Segurança da Comunicação em Redes de Sensores em Malha

Rui Miguel Pinheiro Pires

Dissertação para obtenção do Grau de Mestre em

Telecomunicações e Informática

Orientador: Prof. Miguel Nuno Dias Alves Pupo Correia

Júri

Presidente: Prof. Paulo Jorge Pires Ferreira
Orientador: Prof. Miguel Nuno Dias Alves Pupo Correia
Vogal: Prof. Henrique João Lopes Domingos

Novembro 2015

Dedicada aos meus pais e irmão.

Agradecimentos

Em primeiro lugar quero agradecer à minha Mãe e ao meu Pai por me acompanharem desde sempre, por estarem sempre presentes e por serem o meu exemplo. Ao meu Irmão, também um obrigado muito especial.

Aos meus Avós por tudo o que me ensinaram e pelo seu exemplar percurso de vida. Aos meus tios pelo apoio, que nunca me faltou, e pelo encorajamento. Enfim, agradeço a toda a minha família.

Agradeço a todos os meus colegas e amigos por todos os momentos de entusiasmo partilhados em conjunto.

A toda a equipa da Sensefinity, pelo apoio e suporte.

E finalmente um agradecimento muito especial ao professor Miguel Correia pela sua orientação, pela sua disponibilidade, apoio total, pelas suas criticas e colaboração.

Resumo

Graças aos recentes avanços tecnológicos, o fabrico de pequenos sensores com consumo reduzido de energia e com capacidade de comunicação sem fios, tem-se tornado possível e economicamente viável. A combinação das características destes dispositivos como o seu tamanho reduzido, o baixo custo e a capacidade de comunicação de sem fios, faz das redes de sensores sem fios uma área com enormes possibilidades de aplicação. Este tipo de redes poderão ser usadas em vários tipos de aplicações, desde a monitorização geográfica, por exemplo efetuando o *tracking* da localização de animais, a sistemas de estacionamento inteligentes, ou mesmo na área militar.

A proliferação do uso de redes de sensores torna a segurança um campo de investigação prioritário, com o objetivo de garantir a fiabilidade e o correto funcionamento das aplicações, suportadas pelas mesmas. Os dispositivos normalmente utilizados nas redes de sensores possuem limitações de recursos, estando normalmente limitados em termos de capacidade de processamento, de memória, e energética. Estas limitações tornam por vezes impossível a utilização de algumas das tradicionais técnicas de segurança no desenho de uma solução para a proteção deste tipo de redes.

Este trabalho analisa em concreto o problema da segurança da comunicação para sistema composto por várias redes de sensores em malha, que tem como objetivo monitorizar a localização de objetos. Ao longo deste documento é apresentada a solução idealizada para resolver este problema, é descrita a sua implementação e apresentada a análise feita sobre o seu funcionamento.

Palavras-chave: Redes de sensores, segurança, *routing* em redes de sensores.

Abstract

Thanks to the recent technological advances, the manufacture of small sensors with reduced power consumption and wireless communication capabilities, has become possible and economically viable. The combination of the characteristics of these devices, such as their small size, their low cost and wireless communication capability, gives to the wireless sensor networks (WSN) a large area of application possibilities. They can be used in geographical monitoring, performing for example the tracking of an animal 's location, in smart parking systems, or even in military applications.

The proliferation of wireless sensor networks makes security a priority research field in order to ensure the reliability and proper operation of the applications supported by them. The devices usually used in wireless sensor networks have limited resources, normally they are limited in terms of processing, memory and energy capabilities. These limitations make the use of some traditional security techniques impossible.

This work analyses in particular the problem of communication security for a network of this kind, and presents a solution. This solution is implemented and its operation is analysed.

Keywords: Wireless sensor networks, security, routing in wireless sensor networks.

Conteúdo

Agradecimentos v
Resumo vii
Abstract ix
Lista de Tabelas xiii
Lista de Figuras xv
Lista de Símbolos xvii
Glossário xvii

1 Introdução **1**
1.1 Motivação 1
1.2 Problema 2
1.3 Objectivos 3
1.4 Organização 3

2 Trabalho Relacionado **5**
2.1 Segurança em redes de sensores 5
2.1.1 Redes de sensores 5
2.1.2 Ameaças e modelos de confiança 6
2.1.3 Requisitos de segurança 7
2.1.4 Principais ataques 8
2.2 Mecanismos de segurança 12
2.2.1 Comunicação segura 12
2.2.2 Autenticação de mensagens de *broadcast* 16
2.2.3 Gestão de chaves 19
2.3 *Routing* em redes malha e de sensores 23
2.3.1 Principais protocolos de *routing* 23
2.3.2 *Routing* seguro 27
2.4 Resumo do capítulo 34

3 Solução Proposta **37**
3.1 Modelação do problema 37
3.1.1 Requisitos 39

3.2 Visão geral da solução 39
3.2.1 Autenticação na rede 41
3.2.2 Comunicação segura 41
3.2.3 Comunicação em grupo 42
3.2.4 Mensagens de alarme 43
3.2.5 Múltiplos caminhos 43
3.3 Descrição da implementação 43
3.3.1 Criptografia de chave pública 44
3.3.2 Autenticação na rede 45
3.3.3 Comunicação segura 48
3.3.4 Comunicação em grupo 53
3.3.5 Mensagens de alarme 54
3.3.6 Múltiplos caminhos 54
3.4 Resumo do capítulo 55

4 Avaliação Experimental **57**
4.1 Consumo de bateria 57
4.2 Criptografia de curvas elípticas 58
4.3 Comunicação segura 61
4.4 Análise global 63
4.5 Resumo do capítulo 65

5 Conclusão **67**
5.1 Trabalho futuro 68

Bibliografia **69**

Lista de Tabelas

2.1 Características de dispositivos usados em redes de sensores. 6

4.1 Memória requerida, para os três tamanhos de chaves assimétricas disponíveis. 59
4.2 Dados estatísticos, para o uso de chaves assimétricas de 128 bits. 59
4.3 Dados estatísticos, para o uso de chaves assimétricas de 160 bits 59
4.4 Dados estatísticos, para o uso de chaves assimétricas de 192 bits. 60
4.5 Análise ao consumo de bateria imposto pela execução do protocolo de autenticação. . . 63

Lista de Figuras

1.1 Arquitetura do sistema. Nós móveis estão representados pelas bolas pretas, já os quadrados vermelhos representam os nós *gateway*. Cada nó *gateway* recebe mensagens dos nós móveis da sua rede e encaminha as mesmas para unidade central do sistema (UC). 3

2.1 Exemplo de um *wormhole attack*. 10
2.2 Comparação entre o formato original dos pacotes de dados usados no *TinyOs* e os formatos propostos pelos dois métodos apresentados no TinySec, [16]. 14
2.3 *Formato de um pacote MiniSec-U.* 16
2.4 Exemplo do funcionamento do protocolo μTESLA, [15]. 17
2.5 Funcionamento do protocolo ARMS, [18]. 18
2.6 *Exemplo de uma árvore de encaminhamento criada pelo protocolo TinyOS beaconing, [4].* 26
2.7 *Inicio de uma nova ronda de difusão da mensagem de routing.* 30
2.8 [34] Nó comprometido α falsifica contagem anunciando *hop count* igual a um em vez de três. 31
2.9 *[34] Nó comprometido α falsifica contagem anunciando hop count igual ao anunciado por β.* . . . 31

3.1 Arquitetura do sistema. Nós móveis estão representados pelas bolas pretas, já os quadrados vermelhos representam os nós *gateway*. Cada nó *gateway* recebe mensagens dos nós móveis da sua rede e encaminha as mesmas para unidade central do sistema (UC). 38
3.2 Dispositivos usados na implementação do sistema. 39
3.3 Arquitectura do sistema, solução. 40
3.4 Protocolo *ECDH* (*Elliptic curve Diffie–Hellman*). 46
3.5 Protocolo de autenticação mútua implementado. 47
3.6 Protocolo de confirmação de receção das mensagens de alarme. 49
3.7 Modo de cifra *CBC*. Ti e Ci representam blocos do texto e do criptograma respetivamente, E é a função de cifra e D é a função de decifra 50
3.8 Esquema completo para a proteção das mensagens. 52
3.9 Mensagem de *routing* original, a), e mensagem de *routing* protegida, b). 53
3.10 Envio de mensagens por múltiplos caminhos. 55

4.1 Variação da percentagem de bateria ao longo do tempo. 58
4.2 Variação da percentagem de memória usada para os três tamanhos de chaves assimétricas disponíveis. 59
4.3 Variação do tempo médio de execução de cada uma das operações necessárias no protocolo de autenticação, para os diferentes tamanhos de chaves assimétricas. 60
4.4 *Análise ao consumo de bateria imposto por cada operação.* 61
4.5 *Tempos de computação para as operações de cifra e calculo do MAC, decifra e validação do MAC, para diferentes tamanhos de conteúdo.* 62
4.6 *Distribuição do tempo de execução do protocolo de autenticação, teste realizado entre nó e gateway com comunicação direta.* 63
4.7 *Análise ao consumo de bateria imposto pela execução do protocolo de autenticação.* 64

Glossário

AES	Advanced Encryption Standard.
CA	Certification Authority.
CBC-MAC	Cipher Block Chaining Message Authentication Code.
CBC	Cipher Block Chaining.
CMAC	Cipher-based Message Authentication Code.
ECC	Elliptic Curve Cryptography.
ECDH	Elliptic Curve Diffie–Hellman.
IV	Initialization Vector.
MAC	Message Authentication Code.
PBC	Pairing-Based Cryptography.
UC	Unidade Central.

Capítulo 1

Introdução

Graças aos recentes avanços tecnológicos, o fabrico de pequenos sensores com consumo reduzido de energia e com capacidade de comunicação sem fios tem-se tornado possível e economicamente viável. O uso deste tipo de sensores tornou possível a criação de redes de sensores sem fios. A combinação das características destes dispositivos, como o seu tamanho reduzido, o relativo baixo custo de produção e a capacidade de comunicação sem fios, faz das redes de sensores sem fios uma área com enormes possibilidades de aplicação e que, por isso, se encontra em grande desenvolvimento.

Uma rede de sensores sem fios consiste em pequenos dispositivos deste tipo, aleatoriamente e densamente distribuídos no meio e com a possibilidade de comunicação entre eles. Em muitos casos existe também a necessidade de uma organização automática, ou seja, não deve ser necessária intervenção externa posteriormente à fase de disposição dos sensores no meio.

Tendo em conta as suas características, as redes de sensores sem fios, poderão ser aplicadas em várias áreas, desde a área militar para detetar intrusões e monitorizar as condições ambientais do meio, à área industrial, por exemplo, para realizar o controlo do inventário [1], ou para aplicações como a monitorização da localização geográfica de objetos.

A proliferação do uso deste tipo de redes torna a segurança um campo de investigação prioritário, com o objetivo de garantir a fiabilidade e o correto funcionamento das aplicações suportadas por estas redes. Como a comunicação é realizada recorrendo a ligações sem fios (*wireless*), um atacante pode escutar e interferir com a comunicação, e introduzir ou repetir mensagens. Desafios como a mobilidade e as limitadas capacidades dos dispositivos usados tornam por vezes impossível a utilização de algumas das tradicionais técnicas de segurança no desenho de uma solução para a proteção deste tipo de redes. As especificidades que as redes de sensores sem fios apresentam, são portanto desafios que devem ser ultrapassados de modo a garantir as propriedades de segurança desejadas.

1.1 Motivação

As redes de sensores são cada vez mais usadas, mas para garantir a evolução desta tecnologia é necessário torna-las seguras e fiáveis, sendo portanto a segurança um campo de investigação prioritário

nesta área.

As características especiais que este tipo de redes presenta, como as limitadas capacidades dos dispositivos usados, faz com que sejam necessárias soluções de segurança diferentes das tradicionalmente usadas nas redes de computadores mais comuns.

Dadas estas dificuldades, várias são as propostas para garantir a segurança, que apenas se focam em casos limitados ou em garantir apenas determinadas propriedades da segurança, e poucas são as propostas que apresentam uma solução completa para este tipo de redes.

1.2 Problema

Este projeto surge com o objetivo de solucionar as questões de segurança informática associadas a um produto de monitorização da localização de objetos de grandes dimensões. O projeto está a ser desenvolvido pela *Sensefinity*, uma *startup* tecnológica sediada em Lisboa.

O problema consiste em garantir as diversas facetas de segurança da comunicação entre um conjunto de dispositivos que formam várias redes em malha (*mesh networks*), e que comunicam com uma entidade central que se pode encontrar geograficamente distante. Esta entidade é contactável através da Internet ou via rede móvel.

Cada uma destas redes em malha pode ser constituída por dezenas de nós com sensores e um ou mais nós com maiores capacidades, e sem limitação de energia, que funcionam como *gateway*, ponto de saída da rede. O *gateway* de cada rede permite a comunicação entre os elementos da mesma, e a unidade central do sistema. Para tal estes nós *gateway* estão ligados à Internet ou possuem capacidade de comunicação via rede móvel. Na unidade central a informação recebida é agregada e disponibilizada ao observador do sistema.

O sistema possui a arquitetura ilustrada na figura 1.1. O sistema é formado por um conjunto de várias redes em malha, sendo cada rede constituída por vários nós móveis (bolas pretas) e um ou mais nós *gateway* (quadrados vermelhos). Os nós *gateway*, como já referido, possibilitam a comunicação entre os elementos da rede e a unidade central (UC).

Os nós de cada rede consistem em dispositivos de tamanho reduzido, com capacidades computacional e energética limitadas, associados a um objeto de grandes dimensões que possui a capacidade de mobilidade. Ainda que permaneçam estáticos grande parte do tempo, estes objetos poderão alterar a sua posição na rede ou deslocar-se entre as várias redes que pertencem ao sistema.

O objetivo do sistema é realizar a monitorização da localização geográfica e do estado dos objetos a que os dispositivos estão associados. Para tal cada um dos dispositivos envia periodicamente uma mensagem com a sua informação de localização. Assim, se um objeto sair de uma rede, ou se encontrar incontactável, tal é detetado pela falta desta comunicação originada pelo dispositivo que têm associado. Se ocorrer um determinado evento no objeto que os dispositivos têm capacidade de detetar, o observador do sistema é alertado pelo envio de uma mensagem de alarme.

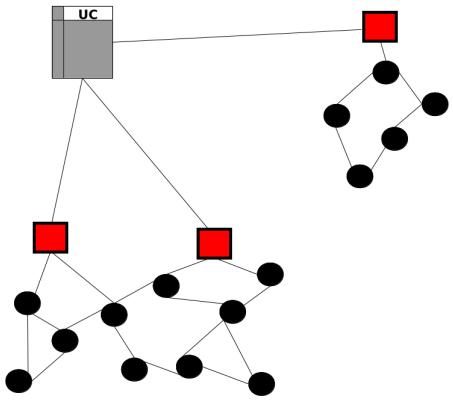

Figura 1.1: Arquitetura do sistema. Nós móveis estão representados pelas bolas pretas, já os quadrados vermelhos representam os nós *gateway*. Cada nó *gateway* recebe mensagens dos nós móveis da sua rede e encaminha as mesmas para unidade central do sistema (UC).

1.3 Objectivos

A intenção deste trabalho é estudar a problemática da segurança para as redes de sensores. Tendo como principais objetivos:

- Estudar o funcionamento das redes de sensores sem fios e analisar o problema de segurança deste tipo de redes.
- Compreender as especificidades do problema enunciado, e desenhar uma solução de segurança que garanta as propriedades básicas para a segurança da comunicação, nomeadamente autenticidade, integridade e confidencialidade.
- Implementar a solução desenhada e analisar o seu funcionamento.

1.4 Organização

Este documento é constituído por cinco capítulos e um anexo. Os capítulos do documento são os seguintes:

- 1. Introdução;
- 2. Trabalho relacionado;
- 3. Solução proposta;
- 4. Avaliação experimental;
- 5. Conclusões.

No capítulo inicial, o atual, é realizada a introdução ao tema abordado pelo documento, é indicada a motivação, é descrito o problema e são apresentados os objetivos deste trabalho.

No capítulo 2 é exposto o trabalho relacionado sobre aspetos de segurança das redes de sensores e redes em malha. Na primeira secção, a 2.1, é feita uma introdução ao tópico das redes de sensores sem fios e são identificadas as principais ameaças e os principais tipos de ataques a que este tipo de redes estão sujeitas. De seguida, na secção 2.2, são apresentadas as principais propostas de mecanismos desenhados para contrariar tais ameaças. Na última secção, a 2.3, é feita uma análise ao problema do *routing* em redes de sensores sem fios, identificando os principais protocolos, e apresentando e analisando as propostas de protocolos de *routing* seguros.

No terceiro capítulo é descrito em maior detalhe o problema em questão e é apresentada a solução desenhada para o mesmo. Na primeira secção do capítulo, a 3.1, é apresentado em maior detalhe o problema, e são apresentados os requisitos de segurança identificados. Posteriormente, na secção 3.2, é apresentada uma visão geral sobre a solução desenhada. Na última secção do capítulo, a 3.3, é apresentada a implementação, de um modo detalhado, para cada um dos pontos da solução.

No capítulo 4 são descritos e avaliados os resultados da avaliação experimental feita sobre a solução implementada. É analisado o impacto da cada uma das técnicas de criptografia usadas pela solução e é também analisado o impacto geral provocado pela solução no sistema.

Por fim, no capítulo 5 é apresentada a conclusão do documento e são feitas algumas considerações sobre o trabalho futuro.

Capítulo 2

Trabalho Relacionado

Este capítulo é constituído por três secções. Na primeira secção, a 2.1, é feita uma introdução ao tópico das redes de sensores sem fios e são identificadas as principais ameaças e os principais ataques a que este tipo de redes estão sujeitas. De seguida na secção 2.2 são identificados os principais mecanismos desenhados para contrariar tais ameaças. Na última secção deste capítulo é feita uma análise ao *routing*, um ponto essencial nestes tipos de redes.

2.1 Segurança em redes de sensores

2.1.1 Redes de sensores

Graças aos recentes avanços tecnológicos, o fabrico de pequenos sensores com consumo reduzido de energia e com capacidade de comunicação sem fios, tem-se tornado possível e economicamente viável. A combinação das características destes dispositivos como o seu tamanho reduzido, o baixo custo e a capacidade de comunicação de sem fios, faz das redes de sensores sem fios uma área com enormes possibilidades de aplicação.

Vários tem sido os dispositivos que recentemente tem sido apresentados, como por exemplo *TelosB* [1], *MICAz* [2] e *Imote2* [3]. Uma rede de sensores sem fios consiste em dispositivos deste tipo, aleatoriamente e densamente distribuídos no meio, e com a necessidade de se auto organizarem, ou seja, não deve ser necessária intervenção externa posteriormente à fase de disposição dos sensores no meio. Na grande maioria das redes de sensores existe um nó, o *gateway*, responsável por efetuar a ligação entre a rede sem fios e uma infraestrutura externa, como por exemplo a Internet.

Como identificado em [2], este tipo de dispositivos poderão ser usados em vários tipos de aplicações como a monitorização geográfica, monitorização de habitat, por exemplo efetuando o *tracking* da localização de animais, ou sistemas de estacionamento inteligentes entre outros. Estes sensores podem medir as condições ambientais do ambiente que os rodeia e, em seguida, transformar estas

[1] http://www.willow.co.uk/TelosB_Datasheet.pdf.
[2] http://www.openautomation.net/uploadsproductos/micaz_datasheet.pdf
[3] http://cps.cse.wustl.edu/images/e/e3/Imote2_Datasheet.pdf.

medições em sinais. Estes sinais podem ser processados localmente, de modo a revelar algumas características, e por exemplo, em caso de alteração de estado de uma determinada caraterística, poderá ser enviada uma mensagem de alerta para uma unidade central.

Como já referido, os dispositivos normalmente utilizados nas redes de sensores possuem limitações de recursos [3], estando normalmente limitados em termos de capacidade de processamento e de memória, e com pouca capacidade energética. A título de exemplo a tabela 2.1 apresenta as características de alguns dos dispositivos normalmente usados nas redes de sensores.

Nome	Microcontrolador	Rádio	Memória- RAM	Memória- ROM
MICAz	8bit Atmel - 8 MHz	802.15.4 - 250 kb/s	4 kB	128 kB+ 512kB
Imote	32bit ARM - 12 MHz	Bluetooth -720 kb/s	64K SRAM	512K Flash
TelosB	16bit MSP430 - 8 MHz	802.15.4 - 250 kb/s	10 K SRAM	48 KB/1024 KB

Tabela 2.1: Características de dispositivos usados em redes de sensores.

As redes de sensores são por vezes designadas também como redes em malha (*mesh networks*), pois partilham algumas semelhanças. A principal semelhança é o facto de em ambas as redes o modelo de comunicação ser *multi-hop*, ou seja, recorrendo a sensores intermédios que têm a capacidade de encaminhar as mensagens.

Porém como identificado em [4], para além da maior limitação em termos dos recursos dos dispositivos usados em redes de sensores, a grande diferença entre ambas as redes é o facto de nas redes de sensores existirem essencialmente os seguintes três fluxos de comunicação distintos:

- Muitos para um - Múltiplos sensores enviam dados que captam do meio para o *gateway*, ou para um ponto de agregação de dados na rede.
- Um para muitos - Um único nó, normalmente o *gateway*, difunde mensagens pela rede, por exemplo a requerer novos dados ou com informação de configuração.
- Comunicação local - Comunicação entre nós vizinhos, ou seja, entre dois nós que se encontram ao alcance um do outro. Este tipo de comunicação é útil para estabelecimento das rotas de encaminhamento e para troca de dados sobre o meio envolvente.

Como se verifica, por norma, nas redes de sensores não existe um fluxo de comunicação entre dois nós da rede que não sejam vizinhos. Esta é a principal diferença entre as redes de sensores e as chamadas redes malha, pois as redes em malha caracterizam-se pela possibilidade de comunicação entre todos os elementos de uma rede.

2.1.2 Ameaças e modelos de confiança

Como a comunicação neste tipo de redes é realizada sem fios (*wireless*), é assumido que o meio de transmissão é inseguro, pelo que um atacante pode escutar a comunicação e introduzir ou repetir mensagens. Tal como a entidade que possui a rede, um atacante pode ainda distribuir pelo meio novos sensores com capacidades semelhantes às dos nós legítimos. Dada a falta de segurança física é

normal assumir também que um atacante consegue capturar os nós de uma rede, e ler e escrever na memória dos mesmos [5].

Uma distinção importante sobre o modelo de ameaças pode ser feita tendo em conta as capacidades dos recursos que o atacante dispõe. Em [4] é apresentada a distinção entre *mote-class attackers* e *laptop-class attackers*. No primeiro caso o atacante utiliza dispositivos com capacidades semelhantes às dos dispositivos usados na rede. Na segunda hipótese o atacante usa dispositivos mais poderosos, como por exemplo um computador portátil. No segundo modelo o atacante tem vantagem sobre os nós da rede pois apresenta maior poder computacional, maior alcance e maior capacidade energética.

Uma segunda diferenciação pode ser feita entre atacantes internos ou externos. A ameaça externa consiste em ataques realizados por elementos que não pertencem à rede nem à entidade que coordena a mesma. Por outro lado, os ataques internos são realizados por nós legítimos que foram comprometidos, ou por elementos externos que através do roubo de informação, como chaves criptográficas de nós legítimos, conseguem participar de modo aparentemente legítimo na rede.

Como já referido a grande maioria das redes em malha e de sensores possuem pelo menos um nó *gateway*, que permite a ligação com o exterior. Dado que se o *gateway* for comprometido, facilmente toda a rede se torna inutilizável, é normal assumir que este é um ponto é de confiança [4]. Ou seja, é assumido que estes se comportam de modo correto e que são confiáveis.

2.1.3 Requisitos de segurança

De seguida são identificadas as propriedades de segurança desejáveis para uma rede de sensores:

- Autenticidade. Esta propriedade permite que um nó confie na identidade de outro nó com quem comunica. Uma mensagem recebida é autêntica se foi realmente originada e enviada pela entidade referida. Quando esta propriedade não é garantida um adversário pode personificar qualquer nó da rede, criar mensagens falsas e modificar mensagens legítimas de outros nós que participem na rede.

- Integridade. A propriedade de integridade assegura que dados são entregues no destino tal como foram enviados inicialmente pela origem. A alteração inválida de uma mensagem pode ser originada por falhas na comunicação que originam erros, ou realizadas de modo malicioso por um atacante que tenta alterar o conteúdo de uma mensagem.

- Confidencialidade. É um requisito básico em qualquer sistema de comunicação pois assegura que dados são apenas acedidos e vistos por membros autorizados para tal. A confidencialidade é normalmente obtida com o uso de criptografia. Um esquema de criptografia deve não só impedir a descoberta do conteúdo de cada mensagem, mas também impedir que os adversários consigam descobrir algum tipo de informação sobre as mesmas. Para tal um esquema de criptografia deve garantir a propriedade de segurança semântica. Esta propriedade assegura, por exemplo, que um atacante não obtém qualquer informação sobre o conteúdo de uma mensagem, mesmo que observe várias vezes o resultado da cifra do mesmo texto [6].

- Disponibilidade. A disponibilidade é a propriedade que garante a capacidade de sobrevivência e correto funcionamento dos serviços da rede, mesmo sob a existência de vários ataques.

As especificidades que as redes de sensores sem fios apresentam, são desafios que devem ser ultrapassados de modo a garantir as propriedades de segurança desejadas [7]:

- A ligação sem fios faz com que estas sejam facilmente alvo de ataques passivos e ativos. Os ataques passivos podem comprometer a confidencialidade da comunicação, enquanto que os ataques ativos podem resultar em violações da disponibilidade, integridade e autenticação da comunicação.

- É possível um nó ser comprometido dada a falta de segurança física que este tipo de redes usualmente possui. Como concluído em [5] é possível ler a memória *ROM* e *RAM* dos dispositivos normalmente usados em redes de sensores sem fios em menos de 1 minuto. Esse estudo prova que se consegue obter o código *assembly* presente nos sensores, e refere que quando são usados sistemas operativos ou protocolos conhecidos é mais fácil localizar em memória os dados mais relevantes, como é o caso das chaves criptográficas.

- Outro desafio importante advém da possível mobilidade dos nós, que podem alterar a sua posição dentro de uma rede, alterando a topologia da mesma, ou alternar entre diferentes redes.

- Por último as já referidas limitadas capacidades dos sensores tornam impossível a utilização de algumas tradicionais técnicas de segurança, sendo por isso também um desafio a ultrapassar.

2.1.4 Principais ataques

Nesta secção são apresentados os principais ataques a que as redes de sensores sem fios e em geral as redes em malha estão sujeitas. É feita uma breve descrição dos diferentes tipos de ataques, que foram identificados em várias fontes [7], [8],[9],[10],[11],[12].

Ataque à informação em trânsito

Um atacante que disponha de meios adequados pode recolher informação que circule na rede, desde que a mesma não esteja devidamente protegida. Se a comunicação entre sensores, ou entre sensores e o *gateway*, for feita sem nenhum tipo de codificação, então um atacante com um recetor apropriado consegue captar facilmente o fluxo de dados. Um atacante conhecendo o tipo de comunicação existente pode, por exemplo, introduzir falsas mensagens na rede tentado personificar outro nó. Este tipo de ataques pode quebrar a autenticidade e integridade das mensagens trocadas na rede.

Um atacante pode ainda repetir mensagens legítimas enviadas por nós legítimos no passado executando assim um *replay attack*.

Sybil attack

Em vários casos nas redes de sensores é necessário o trabalho de vários sensores em conjunto para a realização de determinada tarefa, por exemplo agregação de vários sinais detetados por vários sensores de modo a determinar um estado do meio.

Neste tipo de situações um nó atacante pode tentar fazer-se passar por vários nós usando várias identidades falsas, de modo a alterar o resultado final da agregação dos sinais detetados. Este tipo de ataque em que um nó tenta a personificação de vários nós em simultâneo é conhecido como o *Sybil attack* [13].

Selective forwarding/Blackhole

As redes *multi-hop* assumem que os nós intermédios que participam na rede irão de um modo justo encaminhar a totalidade das mensagens que recebem, o que nem sempre é verdade. O ataque de encaminhamento seletivo (*selective forwarding*) consiste num nó malicioso não realizar tal função para uma parte do tráfego que recebe, ou seja, de um modo seletivo não encaminha parte das mensagens. Um nó intermédio atacante pode também bloquear a totalidade do tráfego que recebe, realizando assim um *blackhole attack*.

Realizando um *blackhole attack* ao fim de algum tempo os seus nós vizinhos facilmente irão suspeitar da realização de tal ataque e irão tentar encaminhar o tráfego por outro nó. Pelo contrário, realizando um ataque de encaminhamento seletivo *selective forwarding*, o nó malicioso levanta menos suspeitas sobre si, pois o número de pacotes perdidos pode ser assumido pelos nós emissores como causa de colisões no meio, erros de transmissão ou perdas devido a sobrecarga na rede.

Sinkhole attack

O objetivo de um ataque de *sinkhole* é atrair para um nó a totalidade do tráfego de uma dada área da rede. Este ataque corresponde normalmente a um nó interno à rede, por exemplo um nó comprometido, fazer-se parecer especialmente atrativo para a realização de encaminhamento de mensagens para o *gateway*. O ataque pode ser efetuado falsificando as mensagens do protocolo de *routing*, por exemplo, simulando que o nó se encontra a pouca distância do *gateway*.

A realização deste tipo de ataque torna mais efetiva a posterior realização de um ataque de *selective forwarding* ou *blackhole*.

Wormhole attack

O *Wormhole attack* [4] consiste em um nó atacante receber uma mensagem em dada zona da rede e repetir a mesma mensagem noutra zona distante. Normalmente este tipo de ataque envolve dois nós maliciosos, distantes, que partilham entre ambos uma ligação externa à rede.

Por exemplo como ilustrado na figura 2.1, um nó atacante A que se encontra perto do *gateway*, representado na figura como retângulo vermelho, ao receber uma mensagem do mesmo, encaminha a mensagem, usando uma ligação externa, para outro nó malicioso B. Nó B ao receber a mensagem

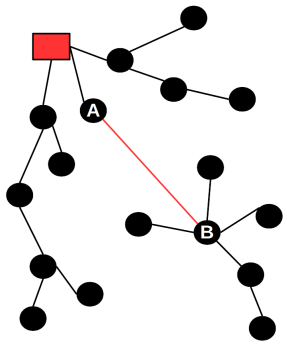

Figura 2.1: Exemplo de um *wormhole attack*.

repete o seu envio para os seus vizinhos. Tal ataque neste caso culmina numa espécie de ataque de negação de serviço, pois os nós vizinhos de B irão assumir que se situam a apenas um salto para o *gateway*, o que não acontece. Estes nós vizinhos irão enviar mensagens diretamente para o *gateway*, mas estas serão perdidas pois estes encontram-se fora do alcance do *gateway*.

Esta técnica permite ainda que atacante efetue outro tipo de ataques. Se o nó B encaminhar as mensagens para o *gateway*, os seus vizinhos irão acreditar que se encontram mesmo perto do *gateway* e assim o nó B terá grande parte da comunicação da rede na sua posse. Situação perfeita para efetuar ataques à informação em trânsito ou encaminhamento seletivo.

Rushing attack

Este ataque apresentado em [14], é um ataque de negação de serviço efetivo em protocolos de *routing on-demand*. Este tipo de protocolos, não mantém proactivamente o estado das ligações com os nós vizinhos, nem troca qualquer tipo de mensagens de *routing*, quando um nó necessita de enviar dados para outro nó, realiza previamente uma fase de descoberta do mesmo. Nesta fase de descoberta, o nó de origem propaga mensagens de descoberta (*route request*), que são difundidas pela rede pelos restantes nós. Ao receber esta mensagem o nó de destino responde com uma mensagem a confirmar a possibilidade de conectividade. Durante este processo, de modo a limitar as mensagens transmitidas na rede e o consumo de energia, cada nó intermédio em geral apenas encaminha a primeira mensagem de descoberta recebida para um dado pedido.

O *rushing attack* tira partido desta particularidade. Este ataque consiste em um atacante fazer com que as restantes mensagens de pedidos de descoberta de encaminhamento sejam bloqueadas, acelerando as mensagens reencaminhadas pelo mesmo. Um atacante que acelere os pedidos de descoberta que passam pelo próprio faz com que estes sejam os primeiros a chegar aos nós vizinhos do nó destino. O que faz com que os nós vizinhos do nó destino encaminhem apenas o pedido recebido pelo atacante e bloqueiem os restantes.

Com a realização deste ataque o atacante consegue impedir que o nó que originou o pedido descubra o nó que pretende. Para tal bloqueia as mensagens de confirmação de descoberta, que têm o trajeto inverso, impedindo assim a troca de informação entre os dois nós. Por outro lado, o atacante

pode optar por deixar terminar o processo de descoberta de rota, e garantir assim que faz parte do trajeto da comunicação futura entre os dois nós. Esta última possibilidade permite que o atacante realize posteriormente outro tipo de ataques, por exemplo o ataque de encaminhamento seletivo, ou alteração do conteúdo das mensagens trocadas.

Para a concretização deste ataque um atacante necessita, portanto, de fazer com que as mensagens de descoberta encaminhadas pelo próprio sejam as primeiras a chegar ao nó destino. Para tal um atacante pode saltar os atrasos introduzidos pela camada de acesso ao meio, ou por exemplo aumentar a potência de transmissão reduzindo assim o número de saltos percorridos até ao nó destino. Mesmo que o protocolo de *routing* usado não encaminhe apenas a primeira mensagem de descoberta, a realização deste ataque faz com que o atacante tenha uma grande probabilidade de se intrometer no caminho que irá ser estabelecido.

Este ataque pode ser realizado ainda de um modo mais efetivo através de um *wormhole attack*. Com a junção destes dois ataques os pedidos enviados pela ligação externa partilhada por dois atacantes, facilmente chegam primeiro a outras partes da rede.

Hello flood attack

Muitos protocolos exigem que os nós da rede enviem mensagens *HELLO* para se anunciarem aos seus vizinhos, pelo que um nó ao receber essa mensagem pode assumir que está ao alcance do nó que enviou essa mensagem.

Esta suposição como apresentado em [4] pode ser falsa. Um atacante que possua um dispositivo com um alcance de comunicação superior aos dispositivos usados na rede (*laptop attack*), pode convencer vários sensores da rede de que é seu vizinho e está ao seu alcance. Por exemplo, um adversário anunciando que está a um número reduzido de saltos para o *gateway* irá enganar vários nós da rede, pois estes irão tentar encaminhar pelo mesmo as mensagens com destino ao *gateway*. Como o nó atacante pode estar a uma grande distância, dado possuir um maior alcance de comunicação, as mensagens enviadas para o mesmo serão perdidas. Sendo a rede deixada num estado de confusão total, pois vários outros nós irão tentar também encaminhar as mensagens pelo atacante.

Jamming

Na camada física um atacante pode atacar a disponibilidade do sistema executando um ataque de *jamming*. Este ataque consiste em um atacante transmitir ruído de modo a impossibilitar a comunicação entre os elementos de uma rede. Ataque pode ser realizado de uma forma abrupta, criando interferências constantes que impeçam a total comunicação dos vizinhos na área de alcance, ou de forma seletiva, por exemplo, transmitindo apenas ruído quando é detetado que determinado nó está a transmitir uma mensagem.

Negação de serviço

O conceito de negação de serviço define um conjunto de ataques que tentam evitar que a rede tenha capacidade de fornecer ou aceder a determinado serviço. É um tipo de ataque que nas redes de sensores sem fios e nas redes em malha pode ocorrer em diferentes camadas. Na camada física pode ocorrer executando um ataque da *jamming*, na camada de acesso ao meio pode ser obtido pelo desrespeito das regras de acesso ao meio, por exemplo, enviando mensagens apenas com o intuito de gerar colisões com mensagens enviadas por outros sensores. Na camada de rede pode ser obtido realizando *blackhole attack*, de tal forma que impeça parte da rede de comunicar com o *gateway*.

Exaustão da bateria

Este ataque é realizado quando um atacante consegue fazer com que um sensor realize trabalho computacional pesado que seja desnecessário. Ataque pode ser efetuado, por exemplo, criando tráfego falso e obrigando o nó a realizar a verificação da autenticidade das mensagens, repetindo mensagens anteriores, que a rede terá de encaminhar até ao *gateway*, entre outros.

O tempo de vida da bateria dos sensores é um ponto critico nas redes de sensores, por isso este tipo de ataque é uma ameaça que pode ser bastante eficiente.

2.2 Mecanismos de segurança

Nesta secção são apresentados os principais mecanismos de segurança propostos para as redes de sensores sem fios, que podem também ser aplicados em redes em malha.

2.2.1 Comunicação segura

Nesta subsecção são identificados protocolos que pretendem garantir a propriedades de segurança básicas ao nível da comunicação entre os nós de uma rede. Estes protocolos não abordam o problema da gestão de chaves e assumem a existência de chaves simétricas pré-partilhadas entre os nós. Os protocolos apresentados nesta subsecção foram desenhados com o objetivo de serem implementados sobre o *TinyOs*, um sistema operativo desenhado para redes de sensores.

SNEP

O SNEP [15], é um protocolo que tem como objetivos garantir a confidencialidade, integridade e autenticidade da comunicação mantida entre os elementos de uma rede de sensores, tentando em simultâneo fazer com que o *overhead* adicionado pela solução de segurança seja baixo. Este protocolo oferece ainda proteção contra o envio de mensagens repetidas.

A confidencialidade é obtida pelo protocolo através da cifra do conteúdo de cada mensagem enviada. De modo a obter a autenticação e garantir a integridade das mensagens trocadas entre os nós, a

chave simétrica partilhada entre dois nós é usada para calcular um *message authentication code* (MAC) para cada mensagem enviada.

Com o objetivo de garantir a proteção contra o envio de mensagens repetidas é incluído no MAC, de 8 bytes, um contador (C). Esta solução permite que o recetor tenha a garantia de que os dados enviados são recentes, pois após uma mensagem ter sido verificada, o recetor sabe que a mensagem foi enviada posteriormente à ultima mensagem recebida para o mesmo sensor, pois esta possuía um valor de contador inferior.

Como é boa prática, este protocolo não utiliza a mesma chave para diferentes primitivas criptográficas.

Por isso a partir da chave mestra X_{AB}, partilhada por dois nós A e B, são derivadas outras chaves. Chaves para cifra, K_{AB} e K_{BA}, uma para cada sentido de comunicação, e chaves usadas no calculo dos MAC, $K^{`}_{AB}$ e $K^{`}_{BA}$, uma vez mais uma para cada sentido de comunicação.

Os dados de uma mensagem (M), enviada a partir do nó A para B são cifrados da seguinte forma:

$$M = \{D\}_{(K_{AB},C_{AB})}, MAC(K^{'}_{AB}, C_{AB}||\{D\}_{(K_{AB},C_{AB})})$$

Ou seja, os dados D são cifrados usando chave de cifra partilhada pelos nós para o sentido especifico (K_{AB}), e usando também o contador de A (C_{AB}). O MAC associado a cada mensagem é calculado sobre o resultado da cifra dos dados, usando $K^{`}_{AB}$, o contador de A (C_{AB}) e o conteúdo da mensagem resultante da cifra.

De referir que protocolo obtém a propriedade de *segurança semântica*, pois é usado um contador que garante que duas mensagens com o mesmo conteúdo irão originar criptogramas diferentes. É assumindo pelos autores que o tamanho para o contador é suficientemente longo de modo a que este não se repita durante a vida útil do nó.

Um dos pontos fracos deste protocolo é a necessidade de cada sensor ter de manter o estado de um contador para cada sensor com o qual queira comunicar. Para além de que, caso uma ou mais mensagens sejam perdidas na rede, será perdida também a sincronização dos contadores entre os nós, sendo necessário sincronizar novamente os mesmos. De modo a sincronizar novamente os contadores pode-se repor os valores iniciais em ambos os nós da rede, ou por exemplo caso A suspeite de que o seu contador local C_{BA} não se encontra sincronizado com o do nó B, A pode pedir a B que lhe envie o seu contador C_{BA}.

TinySec

O TinySec [6], apresenta uma arquitetura de segurança da camada de ligação para redes de sensores sem fios. Tal como no SNEP o mecanismo de distribuição de chaves foi deixado de fora, pelo que é assumido que nós que pretendem comunicar já partilham uma chave simétrica entre ambos.

O protocolo oferece dois modos de operação, um garantido a autenticação e confidencialidade (*TinySec-AE*) e outro em que apenas a autenticação é garantida (*TinySec-Auth*).

O TinySec utiliza o modo de cifra por blocos *cipher-block chaining* (CBC) para garantir a confidencialidade, e de modo a obter a propriedade de *segurança semântica* é utilizado um *initialization vector*

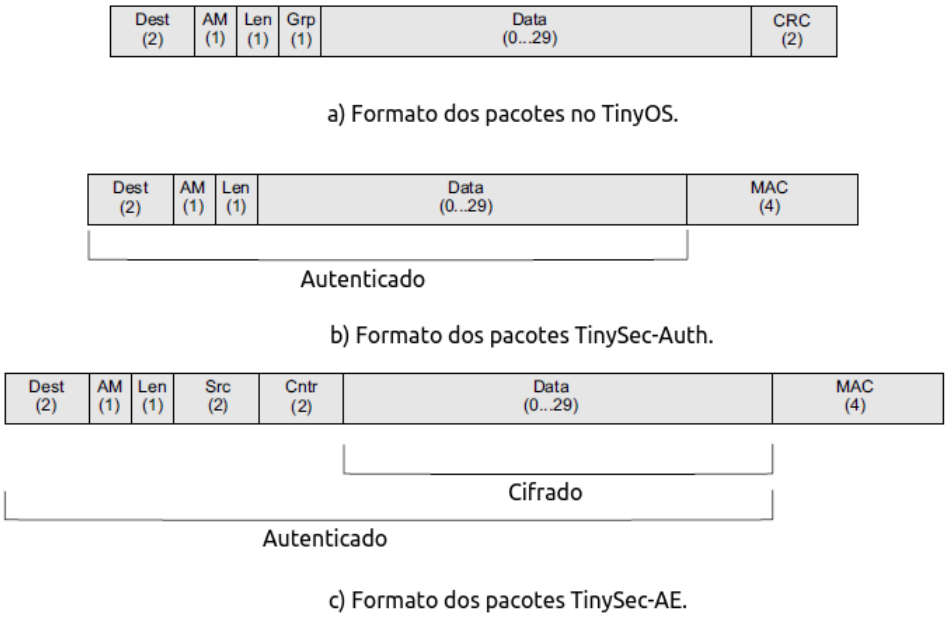

Figura 2.2: Comparação entre o formato original dos pacotes de dados usados no *TinyOs* e os formatos propostos pelos dois métodos apresentados no TinySec, [16].

(IV) que depende de um contador local entre outros elementos. TinySec ao contrário do SNEP não utiliza estes contadores para detetar mensagens repetidas, os seus autores sugerem que a proteção contra o ataque de repetição de mensagens não deve ser feita na camada de ligação.

Tal como o SNEP de modo a garantir a autenticidade e integridade, a cada mensagem será adicionado um *message authentication code* (MAC), calculado sobre o conteúdo cifrado e cabeçalho da mensagem, usando a chave partilhada entre os dois nós. Neste caso usando o modo CBC-MAC gerando um MAC de 4 bytes.

A figura 2.2 mostra os formatos das mensagens para os métodos *TinySec-AE* e *TinySec-Auth*. Ambos os métodos propostos optam por retirar o campo *Grp* usado no *TinyOs* para separar diferentes redes de sensores. Tal opção é possível pois usando o TinySec a separação entre diferentes redes será obtida se cada rede usar diferentes chaves de comunicação. No método TinySec-AE é adicionado o campo *Src* que identifica o nó emissor, este campo foi adicionado com o intuito de oferecer mais entropia aos IV usados, dado que sem este campo todos os nós ao iniciarem o processo de comunicação com o *gateway* iriam usar o mesmo IV.

De modo a não incrementar em demasia o tamanho do pacote final o TinySec utiliza um IV de 8 bytes que é composto por todos os campos de cabeçalho na figura 2.2, ou seja, $IV = dst||AM||l||src||ctr$, sendo *ctr* um contador de 16 bits, e *dst*, *AM* e *l* campos do cabeçalho dos pacotes do *TinyOs*.

Os testes realizados usando o protocolo mostram que o mesmo funciona adicionando menos de 10% de acréscimo de energia, latência e largura de banda, quando comparando com o *TinyOs*.

MiniSec-U

Desenhado posteriormente ao TinySec e SNEP, o MiniSec-U [17] tem como objetivo garantir a confidencialidade, integridade e autenticação da comunicação trocada entre dois nós, e obter proteção contra o envio de mensagem repetidas. Protocolo propõem-se a apresentar uma solução mais eficiente em termos energéticos, melhorando os resultados apresentados pelos seus antecessores.

Como já referido no protocolo SNEP cada nó mantém contadores locais de modo a detetar e mensagens repetidas. Com o objetivo de reduzir o consumo de energia, no SNEP o contador usado em cada mensagem não é enviado juntamente com a mesma. Pelo que em caso de perca de sincronismo é necessário recorrer a um protocolo para repor o mesmo. Por outro lado, no TinySec o contador usado para a cifra de cada mensagem é enviado com a mesma incrementando o tamanho de cada mensagem em 2 bytes.

O protocolo MiniSec-U tenta encontrar uma solução intermédia combinando o melhor das duas soluções. Em cada nó é mantido um contador sequencial para cada sentido da comunicação entre dois nós, por exemplo, o contador C_{AB} para comunicação do nó A para o nó B é usado como IV na cifra das mensagens enviadas. De modo a tolerar 2^x perdas de pacotes seguidas, são associados a cada mensagem os últimos x bits do contador local do emissor, A neste caso.

Ao receber cada mensagem B compara os últimos x bits recebidos na mensagem com o valor do seu contador local C_{AB}, altera o valor do seu contador substituindo no valor atual os últimos x bits recebidos e de seguida tenta decifrar a mensagem. Os autores apontam como sendo razoável enviar em cada mensagem apenas os últimos 3 bits do contador. Assim sendo é sugerido que os mesmos sejam enviados, no pacote do *TinyOs*, nos primeiros 3 bits do campo *lenght*. Como o campo de dados no *TinyOs* é limitado a 30 bytes, os 3 primeiros bits do campo *lenght* não serão necessários.

Por exemplo, com x igual a 3, supondo que cada nó tem o seu contador local $C_{AB} = 0$ e que A envia seis mensagens seguidas para B, mas apenas a sexta, enviada com $C_{AB} = 5$, é recebida. Ao receber esta mensagem, B imediatamente incrementa C_{AB} para 5 e com sucesso decifra a mensagem.

Outra diferença entre este protocolo e os anteriores é o facto de ser usado o modo *offset CodeBook* (OCB), modo de cifra por blocos que garante simultaneamente autenticação e confidencialidade. Os autores optaram por este modo pois usando o mesmo com apenas uma passagem sobre a mensagem o texto é cifrado e é gerado o MAC que será associado a cada mensagem. Este método é mais eficiente do que o utilizado, por exemplo, no TinySec, onde com o modo CBC é cifrada a mensagem e com o método CBC-MAC é posteriormente calculado o respectivo MAC.

Comparando os mecanismos apresentados apenas pelo incremento adicionado a cada mensagem, o MiniSec é aquele que menos informação adiciona às mensagens. Em simultâneo este protocolo diminui o número de vezes em que será necessário recorrer ao protocolo de para sincronizar os contadores locais.

Para usar o MiniSec é necessário que os dois nós partilhem duas chaves K_{AB} e K_{BA}, uma para cada sentido da comunicação. Cada nó terá ainda que possuir um contador de 32 bit para cada sentido da comunicação entre dois nós, por exemplo, contador C_{AB} usado com a chave K_{AB} para cifra das mensagens de A para B.

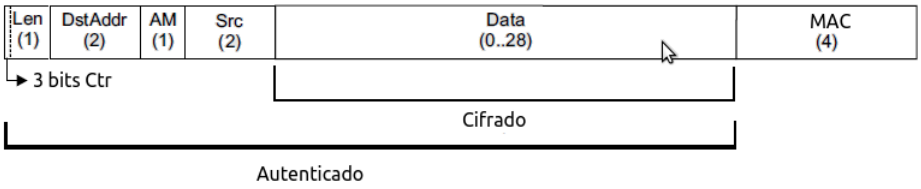

Figura 2.3: *Formato de um pacote MiniSec-U.*

2.2.2 Autenticação de mensagens de *broadcast*

Um modo de comunicação muito usado em redes moveis como em geral nas redes sem fios é o envio de mensagens em *broadcast*. O uso de *broadcast* permite poupar o envio de mensagens de *unicast*, pois com o envio de uma mensagem apenas a informação é difundida para todos os nós vizinhos.

Neste caso de modo a garantir a autenticação de cada mensagem enviada não é razoável usar criptografia simétrica por si só. Por exemplo tendo uma chave simétrica partilhada por todos os nós da rede não é possível garantir esta propriedade, dado que todos os elementos têm conhecimento da chave usada, e por isso capacidade para personificar outros.

Este problema é usualmente ultrapassado com o uso de criptografia assimétrica, mas dadas as limitadas capacidades dos dispositivos usados em redes de sensores, o uso de tais técnicas teria elevados custos computacionais e energéticos. Pelo que nesta secção são apresentados dois protocolos, desenhados para resolver este problema.

μTESLA

O protocolo μTESLA [15], é uma versão mais leve do protocolo TESLA, que foi desenhado para redes computadores e não para ambientes com capacidades computacionais limitadas como as verificadas nas redes de sensores.

O protocolo μTESLA requer que cada nó vizinho partilhe uma chave secreta com o nó que irá realizar o *broadcast*, e que os nós estejam sincronizados.

De modo a simplificar a explicação do protocolo será tomado como exemplo a comunicação entre o nó *gateway* e os seus nós vizinhos.

Inicialmente *gateway* cria uma cadeia de chaves unidirecional (*one-way hash*) de tamanho n, originada a partir de um número aleatório K_{n+1} usando uma função de *hash* unidirecional F, conhecida por todos os elementos.

Como F é uma função *hash* unidirecional, todos os nós poderão calcular $(K0, ..., Kn)$ se conhecerem K_{n+1} mas não o inverso. Toda a cadeia é criada dependendo de um único valor, K_{n+1} neste caso, e apenas é possível calcular os valores da mesma num sentido. A partir do valor inicial K_{n+1} são calculados os restantes elementos da cadeia, da seguinte forma:

$$(F(K_{n+1}) = K_n, F(K_n) = K_{n-1}, ..., F(K_1) = K_0)$$

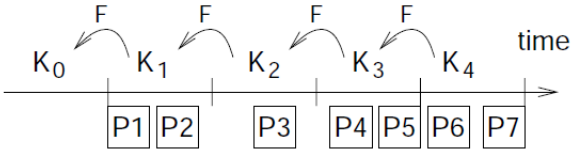

Figura 2.4: Exemplo do funcionamento do protocolo μTESLA, [15].

Sendo em cada passo impossível saber qual a chave aplicada na função F para calcular um dado valor.

O valor K_0 resultado final da cadeia gerado pelo *gateway* é enviado para cada nó de forma autenticada e garantindo a integridade, usando a chaves secretas que o *gateway* partilha com cada nó vizinho.

Cada mensagem posteriormente enviada pelo *gateway* têm associado um *message authentication code* (MAC), de modo a garantir a sua autenticação e integridade. Como ilustrado na figura 2.4 o tempo é dividido em períodos uniformes e o *gateway* associa para cada período uma chave da cadeia. Ou seja, o *MAC* das mensagens enviadas, por exemplo, no período i será calculado usando a chave K_i. Durante cada período cada nó vizinho ao receber as mensagens guarda as mesmas em *cache*, pois ainda não tem conhecimento da chave a usar para efetuar a verificação.

Depois de finalizado cada período, o *gateway* anuncia a chave usada, por exemplo, K_2 para o período 2. Ao receberem a chave cada nó primeiro confirma que a mesma pertence à cadeia inicialmente gerada pelo *gateway*, verificando neste caso se $F(F(K_2)) = K_0$. De seguida cada nó autentica cada mensagem recebida durante o período respetivo, no exemplo da figura 5 verificação da mensagem três (P3) seria feita usando K_2.

O protocolo μTESLA oferece um modo eficiente de autenticação para o envio de mensagens em *broadcast*, sendo baseado em criptografia simétrica. Possui algumas desvantagens, como por exemplo, o facto de os nós terem de guardar as mensagens em memória pois a autenticação das mesmas ser atrasada no tempo, este aspeto é relevante dada a limitada a memória dos dispositivos normalmente usados em redes de sensores. Protocolo necessita ainda que, todos os elementos estejam temporalmente sincronizados, de modo a dividirem de semelhante forma os períodos de tempo.

ARMS

O protocolo ARMS [18], desenhado posteriormente ao protocolo μTESLA, tem como objetivo tornar processo de envio de mensagens autenticadas em *broadcast* mais eficiente.

Tal como o μTESLA este protocolo assume a existência de chaves secretas pré-partilhadas entre cada par de nós e baseia-se no uso de cadeias de *hash* unidirecionais.

No esquema básico deste protocolo o emissor cria inicialmente uma cadeia de *hash* unidirecional com apenas duas chaves. Para tal este gera uma chave aleatória K_1, e calcula $F(K_1)$ usando F, uma função de *hash* unidirecional partilhada por todos os elementos. Posteriormente envia o resultado de $F(K_1)$ ($F(K_1) = K_0$) de modo autenticado para cada vizinho, usando a chave secreta partilhada com

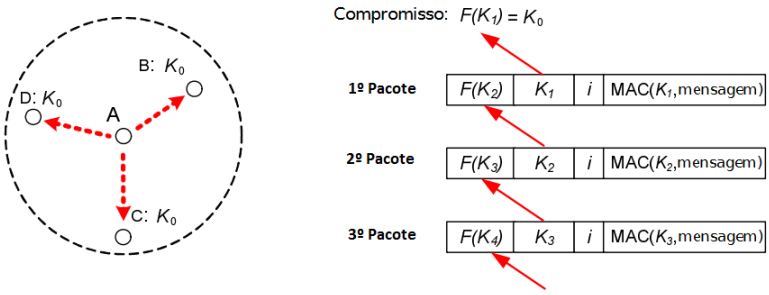

Figura 2.5: Funcionamento do protocolo ARMS, [18].

cada um.

As subsequentes mensagens enviadas em *broadcast*, usando este protocolo, têm a forma ilustrada na figura 2.5. Após o envio de modo autenticado para cada nó vizinho de K_0, cada mensagem seguinte contêm um campo usado como desafio, de modo a autenticar a mensagem seguinte. Neste caso a chave $F(K_2)$ recebida em P1, é o desafio usado para autenticar a mensagem P2

Como se verifica na figura 2.5 a primeira mensagem enviada em *broadcast* tem a seguinte constituição:

$$P1 = F(K_2)|K_1|i|MAC(K_1, menssagem)$$

Sendo $F(K_2)$ o desafio usado para autenticar a mensagem seguinte, que deverá possuir a chave que aplicada à função F iguala o valor de $F(K_2)$, K_1 a chave usada para calcular o MAC da presente mensagem e i o conteúdo da mensagem. Ao receber esta mensagem o nó verifica a autenticidade, pela igualdade $F(K_1) = K_0$ e guarda $F(K_2)$ para verificar a autenticidade da seguinte mensagem de *broadcast* recebida.

Ou seja, para o continuo envio de mensagens autenticadas em *broadcast*, no esquema básico, por cada nova mensagem nó emissor gera uma nova cadeia de chaves e envia a nova chave final da cadeia na mensagem. De modo a comprovar a autenticidade a mensagem seguinte enviada em *broadcast* nó emissor terá de indicar uma chave que aplicada à função F resulte na chave final da cadeia enviada anteriormente.

Com este modo caso um nó não receba uma mensagem não conseguirá autenticar mais nenhuma mensagem recebida, dado que o último desafio que este possui apenas poderia ser verificado pela mensagem que ele não recebeu. De modo a evitar tal desvantagem o tamanho da cadeia de *hash* usada pode ser aumentada, para que seja possível tolerar a perda várias mensagens seguidas.

Em comparação com o μTESLA este método, não requer sincronização temporal e utiliza muito menos memória em cada nó, por outro lado requer a constante criação de um número aleatório de modo a criar uma nova cadeia.

2.2.3 Gestão de chaves

Dadas a limitações dos dispositivos usualmente usados em redes de sensores sem fios a criptografia simétrica é dominante neste tipo de redes. Sendo a distribuição de chaves normalmente realizada através da pré-distribuição das mesmas nos sensores ou através do estabelecimento pró-ativo, ou seja, após a sua disposição na rede.

A maioria das soluções apresentadas para gestão de chaves em redes de sensores, concentram-se basicamente no processo de distribuição de chaves, mas existem outros pontos a ter em conta.

Um desses pontos é a possibilidade de revogação de chaves, por exemplo quando um sensor é comprometido será essencial revogar as chaves associadas ao mesmo. Outro ponto importante é a necessidade de redistribuir chaves, pois cada chave deve ter associada um tempo de vida, e após esse tempo ter expirado é necessário que todos os nós que partilhem a chave revoguem a chave anterior e criem uma nova chave. Este método reduz a possibilidade de um atacante passivo descobrir a chave usada através da escuta da troca de mensagens, pois a mesma chave será usada apenas durante um determinado período de tempo.

Um processo de pré-distribuição de chaves consiste em antes da disposição de cada nó na rede instalar chaves, que irão permitir uma posterior comunicação segura entre os vários nós. O método mais simples consiste na instalação de uma única chave em todos os nós da rede, mas esta solução é inadequada quando se assume que os nós podem ser comprometidos, pois a captura de um único nó da rede compromete a sua totalidade.

A pré-distribuição pode também ser feita instalando inicialmente em cada nó chaves simétricas partilhadas com cada um dos nós da rede, ou seja, para uma rede de n nós cada nó teria de guardar $n-1$ chaves para uma possível posterior comunicação com todos os nós da rede. Esta solução apresenta o espeto negativo de consumir muita memória desnecessariamente, pois nas redes de sensores a comunicação é feita normalmente apenas com os nós vizinhos e com o *gateway* da rede. De modo a reduzir a memória requerida para a pré-distribuição de chaves é por vezes usada a informação sobre a futura localização na rede do nó, por exemplo de modo a diminuir o número de chaves que serão instaladas.

Em contraste com a pré-distribuição de chaves, a distribuição pró-ativa permite a geração dinâmica das chaves após a disposição dos nós na rede, antes de existir troca de dados.

LEAP+

Um exemplo de um sistema que utiliza o método de distribuição pró-ativa é o protocolo LEAP+ (localized encryption and authentication protocol) [11]. Este protocolo tenta resolver o problema de em redes de sensores existirem diferentes fluxos de comunicação com diferentes requisitos de segurança. Para tal usa quatro chaves diferentes para diferentes fluxos de comunicação, que usam mecanismos distintos como: comunicação de cada nó com o *gateway* da rede; comunicação de um nó com outro nó da rede; comunicação em grupo local (vizinhança) e comunicação em grupo global, com todos os nós da rede.

No LEAP+ é assumido que sensores são estáticos e que podem ser comprometidos, mas apenas

ao fim de um dado tempo t após a sua disposição no ambiente. Pelo que durante este período de tempo t cada novo nó tem conhecimento da chave mestra do sistema, sendo capaz de se autenticar perante os seus vizinhos e estabelecer chaves simétricas usadas para posterior comunicação. Ao fim do tempo t o novo nó elimina a chave mestra impedindo que a rede seja totalmente comprometida. Após este período de tempo, é assumido que os nós permanecem estáticos e apenas poderão estabelecer novas ligações com novos nós na rede, que possuam a chave mestra.

A grande desvantagem do LEAP+ é o facto ser assumido que os nós da rede são estáticos, o que limita a possibilidade de utilização do protocolo.

Gestão probabilística

Em [19] é apresentada outra técnica de distribuição pró-ativa, sendo proposto um protocolo probabilístico para gestão básica de chaves. Este protocolo para além da distribuição de chaves aborda também a sua revogação e redistribuição.

A distribuição de chaves realizada em [19] é constituída por três fases que são a pré-distribuição, descoberta de chaves partilhadas e estabelecimento de chaves de percurso, sendo as últimas fases usadas para estabelecer uma ligação segura entre dois nós. Na fase de pré-distribuição, ou seja, antes da disposição dos nós na rede, são realizados os seguintes passos:

- É gerado um grande conjunto de P chaves de criptográficas, denominado de *key pool*, e os seus correspondentes identificadores;
- Recolha de modo aleatório de K chaves da *key pool*, criando o chamado *anel de chaves* para cada nó do sistema;
- O *anel de chaves* de cada nó é guardado em cada sensor correspondente;
- Identificadores das chaves do *anel de chaves* de cada nó são associadas ao respectivo sensor e são guardadas num nó de controlo que se assume seguro;
- Em cada sensor é carregado o identificador do nó de controlo correspondente e a chave partilhada entre ele e o sensor.

Na fase de pré-distribuição, K chaves são carregadas em cada sensor, é assegurado que cada par de sensores irá partilhar pelo menos uma chave, com um certo grau de probabilidade, sendo K um número inferior ao número de nós ativos da rede e menor do que P.

Na fase da descoberta das chaves partilhadas cada nó começa por conhecer inicialmente os seus nós vizinhos com os quais partilha pelo menos uma chave do *anel de chaves*. Para tal cada nó difunde a lista de identificadores das chaves do seu *anel de chaves*.

A descoberta das chaves partilhadas é realizada durante a fase de inicialização da rede sendo estabelecida a topologia e rotas encaminhamento da mesma. Se dois nós partilham uma chave então poderá existir uma ligação com comunicação segura entre ambos, utilizando a chave simétrica partilhada e, por exemplo, um dos mecanismos apresentados em 3.2.1.

Protocolo proposto refere ainda como pode ser feito o encaminhamento de modo a possibilitar a comunicação segura entre dois nós vizinhos que não partilham chaves. Neste caso encaminhamento é realizado utilizando ligações estabelecidas entre nós intermédios.

A revogação de chaves permite a eliminação da rede de nós tenham passado a ser considerados não seguros. Para executar tal processo um nó controlador difunde de modo autenticado pela rede uma mensagem que contêm a lista dos K identificadores das chaves do *anel de chaves* do nó em questão. Ao receber está mensagem cada nó verifica se algum identificador corresponde a alguma chave do seu *anel de chaves*, caso tal aconteça o nó elimina as mesmas do *anel de chaves*, podendo assim ser perdida a ligação com algum nó vizinho.

A grande desvantagem deste protocolo é a existência de falsos negativos, ou seja, o não estabelecimento de chaves para comunicação entre dois nós legítimos da rede, por não partilharem chaves em comum e onde também não é possível a comunicação utilizando ligações intermédias.

Criptografia assimétrica em redes de sensores

Como é conhecido umas das técnicas mais usadas para oferecer o suporte à gestão das chaves simétricas, em redes de computadores, é a criptografia assimétrica. No entanto todos os mecanismos apresentados anteriormente, nas subsecções 2.2.1 e 2.2.2, partiam do pressuposto que tais técnicas não podem ser usadas de um modo eficiente em redes de sensores em malha. Este pressuposto é justificado pelas limitações dos dispositivos usados, nomeadamente em termos energéticos, e pelos elevados custos associados às técnicas normalmente usadas por estes métodos. Por isso protocolos apresentados baseiam-se muitas vezes em chaves simétricas pré-partilhadas entre os nós da rede. Nos anos mais recentes tem surgidos vários estudos que mostram que tal pressuposto pode não ser assim tão verdadeiro.

Em [20] é mostrado que a criptografia de chave pública é computacionalmente realizável em processadores de 8-bits, embora os testes realizados mostrem que os custos energéticos associados são elevados, sendo apenas sugerido o seu uso para distribuição de chaves. Resultados são também confirmados por posteriores estudos como [21] e [22].

Em [23] é realizada uma análise comparativa entre duas técnicas de criptografia assimétrica RSA e ECC (*elliptic curve cryptography*). Os testes efectuados, tal como sugerido anteriormente em [20] e [22], mostram que a técnica de ECC apresenta melhores resultados ao nível do custo computacional. Mas a grande vantagem desta técnica é o reduzido tamanho das chaves usadas quando comparadas, para o mesmo nível de segurança, com uma chave RSA. Por exemplo para obter o mesmo nível de segurança de uma chave RSA de 1024 bits será necessária uma chave ECC de apenas 160 bits [23].

Tal resulta numa menor largura de banda usada, por exemplo, para autenticação mútua, dado os certificados trocados possuírem um tamanho inferior. Este é um ponto essencial pois como também foi provado em [23] o custo de transmissão e receção de dados têm grande relevância nestes dispositivos. Os testes efetuados mostram, por exemplo, que o custo de transmissão de um byte é cerca de dez vezes superior ao custo energético necessário para calcular o resultado da execução da função de

hash SHA-1 sobre o mesmo.

Nos testes realizados em [23], é também mostrado que o custo de um *handshake* inicial entre dois nós, com autenticação mútua e estabelecimento de chave simétrica entre ambos, quando comparado com o custo total após 64kB transmitidos entre ambos, representa apenas cerca de 2% do total da energia consumida.

Por último os autores de [23] afirmam que usando o microcontrolador de baixo custo *Texas Instruments* MSP430, que será usado neste projeto, os resultados serão ainda melhores pois este microcontrolador realizará o mesmo número de instruções consumindo menos de metade da energia.

Em [24] TinyECC, é apresentada uma biblioteca configurável para o uso de ECC, desenhada para redes de sensores e executável por exemplo no TinyOS. TinyECC cobre todos os usos típicos de esquemas *public-key cryptography* (PKC), como estabelecimento de chave entre nós, usando uma variante do método *Diffie-Hellman*, *Elliptic curve Diffie–Hellman* (ECDH), assinatura e verificação de uma assinatura digital.

Pairing-Based Cryptography

Pairing-Based Cryptography (PBC) é uma técnica criptográfica que permite que dois elementos que conhecem os identificadores de ambos calculem de modo autenticado uma chave simétrica partilhada entre ambos.

Consideremos o seguinte cenário, em que cada nó tem um único identificador associado (ID) e um segredo não partilhado com qualquer outro elemento. Dois nós que conhecem o ID um do outro sem comunicação são capazes de derivar uma chave secreta apenas partilhada pelos mesmos. O processo é autenticado pois cada um dos intervenientes tem a garantia que a apenas um nó legítimo conseguiria calcular uma chave idêntica à calculada pelo próprio.

Para a realização do processo é necessária a existência de uma entidade confiável. Esta entidade cria os segredos e os identificadores que serão guardados em cada um dos nós. Este segredo é respetivo ao ID que está associado a cada nó, S_x no caso do nó com ID_x. Para além deste segredo esta entidade oferece a cada nó uma função F ($F(ID)$) que recebe como input o ID de cada nó e retorna o chamado parâmetro publico (P_x), ($F(ID_x) = P_x$).

A autoridade de confiança cria os segredos associados a cada um dos elementos (S_x), usando um segredo global (S), que apenas é conhecido pela própria entidade, e utilizando também o parâmetro público associado ao respetivo elemento ID ($S_A = [S] * P_A$).

O segredo desta técnica está na forma como estes parâmetros são criados, pois tal permite que exista uma função polinomial (e) que possui a seguinte propriedade:

$$Kab = e(S_A, (P_B) = e(S_B, (P_A)$$

É com esta propriedade que cada elemento utilizando apenas o segredo que possui S_A e o parâmetro público do restante elemento P_B, calcula uma chave que será partilhada com o restante elemento.

Embora esta seja uma técnica ainda não muito utilizada, é uma técnica recente que parece encaixar-

se perfeitamente no âmbito da rede de sensores. Oferece um modo de estabelecimento de chave partilhada autenticado entre dois elementos, com baixos custos computacionais e energéticos.

Em [25] é apresentada uma implementação de um esquema de *Pairing-Based Cryptography* para processadores de 8 bits. Os resultados teóricos apresentados mostram que técnica permite a criação de uma chave partilhada de modo autenticado entre duas partes, em menor tempo quando comparado com outras técnicas como ECDH, mas por outro lado os recursos de memória ROM necessários são o dobro.

2.3 *Routing* em redes malha e de sensores

2.3.1 Principais protocolos de *routing*

No desenvolvimento das redes em malha e de sensores, um dos pontos a que foi dada uma grande atenção desde cedo foi o estudo de protocolos de *routing*, protocolos responsáveis pela entrega dos dados que os sensores captam. O *routing* nas redes de sensores é um ponto muito desafiante, dadas as diferenças existentes para as redes de computadores em malha. As principais diferenças são:

- A impossibilidade de existência de um esquema de endereçamento global como o do protocolo IP, para redes com um elevado número de nós;
- Praticamente a totalidade das aplicações requerem a existência de um fluxo de dados para um nó especifico da rede, o *gateway*;
- O tráfego gerado pelos nós para muitas aplicações é redundante e pode ser alvo de agregação;
- Dispositivos usados neste tipo de redes, como já referido, possuem menores capacidades, do que as apresentadas pelos dispositivos usualmente utilizados em redes de computadores em malha;

Os protocolos de *routing* desenvolvidos têm a particularidade de o seu desenho ter sido bastante influenciado pelo tipo de aplicação e pela arquitetura da rede. Os protocolos foram desenhados tendo em conta características especificas como o modelo da rede, com nós móveis ou estacionários, dependendo da sua colocação na rede, se é ou não feita de um modo determinista, tendo também em conta se os nós possuem todos os mesmos tipo de recursos, se a aplicação permite ou não agregação de dados, se é ou não considerada comunicação usando *multi-hop*, e um ponto muito relevante é o modo como os dados são entregues. A troca de dados pode ser feita de um modo continuo, ou seja, periodicamente todos os nós enviam os seus dados para o *gateway*, pode ser despoletada apenas por certos eventos detetados pelos sensores, em resposta a pedidos normalmente do *gateway* da rede, ou pode mesmo ser um modelo híbrido entre as opções anteriores.

Tal como os protocolos de *routing* para redes em malha, os protocolos de *routing* para as redes de sensores sem fios podem ser divididos em dois grandes grupos bastantes distintos, protocolos reativos ou pró-ativos.

Os protocolos reativos, como o nome sugere, apenas é calculada a rota de encaminhamento quando é necessário enviar alguma mensagem, pelo que normalmente antes do envio de uma mensagem para outro nó é necessário primeiro realizar um processo de descoberta do caminho.

Pelo contrário num sistema pró-ativo os nós estão em constante comunicação para criação e atualização das tabelas de *routing* de cada um. Para que cada nó saiba sempre qual o próximo nó para o qual deve enviar uma nova mensagem, sem ser necessário realizar o processo de descoberta do caminho.

Os protocolos podem ainda ser divididos, como sugerido em [1], em três tipos distintos:*planos*, *hierárquicos* ou *baseados na localização*.

Nos protocolos *planos* todos os nós realizam o mesmo tipo de operações e estão em igualdade. Os protocolos *hierárquicos* delegam em nós específicos de tarefas como agregação de dados ou a responsabilidade de encaminhamento de mensagens. Quanto aos protocolos *baseados na localização*, este tipo de protocolos utiliza a informação de posição dos nós para realizar o encaminhamento das mensagens, por exemplo, encaminhando mensagens apenas para uma dada região da rede, em contraste com o envio para a totalidade da mesma.

Uma divisão diferente foi sugerida anteriormente em [26], onde não é apresentando o conceito de uma arquitetura *plana*, mas inclui um novo tipo de protocolos os *data-centric*. Este tipo de protocolos estão muito relacionados com o modelo de transmissão de dados em resposta a pedidos do *gateway*, pois permitem a especificação do tipo de dados requerido, evitando assim o envio de informação desnecessária.

De seguida são apresentados alguns dos protocolos de *routing* mais usados em redes malha e de sensores.

SPIN

O protocolo SPIN [27], baseado numa arquitetura *plana*, foi um dos primeiros protocolos *data-centric* desenhados. A ideia é simples, baseia-se na troca de meta-dados entre nós para definição dos tipos de dados que serão trocados. Cada nó da rede ao receber novos dados anuncia aos seus vizinhos que tem novos dados, estes, caso queiram receber os dados, enviam uma mensagem de pedido dos mesmos para o nó que efetuou o anuncio. Existem portando três tipos de mensagens: mensagem de anuncio de novos dados (ADV), mensagem de pedido de envio (REQ) e mensagem de envio de dados (DATA).

Este protocolo evita, ao contrário dos mecanismos de *flooding*, que um nó receba dados desnecessários, e evita também que os nós recebam informação repetida, possibilitando assim a poupança de energia. Por outro lado, uma grande desvantagem deste protocolo é o facto de não garantir o envio da informação, para os nós mais distantes da origem. Os nós mais distantes, do primeiro nó que anuncia os dados, apenas irão receber os dados se existirem nós intermédios interessados nos mesmos. Pois os nós intermédios apenas realizam a repetição das mensagens de anuncio de novos dados após possuírem os mesmos.

Directed Diffusion

O *Directed diffusion* [28] é também um protocolo do tipo *data-centric* baseado numa arquitetura *plana*, que foi desenhado para o modelo de envio de dados em resposta a pedidos.

Neste protocolo são usados pares atributo-valor para representar e distinguir os diferentes tipos dados e os pedidos de informação são feitos especificando esses pares.

O protocolo tem o seguinte funcionamento. Normalmente o *gateway* da rede define os dados em que está *interessado* receber, especificando os pares atributo-valor. De seguida efetua o *broadcast* dos mesmos para os seus vizinhos. Os nós guardam os *interesses*, efetuam a sua propagação e criam os caminhos de retorno até ao *gateway*. Quando um nó da rede decidir enviar dados para o *gateway* começa por enviar os dados pelos caminhos estabelecidos, no sentido contrário da propagação dos *interesses*. Posteriormente é efetuada a consolidação do caminho que permite a entrega de dados a um ritmo superior dos restantes, passando este a ser o caminho usado com uma maior frequência.

Várias melhorias foram sugeridas ao protocolo inicialmente proposto, como por exemplo a manutenção de vários caminhos de envio de dados, para que em caso de falha a comunicação continue a ser possível sem ser necessária a procura de novo caminho.

O *Directed Diffusion* é um protocolo simples e eficaz, mas apenas faz sentido para redes em que não exista um fluxo constante de dados com destino ao *gateway*.

Gradient-based *routing*

Este protocolo [29] é uma versão alterada do *Directed Diffusion* e por isso é também um protocolo do tipo *data-centric* baseado numa arquitetura *plana*. A ideia básica consiste em cada nó manter a sua distância para o nó que realizou a difusão do interesse, que normalmente é o *gateway* da rede. Assim cada nó fica a conhecer e mantém o seu número mínimo de saltos (*hops*) até ao *gateway*, sendo este valor referido como o seu peso.

A diferença entre o peso de um nó e o peso de um dado vizinho é chamado de gradiente da ligação. Pelo que as mensagens de cada fluxo são encaminhadas pela ligação que oferece o maior gradiente. Quando um nó faz parte de vários caminhos este poderá realizar agregação de dados, sendo tal mecanismo responsabilidade de cada aplicação.

Cada nó pode usar ainda três técnicas diferentes para balancear a distribuição do tráfego e assim aumentar o tempo de vida da mesma:

-*Esquema estocástico*: Quando existem duas ou mais ligações que ofereçam o mesmo gradiente, para cada mensagem um é escolhido de modo aleatório.

- *Esquema baseado na energia*: De modo a poupar energia de um nó intermédio de um ou mais caminhos, quando a restante energia do mesmo atingir um dado valor, o nó pode aumentar o seu peso, na tentativa de deixar de ser *relay* de algum caminho estabelecido.

- *Esquema baseado no fluxo*: Mecanismo que permite que novos fluxos, não sobrecarreguem nós que já fazem parte do trajeto de vários outros fluxos.

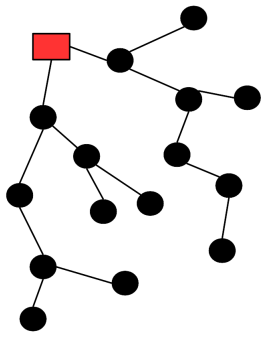

Figura 2.6: *Exemplo de uma árvore de encaminhamento criada pelo protocolo TinyOS beaconing, [4].*

LEACH

Low-energy adaptive clustering hierarchy (LEACH)[30] foi um dos primeiros protocolos de *routing* hierárquicos apresentados e serviu de inspiração para outros.

A ideia é formar *clusters* de sensores escolhendo dentro de cada *cluster* um nó líder e delegar nesses nós a responsabilidade de efetuar o encaminhamento das mensagens até ao *gateway*. Dependendo da aplicação a agregação de dados também pode ser feita. De modo a não sobrecarregar em demasia os nós lideres é definido que estes não serão sempre os mesmos, existindo várias rondas, e no inicio de cada uma cada nó decide de forma aleatória, tendo por base um valor de probabilidade, se tentará ou não assumir-se como líder.

LEACH não pode ser usado em redes de sensores de larga escala pois o protocolo assume que todos os nós conseguem comunicar diretamente com o *gateway*.

TinyOS beaconing

TinyOS beaconing é um protocolo *plano*. Consiste na criação de uma árvore tendo o *gateway* como ponto raiz como mostrada na figura 2.6. Árvore é criada a partir de uma mensagem periodicamente enviada pelo *gateway*. Os nós vizinhos ao receberem a mensagem incrementam um certo valor que representa a distância em número de *hops* para com o *gateway* e reenviam a mensagem em *braoadcast* para os seus vizinhos, que realizam as mesmas operações. Cada nó tendo em conta as mensagens que recebeu define como seu pai o seu nó vizinho que lhe ofereça o menor número de *hops* para o *gateway*.

Este protocolo é bastante simples e por isso muito usado por aplicações que requerem um fluxo continuo de dados para o *gateway* da rede.

GEAR

Geographic and energy-aware routing (GEAR) [31] é um protocolo *baseado na informação de localização*.

Este protocolo tira partido da informação de localização para melhorar a eficiência do encaminhamento das mensagens. Cada nó da rede guarda informação sobre o custo estimado para o envio de

uma mensagem para um dado destino através dos seus vizinhos. O custo associado a cada vizinho é uma combinação da sua energia restante e da sua distância para o destino. Quando recebe uma mensagem para encaminhar o nó tenta encaminhar a mensagem pelo seu vizinho mais próximo do destino, que tenha também níveis razoáveis de energia.

GEAR necessita uma troca constante de mensagens entre vizinhos para atualização quer da informação sobre a localização quer do estado de energia, e requer que todos os nós da rede consigam calcular ou obter informação sobre a sua localização.

2.3.2 *Routing* seguro

Como é mostrado em [4] os protocolos de *routing* para redes em malha e de sensores apresentam vários tipos de falhas de segurança. Tal facto advém de a segurança não ter sido considerada como um ponto essencial no desenho dos mesmos.

Como apresentado em [4], os ataques que tornam os protocolos vulneráveis são os já referidos na secção 3.1:

- *Falsificação de identidade.*
- *Alteração do conteúdo das mensagens.*
- *Repetição de mensagens.*
- *Selective Forwarding attack.*
- *Sinkhole attack.*
- *Sybil attack.*
- *Wormhole attack.*
- *Hello flood attack.*

A grande maioria dos ataques aos esquemas de *routing* têm como grande objetivo diminuir a disponibilidade dos nós e do sistema, tentando impedir ou diminuir a comunicação. Por exemplo, no caso do protocolo *TinyOs beaconig* sem autenticação das mensagens é fácil para um atacante personificar o *gateway* e particionar a árvore de encaminhamento criando outra árvore tendo-o a ele como raiz. Atraindo assim muito tráfego da rede e podendo simplesmente realizar um *blackhole attack*.

Desde [4] que várias sugestões de protocolos de *routing* seguros têm surgido. Em [32] é referido que existem três diferentes abordagens possíveis para os protocolos de *routing* seguros, abordagens que são normalmente usadas de modo exclusivo. Estas três abordagens a seguir são: realizar prevenção de possíveis ataques, implementar mecanismos de deteção e recuperação a ataques, e desenhar e implementar técnicas resistentes a ataques.

A primeira abordagem é a mais usada, e tem como base o uso de algumas das técnicas referidas na secção 3.2. Ou seja, consiste na implementação de mecanismos para fortalecer o protocolo contra ataques, normalmente usando mecanismos criptográficos. Os autores de [32] afirmam que esta abordagem é normalmente a mais eficaz.

A deteção envolve a monitorização em tempo real do comportamento do protocolo e dos participantes. Uma vez descoberto o comportamento malicioso são usadas técnicas de recuperação para possível eliminação de participantes, e restauro do correto funcionamento da rede.

A ultima abordagem de resiliência procura manter um certo nível de disponibilidade, mesmo sobre a presença de ataques.

De seguida são apresentados alguns protocolos de *routing* para redes em malha e de sensores que foram desenhados tendo em conta a segurança. Cada um dos protocolos apresentados utiliza pelo menos uma das abordagens referidas.

Secure Directed Diffusion

Como referido em [4] o protocolo *directed diffusion*, apresentado na secção anterior é vulnerável a vários tipos de ataques, pois no seu desenho não foi tido em conta o aspeto da segurança.

O *secure directed diffusion* (SDD) [33] usa um mecanismo similar ao μTesla apresentado anteriormente, para garantir a autenticidade e integridade das mensagens de *routing* e de dados. O protocolo assume a existência de apenas um *gateway* na rede e que este partilha com cada nó da rede uma chave simétrica secreta.

Entrando em mais pormenor, inicialmente é realizada a criação da cadeia de chaves unidirecional, posteriormente em todos os nós da rede é colocada a chave inicial K_0. Apenas o *gateway* conhece a totalidade da cadeia para poder autenticar as suas mensagens.

O Funcionamento é muito idêntico ao protocolo original, num primeiro passo o *gateway* difunde o chamado *interesse* pelos seus vizinhos enviando a seguinte mensagem.

$$M = (H(interesse1)|MAC(K_1|H(interesse1)))$$

Onde: $H(m)$ é o *hash* da mensagem m, $MAC(k, m)$ é o *Message Authentication Code* de m calculado usando chave k.

Supondo que todos os nós receberam o pedido, posteriormente o *gateway* da rede envia a mensagem $(interesse1|K_1)$. Cada nó pode verificar que a mensagem inicial foi efetivamente enviada pelo *gateway* verificando se $F(K_1) = K_0$. Tendo k_1, cada nó pode verificar a integridade e autenticidade de $H(interesse1)$ e logo obter $interesse1$, tendo sido assim difundido devidamente autenticado o $interesse1$ por toda a rede.

O envio de mensagens feito posteriormente por cada um dos nós para o *gateway* também é feito garantindo a autenticidade e integridade das mesmas.

Numa fase inicial um nó n difunde pelos vários caminhos, pelos quais recebeu a primeira mensagem vinda do *gateway*, a seguinte mensagem M.

$$M = (H(DATA)|MAC(K_n^1|H(DATA))|(K_n^0)S_n|N)$$

Onde: K_0 é a primeira chave de uma cadeia de chaves unidirecional gerada pelo nó n, $(K_n^0)S_n$ a cifra de K_n^0 usando a chave S_n partilhada pelo nó n e o *gateway*.

Ao receber esta mensagem o *gateway* consegue obter K_n^0, usando a chave partilhada com n, e assim obter a primeira chave da cadeia de chaves gerada pelo nó n. Posteriormente este informa, segundo os mecanismos do protocolo μTESLA, todos os nós de que K_n^0 é a primeira chave da cadeia gerada pelo nó n.

Depois da fase de escolha e reforço de um dado caminho, para enviar os dados ($DATA1$) o nó n envia $\{H(DATA1)|MAC(K_n^2, H(DATA1))\}$ para os seus vizinhos. Passado um dado tempo t envia $\{DATA1|K_n^2\}$. Cada nó ao receber verifica se $DATA1$ foi efetivamente enviada por n, usando o mecanismo μTESLA, e procura na sua *cache* por nós vizinhos interessados nos dados por este enviados.

Secure directed diffusion é um protocolo que utiliza uma abordagem preventiva e é resistente a vários ataques bem conhecidos. Por outro em relação ao protocolo original deixa de oferecer a funcionalidade de agregação de dados, pois impossibilita que nós intermédios alterem os dados. SDD impossibilita ainda a recuperação de um caminho já estabelecido, pois em caso de falha de um nó intermédio será necessário reiniciar o processo.

SeRINS

Este protocolo, apresentado em [34], tem como objetivo proteger a rede contra possíveis ataques realizados por nós que fazem parte da mesma. Como referido na secção 3.1 este tipo de ameaça pode gerar ataques como *selective forwarding* ou *blackhole attack*.

Os autores de SeRINS assumem que os nós podem ser comprometidos, mas apenas um pequeno número em relação ao número total de nós da rede. É assumido também que cada nó da rede partilha uma chave secreta com o *gateway*, e que cada par de nós vizinhos possuem uma chave secreta partilhada entre ambos.

Protocolo consiste em três esquemas distintos: *alternate path scheme*, *neighbor report system* e *neighbor authentication*. O *Alternate path scheme* estabelece o esquema de *routing*, similar ao *TinyOs beaconing* referido na secção anterior, ou seja, o *gateway* cria uma árvore sendo ele a raiz da mesma.

Árvore é criada tal como no *TinyOs beaconing* realizando difusões periódicas de mensagens com informação de *routing*, contendo um número de sequência (SN), para identificar as diferentes rondas, e um contador do número de saltos (HC). Em contraste com o *TinyOS beaconing*, as mensagens do SeRINS são autenticadas e cada nó guarda mais do que um nó pai, mantendo assim vários caminhos possíveis para o *gateway*.

Cada nó considera todos os seus pais de igual forma, sendo a escolha do próximo nó para encaminhamento de cada mensagem baseada num sistema *round-robin*. Assim nós que teriam no seu caminho mais próximo para o *gateway* um nó comprometido por exemplo a realizar o ataque de *selective forwarding*, irão ter um maior número de mensagens entregues ao *gateway*, dado o menor número de vezes em que o nó comprometido faz parte do caminho usado para encaminhar a mensagem até ao *gateway*.

Na disseminação das mensagens de *routing* de modo a garantir a autenticidade é usado o protocolo μTESLA entre o *gateway* da rede e os seus vizinhos, para que seja impossível personificar o *gateway*.

Nos saltos seguintes da difusão a autenticação é garantida usando o *neighbor authentication scheme*.

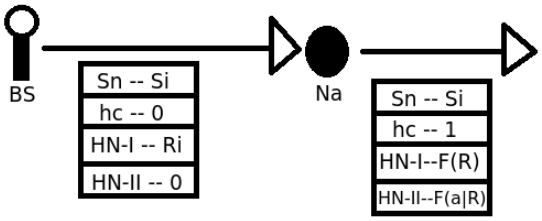

Figura 2.7: *Inicio de uma nova ronda de difusão da mensagem de routing.*

Este método usa o protocolo ARMS apresentado anteriormente, lembrar que em relação ao μTESLA o protocolo ARMS requer o uso de menos memória em cada nó e não necessita de sincronização de relógios entre os nós.

Como já referido o método *Alternate path scheme* torna a rede resiliente a nós comprometidos que realizem, por exemplo, o ataque *selective forwarding*. Mas um nó comprometido pode ainda mentir sobre a sua distância para o *gateway* e por isso o SeRINS usa ainda o mecanismo *neighbor report system*. O *neighbor report system* permite que cada nó verifique a informação de *routing* enviada por outro nó e em caso de inconsistência ou suspeita da veracidade da informação pode reportar o acontecimento ao *gateway* da rede. O *gateway* posteriormente decide se nó suspeito é eliminado da rede, com base no número de acusações feitas sobre o mesmo.

Para permitir a verificação das mensagens de *routing* recebidas foram adicionados mais dois campos às mensagens trocadas. Estes dois campos são o *hopping number I* ($HN-I$) e *Hopping number II* ($HN-II$). O *Hopping number I* é usado para prevenir a falsificação da contagem do número de saltos, enquanto o *Hopping number II* é usado para evitar o ataque de igual distância.

Em determinada ronda (Si) o *gateway* escolhe um número aleatório R e inicia a difusão enviando a mensagem M.

$$M = \{SN = Si, HC = 0, HN - I = R, HN - II = 0\}$$

Cada nó Na ao receber uma nova mensagem de *routing*, faz *update* aos seus nós pais, se receber uma mensagem com HC menor e guarda HC e $HN - I$ recebidos. De seguida, tal como ilustrado na figura 2.7, nó Na difunde uma nova mensagem para os seus vizinhos. Esta mensagem contêm o campo HC incrementando em um valor, um novo *hopping number I* calculado aplicando uma função, de *hash*, F ao *hopping number I* ($HN - I$) recebido ($F(HN - I)$), e com o *hopping number II* igual a $F(ID_n|HN - I)$, onde ID_n é o identificador do nó Na. A função F usada é uma função de *hash unidirecional*, que todos os nós da rede conhecem.

Assim ao receber uma nova mensagem de *routing*, cada nó verifica os valores recebidos para *hopping number I e II* antes de realizar nova difusão. O modo como é feita a verificação e deteção de mensagens maliciosas é explicado, de seguida, em dois casos distintos.

Caso a. Ao receber mensagem $M1$ ($M1 = \{SN = Si, HC = h', HN - I = I', HN - II = II'\}$) com valor de HC menor ou igual que o HC (h) do seu primeiro nó pai, que tem I como *hopping number-I*, nó verifica se $F^{h-h'}(I') = I$, como ilustrado na figura 2.8. Se a condição se verificar nó aceita mensagem

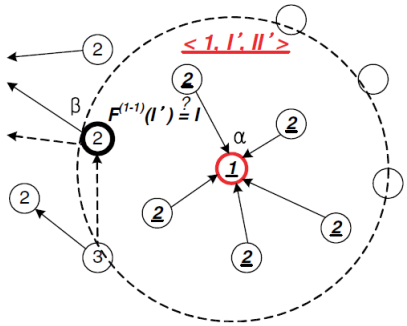

Figura 2.8: [34] Nó comprometido α falsifica contagem anunciando *hop count* igual a um em vez de três.

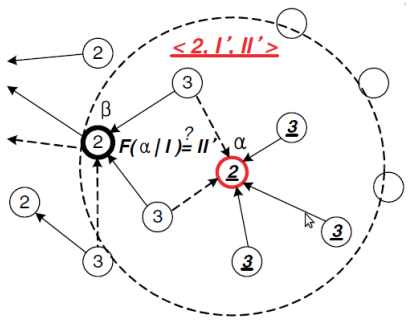

Figura 2.9: *[34] Nó comprometido α falsifica contagem anunciando hop count igual ao anunciado por β.*

e realiza difusão, caso contrário nó alerta o *gateway* indicando o nó de quem recebeu a mensagem.

No exemplo da figura 2.8, o nó comprometido α falsifica a contagem anunciando o *hop count* igual a um em vez de três. Como o valor de *hopping number-I* em cada passo é alterado, sendo calculado a partir de uma função de *hash* unidirecional, nó α não consegue saber o valor legítimo para *hopping number-I* associado ao HC igual a 1. Assim nó β ao realizar a verificação da informação de *routing* suspeita do comportamento de α.

Caso b. Este ponto mostra como é detetado o ataque de igual distância. Quando um nó β recebe de α uma mensagem de *routhig* idêntica à que ele anunciou, ou seja, com idêntico *hop count* (h') e *hopping number- I* (I') idênticos, nó β verifica *hopping number-II* (II') recebido de α. Neste caso nó β usa os *hopping number-I* do seu nó pai (I) e verifica se é válida a condição $F(\alpha|I) = II'$, ver figura 2.9. Caso condição não seja verificada β reporta suspeita sobre α ao *gateway*. Este esquema irá criar falsas suspeitas, falsos positivos, pois é assumido que nó α terá de ter em comum com β o mesmo pai para poder oferecer a mesma distância.

No exemplo da figura 2.9, nó comprometido α falsifica contagem anunciando *hop count* igual ao anunciado por β. Neste caso como α não conhece o valor legítimo para *hopping number-I* do primeiro pai de β, será detetado.

Com o objetivo de aumentar a eficiência energética as verificações só são efetuadas após um número definido de mensagens recebidas de modo sequencial do mesmo nó com o mesmo *hop count*.

SeRins é resistente a todos os ataques referidos na secção 3.1 como *Sybil* e *Sinkhole*. Para além de

uma abordagem preventiva este protocolo ao permitir a existência de vários caminhos para o *gateway* está a usar também uma abordagem resiliente, pois o funcionamento da rede é mantido mesmo na presença de nós comprometidos.

Protocolo garante ainda proteção contra ataques de nós exteriores à rede pois toda a comunicação de *routing* é realizada garantindo a sua autenticidade e integridade.

A clean slate approach

Este protocolo, apresentado em [32] foi desenhado para redes em malha, dado que permite o modelo de comunicação nó a nó. Ou seja, cada nó pode comunicar com qualquer outro nó da rede, e não necessariamente apenas com os seus vizinhos ou com o *gateway*.

Protocolo combina as três diferentes abordagens por ele apresentadas: prevenção, deteção e recuperação, e também a abordagem resiliente. Usa criptografia de chave pública pois os autores assumem que o processo de verificação pode ser realizado de modo eficiente por cada nó.

Protocolo assume a existência de uma autoridade da rede *network authority* (NA), que possui um par de chaves pública e privada. Nos restantes nós do sistema são pré-carregados os seguintes elementos: a chave pública da NA, um identificador ID_a, único para cada nó, e um certificado $\{ID_a\}_{K_{NA}^{-1}}$, assinado digitalmente pela NA.

A NA fornece também a cada nó um conjunto de k desafios($C_0, C_1, ..., C_k$) verificáveis, usados para autenticação. Estes desafios são uma cadeia de chaves unidirecional, $Y = \{(C_0, C_k) : H^k(C_k) = C0\}$, onde $H^k(y)$ significa aplicar a função de *hash* k vezes, usando y como chave inicial. Em cada nó é carregado ainda o desafio inicial (C_0), juntamente com o identificador do nó assinado pela NA com a sua chave privada ($\{C_0||ID_a\}_{K_{NA}^{-1}}$)). Assim para responder ao um desafio de autenticação durante a ronda i, o nó envia o desafio C_i ($H^{k-i}(C_k)$), juntamente com o iniciador C_0 o seu identificador na rede e o certificado $\{C_0||ID_a\}_{K_{NA}^{-1}}$.

Qualquer nó com estes dados pode verificar o desafio, usando a chave pública de NA para confirmar a identidade do nó. Posteriormente usando o correspondente iniciador é verificado se $H^i(C_i) = C_0$.

Este processo de autenticação descrito é utilizado inicialmente para descoberta de nós vizinhos devidamente autenticados. Processo é realizado pela trocada dos ID's e certificados entre os nós, sendo localmente formado um grupo de nós vizinhos devidamente autenticados. Protocolo posteriormente atribui endereços de rede exclusivos para cada nó e estabelece as tabelas de *routing* para permitir encaminhamento entre os nós do grupo. Nesta fase o *sybil attack* é impedido pois são usados os certificados assinados pela NA.

Utilizando agrupamento recursivo (*recursive grouping*), os diferentes grupos locais de nós são recursivamente agrupados, utilizando um método semelhante ao anterior, até ser formado um único grupo na rede. Durante o algoritmo de agrupamento, rede é hierarquizada e endereços de cada nó são constantemente atualizados. Após este processo cada nó consegue encaminhar mensagens para qualquer outro nó da rede, mas processo não permite a posterior entrada de novos nós para a rede.

Protocolo utiliza também a abordagem resiliente pois possibilita o uso de vários caminhos para o encaminhamento de cada mensagem, dado que cada nó pode guardar mais do que um nó como sendo

o seguinte para uma dada gama de endereços. Protocolo possibilita ainda que nó inicial possa associar a cada mensagem uma indicação sobre o encaminhamento, indicando como cada nó intermédio deve encaminhar a mesma.

Este protocolo usa ainda técnicas para detetar e eliminar nós com comportamento malicioso. O algoritmo de verificação do agrupamento em árvore *grouping verification Tree* (GVT) deteta comportamento malicioso durante a fase de agrupamento recursivo. GVT é baseado nas árvores de *hash* [35] e impede que um nó malicioso faça parte do grupo ou impeça o correto funcionamento do algoritmo.

A segurança deste algoritmo é garantida pela fase de descoberta de modo autenticado de nós vizinhos e pelo uso do algoritmo de agrupamento recursivo que está protegido pelo GVT.

INSENS

Dada a simplicidade do modo básico deste protocolo será apenas aqui apresentada a versão corrigida do protocolo [36].

Este protocolo de *routing*, é apresentado como sendo tolerante a intrusões. O funcionamento do protocolo de *routing* é semelhante aos protocolos *TinyOs beaconing* e *gradient based*. Ou seja, consiste na criação de uma árvore de encaminhamento tendo como raiz um nó *gateway*, sendo neste caso possível a existência de vários nós *gateway* para a mesma rede.

De modo a tolerar falhas de nós e ataques internos como o *blackhole* e o encaminhamento seletivo, para além de ser permitida a existência de vários nós *gateway*, são mantidos sempre que possível vários caminhos entre os nós da rede e cada um dos nós *gateway*. Assim o protocolo é tolerante a intrusões, pois a existência de caminhos redundantes para os vários nós *gateway* limitam possíveis ataques realizados por nós atacantes.

Cada um dos nós *gateway* da rede difunde periodicamente pela rede uma mensagem de *route request* (REQ), mensagem responsável pela criação da árvore de encaminhamento. Esta mensagem é difundida pela rede de modo autenticado, usando um mecanismo semelhante ao protocolo *ARMS*. Para tal cada nó mantém uma chave de grupo de vizinhança (*cluster key*), com os seus nós vizinhos e é usada de modo a impedir que um atacante consiga interferir na comunicação. De modo a evitar o *wormhole attack* o *MAC* (*message authentication code*) associado a cada mensagem recebida depende de informação de cada um dos nós intermédios. Para tal o *MAC* é recalculado em cada nó usando a chave de grupo de vizinhança (*cluster key*). Para além da autenticação e integridade as mensagens *REQ* são também trocadas garantindo a confidencialidade das mesmas segundo o protocolo *ARMS*, ou seja, usando a mesma chave de grupo de vizinhança.

O protocolo *INSENS* recorre ao protocolo *LEAP* para o processo de gestão de chaves, ou seja, para o estabelecimento de pares de chaves entre os nós legítimos vizinhos na rede. É durante este processo inicial de estabelecimento das chaves entre os pares que são também estabelecidas as chaves de grupos de vizinhança. O Recurso ao protocolo *LEAP* permite que um novo nó estabeleça chaves com os seus vizinhos e com os nós *gateway* da rede, mas em contrapartida limita o seu uso a redes estáticas. Esta limitação impede que os nós sejam móveis pois um nó que já tenha realizado a fase de autenticação e já tenha eliminado a *global Key* do sistema está impossibilitado de se autenticar perante

novos vizinhos, descobertos pela alteração da sua posição na rede.

TARF

O TARF [37] pertence ao grupo dos protocolos de *routing* que apresentam apenas uma abordagem resiliente para resolver o problemas de segurança.

O protocolo assume que as mensagens de *routing* já são enviadas garantindo a sua autenticidade e integridade, assume também que é usada uma arquitetura *plana* e um protocolo como o *TinyOs beaconig* já se encontra implementado sobre a rede.

Neste protocolo cada nó guarda sobre os seus vários nós vizinhos, um nível de confiança e um custo energético.

Nível de confiança associado a cada nó vizinhos é a probabilidade, do ponto de vista de um nó, de dada mensagem ser efetivamente entregue ao *gateway*, usando esse nó vizinho como próximo salto. Custo energético é a estimativa para a energia que deverá ser consumida pelo conjunto de todos os nós de um dado caminho, que inclui o nó vizinho e encaminha mensagens até ao *gateway*. Esta estimativa tem por base o número de saltos restantes para o *gateway* e o custo médio de transmitir um pacote de tamanho uniforme entre dois nós. É com base nestes dois pontos, nível de confiança e custo energético, que são baseadas as decisões de encaminhamento.

Para que os nós consigam obter o nível de confiança associado a cada nó vizinho, o *gateway* periodicamente difunde uma mensagem, indicando para cada nó da rede as mensagens que este não recebeu. Cada nó guarda informação sobre que nó vizinho usou para enviar cada mensagem até ao *gateway*. Assim ao receber a mensagem difundida pelo *gateway* cada nó da rede consegue calcular a percentagem de mensagens enviadas com sucesso, através de cada um dos seus nós vizinhos. Esta percentagem representa o nível de confiança associado a cada nó vizinho.

2.4 Resumo do capítulo

Ao longo deste capítulo foi exposto o trabalho relacionado sobre aspetos de segurança das redes de sensores e redes em malha.

Na primeira secção, foi feita uma introdução ao tópico das redes de sensores sem fios, sendo mencionadas as possíveis aplicações para esta tecnologia. Foi também identificado o tipo de *hardware* normalmente usado nestes sistemas, que como referido, apresenta limitações e termos de capacidade de memória, capacidade computacional e dependendo da aplicação capacidade energética. De seguida foram identificadas as principais ameaças e requisitos de segurança para redes deste tipo, e foram também analisados os principais ataques a que as redes de sensores estão sujeitas.

Posteriormente, na secção 2.2, foram apresentadas as principais propostas de mecanismos desenhados para contrariar o problema da segurança da comunicação em redes de sensores em malha. Foram identificados os principais protocolos que pretendem garantir as propriedades de segurança básicas ao nível da comunicação entre os nós de uma rede: Protocolos que permitem a comunicação segura entre dois nós que partilhem uma chave simétrica, e protocolos que garantem a autenticidade

de mensagens enviadas em *broadcast*. Por fim na subsecção 2.2.3 foi analisada a problemática da gestão de chaves, tendo como principal foco o processo de distribuição das mesmas.

Na secção 2.3, foi analisado um dos pontos fundamentais das redes de sensores, o protocolo de *routing*. Inicialmente foram identificados os principais protocolos usados nas redes de sensores e em redes em malha. Posteriormente foram apresentados protocolos de *routing* que oferecem garantias de segurança, usando algumas das técnicas identificadas anteriormente.

De forma geral como é descrito ao longo do capitulo devidos às especificidades das redes de sensores e às limitadas capacidades dos dispositivos usados, não é normalmente possível o uso de alguns mecanismos de segurança mais usados em redes de computadores, sendo por isso apresentadas outro tipo de soluções.

Em termos da segurança da comunicação as técnicas apresentadas na subsecção 2.2.1 oferecem boas garantias para este tipo de redes. Por outro lado, o processo de gestão de chaves é o ponto em as redes de sensores encontram maiores desafios, nomeadamente para sistemas em que é requerido que o processo de distribuição de chaves seja realizado posteriormente à fase de disposição dos nós no meio, e quando devido às limitações dos sensores é dificultado o uso de criptografia assimétrica.

Capítulo 3

Solução Proposta

Este capítulo tem como objetivo descrever em maior detalhe o problema associado a este trabalho e apresentar a solução implementada para o mesmo. O capítulo é constituído por três secções. Na primeira secção é detalhado o problema e são apresentados os requisitos de segurança identificados para o mesmo. Na secção 3.2, é apresentada a solução desenhada no seu todo e por fim na última secção, a 3.3, é apresentada a implementação, de um modo detalhado, para cada um dos pontos da solução.

3.1 Modelação do problema

Como referido na introdução deste documento, o problema que se pretende resolver consiste em garantir as diversas facetas da segurança da comunicação entre um conjunto de dispositivos que formam várias redes em malha e que comunicam com uma entidade geograficamente distante, através da Internet ou via rede móvel.

Cada uma destas redes em malha pode ser constituída por dezenas de nós com sensores e um ou mais nós com maiores capacidades, e sem limitação de energia, que funcionam como *gateway*, ponto de saída, da rede. O *gateway* de cada rede permite a comunicação entre os elementos da mesma rede, e a unidade central do sistema. Para tal, estes estão ligados à Internet ou possuem capacidade de comunicação via rede móvel. Na unidade central a informação recebida é agregada e disponibilizada ao observador do sistema.

O sistema possui a arquitetura ilustrada na figura 3.1. É composto por várias redes em malha, sendo cada rede constituída por vários nós móveis, bolas pretas, e um ou mais nós *gateway*, quadrados vermelhos. Nós *gateway*, como referido, possibilitam a comunicação entre os elementos da rede e a unidade central (UC).

Os nós de cada rede consistem em dispositivos de tamanho reduzido, com capacidades computacional e energética limitadas, associados a um objeto de grandes dimensões. Estes objetos possuem a capacidade de mobilidade, ainda que permaneçam estáticos grande parte do tempo poderão alterar a sua posição na rede ou deslocar-se entre as várias redes que pertencem ao sistema.

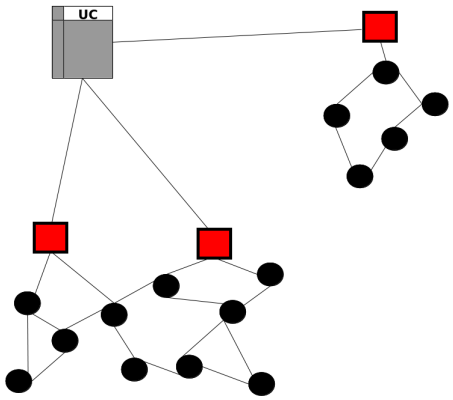

Figura 3.1: Arquitetura do sistema. Nós móveis estão representados pelas bolas pretas, já os quadrados vermelhos representam os nós *gateway*. Cada nó *gateway* recebe mensagens dos nós móveis da sua rede e encaminha as mesmas para unidade central do sistema (UC).

O objetivo do sistema é realizar a monitorização da localização geográfica e o estado dos objetos a que os dispositivos estão associados. Para tal os nós enviam periodicamente uma mensagem para a unidade central, através de um *gateway*, anunciando a sua posição geográfica e confirmando assim a sua presença em determinada rede.

Cada nó de uma rede comunica com os seus vizinhos, para efetuar a troca de dados e terá também de comunicar com um dos nós *gateway* da rede. Uma versão preliminar do protocolo de *routing* já se encontrava inicialmente desenvolvida, sendo semelhante aos protocolos *gradient-based* e *TinyOS beaconig* apresentados no ponto 3.3.2. Assim a camada de segurança é desenvolvida sobre esta camada, que já permite a comunicação entre cada nó e os nós *gateway* da cada rede.

Cada nó tem ainda a capacidade de detetar determinados eventos especiais, que dada a sua importância é necessário o alerta imediato do observador do sistema. Estes alertas correspondem à alteração de determinado estado no objeto. Ao detetar este tipo de eventos cada nó cria uma mensagem especial, que encaminha de igual modo para a unidade central, para que o observador do sistema seja alertado.

Os dispositivos usados no sistema, apresentados na figura 3.2, possuem as seguintes características:

- Processador *Texas Instruments* MSP430 (16 *bits*), 12 Mhz- ciclo de relógio.
- 128 kBytes de memória *ROM*.
- 16 kBytes de memória RAM.
- Alcance máximo de comunicação de aproximadamente de 10 metros.

O protocolo de comunicação já implementado apresenta algumas limitações que dificultaram a implementação da solução de segurança:

- O tamanho máximo para o conteúdo de um pacote de dados é de 40 *bytes*.
- Não é possível o envio de mensagens, a partir do *gateway*, com destino um nó especifico de cada rede.

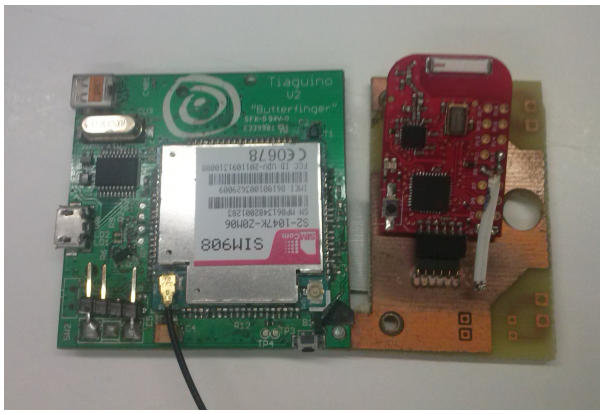

Figura 3.2: Dispositivos usados na implementação do sistema.

- Não é possível, para já, o envio de mensagens, a partir da unidade central, para um *gateway* especifico

Dado os fluxos de comunicação existentes e as limitadas capacidades dos dispositivos usados, este problema enquadra-se no conceito de redes de sensores.

3.1.1 Requisitos

A principal ameaça no ambiente considerado neste trabalho é o facto de os elementos da rede estarem particularmente vulneráveis a atacantes localizados fisicamente perto da rede, podendo estes facilmente escutar ou interferir com a comunicação.

Sendo à partida esperado que o sistema possa sofrer tentativas de ataques externos, é um requisito garantir a confidencialidade, integridade e autenticidade das trocas de dados existentes entre os elementos da rede. Para tal é necessária a implementação de um mecanismo de autenticação dos nós do sistema, para que apenas os nós legítimos consigam enviar para a unidade central mensagens válidas.

Devido à falta de segurança física não se pode assumir que é impossível um atacante aceder a um dispositivo, e ler o seu conteúdo. Com o possível roubo de informação das chaves criptográficas de nós legítimos, o sistema pode ser vítima de ataques internos em cada uma das redes. Por isso, o sistema terá de funcionar assumindo que alguns dos nós das redes em malha podem ter comportamento erróneo, como por exemplo o bloqueio de tráfego com destino ao *gateway*.

Por último é esperado, e essencial, que a solução apresentada seja eficiente, de modo a que o tempo de vida esperado para a rede não seja diminuído em demasia com a implementação da mesma.

3.2 Visão geral da solução

O objetivo desta secção é apresentar a solução desenhada de um modo simples, para que a mesma seja facilmente compreendida no seu todo.

Tal como referido anteriormente, no ponto 2.1.2, no desenho de uma solução para uma rede de sensores é usual assumir que os nós responsáveis por fazer a ligação entre a rede e o exterior, nós

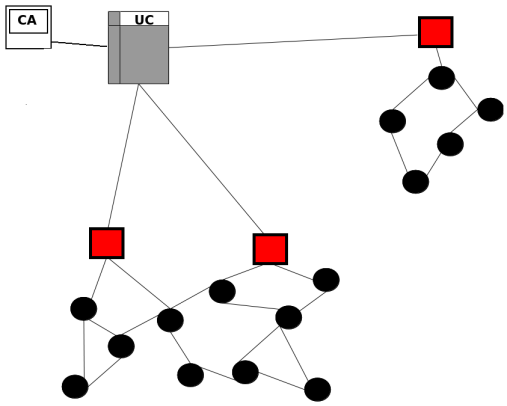

Figura 3.3: Arquitectura do sistema, solução.

gateway, são de confiança. Neste caso será também assumido que os nós *gateway* são confiáveis e não podem ser comprometidos.

De modo a garantir a autenticidade de cada um dos nós, é realizado um protocolo de autenticação recorrendo uma biblioteca de criptografia assimétrica, que usa a técnica de criptografia de curvas elípticas. A criptografia assimétrica, como discutido anteriormente, oferece garantias de segurança no estabelecimento de uma chave de sessão, partilhada entre dois nós, e permite que esta seja distinta por cada processo, o que não ocorre nas restantes opções apresentadas.

O uso de criptografia assimétrica é muitas vezes considerado inadequado para as redes de sensores. No entanto, no ambiente considerado o uso deste tipo de criptografia é viável. Cada nó apenas envia mensagens para a unidade central pelo que apenas será necessário um protocolo de autenticação, para estabelecimento de uma chave de sessão partilhada apenas entre cada nó e a unidade central. Esta chave será usada para garantir a segurança das mensagens posteriormente enviadas por cada nó para a unidade central. Neste caso embora os nós sejam móveis, podendo alterar a sua posição dentro de cada rede ou mudar mesmo de rede, o fluxo de comunicação é sempre o mesmo, as mensagens serão enviadas para unidade central.

A escolha, entre as várias técnicas de criptografia assimétrica, recaiu sobre a técnica baseada em curvas elípticas. Esta escolha é justificada por esta técnica ser a que, tendo em conta os estudos considerados anteriormente neste documento, apresenta os melhores resultados quanto ao consumo energético e em termos de capacidades computacionais e de memória requeridas.

A solução possui a arquitetura ilustrada na figura 3.3. À arquitetura original, figura 3.1, foi adicionada uma entidade, a autoridade de certificação (*certification authority* - CA), entidade de confiança responsável por gerar os pares de chaves para cada nó e criar os respetivos certificados.

Cada nó, quando instalado no sistema, possui:

- Um par de chaves pública e privada.
- A chave pública da *CA*.
- Um certificado assinado pela *CA*, atestando a veracidade da chave pública do nó.

De modo a garantir a segurança das mensagens do protocolo e *routing* foi definido que as mesmas serão seguras recorrendo a uma chave de grupo. Esta chave deverá ser partilhada por todos os nós da mesma rede que estejam devidamente autenticados.

Esta secção está dividida em cinco subsecções, que abordam os principais pontos da solução. Primeiro é abordada a questão do protocolo de autenticação. Na segunda subsecção é abordado o tema da comunicação segura entre entre cada nó e a unidade central, recorrendo à chave estabelecida no protocolo de autenticação. Na subsecção 3.2.3 são feitas algumas considerações sobre a proteção das mensagens da comunicação em grupo, que se resumem apenas às mensagens do protocolo de *routing*. Na quarta subsecção é referido de que modo é garantido o envio das mensagens de alarme. Por fim e na subsecção 3.2.5 é apresentado o modelo que permite o envio de mensagens por múltiplos caminhos.

3.2.1 Autenticação na rede

Ao se ligar pela primeira vez e estando na presença de uma rede, cada nó deverá realizar um protocolo de autenticação e estabelecer uma chave de sessão partilhada entre ele e a unidade central, de modo a garantir a segurança da comunicação entre ambos.

Como a mesma rede pode ser constituída por vários nós *gateway*, e como cada nó pode se movimentar entre as diferentes redes, de modo a evitar diferentes processos de autenticação perante diferentes nós *gateway*, os processos de autenticação serão validados pela unidade central (UC). Deste modo no fim do processo de autenticação um nó possui uma chave que ele partilha apenas com a unidade central, podendo assim mudar de rede, ou enviar mensagens por outro *gateway* da mesma rede usando sempre a mesma chave simétrica para garantir a segurança das mesmas.

O protocolo de autenticação mútua, é realizado recorrendo à criptografia assimétrica, usando a técnica de curvas elípticas. E neste caso poderá ser realizado sem a troca de certificados entre os elementos, pois cada nó possui um certificado assinado pela *CA* indicando qual a chave pública associada à unidade central, e a unidade central deverá possuir o certificado de cada nó.

A chave de sessão estabelecida, no protocolo de autenticação, terá de ter um tempo de vida associado, tempo ao fim do qual deve ser efetuada a sua renovação, ou seja, será necessário a realização de outro processo de autenticação.

3.2.2 Comunicação segura

Recorrendo à chave de sessão estabelecida, cada nó envia mensagens de modo seguro para a unidade central. As mensagens contêm dados específicos para cada aplicação, neste caso é enviada a informação de localização geográfica periodicamente. Para estas mensagens serão garantidas as seguintes propriedades: confidencialidade, integridade, autenticidade e proteção contra repetição de mensagens.

De modo a garantir a confidencialidade da comunicação é usado esquema de cifra simétrica. A chave usada é partilhada apenas entre cada nó e a unidade central e obtida pelo protocolo de autenticação

enunciado no ponto anterior.

Para garantir autenticidade e integridade da comunicação, a cada mensagem será associado um *MAC* (*message autentication code*). Este *MAC* é baseado em informação presente na cabeçalho de cada mensagem e no seu próprio conteúdo. Este mecanismo é também calculado recorrendo à chave simétrica partilhada entre cada nó e a unidade central. Assim ao receber cada mensagem a unidade central, antes de verificar o seu conteúdo, validada a autenticidade e integridade da mesma.

A proteção contra a repetição de mensagens é obtida com o uso do numero de sequencia de cada mensagem, presente no cabeçalho de cada uma e usado no calculo do *MAC*. Ou seja, caso um atacante tente repetir uma mensagem antiga alterando o seu numero de sequencia para um numero superior, ao receber esta mensagem a unidade central verificará que o *MAC* associado à mesma é inválido.

3.2.3 Comunicação em grupo

Chamamos grupo ao conjunto de nós que, por estarem fisicamente próximos, formam uma rede em malha que pode possuir um ou mais nós *gateway*. A comunicação entre os nós de cada grupo, nomeadamente a troca das mensagens de *routing*, são protegidas por um *MAC* associado a cada mensagem. Este é calculado tendo como base o conteúdo da mensagem e usando a chave de grupo, chave partilhada por todos os nós de um determinado grupo. Ou seja, a difusão das mensagens de *routing* tem a sua integridade e autenticidade garantida, pela associação de um *MAC* calculado usando a chave de grupo.

Esta chave de grupo é recebida por cada nó da rede no fim do seu processo de autenticação, sendo assim apenas partilhada pelos nós da rede corretamente autenticados.

Cada chave de grupo terá de ser idêntica dentro do mesmo grupo, para que as mensagens de *routing* mesmo que iniciadas pelos diferentes nós *gateway*, do mesmo grupo, sejam validadas por todos os elementos da rede em malha. Para tal cada grupo físico terá de estar definido na unidade central, para que dependendo da rede em que o nó se encontra, cada nó receba a correspondente chave de grupo no fim do processo de autenticação. Ao mudar de rede cada nó terá de pedir a chave usada pelo novo grupo a que este pertence. Esta ser-lhe-á enviada pela unidade central de modo seguro, usando a chave de sessão partilhada entre ambos.

Tendo como exemplo as mensagens de *routing* a única alteração, na comparação entre as mensagens originais e as novas usadas pelo sistema, é o acrescento do *MAC*, de 4 *bytes*, a cada mensagem. Este conjunto de mensagens estão deste modo protegidas quanto à sua autenticidade e integridade, para elementos exteriores ao grupo. Neste caso não é garantida a confidencialidade das mensagens, sendo o seu conteúdo é enviado em claro. Deste modo novos nós ainda não autenticados conseguem descobrir os nós vizinhos e comunicar com um dos *gateways* da rede, de modo realizar o processo de autenticação. No fim deste processo cada nó, ao possuir a chave de grupo, consegue distinguir entre mensagens originadas por nós pertencentes ao grupo e mensagens falsificadas.

3.2.4 Mensagens de alarme

Para as mensagens de alarme foi definido que para além de garantir as propriedades já referidas na subsecção referente à comunicação segura, seria necessário obter uma confirmação de receção da mensagem por parte da unidade central. Para tal a unidade central ao receber cada mensagem de alarme terá de enviar de retorno uma mensagem para o nó emissor a confirmar a receção do alerta. O nó emissor original ao não receber a mensagem de retorno dentro de um determinado período de tempo irá repetir o envio do alerta. Assim fica garantido que mesmo em caso de ocorrência de erro, ou caso algum nó esteja a bloquear ou corromper o tráfego, as mensagens perdidas serão reenviadas. Temos assim a garantia, no fim do processo, que a mensagem de alarme chegou pelo menos uma vez à unidade central.

3.2.5 Múltiplos caminhos

Para além das medidas de prevenção, para a proteção das mensagens, serão utilizadas também técnicas de resiliência, de modo a que o sistema tenha a capacidade de resistir a ataques internos à rede. Este tipo de ataque pode ser, por exemplo, um nó comprometido que efetue ataques como o *balckhole* ou encaminhamento seletivo. Para resistir a estes ataques é necessário que cada nó, ao encaminhar tráfego para o *gateway*, não use sempre o mesmo trajeto. A ideia será que cada nó tenha um conjunto de nós vizinhos especial, e ao encaminhar cada mensagem escolha o próximo nó com base neste conjunto, alterando regularmente a escolha. Deste modo o tráfego terá múltiplos caminhos, evitando possíveis nós atacantes, e obtém-se uma melhor distribuição do tráfego pela rede, deixando de sobrecarregar sempre os mesmo nós intermédios.

3.3 Descrição da implementação

Ao longo desta secção é especificado de que modo foi implementada a solução idealizada. A secção é divida em seis subsecções: na primeira é explicado o esquema de criptografia de chave pública usado; na segunda é detalhado o protocolo de autenticação; na terceira subsecção é apresentado o esquema que garante a segurança das mensagens enviadas por cada nó; na quarta subsecção é explicado de que modo são garantidas a autenticidade de integridade das mensagens de comunicação em grupo; na quinta subsecção é explicada o tratamento especial dado às mensagens de alarme; e por fim na última subsecção é explicada técnica que faz com que os nós alterem entre os vários caminhos, no envio de mensagens para os nós *gateway* de cada rede.

Durante a realização deste trabalho não foi possível estabelecer a comunicação no sentido da unidade central para os vários nós *gateway*. Pelo que é apenas possível a comunicação no sentido de cada um dos nós *gateway* para a unidade central. Deste modo a solução original teve de ser alterada. O protocolo de autenticação não pôde ser gerido pela unidade central, pelo que o mesmo ocorre entre cada nó e o nó *gateway* da rede. Na unidade central foi implementado o funcionamento necessário de modo que esta possa realizar os processos de autenticação, e validar as mensagens recebidas, mas

como explicado não é ainda possível implementar na totalidade a solução idealizada.

Assim, para já, o protocolo de autenticação é gerido pelo nó *gateway* de cada rede, e como a chave de sessão estabelecida é apenas partilhada entre o *gateway* e cada nó, cada rede apenas pode possuir um nó *gateway*.

Esta limitação não tem influência na implementação dos restantes pontos da solução, na perspetiva de cada um dos nós. A chave de sessão estabelecida, no protocolo e autenticação realizado com o *gateway*, é usada para garantir a segurança das mensagens enviadas para o mesmo. Quanto à comunicação em grupo, as mensagens de *routing* continuam a ser seguras recorrendo à chave de grupo, recebida por cada nó no fim do processo de autenticação. As mensagens de alarme são enviadas para o *gateway*, e este envia a mensagem de retorno a confirmar a receção. Quanto ao envio das mensagens por múltiplos caminho o processo, funciona de igual forma.

Embora esta limitação não permita implementar a solução desenhada, na perspetiva da avaliação experimental esta alteração não irá afetar os resultados uma vez que o trabalho computacional realizado por cada nó é idêntico ao necessário pela solução desenhada.

3.3.1 Criptografia de chave pública

O protocolo de autenticação é realizado recorrendo a uma biblioteca de certificados digitais, *simple ECC certificate library*, baseada na técnica de criptografia de curvas elípticas, [1]. Esta biblioteca foi desenvolvida de modo a permitir o seu uso em redes de sensores, tentando reduzir a capacidade de memória e computacional requeridas. Como exemplo, os certificados usados são bastante simples, não possuindo vários dos campos presentes nos certificados X.509, como por exemplo a data de validade. Esta biblioteca oferece as operações básicas para a criptografia de chave pública como são: geração de certificados, geração de assinaturas, usando o método *ECDSA*; validação deste tipo de assinaturas. Esta operações são usadas para assinatura dos certificados e para a validação dos mesmos. Biblioteca disponibiliza ainda o protocolo *ECDH* (*Elliptic curve Diffie–Hellman*) usado para o estabelecimento de uma chave simétrica entre dois elementos.

Como em qualquer esquema de criptografia assimétrica existem dois tipos de chaves:

- Chave privada - Chave que deverá ser conhecida apenas pelo nó e usada para gerar assinaturas, de modo a garantir a sua autenticidade.

- Chave pública - Chave que é partilhada com restantes elementos no processo de autenticação, de modo a que estes validem as assinaturas geradas pela chave privada.

Esta biblioteca, como referido, permite o uso de certificados assinados digitalmente. Um certificado é usado para associar uma entidade a uma chave pública. O uso de certificados é essencial pois é um modo de garantir que a chave pública que um nó diz possuir é efetivamente a sua, impedindo que um nó ofereça a sua chave pública mas tentando-se fazer passar por outro nó.

[1] https://github.com/nist-emntg/ecc-light-certificate

A escolha entre esta e outras bibliotecas de criptografia assimétrica baseada em curvas elípticas, deveu-se essencialmente ao facto de esta possuir um esquema básico de certificação digital. Algo que não acontece nas restantes bibliotecas analisadas, como por exemplo em, [2] e [24].

Cada certificado de chave pública é constituído por:

- Chave pública.
- *Hash* da chave pública da entidade que assina o certificado.
- Resultado da assinatura do certificado por parte da entidade assinante.

Na implementação deste sistema cada nó, quando instalado no sistema, possui:

- Chave privada, 18 *bytes*.
- Certificado, com a sua chave pública, assinado pela autoridade de certificação da empresa, 105 *bytes*.
- Um certificado de confiança, que possui a chave pública da autoridade de certificação da empresa, também de 105 *bytes*. Este certificado é usado para validar a assinatura presente nos certificados públicos que um nó recebe no processo de autenticação. Para validar um certificado que possua a chave pública de outro nó, o nó usa a chave pública do certificado de confiança de modo a verificar se assinatura presente no certificado público recebido foi efetivamente gerada pela chave privada correspondente ao certificado de confiança que este possui.

Ou seja, em apenas 228 *bytes* cada nó guarda toda a informação requerida para a gestão de chaves do esquema de criptografia assimétrica.

A autoridade de certificação é a entidade responsável por criar os pares de chaves públicas e privadas para cada um dos nós da rede, e é responsável também por gerar os certificados de cada nó e assinar os mesmos. Ou seja, é usado um modelo de certificação hierárquico de monopólio, pois existe apenas uma entidade raiz de toda a cadeia de certificação. A chave pública desta entidade está presente no certificado de confiança que todos os elementos do sistema possuem.

De modo a distinguir um nó normal de um nó *gateway*, a biblioteca foi alterada passando esta informação a estar incluída em cada certificado. Ou seja, no processo de assinatura de um certificado público, esta informação é também usada. Assim caso um atacante consiga obter o certificado de um nó de uma rede, não se conseguirá fazer passar por um nó *gateway* perante outros nós, pois ao validar o seu certificado os restantes nós irão detetar que o mesmo não é um *gateway*. Em caso de alteração deste parâmetro no certificado público, tal será também detetado pelos restantes nós, pois a assinatura do certificado, realizada pela entidade de confiança, será inválida.

3.3.2 Autenticação na rede

O estabelecimento de uma chave simétrica entre nó e gateway é realizado recorrendo ao protocolo ECDH (Elliptic curve Diffie–Hellman), representado na figura 3.4. Este protocolo tal como o protocolo

[2] https://github.com/iSECPartners/nano-ecc

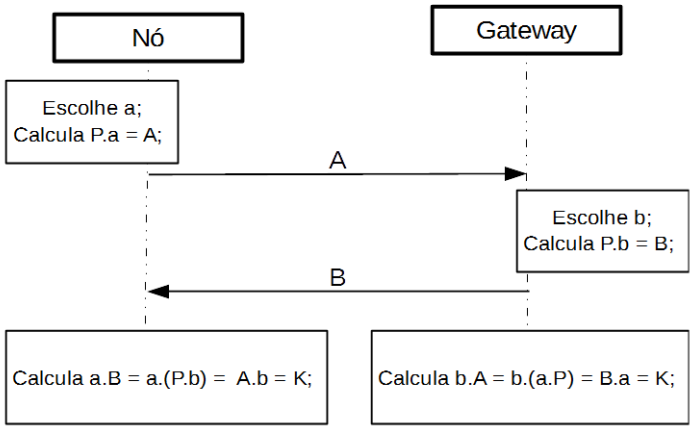

Figura 3.4: Protocolo *ECDH* (*Elliptic curve Diffie–Hellman*).

Diffie-Helman permite que duas partes desconhecidas estabeleçam uma chave secreta sob um canal de comunicação inseguro. Esta chave é usada para cifrar posteriores mensagens, usando um esquema de cifra de chave simétrica.

O protocolo é constituído pelas seguintes fases:

- Os Intervenientes escolhem de modo aleatório um inteiro (a e b) (chave privada efémera).
- Cada elemento multiplica o inteiro escolhido anteriormente pelo ponto gerador P, ponto partilhado por todos os elementos do sistema, e obtêm respetivamente os elementos A e B ($A = a.P, B = b.P$) (chaves públicas efémeras).
- Intervenientes trocam os respetivos pontos calculados (A e B).
- Cada elemento multiplica o ponto recebido do restante com o seu inteiro escolhido inicialmente($a.B$, $b.A$).
- Resultado é idêntico pois $a.B = a.(b.P) = b.(a.P) = b.A$, esta valor é usado para derivar a chave simétrica.

Cada elemento expõe apenas a sua chave pública efémera, pelo que qualquer atacante de modo a descobrir a chave simétrica obtida, teria de resolver o problema do logaritmo discreto em curvas elípticas. Este problema consiste de modo simplificado em descobrir o número inteiro (a) que multiplicado pelo ponto gerador P resulta no parâmetro público A.

Por si só, um protocolo de autenticação baseado na simples troca destas mensagens não é seguro, pois este pode ser vítima de por exemplo o ataque de homem-no-meio (*man-in-the-middle*). Deve por isso ser garantida a autenticidade das mensagens trocadas entre os elementos. É necessário também implementar técnicas que impeçam a repetição de mensagens, para que um atacante, ao repetir uma mensagem anterior de outro nó, não consiga fazer com que o *gateway* realize o protocolo desnecessariamente. Assim é impedida a realização de um ataque que tem como objetivo desgastar

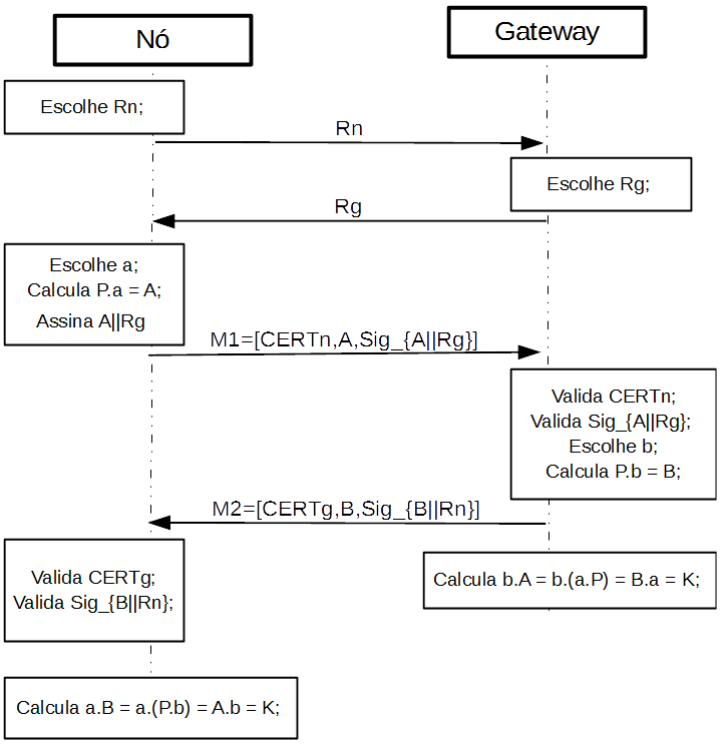

Figura 3.5: Protocolo de autenticação mútua implementado.

apenas os elementos, fazendo com que estes realizem trabalho computacional desnecessário. Para tal o protocolo foi alterado, tal como está representado na figura 3.5.

De modo a garantir a autenticação das mensagens trocadas, estas são assinadas pela chave privada do elemento de origem e enviadas em conjunto com o correspondente certificado. Assim um nó ao receber a mensagem verifica a validade da chave pública utilizando o certificado de confiança e posteriormente verifica se a mensagem foi devidamente assinada pela chave privada, par da chave pública presente no certificado.

Para impedir o ataque de repetição destas mensagens foi adicionada uma fase anterior a esta onde os elementos apenas trocam números aleatórios entre ambos. Assim a cada pedido de autenticação numa fase inicial o *gateway* recebe apenas um número aleatório (R_n) e responde de igual forma, apenas com um número aleatório (R_g). Na fase posterior o *gateway* espera que o nó assine a mensagem do protocolo *ECDH* com este número, garantindo assim que esta mensagem corresponde ao pedido iniciado anteriormente.

Com estas alterações numa fase inicial é diminuída a expectativa de ataques de negação de serviço ao *gateway*, pois este apenas terá de gerar um número aleatório. São evitados também ataques de repetição de processos de autenticação pois cada nó terá de responder com o número aleatório escolhido pelo *gateway*. Ou seja, um atacante apenas poderá repetir mensagens de autenticação entre os dois elementos quando o *gateway* repetir o número aleatório pedido. Procedimento idêntico ocorre também no sentido inverso. Assim, um atacante não poderá repetir mensagens antigas de resposta do *gateway*, pois o nó espera por uma assinatura sobre o número aleatoriamente por si gerado anterior-

mente.

A posterior comunicação entre cada nó e o *gateway* da rede será realizada, usando a chave simétrica estabelecida de modo a garantir a segurança da mesma, como será explicado na próxima subsecção. No fim do protocolo de autenticação cada nó recebe, juntamente com a mensagem *M2*, a chave de grupo usada para a proteção da difusão das mensagens de *routing*. Esta chave esta presente na mensagem *M2*, devidamente cifrada usando a chave simétrica que cada nó irá derivar.

Detalhes da implementação

Dado o limite de 40 *bytes* para o tamanho máximo do conteúdo de cada mensagem trocada entre os nós da rede, as mensagens *M1* e *M2* do protocolo de autenticação são divididas em 5 sub-mensagens. O tamanho total de cada uma destas mensagens é de 177 *bytes*, divididos por: certificado com 105 *bytes*; parâmetro público de 36 *bytes*; assinatura com 36 *bytes*.

Os nós intermédios, a um nó e o *gateway*, no processo de autenticação não encaminham estas sub-mensagens assim que as recebem, estas apenas são encaminhadas quando um nó estiver na posse da sua totalidade. Deste modo, em caso de falha ou erro na transmissão de uma sub-mensagem as restantes sub-mensagens deixam também de ser propagadas, evitando assim transmissões desnecessárias.

Como referido, na modelação do problema, o fluxo de mensagens entre o *gateway* e um nó especifico da rede não existe. Pelo que foi necessário implementar um protocolo que permitisse ao *gateway* responder a cada nó no processo de autenticação. Para tal ao encaminhar mensagens do protocolo de autenticação para o *gateway*, cada nó intermédio guarda informação sobre o correspondente pedido e o endereço do nó vizinho de quem recebeu a mensagem. Assim, e começando pelo *gateway*, cada nó encaminha a mensagem de resposta pelo correspondente nó que encaminhou até si a mensagem do pedido inicial. Ao seja o trajeto da mensagem de resposta terá o caminho inverso ao da primeira mensagem, como ilustrado na figura 3.6. A informação guardada por cada nó intermédio, para a realização deste processo, tem um tempo de vida associado. Deste modo em caso de erro nós irão apagar a informação que deixou de ser necessária.

Como será explicado no ponto 3.3.6, para que a rede consiga resistir a ataques como o bloqueio de tráfego, foram implementados mecanismos que permitem o envio das mensagens por múltiplos caminhos. Mas neste caso para que as mensagens do processo de autenticação, enviadas por um nó, sejam encaminhadas para o mesmo *gateway*, e de modo a possibilitar a respostas por parte do *gateway*, para cada processo as mensagens são encaminhadas pelo mesmo trajeto. Ou seja, para o mesmo processo de autenticação todas as mensagens passam pelos mesmos nós intermédios.

3.3.3 Comunicação segura

Ao logo desta secção são explicados e analisados vários aspetos do esquema de cifra e cálculo do *MAC*, associado a cada mensagem, sendo no fim apresentado o esquema no seu todo. Este esquema é responsável por garantir a autenticidade, integridade e confidencialidade da comunicação entre cada

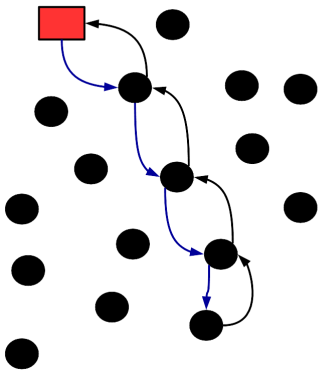

Figura 3.6: Protocolo de confirmação de receção das mensagens de alarme.

nó da rede e o *gateway*. Para tal é usada a chave simétrica calculada, como explicado anteriormente, através de um protocolo de autenticação mutua.

Cifra por blocos

A confidencialidade da comunicação é garantida cifrando o conteúdo de cada mensagem. Como é mais usual nos esquemas de segurança atuais, e mais recomendado, a cifra é feita recorrendo à cifra por blocos, usando chaves simétricas. Para tal é usado o algoritmo de cifra *AES*, definido em [38]. Esta foi uma escolha óbvia pois este é o algoritmo mais recomendado e mais usado atualmente.

O algoritmo *AES* usa blocos de 128 *bits* com três tipos de tamanhos diferentes de chaves (128 *bits*, 192 *bits* e 256 *bits*). Quanto maior a chave usada maior o tempo de computação da cifra e por consequência maior o consumo energético. Por essa razão, para este sistema, são usadas as chaves de 128 *bits*. De referir que este tamanho de chaves simétricas, 128 *bits*, é considerado seguro e recomendado [39] para aplicações *near term*, com um prazo de 10 anos.

Modo de cifra

O modo de cifra estabelece o modo como o algoritmo de cifra é usado para cifra de mensagens que têm tamanho arbitrário, nomeadamente mensagens com conteúdo superior ao tamanho de cada bloco (128 *bits*, 16 *bytes*). Neste caso o modo de cifra *CBC* (*Cipher Block Chaining*) é o modo usado para a cifra das mensagens.

Este modo define que a cada bloco de texto em claro a cifrar é previamente aplicada uma função XOR com o bloco anterior do criptograma, figura 3.7. Desta forma, cada bloco cifrado fica dependente de todos os blocos de texto anteriores. Para que cada mensagem seja única, mesmo quando é realizada a cifra de textos idênticos, é usado o vetor de inicialização (*initialization vector*, *IV*) como realimentação para o primeiro bloco.

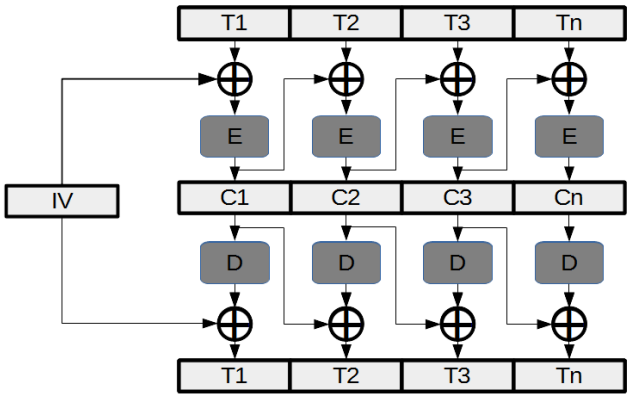

Figura 3.7: Modo de cifra *CBC*. Ti e Ci representam blocos do texto e do criptograma respetivamente, E é a função de cifra e D é a função de decifra

Vector de inicialização (IV)

Como referido, no capítulo anterior, garantir a segurança semântica implica que a cifra de duas mensagens idênticas resulte em criptogramas distintos. O uso de um *IV initialization vector* tem esse mesmo objetivo, sendo este usado para adicionar entropia ao algoritmo de cifra.

O *IV* pode ser visto como parte do $input$ usado pelo algoritmo de cifra que deve ser usado de um modo único, ou seja, deve ser distinto por cada mensagem cifrada. Como o recetor das mensagens necessita de conhecer o *IV* usado na cifra de cada mensagem, o mesmo é normalmente enviado em claro junto com a mensagem. O vetor de inicialização pode ou não ser secreto, a segurança de um modo de cifra não depende do conhecimento do *IV*, pois este é usado para introduzir entropia ao criptograma.

Usualmente o *IV* é aleatório, e é enviado juntamente com cada mensagem. Como este processo incrementa o tamanho de cada mensagem em 16 *bytes*, tamanho de um *IV*, foi necessário de recorrer a outra solução. Foi decidido diminuir o tamanho do *IV* e não usar um *IV* aleatório, mas sim um número sequencial. Este número é o número de sequencia de cada mensagem, constituído por 2 *bytes*, e que já era enviado juntamente com cada mensagem. Para além desta abordagem, de modo a distinguir o *IV* usado pelos diferentes nós da rede, cada *IV* é também constituído por outros elementos já presentes em cada mensagem. Estes elementos são o endereço do nó de origem e o tipo de mensagem enviada. Deste modo não foi necessário aumentar o tamanho das mensagens enviadas, pois o *IV* usado é constituído por elementos já presentes em cada mensagem.

Como explicado, o *IV* neste caso não é aleatório, mas sim sequencial, e por isso previsível. Para permitir o uso de um *IV* previsível foi adicionado um passo extra ao modo CBC normal. Este passo consiste na pré-cifra do *IV*, usando AES e uma chave distinta da chave usada na cadeia CBC. Esta chave é derivada a partir de uma chave simétrica, partilhada entre os elementos da comunicação.

MAC (Message authentication code)

Esta técnica é usada de modo a garantir a autenticidade e integridade de cada mensagem trocada. O *MAC* funciona como um *checksum* calculado de forma criptográfica e segura, sendo para tal necessário que os elementos partilhem uma chave secreta simétrica.

Cada nó emissor calcula o *MAC* com base no conteúdo da mensagem e usando a chave partilhada. O *MAC* calculado é associado posteriormente à mensagem, sendo enviado juntamente com a mesma.

O recetor, o *gateway* neste caso, ao receber cada mensagem calcula um *MAC* de igual forma e compara-o ao recebido com a mensagem. Caso sejam idênticos significa que o conteúdo da mensagem não sofreu alterações e que foi usada a chave secreta partilhada pelo emissor e recetor. Neste caso a mensagem é aceite, caso contrário é descartada pelo recetor por não estar assegurada a sua autenticidade e integridade.

Para o calculo do *MAC* é usado o modo *CMAC* [40]. Este modo é baseado no modo de cifra *CBC* e por isso quase não acrescenta código à solução, e permite que com a apenas uma passagem pelo conteúdo da mensagem se realize a cifra e calculo do *MAC* associado à mesma. Para o calculo do *MAC* é também usado o *IV* da mesma forma como foi explicado anteriormente.

O *CMAC* foi apresentado para resolver problemas do *CBC-MAC*, nomeadamente quando é necessário acrescentar *padding* à mensagem. Para o uso do modo *CMAC* são necessárias mais duas chaves, *K1* e *K2*, sendo apenas usada uma por mensagem. Uma destas chaves é escolhida de modo exclusivo e é realizada a operação lógica disjunção exclusiva(XOR) juntamente com o último bloco do conteúdo da mensagem, como é ilustrado na figura 3.8. *K1* é usada quando o último bloco do conteúdo da mensagem está completo, ou seja, é composto exatamente pelos 16 *bytes*. Em contraponto *K2* é usada quando o último bloco não está completo e por isso contêm *padding* de modo a preencher o mesmo.

Estas chaves são derivadas a partir da chave simétrica partilhada, como será explicado no último ponto desta subsecção.

Padding

Como na maioria dos casos o conteúdo de uma mensagem não é múltiplo do tamanho de um bloco (128 *bits*, 16 *bytes*) é necessário acrescentar informação à mensagem de modo a que o conteúdo cifrado seja múltiplo do tamanho de cada bloco. Realizando assim o *padding* da mensagem.

A técnica de *padding* usada, na implementação deste sistema, consiste em antes da cifra preencher todos *bytes* em falta do último bloco com o valor do número de *bytes* em falta, seguindo a recomendação do RFC [41].

Exemplos:

$$xxxx|xxxx|xxxx|xx0202$$

$$xxxx|xxxx|xx0606|06060606$$

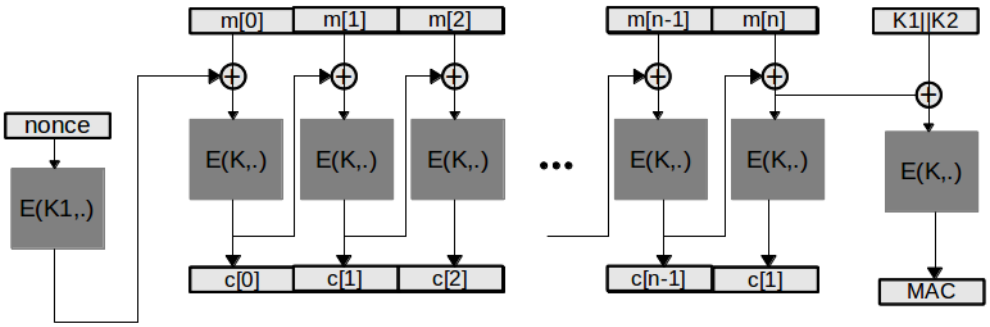

Figura 3.8: Esquema completo para a proteção das mensagens.

Resumo do esquema

Na figura 3.8 está representado o esquema completo, usado para garantir a autenticidade, integridade e confidencialidade de cada mensagem enviada por cada nó para o *gateway*.

O esquema utiliza o algoritmo de cifra de blocos *AES* sobre o modo de operação *CBC*. De modo a permitir a reutilização de código e uma maior eficiência, o modo *CMAC* é usado para o calculo do *MAC* associado cada mensagem. O uso do modo *CBC*, usado pelo modo de cifra e pelo modo *CMAC*, permite que com apenas uma passagem pelos dados estes sejam cifrados e que seja calculado o respetivo *MAC*.

Como referido é de notar a pré-cifra do *IV*, que permite que este seja previsível. Para tal é usada a chave *K1* e o algoritmo *AES*.

Como explicado o modo *CMAC* necessita de duas chaves adicionais, *K1* e *K2*, que são usadas de modo exclusivo em cada processo. *K2* é usada quando o último bloco *M[n]* contém *padding*. Por outro lado quando o último bloco é completo, é usada a chave *K1*. Este último caso é o menos provável, é não acontece neste sistema, sendo esta a razão pela qual a chave *K1* é também usada na pré-cifra do *IV*.

Derivação das chaves

O processo de derivação de chaves, permite derivar as chaves *K1* e *K2*, a partir de uma chave simétrica *K*. Neste caso a chave *K* pode ser a chave estabelecida, entre cada nó e o *gateway*, no processo de autenticação, ou pode ser a chave de grupo, recebida por cada nó no final do mesmo processo.

A função de derivação de chaves usada é definida em [42] como *KDF in Feedback Mode* (função de derivação de chaves em modo de realimentação).

De forma simplificada a função segue o seguinte modelo:

$$K1 = F(K, (CTX||0))$$

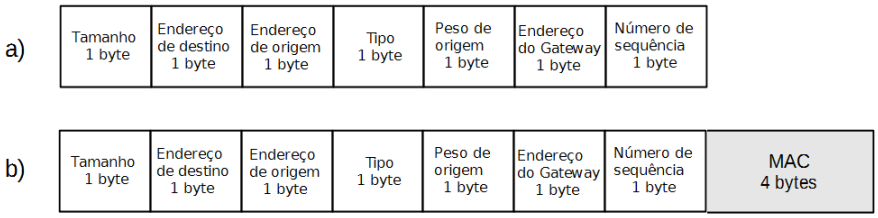

Figura 3.9: Mensagem de *routing* original, a), e mensagem de *routing* protegida, b).

$$K2 = F(K1, (CTX||1))$$

$$Kn = F(K_{n-1}, (CTX||n))$$

- *F* - Função pseudoaleatória, neste caso é usado o algoritmo de cifra *AES*, para 128 *bits*, de modo a reutilizar código.
- *K* - Chave original.
- *CTX* - Contexto, variável conhecida por todos os elementos do sistema, e que deve ser distinta entre as diferentes aplicações. Neste caso existe um contexto usado para derivar as chaves usadas para a comunicação entre nó e *gateway*, e outro para derivar chaves usadas na proteção das mensagens de grupo.
- *n* - número de chaves a derivar.

3.3.4 Comunicação em grupo

Neste momento o único tipo de mensagens que usam o paradigma de comunicação em grupo, são apenas as mensagens do protocolo de *routing*, mas a implementação aqui apresentada pode ser usada por outro tipo de aplicação.

As mensagens de *routing* passaram a ter garantias de autenticidade e de integridade. Para tal possuem um *MAC* calculado com base na chave de grupo, partilhada pelos vários elementos de uma rede, recebida por cada nó no fim do processo de autenticação. Como se verifica na figura 3.9 a única alteração que estas mensagens sofreram foi o acrescento do *MAC*, de 4 *bytes*, a cada mensagem trocada entre os nós da rede.

O *MAC* é calculado recorrendo à técnica *CMAC*, explicada anteriormente. O *IV* usado é constituído pelo número de sequência da mensagem, pelo endereço de origem do *gateway* que iniciou o processo, e pelo endereço do nó intermédio que difunde a mensagem.

O *IV* é constituído por estes três elementos de modo a garantir a unicidade do mesmo, evitando a sua repetição. Por cada ronda de difusão o número de sequencia é o mesmo, mas com o uso do endereço do nó que faz a difusão o *IV* usado por cada um dos nós da rede é diferente. Por fim o uso do endereço do *gateway*, que iniciou o processo, faz com que cada *IV* seja único mesmo quando a rede possui vários nós *gateway*.

As chaves extra, *K1* e *K2*, necessárias ao funcionamento do *CMAC* são derivadas da forma apresentada na subsecção anterior, a partir da chave de grupo.

Cada nó que participa na rede ao receber cada mensagem, do protocolo de *routing*, apenas a aceita após validar o *MAC* que cada uma tem associado. Para tal, o nó recetor calcula o *MAC* para a respetiva mensagem tendo em conta o seu cabeçalho, conteúdo e recorrendo à chave de grupo. Comparando o *MAC* calculado com o recebido cada nó valida a mensagem de *routing* recebida. De seguida ao propagar a mensagem de *routing*, nó calcula de igual forma o respetivo *MAC* para associar à mesma.

3.3.5 Mensagens de alarme

Para as mensagens de alarme foi definido que, para além de garantir as propriedades de segurança idênticas das restantes mensagens, é necessário também resistir a ataque internos, como o bloqueio de tráfego. Para tal o *gateway* ao receber cada mensagem, deste tipo, envia de retorno uma mensagem para o nó emissor a confirmar a receção do alerta. Esta mensagem de retorno está também protegida quanto à sua confidencialidade, integridade e autenticidade de modo idêntico às restantes mensagens, usando a chave partilhada pelo nó e *gateway*.

O nó emissor do alarme espera pela mensagem de confirmação de receção um determinado tempo t. Este tempo é dependente do número de saltos intermédios a que o mesmo se encontra do *gateway*, quanto mais saltos mais tempo de espera. Caso não receba a mensagem de retorno dentro deste tempo t, o nó reenvia o alerta. Este processo repete-se até que o nó receba a mensagem de confirmação por parte do *gateway* da rede. Como será explicado no próximo ponto, o reenvio da mensagem deverá fazer com que o caminho usado seja distinto do anterior.

De modo a possibilitar o envio da mensagem de resposta por parte do *gateway*, é usado o esquema idêntico ao implementado para possibilitar a execução do protocolo de autenticação, como explicado em 3.3.2. Ou seja, como representados na figura 3.6, o caminho da mensagem de confirmação é o inverso ao de envio da mensagem de alerta alerta.

3.3.6 Múltiplos caminhos

A opção de encaminhar o tráfego usando vários nós vizinhos e não escolhendo sempre o mesmo tem dois objetivos. O primeiro é não sobrecarregar sempre os mesmos nós intermédios, de modo a realizar uma divisão do tráfego pela rede mais equitativo. O segundo objetivo é capacitar a rede com a capacidade de resiliência, de modo a que o protocolo consiga resistir a ataques internos da rede. Este tipo de ataques, pode ser por exemplo um nó comprometido que efetue ataques como o bloqueio de tráfego ou o encaminhamento seletivo, ou um ataque de *jamming* localizado apenas em parte da rede.

Na solução implementada cada nó seleciona um conjunto máximo três nós vizinhos preferíveis. Estes nós vizinhos são os que se encontram a menos saltos (*hops*) para o *gateway*, e que não ofereçam uma diferença superior a 1 valor, em relação ao melhor nó, o que oferece um menor número de saltos. Ao encaminhar cada mensagem, cada nó varia a escolha do próximo nó, com base nesta lista de vizinhos de forma sequencial.

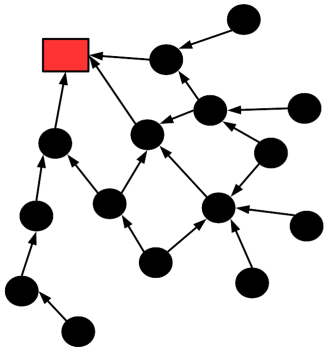

Figura 3.10: Envio de mensagens por múltiplos caminhos.

Neste momento fazem parte desta lista nós que não ofereçam uma diferença superior a 1 valor, em relação ao melhor nó vizinho. Mas este valor é configurável, podendo mesmo ser zero, de modo a que cada nó encaminha tráfego apenas pelos nós vizinhos mais próximos do *gateway*, e que se encontram a um número de saltos idêntico, como representado na figura 3.10.

3.4 Resumo do capítulo

Ao longo deste capítulo foi descrito em maior detalhe o problema associado a este trabalho e foi apresentada a solução implementada para resolver os problemas de segurança do mesmo.

Como referido o problema apresentado é um problema típico de redes de sensores, que apresenta algumas especificidades como é o caso da capacidade de mobilidade dos nós do sistema.

Na solução apresentada a segurança da comunicação é baseada no uso uma chave simétrica partilhada entre cada nó e a unidade central, *gateway* na corrente implementação. Esta chave é obtida realizando um protocolo de autenticação entre os elementos, recorrendo a uma biblioteca de criptografia assimétrica que usa a técnica de criptografia de curvas elípticas.

Como referido no capítulo anterior o uso de criptografia assimétrica é muitas vezes considerado inadequado para as redes de sensores. No entanto, no ambiente considerado o uso deste tipo de criptografia é viável.

As mensagens enviadas por cada nó para a unidade central, estão seguras quanto à sua autenticidade, integridade e confidencialidade, usando mecanismos de criptografia recomendados, como é o caso do modo de cifra *CBC* e da técnica *CMAC*.

Técnicas como o reenvio das mensagens de alarme, e o envio de mensagens por múltiplos caminhos oferecem à rede a capacidade de resiliência, pois permitem que a rede seja resistente a ataques internos, como o bloqueio de tráfego, nós com comportamento erróneo, ou mesmo a ataques de *jamming* localizados em parte da rede.

As mensagens do protocolo de *routing* são enviadas em claro, sendo apenas garantidas as propriedades de autenticidade e integridade, para tal é usada uma chave simétrica partilhada apenas pelos nós do mesmo grupo, devidamente autenticados.

Capítulo 4

Avaliação Experimental

Neste capítulo são apresentados os resultados dos testes realizados que tentam responder a duas questões essenciais, “Qual o impacto provocado pela solução em cada nó” e “Tendo em conta esse impacto é aceitável o seu uso?”. Os testes realizados tentam analisar o impacto da solução essencialmente em três aspetos, memória usada, tempo de execução acrescentado e análise ao consumo de bateria.

O capítulo está dividido em quatro secções: na primeira é explicada e analisada a forma como foram realizados os testes ao consumo energético; na segunda secção são realizados testes ao ponto essencial e mais problemático da solução, o uso da biblioteca de criptografia de curvas elípticas; na terceira secção é analisado o impacto da criptografia usada para proteção das mensagens; por fim na quarta secção é analisado o impacto geral da solução.

4.1 Consumo de bateria

As baterias usadas por cada nó têm a capacidade de 850 mA/h (miliampere/hora). Cada nó tem a possibilidade de a cada momento obter a percentagem de bateria que possui, com uma granularidade de 1%. Esta informação é usada pelo sistema, o seu valor é também enviado para a unidade central, mas com uma frequência muito menor do envio da informação de localização. Foi com este valor que foram também realizados os testes ao consumo energético, que são apresentados ao longo deste capítulo.

Antes da realização dos testes ao consumo energético de cada operação criptográfica, foi analisado o grau de precisão desta medida. Para tal foi executado um programa com execução linear, captura periódica da posição de *GPS*, e foram feitas medições à percentagem de bateria, com uma frequência de quinze minutos. Foi obtido o gráfico apresentado na figura 4.1.

Como se verifica o decréscimo da percentagem de bateria não tem uma representação linear. Se no inicio a percentagem decresce de forma, aproximadamente, linear, para valores medianos o decréscimo é mais lento, enquanto que para valores próximos de zero o decréscimo é mais acentuado.

Tendo como base esta análise os testes ao consumo da bateria foram realizados sobre a mesma

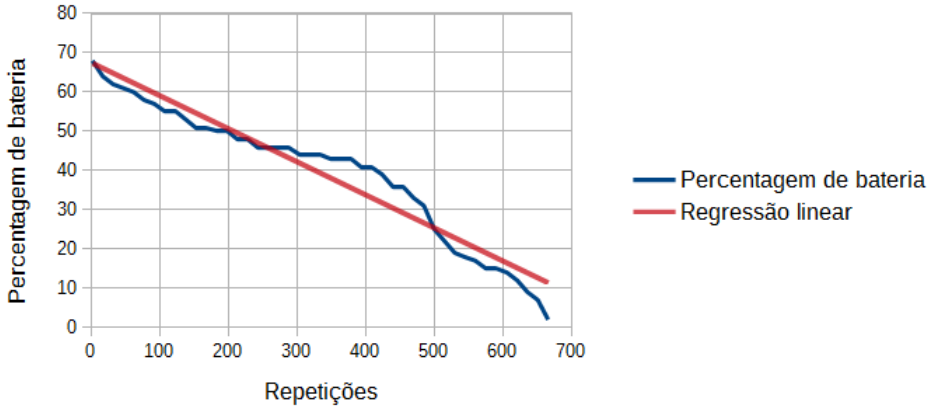

Figura 4.1: Variação da percentagem de bateria ao longo do tempo.

gama de percentagens e preferencialmente para valores mais elevados, de modo a obter resultados que pudessem ser comparáveis.

4.2 Criptografia de curvas elípticas

Nesta secção são apresentados os resultados dos testes realizados sobre o impacto do uso da biblioteca de criptografia de curvas elípticas. Um dos pontos de análise essencial era comparar a influência do tamanho das chaves assimétricas disponíveis pela biblioteca, 128 bits, 160 bits e 192 bits. Parte desta análise foi realizada antes da implementação do sistema, com o objetivo de selecionar o tamanho das chaves assimétricas que seriam usadas na implementação do sistema.

Numa primeira fase fez-se uma análise à memória requerida pela biblioteca de criptografia de curvas elípticas para os diferentes tamanhos de chaves disponíveis. Para tal foi verificada a quantidade de memória ROM e RAM usada pela biblioteca, para os três tamanhos disponíveis. Os valores apresentados no gráfico da figura 4.1 foram obtidos recorrendo ao *software*, *Code Composer Studio* [1], usado para a programação dos nós dos sistema.

Como se verifica na análise da tabela 4.1 e do gráfico presentado na figura 4.2, a variação da memória usada em relação ao tamanho da chaves assimétricas usadas, tem uma maior relevância na memória RAM requerida. Em contraponto a memória ROM, necessária para guardar código da biblioteca, não regista uma variação significativa. Em termos de memória RAM necessária observamos uma variação relevante, de 14,8% para 18,1%, quando comparamos o uso de chaves com 128 bits para chaves de 192 bits. Por outro lado, quanto à memória ROM necessária a variação é marginal, para os três tamanhos disponíveis o valor é sensivelmente idêntico (cerca 16,7%). Ou seja, a variação da memória usada é mais sentida na execução das operações do processo de autenticação do que pelo tamanho total do código.

Em termos globais estes resultados mostram que o uso desta biblioteca consome uma percentagem

[1] http://www.ti.com/tool/ccstudio

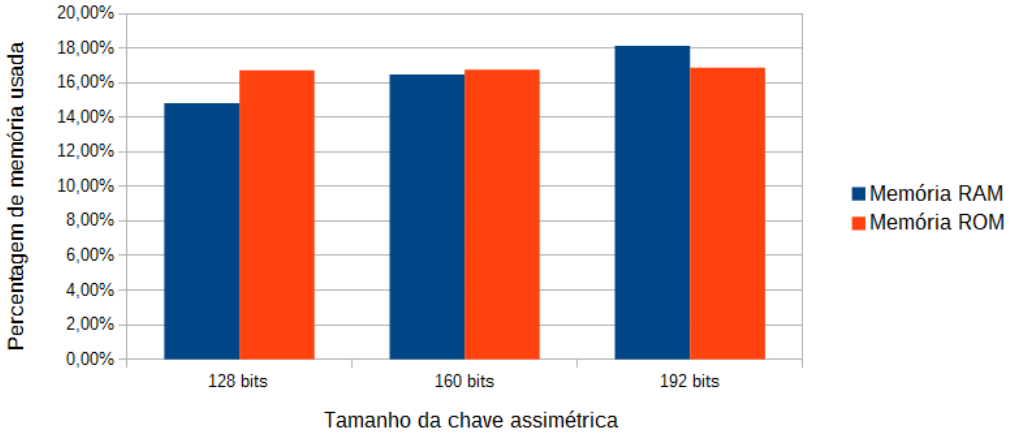

Figura 4.2: Variação da percentagem de memória usada para os três tamanhos de chaves assimétricas disponíveis.

muito relevante da memória total disponível, tanto em termos de memória RAM como de ROM.

Chave assimétrica	RAM	ROM
128 Bits	2.426 kB	21.907 kB
160 Bits	2.698 kB	21.933 kB
192 Bits	2.970 kB	22.087 kB

Tabela 4.1: Memória requerida, para os três tamanhos de chaves assimétricas disponíveis.

Em termos temporais foram realizados testes de modo a obter os tempos médios de execução para cada uma das operações do processo de autenticação, uma vez mais comparando os três tamanhos de chaves disponibilizados pela biblioteca. Os resultados são apresentados na figura 4.3 e nas tabelas 4.2, 4.3 e 4.4, por cada operação foram realizadas 40 repetições, para cada um dos tamanhos das chaves assimétricas.

	1.	2.	3.	4.	5.
Média	10,8406s	2,1725s	2,9149s	4,1196s	10,7919s
Erro padrão	0,0334s	0,0088s	0,0369s	0,0188s	0,0210s
Variância	0,0111s	0,00077s	0,0136s	0,003s5	0,0044s
Desvio padrão	0,1055s	0,0278s	0,1168s	0,0595s	0,0664s

Tabela 4.2: Dados estatísticos, para o uso de chaves assimétricas de 128 bits

	1.	2.	3.	4.	5.
Média	13,594s	2,9146s	3,5485s	5,6233s	13,6109s
Erro padrão	0,0437s	0,0171s	0,0581s	0,0236s	0,0337s
Variância	0,0191s	0,0029s	0,0338s	0,0056s	0,0114s
Desvio padrão	0,1381s	0,0539s	0,1838s	0,0746s	0,1066s

Tabela 4.3: Dados estatísticos, para o uso de chaves assimétricas de 160 bits

Como se verifica, as operações com uma execução mais demorada são a computação de assinaturas e, especialmente, a sua validação. De notar que neste caso, no protocolo de autenticação, a

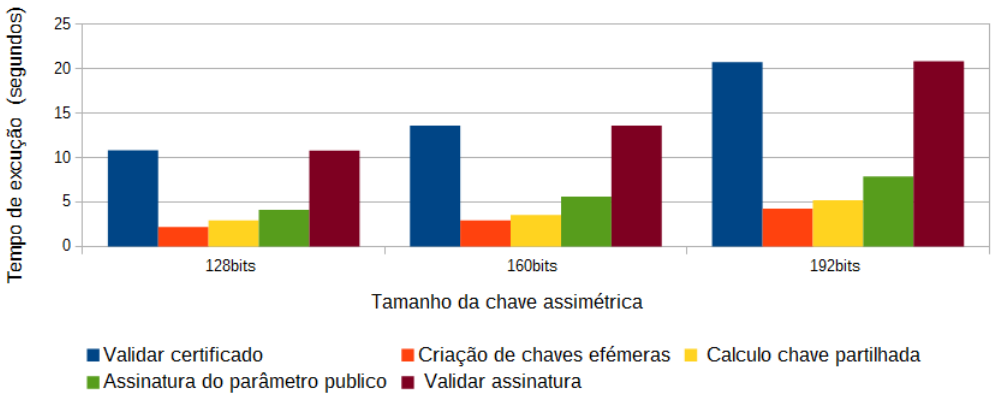

Figura 4.3: Variação do tempo médio de execução de cada uma das operações necessárias no protocolo de autenticação, para os diferentes tamanhos de chaves assimétricas.

	1.	2.	3.	4.	5.
Média	20,7639s	4,2361s	5,2071s	7,8898s	20,8430s
Erro padrão	0,0228s	0,0180s	0,0666s	0,0151s	0,0388s
Variância	0,0052s	0,0032s	0,0444s	0,0023s	0,0151s
Desvio padrão	0,0720s	0,0569s	0,2107s	0,0478s	0,1227s

Tabela 4.4: Dados estatísticos, para o uso de chaves assimétricas de 192 bits

validação de assinaturas ocorre em duas situações: validação da assinatura do parâmetro público do protocolo *ECDH*; e validação da assinatura da *CA*, que compõem o certificado público de cada nó.

Em contraponto com os resultados para a memória necessária, para os resultados do tempo de execução de cada operação é mais fácil notar a relevância do tamanho das chaves assimétricas usadas. Resultados mostram que a diferença entre o tempo médio de cada operação é sensivelmente o dobro, quando comparamos o uso de chaves assimétricas de 128 bits com chaves de 192 bits.

É de referir que os testes temporais foram realizados com o ciclo máximo de relógio possível, 8MHz, sendo esta a frequência usada para o processo de autenticação de modo a obter tempos de execução aceitáveis. Esta frequência é apenas usada para a realização destas operações, pelo que a frequência volta ao seu ciclo normal, 1MHz, após a sua realização.

Estes testes, como referido, foram realizados também, não de forma tão exaustiva, numa fase precedente à implementação do sistema, de modo a decidir qual o tamanho de chave assimétrica que seria usado. Dados os resultados obtidos, e dada a relevância essencialmente presente no tempo de execução de cada operação, optou-se pelo uso de chaves de 128 bits.

Por fim foram realizados testes ao consumo de bateria. Estes testes consistiram em medir a percentagem de bateria no intervalo da execução de blocos de 100 unidades de cada operação. Neste caso o teste foi apenas realizado para chaves assimétricas de 128 bits, tamanho das chaves assimétricas usadas. Dado o tempo necessário para a execução de cada operação esta experiência foi apenas realizada uma única vez.

Pelo declive das diferentes curvas obtidas, gráfico da figura 4.4, comprova-se que as operações mais demoradas, gerar e validar assinaturas, são as que provocam um maior desgaste na bateria. Mas

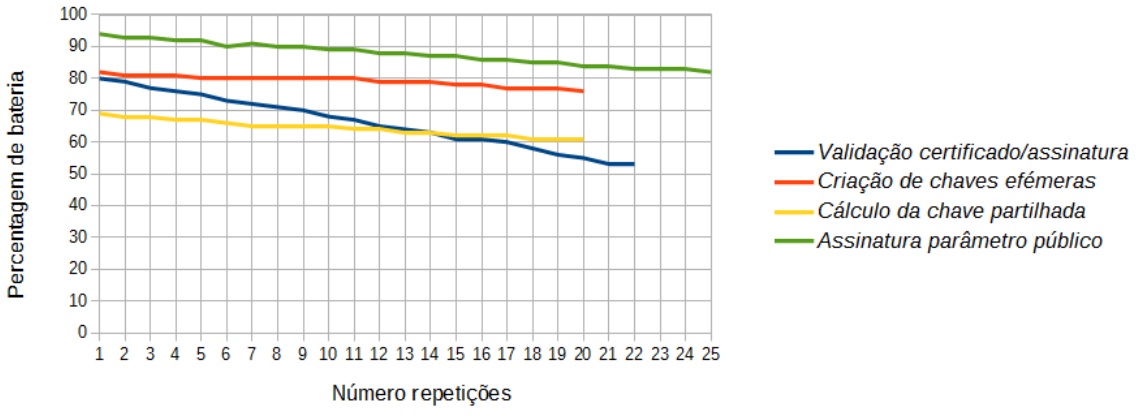

Figura 4.4: *Análise ao consumo de bateria imposto por cada operação.*

mais importante, os testes mostram o impacto da execução de cada operação. Quanto a este ponto verifica-se que o consumo de bateria é diminuto, pois por exemplo a execução de duas mil validações de assinatura, operação mais demorada, provocam apenas um consumo de 15% da bateria.

Como esperado, pelo estudo aos resultados das experiências realizados, se é verdade que o uso desta técnica de criptografia assimétrica tem um impacto significativo, principalmente em termos de memória requerida e tempo de execução, fica também provado que é possível e até aceitável a sua utilização, sempre em cenários em que o seu uso é escasso.

4.3 Comunicação segura

Nesta secção é analisado o impacto das operações realizadas para garantir propriedades como a autenticidade, confidencialidade e integridade para os dois tipos de mensagens trocadas pelos nós, mensagens de dados enviadas para o *gateway* da rede e mensagens do protocolo de *routing*.

Para garantir estas propriedades na comunicação entre cada nó e o *gateway* da rede, são executadas dois tipos de funções distintas, a cifra e cálculo de *MAC* e a decifra e validação do *MAC*. Quando um nó envia uma mensagem para o *gateway*, realiza a cifra da mesma e acrescenta à mensagem o *MAC* derivado. Como explicado, o processo de cifra e de cálculo do *MAC* são realizados por um método idêntico, ambos usam o modo *CBC*, pelo que estes processos são realizados em simultâneo.

O recetor de uma mensagem de modo a validar a sua autenticidade e integridade tem de realizar duas operações. Primeiro decifra a mensagem e de seguida valida o *MAC* associado à mesma. A decifra do conteúdo da mensagem consiste em realizar o esquema de modo inverso ao da cifra. Já para validar o *MAC* associado à mensagem o recetor executa a função de calculo do *MAC*, sobre o conteúdo da mensagem decifrada, e compara o *MAC* recebido juntamente com a mensagem, com o *MAC* por si calculado. Deste modo o recetor das mensagens tem de realizar um esforço superior ao do emissor da mensagem.

Em termos de memória requerida para realizar este tipo de operações, os testes mostram que a

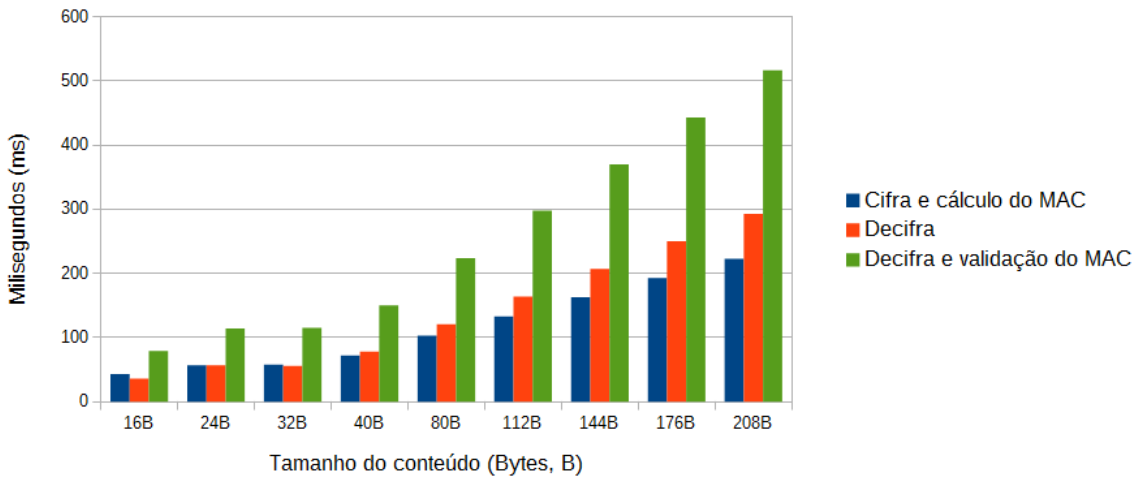

Figura 4.5: *Tempos de computação para as operações de cifra e calculo do MAC, decifra e validação do MAC, para diferentes tamanhos de conteúdo.*

memória RAM necessária é irrelevante (cerca de 50 *bytes*) enquanto a memória ROM requerida é cerca de 3kB, cerca de 3% da memória ROM total disponível. Valores que se consideram aceitáveis.

Quanto aos resultados temporais, ao analisar o gráfico da figura 4.5 observamos a influência do tamanho do conteúdo no tempo de computação de cada uma das operações, cifra e calculo do *MAC*, e decifra e validação do *MAC*.

O atual tamanho do conteúdo das mensagens da aplicação é no máximo de 14 bytes, neste caso os tempos de execução de cada uma operações são os seguintes:

- Cifra e calculo do MAC: 43 ms
- Decifra e Validação do MAC: 79 ms

Esta experiência foi repetida 40 vezes para cada um dos valores obtidos, tendo sido obtido um desvio padrão irrelevante para cada um dos resultados.

Quer para o tamanho atual das mensagens quer para um eventual aumento do tamanho das mesmas os resultados são aceitáveis. Como explicado, estas operações são realizadas para as mensagens enviadas por cada nó da rede para unidade central. Neste caso o nó emissor realiza e sua cifra e calculo do MAC, e o conjunto de operações mais pesadas, decifra e validação do *MAC*, são realizadas na implementação actual pelo *gateway*.

Quanto às mensagens do protocolo de *routing*, é apenas garantida a sua autenticidade e integridade associando para tal a cada uma um *MAC*. Ao receber cada mensagem o nó receptor valida o *MAC* recebido, e ao propagar a sua mensagem de *routing* cria um *MAC* para associar à mesma. O tempo necessário para um nó calcular um *MAC*, para este tipo de mensagens, é em média de 33 ms. Na difusão das mensagens do protocolo de *routing*, cada nó realiza esta operação duas vezes. Ao receber cada mensagem nó calcula o *MAC* sobre o conteúdo da mesma e compara-o com o *MAC* recebido, de modo a validar a mensagem. Ao propagar esta mensagem como cada nó altera o conteúdo da

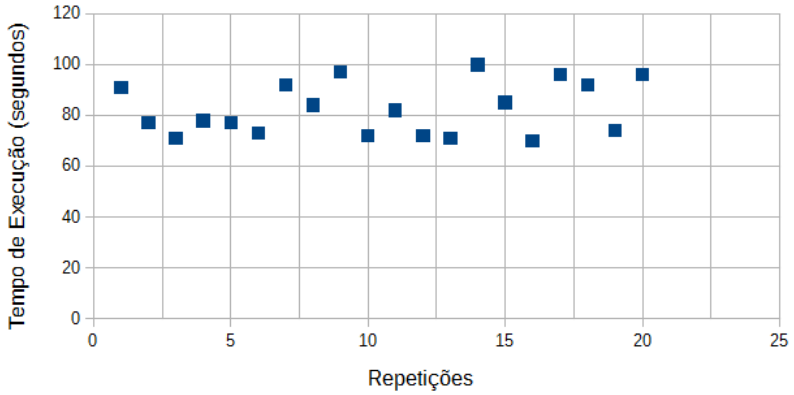

Figura 4.6: *Distribuição do tempo de execução do protocolo de autenticação, teste realizado entre nó e gateway com comunicação direta.*

mensagem o *MAC* é também recalculado. Este é o tipo de operação criptográfica que cada nó efetua um maior número de vezes, pois cada difusão de mensagens de *routing* por parte de cada *gateway*, para esta aplicação, tem a frequência de 3 minutos.

Notar, que neste caso, estas operações são realizadas com o ciclo de relógio a 1Mhz, a frequência normal de funcionamento de cada nó. Dado os resultados temporais obtidos e a alta granularidade para a medida do consumo de energético, é impossível obter resultados sobre o consumo de bateria, pois este para estas operações é marginal.

4.4 Análise global

Ao longo desta secção são apresentados e analisados os testes realizados que incidem sobre o funcionamento geral de toda a solução.

Um ponto de importante análise é o funcionamento do protocolo de autenticação. Para tal foram realizados testes ao tempo de execução e ao consumo de bateria impostos por este processo. O gráfico da figura 4.6 mostra a distribuição temporal para a realização do protocolo de autenticação. Processo realizado entre um *gateway* e um nó com comunicação direta. Na tabela 4.2 estão presentes as propriedades estatísticas mais relevantes obtidas para esta experiência.

Propriedades Estatísticas	Segundos
Média	82,5s
Mediana	80s
Desvio padrão	10,32s
Mínimo	70s
Máximo	100s
Intervalo	30s

Tabela 4.5: Análise ao consumo de bateria imposto pela execução do protocolo de autenticação.

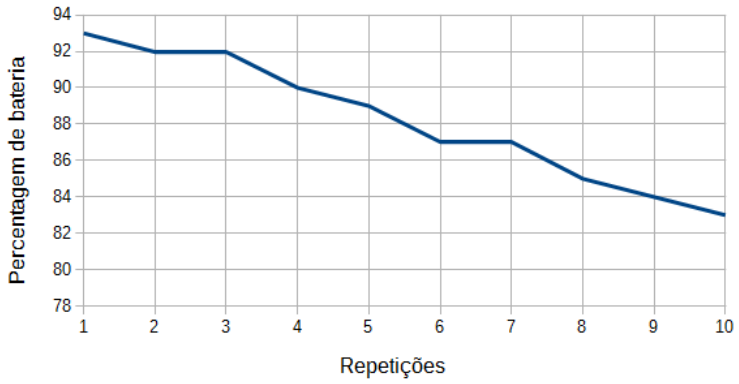

Figura 4.7: *Análise ao consumo de bateria imposto pela execução do protocolo de autenticação.*

Como se verifica os valores obtidos, para 20 repetições, apresentam uma intervalo de resultados significativo. O motivo para tal, está no facto de neste teste estar contabilizado o tempo de descoberta do *gateway*, tempo em que o nó já iniciou o processo, mas ainda não possui conectividade ao *gateway*. Por isso o valor a ter como referência para o tempo do protocolo de autenticação é o valor mínimo, pois este tempo é obtido quando a fase de descoberta é precedente ao inicio do protocolo de autenticação.

O valor obtido, 70 segundos, é um valor expectável tendo em conta os resultados temporais para a execução de cada operação do protocolo. Este tempo pode ser reduzido, pois neste caso dadas limitações técnicas já referidas, as mensagens *M1* e *M2* do protocolo são divididas em 5 sub-mensagens enviadas com um intervalo temporal de 1 segundo, limitação que deixará de existir numa futura versão do protocolo de comunicação.

Em termos gerais um acréscimo de 70 segundos não é significativo, uma vez que este processo é realizado apenas quando um nó entra pela primeira vez no sistema e não será necessária a sua repetição no tempo de vida expectável de um nó.

Os testes ao consumo energético imposto pelo protocolo de autenticação foram realizados de forma semelhante aos testes realizados para a execução individual de cada uma das operações. Com uma diferença, neste caso a percentagem de bateria é medida entre a execução de blocos de 50 execuções do protocolo em cada nó.

Tendo em conta os resultados obtidos na execução de cada uma das operações individualmente, os resultados, apresentados no gráfico da figura 4.7, eram expectáveis. O consumo é significativo, no total das 500 execuções a percentagem de bateria desce cerca de 10%.

Estes resultados reforçam a necessidade de os nós *gateway* não poderem estar limitados em termos energéticos, pois estes terão de realizar este processo, pelo menos uma vez, para cada um dos nós da rede. Em contraponto os resultados mostram que para os restantes nós a execução de um número reduzido de vezes do protocolo de autenticação não desgasta significativamente a sua bateria.

Quanto à largura de banda, o impacto imposto pela solução é o seguinte:

- Cada processo de autenticação acrescenta na rede a troca de cerca de 500 bytes, divididos pelas várias mensagens.

- A cada mensagem de dados enviada para o *gateway*, são acrescentados em média 8 bytes, 4 para o *MAC* associado a cada uma, e 4 de modo a que o conteúdo da mensagem seja múltiplo de 16 bytes.

- Para cada mensagem do protocolo de *routing* é acrescentado um *MAC* de 4 bytes.

Comparando com os tamanhos anteriores para cada tipo de mensagem, as mensagens de dados têm um aumento médio de 42%, enquanto que as mensagens de *routing* tiveram um aumento de 57%. Dada a alta granularidade dos testes ao consumo de energético, não é possível quantificar o consumo energético associado a este incremento nos tamanhos de cada mensagem.

No caso da aplicação em causa, é de referir que em termos energéticos a recolha da informação de localização tem um impacto muito mais efetivo no tempo de vida de um nó do que os custos inferidos pela solução. Como exemplo o gráfico apresentado na figura 4.1, foi obtido com apenas 12 horas de execução.

A título de exemplo foram realizados cálculos teóricos para uma aplicação ideal em que a obtenção de dados não requeria o consumo de energia. Os cálculos mostram que a influência da solução de segurança é diminuta. O tempo de vida de um nó seria reduzido em cerca de 15 dias. Valor pouco significativo, pois representa uma redução de cerca de 4% do tempo de vida expectável, de 415 para 400 dias.

4.5 Resumo do capítulo

Neste capítulo foram apresentados os resultados dos testes realizados sobre a implementação da solução proposta para garantir a segurança da comunicação do sistema associado a este trabalho.

Os testes realizados analisaram o impacto da solução essencialmente em três aspetos: memória usada, tempo de execução acrescentado e análise ao consumo de bateria.

Na segunda secção os testes realizados ao uso da biblioteca de certificados digitais baseada em criptografia de curvas elípticas confirmaram o impacto significativo imposto pelo uso desta técnica. Este impacto é mais significativo em termos de memória requerida para o seu funcionamento e em termos temporais, nomeadamente para o tempo de execução de cada uma das operações oferecidas pela biblioteca. Relativamente ao consumo de bateria verificou-se que o mesmo é diminuto tendo em conta a execução individual de cada uma das operações. Ficou assim provado que é possível e até relativamente aceitável a utilização desta técnica, sempre em cenários em que o seu uso é escasso.

De seguida na terceira secção foi analisado o impacto da criptografia usada para proteção da comunicação. Para este ponto os testes mostram que quer em termos de memória requerida, quer em tempo de execução, o impacto é diminuto.

Por fim na quarta secção foi analisado o impacto geral da solução.

Capítulo 5

Conclusão

As redes de sensores em malha são um tipo específico de dentro do conjunto de redes moveis. Este tipo de redes consistem em pequenos dispositivos, distribuídos de modo a possibilitar comunicação entre os elementos, com a capacidade de recolher informação do ambiente que os rodeia e de monitorizar eventos e objetos.

Os dispositivos normalmente usados em redes de sensores possuem limitações a nível capacidade de memória, de energia e de processamento. Estas limitações em conjunto com o uso de um meio de comunicação sem fios, e a falta de vigilância sobre os sensores posteriormente à sua distribuição no meio, tornam as redes de sensores suscetíveis a vários tipos de ataques.

Com uma enorme gama de aplicações possíveis, a falta de garantias para a segurança da comunicação é um facto que pode impedir o uso das redes de sensores, sendo por isso este um campo muito relevante no desenho de uma rede deste tipo.

Dadas as limitadas capacidades dos dispositivos usados várias são as medidas de segurança tradicionais por vezes impossíveis de usar em redes de sensores. Uma solução de segurança para este tipo de redes deve ser ponderada entre a sua eficácia e a sua eficiência, relativamente ao consumo de recursos dos dispositivos.

Este trabalho teve como objetivo resolver o problema da segurança da comunicação para um sistema que tem como fim monitorizar a localização geográfica de objetos móveis.

Este trabalho propôs e implementou uma solução que garante a segurança da comunicação da rede, quer a comunicação ponto a ponto, entre cada nó e a unidade central, quer comunicação em grupo, nomeadamente garantindo o correto funcionamento do protocolo de *routing* usado. A solução implementada usa ainda mecanismos que oferecem à rede a capacidade de resiliência a certos ataques.

O processo que garante a autenticação de cada um dos elementos do sistema e que efetua a distribuição de chaves é feito recorrendo a uma técnica de criptografia assimétrica, criptografia de curvas elípticas.

Por isso este trabalho é também um teste sobre a possibilidade de uso desta técnica, em redes de sensores com características semelhantes às do problema em questão.

Os testes realizados mostraram que o uso de criptografia assimétrica, nomeadamente utilizando a técnica de criptografia baseada em curvas elípticas é possível, quando usada de modo limitado. Como é o caso da solução apresentada, onde esta técnica é usada apenas para o processo de autenticação e distribuição de chaves.

Com as futuras melhorias das capacidades dos dispositivos usados em redes de sensores, é de esperar que a técnica de criptografia de curvas elípticas, seja mais usada em redes de sensores. Um exemplo é o caso do *bluetooth low energy*, protocolo muito usado em redes de sensores e na chamada Internet das coisas, que na sua última versão, a 4.2, utiliza o também método ECDH para estabelecimento de uma chave de sessão entre dois elementos.

5.1 Trabalho futuro

Como trabalho futuro, a fim de melhorar o sistema implementado, existem alguns tópicos que podem ser melhorados ou concluídos, a saber:

- Conclusão da implementação da solução inicialmente proposta, nomeadamente a realização do protocolo de autenticação diretamente com a unidade central do sistema.
- Realizar de testes ao consumo energético para o uso de chaves assimétricas com tamanho superiores (160 bits e 192 bits).
- Implementação de protocolo para revogação de chaves que implique novo processo de autenticação.
- Implementação do protocolo que distribui a chave de grupo, a um novo nó na rede que já realizou o processo de autenticação anteriormente.

Bibliografia

[1] J. N. Al-Karaki and A. E. Kamal. Routing techniques in wireless sensor networks: a survey. *Wireless Communications, IEEE*, 11(6):6–28, 2004.

[2] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer networks*, 52 (12):2292–2330, 2008.

[3] V. C. Gungor and G. P. Hancke. Industrial wireless sensor networks: Challenges, design principles, and technical approaches. *Industrial Electronics, IEEE Transactions on*, 56(10):4258–4265, 2009.

[4] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Ad hoc networks*, 1(2):293–315, 2003.

[5] C. Hartung, J. Balasalle, and R. Han. Node compromise in sensor networks: The need for secure systems. Technical report, 2005.

[6] C. Karlof, N. Sastry, and D. Wagner. Tinysec: a link layer security architecture for wireless sensor networks. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, pages 162–175. ACM, 2004.

[7] Y. Wang, G. Attebury, and B. Ramamurthy. A survey of security issues in wireless sensor networks. *IEEE Communications Surveys & Tutorials*, 8(2):2–23, 2006.

[8] A.-S. K. Pathan, H.-W. Lee, and C. S. Hong. Security in wireless sensor networks: issues and challenges. In *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*, volume 2, pages 6–pp. IEEE, 2006.

[9] H. Redwan and K.-H. Kim. Survey of security requirements, attacks and network integration in wireless mesh networks. In *New Technologies, Mobility and Security, 2008. NTMS'08.*, pages 1–5. IEEE, 2008.

[10] M. S. Siddiqui and C. S. Hong. Security issues in wireless mesh networks. In *Multimedia and Ubiquitous Engineering, 2007. MUE'07. International Conference on*, pages 717–722. IEEE, 2007.

[11] S. Zhu, S. Setia, and S. Jajodia. Leap+: Efficient security mechanisms for large-scale distributed sensor networks. *ACM Transactions on Sensor Networks (TOSN)*, 2(4):500–528, 2006.

[12] S. Seth and A. Gankotiya. Denial of service attacks and detection methods in wireless mesh networks. In *Recent Trends in Information, Telecommunication and Computing (ITC), 2010 International Conference on*, pages 238–240. IEEE, 2010.

[13] J. Newsome, E. Shi, D. Song, and A. Perrig. The sybil attack in sensor networks: analysis & defenses. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 259–268. ACM, 2004.

[14] Y.-C. Hu, A. Perrig, and D. B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Proceedings of the 2nd ACM Workshop on Wireless Security*, pages 30–40. ACM, 2003.

[15] A. Perrig, R. Szewczyk, J. Tygar, V. Wen, and D. E. Culler. Spins: Security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, 2002.

[16] I. Krontiris, T. Dimitriou, H. Soroush, and M. Salajegheh. WSN link-layer security frameworks. *Wireless Sensor Network Security*, 1:142, 2008.

[17] M. Luk, G. Mezzour, A. Perrig, and V. Gligor. Minisec: a secure sensor network communication architecture. In *Information Processing in Sensor Networks, 2007. IPSN 2007. 6th International Symposium on*, pages 479–488. IEEE, 2007.

[18] S.-B. Lee and Y.-H. Choi. Arms: An authenticated routing message in sensor networks. In *Secure Mobile Ad-hoc Networks and Sensors*, pages 158–173. Springer, 2006.

[19] L. Eschenauer and V. D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 41–47. ACM, 2002.

[20] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz. Comparing elliptic curve cryptography and rsa on 8-bit cpus. In *Cryptographic Hardware and Embedded Systems-CHES 2004*, pages 119–132. Springer, 2004.

[21] G. Gaubatz, J.-P. Kaps, and B. Sunar. Public key cryptography in sensor networks—revisited. In *Security in Ad-hoc and Sensor Networks*, pages 2–18. Springer, 2005.

[22] D. J. Malan, M. Welsh, and M. D. Smith. A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography. In *Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004. 2004 First Annual IEEE Communications Society Conference on*, pages 71–80. IEEE, 2004.

[23] A. S. Wander, N. Gura, H. Eberle, V. Gupta, and S. C. Shantz. Energy analysis of public-key cryptography for wireless sensor networks. In *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*, pages 324–328. IEEE, 2005.

[24] A. Liu and P. Ning. Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. In *Information Processing in Sensor Networks, 2008. IPSN'08. International Conference on*, pages 245–256. IEEE, 2008.

[25] L. B. Oliveira, D. F. Aranha, C. P. Gouvêa, M. Scott, D. F. Câmara, J. López, and R. Dahab. Tinypbc: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. *Computer Communications*, 34(3):485–493, 2011.

[26] K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Ad hoc networks*, 3(3):325–349, 2005.

[27] W. R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 174–185. ACM, 1999.

[28] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 56–67. ACM, 2000.

[29] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, pages 10–pp. IEEE, 2000.

[30] M. Handy, M. Haase, and D. Timmermann. Low energy adaptive clustering hierarchy with deterministic cluster-head selection. In *Mobile and Wireless Communications Network, 2002. 4th International Workshop on*, pages 368–372. IEEE, 2002.

[31] Y. Yu, R. Govindan, and D. Estrin. Geographical and energy aware routing: A recursive data dissemination protocol for wireless sensor networks. Technical report.

[32] B. Parno, M. Luk, E. Gaustad, and A. Perrig. Secure sensor network routing: A clean-slate approach. In *Proceedings of the 2006 ACM CoNEXT conference*. ACM, 2006.

[33] X. Wang, L. Yang, and K. Chen. Sdd: secure directed diffusion protocol for sensor networks. In *Security in Ad-hoc and Sensor Networks*, pages 205–214. Springer, 2005.

[34] S.-B. Lee and Y.-H. Choi. A secure alternate path routing in sensor networks. *Computer Communications*, 30(1):153–165, 2006.

[35] R. C. Merkle. Protocols for public key cryptosystems. In *IEEE Symposium on Security and privacy*, volume 1109, pages 122–134, 1980.

[36] J. Deng, R. Han, and S. Mishra. Insens: Intrusion-tolerant routing for wireless sensor networks. *Computer Communications*, 29(2):216–230, 2006.

[37] G. Zhan, W. Shi, and J. Deng. Tarf: A trust-aware routing framework for wireless sensor networks. In *Wireless Sensor Networks*, pages 65–80. Springer, 2010.

[38] Federal information processing standards publication (FIPS 197). Advanced Encryption Standard (AES), 2001.

[39] ENISA. Algorithms, key sizes and parameters report, 2013 recommendations. October 2013.

[40] J. Song, R. Poovendran, J. Lee, and T. Iwata. The AES-CMAC algorithm. RFC 4493, RFC Editor, June 2006.

[41] R. Housley. Cryptographic message syntax (CMS). STD 70, RFC Editor, September 2009.

[42] L. Chen. Nist sp 800-108. recommendation for key derivation using pseudorandom functions (revised). Technical report, Gaithersburg, MD, United States, 2009.