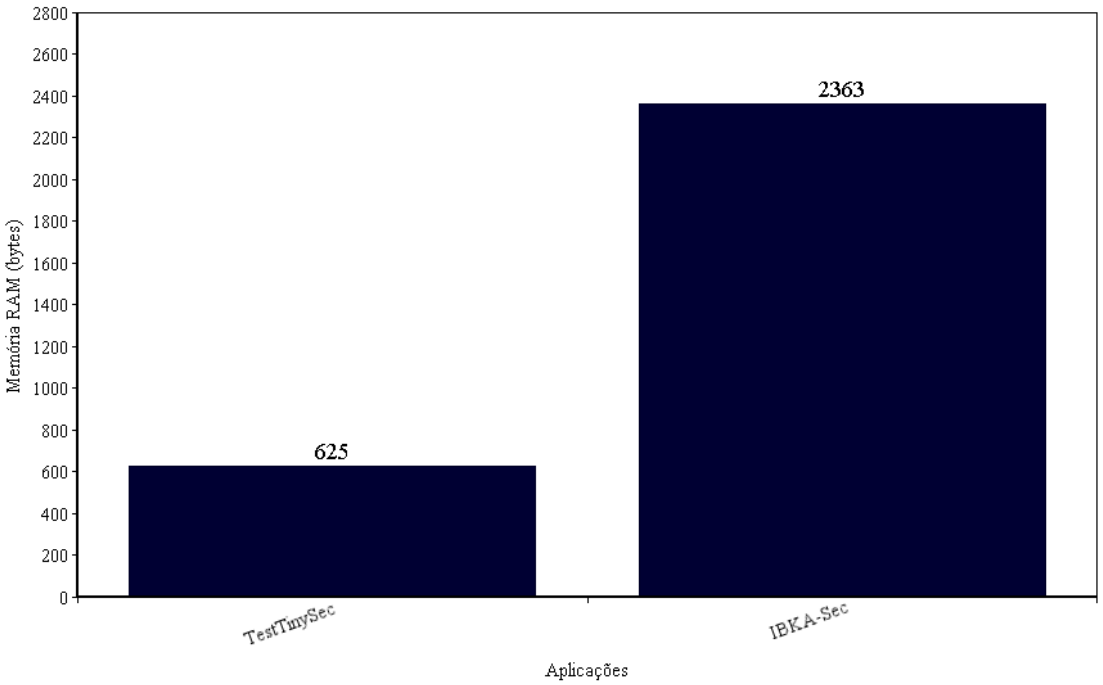

Figura 5.5: *Quantidade de memória RAM exigida pelas Aplicações*

- Quantidade de memória *RAM* e *ROM*;
- Tempo para execução;

A aplicação TestTinySec foi utilizada para comparação dos resultados. A aplicação TestTinySec apenas testa o funcionamento do TinySec ao enviar um contador de forma encriptada para outro nó. A aplicação IBKA-Sec também envia este contador de forma encriptada porém utilizando a chave calculada pelo protocolo de estabelecimento de chaves ao invés da chave pré-compartilhada antes da fase de implantação.

5.7.1 Quantidade Exigida de Memória

Ao se compilar o código da aplicação IBKA-Sec para o *hardware* Mica2 (através do comando *make* mica2), o compilador do TinyOS gera um relatório que contêm a ocupação de memória *ROM* (*FLASH*) e de memória *RAM* (*SRAM*). Logo, os resultados mostrados a seguir foram baseados nesse relatório.

As Figuras 5.5 e 5.6 mostram a quantidade de memória requerida pelas aplicações. A aplicação TestTinySec, configurada para prover autenticação e encriptação das mensagens enviadas, ocupa cerca de 49826 *bytes* de memória *ROM* e 625 *bytes* de me-

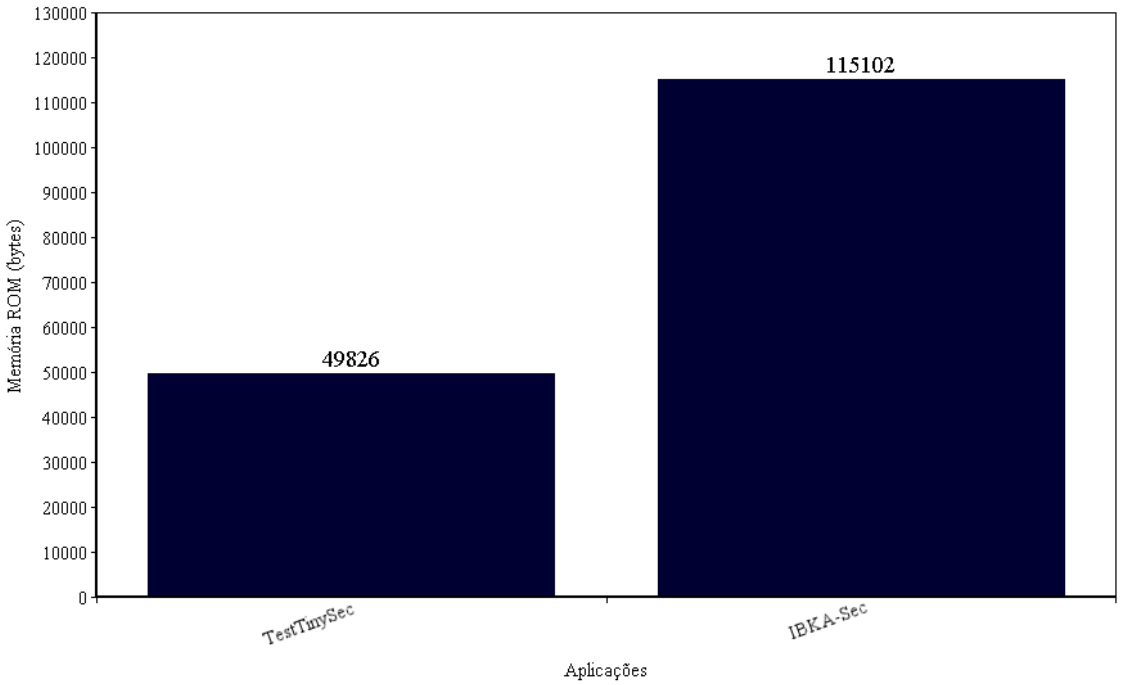

Figura 5.6: *Quantidade de Memória ROM exigida pelas Aplicações*

mória *RAM*. A aplicação IBKA-Sec ocupa 115102 *bytes* de memória *ROM* e 2363 *bytes* de memória *RAM*.

Os valores da quantidade de memória são um pouco alto, porém já esperado, pois a aplicação IBKA-Sec primeiramente executa o processo de estabelecimento de chaves para depois recorrer ao *framework* TinySec para autenticar/encriptar os dados. Além disso, algumas tarefas são realizadas na aplicação, tais como a geração das chaves privadas da AC e dos nós sensores. Essas tarefas não são realizadas pela aplicação em implementações reais (ocorrem antes da fase de implantação), o que diminui a quantidade de memória requerida.

5.7.2 Tempo Necessário Para Execução da Aplicação IBKA-Sec

Para descobrir o tempo total necessário para execução da aplicação IBKA-Sec, foi utilizada a interface SysTime provida pelo componente SysTimeC. No arquivo de configuração da aplicação IBKA-Sec, foi adicionado o componente SysTimeC. Na implementação da configuração foi adicionada a ligação entre o módulo da aplicação e a interface SysTime provida pelo componente SysTimeC.

Para verificar a quantidade de ciclos necessários para que um nó consiga executar

Tabela 5.2: *Resultados da Simulação da Aplicação IBKA-Sec*

Variáveis Analisadas	Valores Obtidos	Ferramenta Utilizada
Memória *RAM* (*bytes*)	2363	Compilador do TinyOS
Memória *ROM* (*bytes*)	115102	Compilador do TinyOS
Tempo de Execução (segundos)	1.94	Monitor c-print do Avrora

a aplicação IBKA-Sec, foi inserido um *timestamp* incial no início da aplicação. No final da execução da aplicação foi inserido outro *timestamp*. A diferença entre os dois *timestamps* foi então colocada como parâmetro do comando printInt32(), sentença para impressão de mensagens através do monitor c-print do simulador Avrora. Veja no Código 5.6 a posição dos *timestamps* inicial e final com o objetivo de verificação do tempo de execução da aplicação.

O simulador Avrora então foi configurado para rodar a aplicação com os seguintes parâmetros: avrora -platform=mica2 -simulation=sensor-network -nodecount=2 -printlogfile=logfile -monitors=c-print,leds nomeAplicação.elf

Cada parâmetro é especificado a seguir:

- -platform=mica2 - Indica a plataforma de experimentação;
- -simulation=sensor-network - Indica que é uma RSSF ao invés de um rede com um único nó;
- -nodecount=2 - Indica que a RSSF é composta por dois nós;
- -printlogfile=logfile - Indica o nome do arquivo que gravará os resultados das sentenças de impressão.
- -monitors=c-print,leds - Indica os monitores que estão sendo utilizados pelo simulador;
- nomeAplicação.elf - Indica qual aplicação que será simulada.

Note na Figura 5.7 que a interface SysTime, provida pelo componente SysTimeC, fornece um temporizador de 921.6 KHz para o mica2, ou seja, a unidade de medida para essa interface é *ticks* de processamento. A cada segundo tem-se 921600 *ticks* de processamento. Para execução da aplicação IBKA-Sec, desde a inicialização foram necessários 1787904 *ticks*, o que equivale a aproximadamente a 1.94 segundos de execução. Veja na Tabela 5.2 os principais resultados obtidos com a simulação da aplicação IBKA-Sec.

5.8 Requisitos de Segurança Adicionados

A arquitetura de segurança TinySec fornece os seguintes requisitos de segurança na camada de enlace dos dados:

- Confidencialidade.

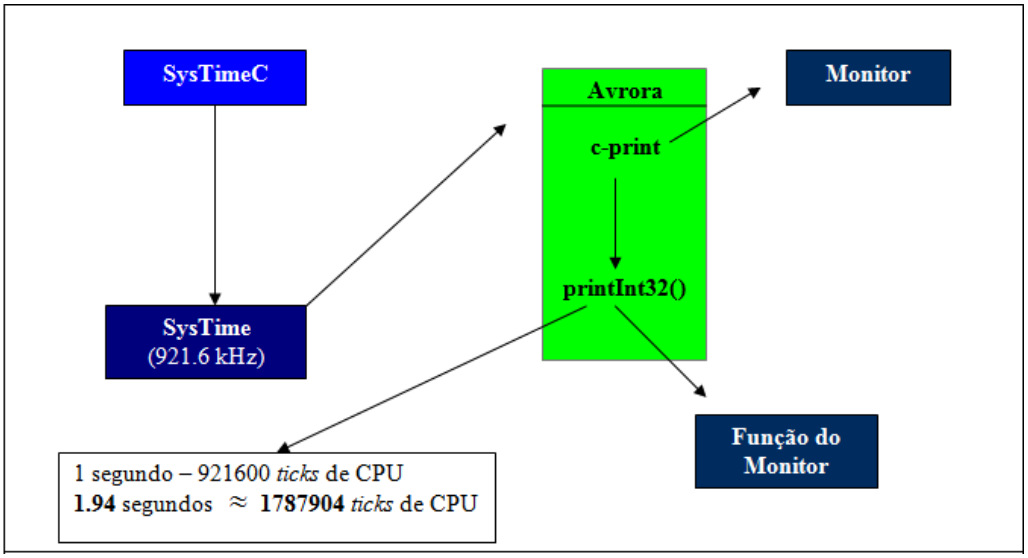

Figura 5.7: *Tempo Necessário para Execução da Aplicação IBKA-Sec*

- Integridade.
- Autenticidade

Em RSSF, evitar o comprometimento dos nós é essencialmente tratar das chaves criptográficas que serão utilizadas pelos algoritmos de criptografia. Uma chave global compartilhada entre os nós da rede, se obtida através de ataques físicos, comprometeria a segurança de toda a rede.

A fragilidade da arquitetura de segurança TinySec se baseia principalmente na utilização do pré-compartilhamento de uma chave global para autenticação e/ou encriptação dos dados. A existência de um protocolo específico de gerenciamento de chaves dificilmente possibilita que um atacante quebre os mecanimos de segurança utilizados pelo TinySec.

A adoção do mecanismo de distribuição de chaves proposto neste trabalho, o IBKA-Sec, faz com que qualquer ataque a um determinado nó se torne estritamente local, o comprometimento de um nó não afeta a comunicação entre os nós restantes da RSSF.

Apenas utilizando o TinySec como solução de segurança, o comprometimento de um único nó pode representar o comprometimento de toda a RSSF, uma vez que todos os nós, nesse contexto, são carregados com o mesmo segredo criptográfico. Note na Figura 5.3 os requisitos de seguranças que são adicionados com a proposta de junção IBKA-Sec.

A escolha adequada de um esquema de distribuição de chaves, neste caso de um esquema de distribuição de chaves baseado em identidade, aumenta o nível de segurança do *framwork* por utilizar primitivas assimétricas para distribuição das chaves criptográficas ao invés de primitivas simétricas.

A adoção da proposta IBKA-Sec ainda permite que um nó sensor envie informações para outro nó sem que o mesmo tenha calculado o segredo criptográfico para

Tabela 5.3: *Requisito de Segurança x Mecanismo de Segurança*

Requisito de Segurança	**Mecanismo de Segurança**	
	Apenas o TinySec	**Proposta IBKA-Sec**
Confidencialidade		
Integridade		
Autenticidade		
Proteção contra nós compromissados		
Encriptação Para Frente		

decriptar a informação. Essa característica se chama encriptação para frente (*Forward Encryption*).

A característica de encriptação para frente é muito importante uma vez que os nós sensores podem ter diferentes padrões de desligamento, ser empregados em momentos diferentes ou se tornarem temporariamente indisponíveis devido a obstáculos ou mau funcionamento.

A proposta de junção adotada neste trabalho é justificada para aplicações de RSSF no qual a segurança dos dados é um requisito prioritário, tais como na área militar ou na área da saúde.

A adoção do IBKA-Sec também é indicada para aplicações empregadas em locais públicos onde a probabilidade de ataque físico e uma possível recuperação de dados sigilosos armazenados na memória dos nós é maior que em ambientes particulares ou monitorados.

CAPÍTULO 6

Conclusões

As RSSF são compostas por pequenos sensores cujos recursos computacionais são extremamente limitados. O baixo custo e a rapidez na implantação fazem com que as mesmas sejam atrativas para diversos cenários. São utilizadas com vários objetivos, um deles é a possibilidade de monitoramento de regiões, oferecendo dados sobre a área que está sendo monitorada, de uma forma muito simples e eficiente.

Devido a um possível vazamento de informações e violação dos requisitos de segurança das aplicações de RSSF, é necessário que os desenvolvedores de aplicações se preocupem em aplicar técnicas seguras que previnem ou dificultam a ação de pessoas maliciosas. Embutir segurança em RSSF é uma tarefa complexa e desafiadora dado as limitações inerentes nesses cenários.

Uma das técnicas para prevenir os ataques em RSSF é o uso de criptografia. Neste sentido, diversas arquiteturas de segurança foram desenvolvidas, dentre elas destaca-se o TinySec, um *framework* que provê autenticação e encriptação na camada de enlace dos dados.

Um aspecto crucial no uso dos algoritmos de criptografia é como as chaves criptográficas são distribuídas. Idealmente, protocolos de estabelecimento de chaves devem fornecer perfeita conectividade entre os nós e poder de resiliência contra captura dos mesmos. Além disso, estes esquemas devem ser de baixo custo tanto em termos de processamento, como de comunicação e armazenamento.

O baixo poder computacional dos sensores inviabiliza o emprego de criptossistemas de chave pública tradicionais para solucionar o problema de distribuição de chaves criptográficas em RSSF. O emprego de técnicas baseadas na criptografia simétrica se torna mais interessante, porém a maioria destes esquemas não são adequados ou dependem de alguma interação entre os sensores para efetuar o acordo de chaves.

A arquitetura de segurança TinySec não contempla mecanismos para troca de chaves, o que sacrifica consideravelmente o seu nível de segurança. Envisiona-se que esquemas de distribuição IBKA são a solução ideal para o estabelecimento de chaves criptográficas em RSSF, simplificando o processo de distribuição e resolvendo a

forma inadequada que o *framework* simétrico TinySec utiliza para estabelecimento destas chaves.

O objetivo deste trabalho foi propor uma junção entre um mecanismo de distribuição de chaves baseado em identidade e o *framework* de segurança TinySec: o IBKA-Sec. A adoção de um mecanismo adequado de distribuição para ser utilizado juntamente com o TinySec é uma opção muito mais segura que o pré-compartilhamento de chaves criptográficas antes da fase de implantação.

Para provar a possibilidade de junção, apresentamos uma aplicação capaz de calcular uma chave de sessão em comum entre dois nós sensores e logo depois utilizar essa nova chave na arquitetura de segurança TinySec para encriptação dos dados.

Para promover a junção entre a aplicação TinyPBC e o framework de segurança da camada de enlace TinySec, alguns problemas foram enfrentados. O principal problema encontrado foi a incompatibilidade entre as duas tecnologias. Ambas foram projetadas para versões diferentes do TinyOS, o que dificulta a integração já que as mesmas utilizam ferramentas diferentes. O TinyOS 1.x requer ferramentas antigas para o correto funcionamento (GCC, por exemplo), já a RELIC-*toolkit* necessita de ferramentas mais novas para o seu funcionamento correto.

6.1 Trabalhos Futuros

Em relação aos trabalhos futuros, podem-se citar:

1. O estudo de novas curvas, tais como as curvas BN (Barreto-Naehrig), para mapeamento da identidade dos nós devido ao ataque (disponível no formato de relatório técnico em [6]) sobre as curvas supersingulares via cálculo do logaritmo discreto no corpo de extensão;
2. Verificação de possíveis otimizações na biblioteca RELIC-*toolkit* utilizada para diminuição da quantidade de memória requerida;
3. Analisar o custo do uso de outras cifras simétricas no TinySec juntamente com o esquema IBKA proposto devido á arquitetura ser independente de cifra;
4. Realizar a portabilidade do código do *framework* de segurança TinySec para o TinyOS 2.x, afim de evitar as incompatibilidades e dificuldades encontradas neste trabalho;

6.2 Publicações

Como fruto de pesquisas e estudos realizados durante a realização desse trabalho de dissertação podem-se citar uma apresentação oral em evento científico, uma publica-

ção/apresentação de artigo científico em simpósio nacional e um aceite para publicação em conferência internacional.

6.2.1 Apresentação de Trabalho em Evento Científico

- **LEMES, M. T.**; BULCÃO NETO, R. F.; OLIVEIRA, L. L. G.; RODRIGUES FILHO, R. V.; SENE JÚNIOR, I. G. **Um Esquema de Acordo de Chave Baseado em Identidade para o *Framework* de Segurança TinySec**. X CONPEEX - Congresso de Pesquisa, Ensino e Extensão. UFG, 2013.

6.2.2 Publicação e Apresentação em Simpósio Nacional

- **LEMES, M. T.**; BULCÃO NETO, R. F.; OLIVEIRA, L. L. G.; RODRIGUES FILHO, R. V.; SENE JÚNIOR, I. G. **Uma Nova Abordagem de Distribuição de Chaves Criptográficas para o *Framework* de Segurança TinySec**. Manaus-AM. Em: XIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais, v.1 p. 1-5, 2013.

6.2.3 Artigo Aceito para Publicação em Conferência Internacional

- **LEMES, M. T.**; BULCÃO NETO, R. F.; OLIVEIRA, L. L. G.; RODRIGUES FILHO, R. V.; SENE JÚNIOR, I. G. **A Identity Based Key Agreement for the Security Framework TinySec**. *In: The Third International Conference on Advanced Communications and Computation*. INFOCOMP 2013. Lisboa, Portugal.

Referências Bibliográficas

[1] AMIN, F.; JAHANGIR, A. H.; RASIFARD, H. **Analysis of Public-Key Cryptography for Wireless Sensor Networks Security**. *Proceedings of World Academy of Science: Engineering & Technology*, 2008.

[2] ANJUM, F.; MOUCHTARIS, P. **Security for Wireless Ad Hoc Networks**. John Wiley & Sons, 2007.

[3] ARANHA, D. F.; GOUVÊA, C. P. L. **RELIC is an Efficient LIbrary for Cryptography**. http://code.google.com/p/relic-toolkit/, Data do Último Acesso: 05/02/2014.

[4] ARANHA, D. F.; OLIVEIRA, L. B.; LÓPEZ, J.; DAHAB, R. **NanoPBC: Implementing Cryptographic Pairings on an 8-bit Platform**. In: *Conference on Hyperelliptic Curves, Discrete Logarithms, Encryption*, 2009.

[5] BANDIRMALI, N.; ERTURK, I. **WSNSec: A Scalable Data Link Layer Security Protocol for WSNs**. *Ad Hoc Networks*, 2012.

[6] BARBULESCU, R.; GAUDRY, P.; JOUX, A.; THOMÉ, E. **A Quasi-Polynomial Algorithm for Discrete Logarithm in Finite Fields of Small Characteristic**. Cryptology ePrint Archive, 2013.

[7] BARRETO, P. S. L. M.; GALBRAITH, S. D.; 'HÉIGEARTAIGH, C. Ó.; SCOTT, M. **Efficient Pairing Computation on Supersingular Abelian Varieties**. *Designs, Codes, and Cryptography*, 2007.

[8] BLAKE, I.; SEROUSSI, G.; SMART, N.; CASSELS, J. **Advances in Elliptic Curve Cryptography (London Mathematical Society Lecture Note Series)**. Cambridge University Press, 2005.

[9] BONEH.; LYNN.; SHACHAM. **Short Signatures from the Weil Pairing**. *JCRYPTOL: Journal of Cryptology*, 2004.

[10] BONEH, D.; BOYEN, X. **Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups**. *Journal of Cryptology*, 2008.

[11] BONEH, D.; FRANKLIN, M. **Identity-Based Encryption from the Weil Pairing**. *SIAM J. Comput.*, 2003.

[12] BOUJELBEN, M.; YOUSSEF, H.; MZID, R.; ABID, M. **IKM - An Identity Based Key Management Scheme for Heterogeneous Sensor Networks**. *Journal of Communications*, 2011.

[13] CULLER, D.; ESTRIN, D.; SRIVASTAVA, M. **Guest Editors' Introduction: Overview of Sensor Networks**. *Computer*, 2004.

[14] DE MEULENAER, G.; GOSSET, F.; STANDAERT, F.-X.; PEREIRA, O. **On the Energy Cost of Communication and Cryptography in Wireless Sensor Networks**. In: *IEEE International Conference on Wireless and Mobile Computing*, 2008.

[15] DONALD, E. K. **The Art of Computer Programming**. *Sorting and Searching*, 1999.

[16] DOYLE, B.; BELL, S.; SMEATON, A. F.; MCCUSKER, K.; CONNOR, N. E. O. **Security Considerations and Key Negotiation Techniques for Power Constrained Sensor Networks**. *The Computer Journal*, 2006.

[17] DU, W.; WANG, R.; NING, P. **An Efficient Scheme for Authenticating Public Keys in Sensor Networks**. *The 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*.

[18] FONG, K.; HANKERSON, D.; LÓPEZ, J.; MENEZES, A. **Field Inversion and Point Halving Revisited**. *Computers, IEEE Transactions*, 2004.

[19] GALINDO, D.; ROMAN, R.; LOPEZ, J. **A Killer Application for Pairings: Authenticated Key Establishment in Underwater Wireless Sensor Networks**. In: *Cryptology and Network Security*. 2008.

[20] GAY, D.; LEVIS, P.; VON BEHREN, R.; WELSH, M. A. **The nesC Language: A Holistic Approach to Networked Embedded Systems**.

[21] GENG YANG, CHUNMING RONG, C. V. J. W.; CHENG, H. **Identity-based key agreement and encryption for wireless sensor networks**. *JCSNS International Journal of Computer Science and Network Security*, 2006.

[22] GOYA, D.; MISAGHI, M.; RUFINO, V.; TERADA, R. **Modelos de criptografia de chave pública alternativos**. *Livro de Mini-cursos do IX Simpósio Brasileiro de Segurança da Informaçao e de Sistemas Computacionais*, 2009.

[23] HENRIQUE, C.; JARDIM, O.; AO NETO, R. F. B.; GODOY, R. P.; MÁXIMO, H.; RIBAS, B.; MUNSON, E. V.; GRAÇA, M.; PIMENTEL, C. **Web Services Enabling Ubiquitous Computing Applications: Lessons Learned by Integrating Ubiquitous e-Learning Applications**. *International Journal of Web Services Practices*, 2005.

[24] HESS, F.; SMART, N. P.; VERCAUTEREN, F. **The Eta Pairing Revisited**. *Information Theory: IEEE Transactions*, 2006.

[25] HOUSLEY, R.; YEE, P.; NACE, W. **Encryption Using KEA and SKIPJACK**. Technical report, 2000.

[26] HU, F.; SHARMA, N. K. **Security Considerations in Ad Hoc Sensor Networks**. *Ad Hoc Networks*, 2005.

[27] IYAMA, T.; KIYOMOTO, S.; FUKUSHIMA, K.; TANAKA, T.; TAKAGI, T. **Efficient Implementation of Pairing on BREW Mobile Phones**. In: *Advances in Information and Computer Security*. 2010.

[28] JR., M. A. S.; BARRETO, P. S. L. M.; MARGI, C. B.; CARVALHO, T. C. M. B. **A Survey on Key Management Mechanisms for Distributed Wireless Sensor Networks**. *Computer Networks*, 2010.

[29] KALLIO, P.; LATVAKOSKI, J. **Challenges and Requirements of Ubiquitous Computing**. *WSEAS Transactions on Information Science & Applications*, 2004.

[30] KARLOF, C.; SASTRY, N.; WAGNER, D. **TinySec: A Link Layer Security Architecture for Wireless Sensor Networks**. In: *Proceedings of the 2nd international conference on Embedded networked sensor systems*, 2004.

[31] KENT, S.; ATKINSON, R. **Security Architecture for the Internet Protocol**. RFC 2401 (Proposed Standard), November 1998. Obsoleted by RFC 4301, updated by RFC 3168.

[32] KIM, H. **Efficient and Non-Interactive Hierarchical Key Agreement in WSNs.** *International Journal of Security & Its Applications*, 2013.

[33] LEONARDO B. OLIVEIRA, FELIPE DAGUANO, R. D. **Avaliando Protocolos de Criptografia Baseada em Emparelhamentos em Redes de Sensores Sem Fio**. *VII Simpósio Brasileiro em Segurança da Informaç ao e de Sistemas Computacionais*, 2007.

[34] LEVIS, P. **TinyOS Programming**. *Available in http://www.tinyos.net/tinyos-2.x/doc/pdf/tinyos-programming.pdf*, 2006.

[35] LIU, A.; NING, P. **TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks**. In: *Information Processing in Sensor Networks. International Conference on*, 2008.

[36] LÓPEZ, J.; DAHAB, R. **Improved Algorithms for Elliptic Curve Arithmetic in GF(2n)**. Springer Berlin Heidelberg, 1999.

[37] LÓPEZ, J.; DAHAB, R. **High-Speed Software Multiplication in F(2m)**. In: *Progress in Cryptology*. 2000.

[38] MAAS, M. **Pairing-Based Cryptography**. PhD thesis, 2004.

[39] MADDEN, S.; SZEWCZYK, R.; FRANKLIN, M.; CULLER, D. **Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks**. In: *Mobile Computing Systems and Applications. Proceedings Fourth IEEE Workshop on*, 2002.

[40] MADDEN, S.; FRANKLIN, M. J.; HELLERSTEIN, J. M.; HONG, W. **TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks**. *ACM SIGOPS Operating Systems Review*, 2002.

[41] MALAN, D.; WELSH, M.; SMITH, M. **A Public-Key Infrastructure for Key Distribution in TinyOS based on Elliptic Curve Cryptography**. In: *Sensor and Ad Hoc Communications and Networks. First Annual IEEE Communications Society Conference on*, 2004.

[42] MARGI, C.; SIMPLICIO, M.; BARRETO, P.; CARVALHO, T. **Segurança em redes de sensores sem fio**. *Livro de Minicursos do IX Simpósio Brasileiro em Segurança da Informaç ao e de Sistemas Computacionais*, 2009.

[43] MILLER, V. **Short Programs for Functions on Curves**. *Unpublished Manuscript*, 1986.

[44] OLIVEIRA, L. B.; ARANHA, D. F.; GOUVÊA, C. P.; SCOTT, M.; CÂMARA, D. F.; LÓPEZ, J.; DAHAB, R. **TinyPBC: Pairings for Authenticated Identity-Based Non-Interactive Key Distribution in Sensor Networks**. *Computer Communications*, 2011.

[45] OLIVEIRA, L. B.; ARANHA, D. F.; MORAIS, E.; DAGUANO, F.; LÓPEZ, J.; DAHAB, R. **Tinytate: Computing the Tate Pairing in Resource-Constrained Sensor Nodes**. In: *Network Computing and Applications. Sixth IEEE International Symposium on*, 2007.

[46] OLIVEIRA, L. B.; DAHAB, R.; LÓPEZ, J.; DAGUANO, F.; LOUREIRO, A. A. F. **Identity-Based Encryption for Sensor Networks**. In: *PerCom Workshops*, 2007.

[47] OLIVEIRA, L. B.; SCOTT, M.; LÓPEZ, J.; DAHAB, R. **TinyPBC: Pairings for Authenticated Identity-Based Non-Interactive Key Distribution in Sensor Networks**. In: *In Networked Sensing Systems. 5th International Conference on*, 2008.

[48] PERRIG, A.; SZEWCZYK, R.; TYGAR, J.; WEN, V.; CULLER, D. E. **SPINS: Security Protocols for Sensor Networks**. *Wireless networks*, 2002.

[49] PHILIP LEVIS, SAM MADDEN, J. P. R. S. K. W. A. W. D. G. J. H. M. W. E. B. D. C. **TinyOS: An Operating System for Sensor Networks**. Technical report, 2005.

[50] REN, J.; LI, T.; ASLAM, D. **A Power Efficient Link-Layer Security Protocol (LLSP) for Wireless Sensor Networks**. In: *Military Communications Conference*, 2005.

[51] RODRIGUES, I. A. L. **Distribuição Segura de Chaves Criptográficas em Redes de Sensores Sem Fio de Grande Escala: Estudo e Avaliação Experimental em Ambiente de Simulação**. PhD thesis, 2011.

[52] ROSLI, R.; YUSOFF, Y. M.; HASHIM, H. **A Review on Pairing Based Cryptography in Wireless Sensor Networks**. In: *Wireless Technology and Applications. IEEE Symposium on*, 2011.

[53] SAKAI R., OHGISHI K., K. M. **Cryptosystems Based on Pairing**. In: *In Symposium on Cryptography and Information Security*, 2000.

[54] SCHNEIER, B. **Applied Cryptography: Protocols, Algorithms, and Source Code in C**.

[55] SCOTT, M. **Optimal Irreducible Polynomials for GF (2m) Arithmetic**. *IACR Cryptology ePrint Archive*, 2007.

[56] SHAMIR. **Identity-Based Cryptosystems and Signature Schemes**. In: *CRYPTO: Proceedings of Crypto*, 1984.

[57] STALLINGS, W. **Criptografia e Segurança de Redes: Princípios e Práticas**. Pearson Prentice Hall, 2008.

[58] SZCZECHOWIAK, P.; COLLIER, M. **TinyIBE: Identity-Based Encryption for Heterogeneous Sensor Networks**. In: *The 5th International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, 2009.

[59] SZCZECHOWIAK, P. **Cryptographic Key Distribution In Wireless Sensor Networks Using Bilinear Pairings**. PhD thesis, 2010.

[60] SZCZECHOWIAK, P.; COLLIER, M. **Practical Identity-Based Key Agreement For Secure Communication in Sensor Networks**. In: *Computer Communications and Networks*, 2009.

[61] SZCZECHOWIAK, P.; KARGL, A.; SCOTT, M.; COLLIER, M. **On the Application of Pairing Based Cryptography to Wireless Sensor Networks**. In: *Proceedings of the Second ACM Conference on Wireless Network Security*, 2009.

[62] SZCZECHOWIAK, P.; OLIVEIRA, L.; SCOTT, M.; COLLIER, M.; DAHAB, R. **NanoECC: Testing the Limits of Elliptic Curve Cryptography in Sensor Networks**. In: *Wireless Sensor Networks*. 2008.

[63] TITZER, B. L.; LEE, D. K.; PALSBERG, J. **Avrora: Scalable Sensor Network Simulation with Precise Timing**. In: *Information Processing in Sensor Networks. Fourth International Symposium on*, 2005.

[64] WALTERS, J. P.; LIANG, Z.; SHI, W.; CHAUDHARY, V. **Wireless Sensor Network Security: A Survey in Book Chapter of Security**. In: *Distributed, Grid, and Pervasive Computing, Yang Xiao Eds*, 2007.

[65] WATRO, R.; KONG, D.; CUTI, S.-F.; GARDINER, C.; LYNN, C.; KRUUS, P. **TinyPK: Securing Sensor Networks with Public Key Technology**. In: *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, 2004.

[66] WEISER, M. **The computer for the 21st century**. *SIGMOBILE Mob. Comput. Commun. Rev.*, 1999.

[67] XIONG, X.; WONG, D. S.; DENG, X. **TinyPairing: A Fast and Lightweight Pairing-Based Cryptographic Library for Wireless Sensor Networks**. In: *Wireless Communications and Networking Conference*, 2010.

[68] YIXIN JIANG, CHUANG LIN, M. S.; SHEN, X. S. **Key Management Schemes for Wireless Sensor Networks**. *Security in Sensor Networks*, 2007.

[69] YLONEN, T. **SSH: Secure Login Connections Over the Internet**. In: *Proceedings of the 6th conference on USENIX Security Symposium, Focusing on Applications of Cryptography*, 1996.

[70] YOUNG, E. A.; HUDSON, T. J.; ENGELSCHALL, R. S. **OpenSSL**. *World Wide Web, http://www. openssl. org/, Data do Último Acesso: 05/02/2014*.

[71] ZHU, S.; SETIA, S.; JAJODIA, S. **LEAP: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks**. *ACM Transactions on Sensor Networks*, 2006.