

UNIVERSIDADE FEDERAL DO PAMPA

Juliano Rodovalho Macedo

**Detecção de Nós Injetores de Falsa
Informação utilizando a Comparação entre
Agrupamentos em Redes de Sensores Sem
Fio**

Alegrete
2017

Juliano Rodovalho Macedo

**Detecção de Nós Injetores de Falsa Informação
utilizando a Comparação entre Agrupamentos em
Redes de Sensores Sem Fio**

Trabalho de Conclusão de Curso apresentado
ao Curso de Graduação em Engenharia de
Software da Universidade Federal do Pampa
como requisito parcial para a obtenção do tí-
tulo de Bacharel em Engenharia de Software.

Orientador: Profa. Dr^a. Aline Vieira de Mello


Alegrete
2017

Juliano Rodovalho Macedo

**Detecção de Nós Injetores de Falsa Informação
utilizando a Comparação entre Agrupamentos em
Redes de Sensores Sem Fio**

Trabalho de Conclusão de Curso apresentado
ao Curso de Graduação em Engenharia de
Software da Universidade Federal do Pampa
como requisito parcial para a obtenção do tí-
tulo de Bacharel em Engenharia de Software.

Trabalho de Conclusão de Curso defendido e aprovado em 29 de Junho de 2017.
Banca examinadora:



Profa. Dr^a. Aline Vieira de Mello
Orientadora
UNIPAMPA



Profa. Ma. Alice Fonseca Finger
UNIPAMPA



Prof. Me. Cristian Müller
UNIPAMPA

Este trabalho é dedicado a minha família...

Y a ti hermano, Arthur Villar Meza.

“Soy lo que me enseñó mi padre

El que no quiere a su patria,

no quiere a su madre.”

– Calle 13, Latinoamérica.

AGRADECIMENTOS

Agradeço primeiramente a Deus e Nossa Senhora do Rosário.

Aos meus familiares, pelo apoio constante, além do envio incrível de boas energias. Não existe felicidade sem família!

A orientadora Professora Dr^a. Aline Vieira de Mello, por sua paciência, dedicação e suporte no desenvolvimento deste Trabalho de Conclusão de Curso (TCC).

A Professora Ma. Alice Fonseca Finger e ao Professor Me. Crístian Müller que aceitaram o convite para compor a banca examinadora.

Ao Professor Dr. Juliano Fontoura Kazienko, por acreditar no meu trabalho. Suas orientações e conselhos de estimada valiosidade, permitiram a concepção da pesquisa que originou esse TCC.

Ao amigo João Otávio Massari Chervinski, companheiro no grupo de pesquisa SEDES (Segurança e Desempenho em Computação Ubíqua e Pervasiva), e que também contribuiu muito com a pesquisa.

E a uma pessoa muito especial!

“Ninguém caminha sem aprender a caminhar,
sem aprender a fazer o caminho caminhando,
refazendo e retocando o sonho pelo qual se pôs a caminhar.”
(Freire, Paulo)

“Felicidad no es hacer lo que uno quiere sino querer lo que uno hace.”
“O homem é aquilo que faz de si mesmo.”
(Sartre, Jean-Paul)

RESUMO

O ataque da Injeção de Falsa Informação, do inglês *False Data Injection*, consiste na introdução indevida de falsas informações, confundindo a rede e os sistemas que dela derivam. A detecção e mitigação desse ataque é altamente complexa, pois todos os nós infectados estão autenticados na rede, cumprindo suas funções originais, repassando os pacotes e realizando coletas, permanecendo assim despercebidos aos sistemas tradicionais de segurança. Nesse contexto, o objetivo geral do presente trabalho é implementar e avaliar um mecanismo capaz de detectar nós Injetores de Falsa Informação, utilizando a comparação entre agrupamentos em Redes de Sensores Sem Fio (RSSFs). Nesse sentido, o mecanismo *Cluster-based False Data Detection (CFDD)* foi implementado e testado através de simulações executadas no simulador TOSSIM do TinyOS. Os resultados demonstram que o mecanismo CFDD foi superior aos demais trabalhos da literatura em todos os cenários analisados, principalmente quando a RSSF apresentava altas taxas de nós maliciosos. Destaca-se a resiliência do CFDD ao suportar infestações superiores a 40% da rede, sem zerar sua detecção mesmo nos cenários com 75% de nós maliciosos.

Palavras-chave: Redes de Sensores Sem Fio, Segurança, Detecção de Nós Maliciosos, Injeção de Falsa Informação, Comparação entre Agrupamentos.

ABSTRACT

One of the major challenges regarding sensors use is to ensure that the information being forwarded hasn't been modified, even when several devices are compromised. The False Data Injection attack modifies the sensed data in Wireless Sensor Networks (WSNs). This type of attack is hard to defend against when using existing approaches, because most of the mechanisms will drop the fake report without verifying from which sensor node that false information is coming from. This work presents a Cluster-based False Data Detection (CFDD) mechanism for malicious nodes detection, based mainly on the comparison among neighborhood clusters. Simulation results show a detection efficiency higher than those of NFFS, GFFS and SEF mechanisms, being capable of identifying more than 30% of the malicious devices in a 255 nodes network, even when 75% of those are compromised.

Key-words: Wireless Sensor Network, Security, Malicious Node Detection, False Data Injection, Neighborhood Clusters Comparison.

LISTA DE FIGURAS

Figura 1 – Representação esquemática geral de um nó sensor MICAz MPR 2400. . .	28
Figura 2 – Imagem real de um nó sensor MICAz MPR 2400.	28
Figura 3 – Representação da arquitetura descentralizada.	29
Figura 4 – Representação da arquitetura hierárquica.	29
Figura 5 – Processo de pesquisa.	42
Figura 6 – Organização da RSSF em arquitetura hierárquica, composta por uma central e os nós sensores.	50
Figura 7 – Etapas do mecanismo CFDD. Ressalta-se que a maioria das etapas do mecanismo são executadas somente na central.	51
Figura 8 – Modificação na estrutura dos pacotes que são transmitidos pela RSSF.	53
Figura 9 – Representação do rastro do pacote.	53
Figura 10 – Detecção dos agrupamentos da rede.	53
Figura 11 – Detecção dos nós maliciosos presentes na RSSF pelo mecanismo CFDD. Utilizando as médias unitárias contra a média do agrupamento ao qual cada nó pertence.	54
Figura 12 – Detecção do agrupamento malicioso, fortemente comprometido, pelo CFDD, utilizando as médias dos agrupamentos vizinhos para comparação e detecção de anomalias.	54
Figura 13 – Representação das camadas necessárias para simular o mecanismo CFDD. Imagem meramente ilustrativa.	56
Figura 14 – Comparação dos resultados da eficiência do CFDD.	60
Figura 15 – Comparação dos resultados do CFDD utilizando linhas para analisar suas tendências.	60
Figura 16 – Comparação do mecanismo CFDD com mecanismos da literatura. . . .	61

LISTA DE TABELAS

Tabela 1 – Comparação entre os mecanismos da literatura, onde ND significa informação Não Disponível.	38
Tabela 2 – Demonstração dos resultados do cálculo da Distribuição Binomial para cada variação de X.	47
Tabela 3 – Parâmetros imutáveis utilizados nas simulações do mecanismo.	48
Tabela 4 – Número de nós maliciosos por rede, baseados na porcentagem de infecção, 20%, 45% e 75%.	49
Tabela 5 – Parâmetros utilizados nas simulações do mecanismo proposto.	55
Tabela 6 – Parâmetros utilizados na análise dos resultados obtidos.	59

LISTA DE SIGLAS

ACM Association Computing Machinery

AES Advanced Encryption Standard

BS Base Station

CAPES Coordenação de Aperfeiçoamento de Pessoal de Nível Superior

CBC-MAC Cypher Block Chaining Message Authentication Code

CFDD Cluster-based False Data Detection

CoS Center-of-Stimulus

EEMDTS Energy Efficient Multipath Data Transfer Scheme

GFFS Geographical Information based False Data Filtering Scheme

GPS Global Positioning System

ID Identificador

IEEE Institute of Electrical and Electronics Engineers

IoE Internet of Everything

IoT Internet of Things

MAC Message Authentication Code

MANET Mobile Ad Hoc Network

nesC network embedded systems C

NFFS Neighbor-Information based False Data Filtering Scheme

RSSF Rede de Sensores Sem Fio

SBSeg Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais

SEF Statistical En-route Filtering

SisGAAz Sistema de Gerenciamento da Amazônia Azul

SO Sistema Operacional

SPAIS Self-checking Pollution Attackers Identification Scheme

TCC Trabalho de Conclusão de Curso

TOSSIM TinyOS SIMulator

UNIPAMPA Universidade Federal do Pampa

WSN Wireless Sensor Network

SUMÁRIO

1	INTRODUÇÃO	23
1.1	Objetivos	24
1.2	Problema de Pesquisa	24
1.3	Organização do Trabalho	24
2	FUNDAMENTAÇÃO TEÓRICA	27
2.1	Redes de Sensores sem Fio (RSSFs)	27
2.1.1	Sensores sem Fio	27
2.1.2	Exemplos da Aplicação das RSSFs	27
2.1.3	Tipos de Arquiteturas em RSSFs	28
2.2	Ataques em RSSFs	30
3	TRABALHOS RELACIONADOS	33
3.1	Critérios para Seleção de Trabalhos Relacionados	33
3.2	Visão Geral Sobre a Segurança em Redes de Sensores Sem Fio (RSSFs)	34
3.3	O Ataque da Injeção de Falsa Informação em RSSFs	34
3.4	Conclusões do Capítulo	37
4	MÉTODO	41
5	DESENVOLVIMENTO	45
5.1	Definição das Hipóteses	45
5.2	Verificação Formal	46
5.3	Configuração	48
5.3.1	Definição do Ambiente	48
5.3.2	Definição do Cenário	48
5.4	Projeto e Codificação	49
5.4.1	Projeto do Mecanismo	49
5.4.2	Codificação do Mecanismo	52
5.5	Simulação	55
6	RESULTADOS	59
6.1	Limitações do Trabalho	62
7	CONCLUSÃO	65
7.1	Trabalhos Futuros	66
	REFERÊNCIAS	67

APÊNDICES	71
APÊNDICE A – ARQUIVO <i>SIMULA.PY</i> , COM O <i>SCRIPT</i> QUE GUIA AS SIMULAÇÕES.	73
APÊNDICE B – ARQUIVO <i>MONTA-RASTRO.PY</i> , COM O <i>SCRIPT</i> QUE DETECTA O RASTRO DOS PACOTES.	75
APÊNDICE C – ARQUIVO <i>IDENTIFICA-AGRUPAMENTOS.PY</i> , COM O <i>SCRIPT</i> QUE IDENTIFICA OS AGRUPAMENTOS PERTENCENTES A REDE.	77
APÊNDICE D – ARQUIVO <i>DETECTA-NODES-MALICIOSOS.PY</i> , COM O <i>SCRIPT</i> QUE DETECTA OS NÓS MALICIOSOS.	81

1 INTRODUÇÃO

Nas últimas três décadas o avanço científico e a popularização da tecnologia transformaram o modo como interagimos com o mundo. Prevê-se, por exemplo, que o número de objetos inteligentes conectados exceda 7 trilhões em 2025, concretizando o termo conhecido como *Immersed Human*, onde a sociedade estará completamente imersa no mundo cibernético (VINCENTELLI, 2014) (BORGIA, 2014).

Uma das principais tecnologias são os sensores sem fio, que permitem a coleta dos mais diversos tipos de dados, mas que apresentam severas limitações de bateria, memória, capacidade computacional e demais recursos. Esses sensores podem ser organizados em redes, que são chamadas de Redes de Sensores Sem Fio (RSSFs), do inglês *Wireless Sensor Networks* (WSNs), cujo intuito é monitorar uma determinada área (KAZIENKO, 2013) (BORGIA, 2014) (FERRAZ; VELLOSO; DUARTE, 2014).

Nesse sentido, por se tratarem de redes que são comumente implantadas em ambientes hostis, às RSSFs estão expostas a diversos tipos de ataques, afetando a segurança da rede, as aplicações que operam ou derivam dela, e conseqüentemente as pessoas que dependem do bom funcionamento da mesma (MARGI et al., 2009).

Um exemplo real desse risco ocorreu em Abril de 2017, na região metropolitana de Dallas-Fort Worth, no Texas, Estados Unidos, onde o sistema de notificações de emergência contra tornados e furacões foi invadido e suas 156 sirenes ativadas por mais de 2 horas em uma madrugada, gerando pânico em uma população de quase 8 milhões de habitantes (BALLOR; WILONSKY; STEELE, 2017) (ROSENBERG; SALAM, 2017) (KUMAR, 2017).

Dentre os ataques no âmbito das RSSFs, a adulteração do dado coletado por um sensor é um dos mais complexos existentes atualmente. Por exemplo, um dispositivo que possua sensor de fumaça deve gerar alertas sempre que detectar a presença dela no ambiente. Entretanto, se esse dispositivo for comprometido, ele passará a emitir falsos alertas, confundindo o sistema, e o impedindo de tomar corretas decisões perante a situações reais (WANGHAM; DOMENECH; MELLO, 2013).

Esse tipo de ataque é conhecido na literatura como Injeção de Falsa Informação, ele ocorre quando um nó malicioso produz informações falsas sobre suas coletas, omitindo o dado real, e assim comprometendo a rede. Contramedidas à Injeção de Falsa Informação são essenciais, porém não triviais. Esse é um ataque difícil de ser detectado e controlado, pois todos os nós infectados estão autenticados na rede, cumprindo suas funções originais, repassando os pacotes e realizando coletas; permanecendo despercebido aos sistemas tradicionais de segurança (RIECKER et al., 2012).

A Injeção de Falsa Informação é uma ataque relativamente recente na literatura, com seus primeiros registros realizados em meados da última década. Embora existam diversos trabalhos que abordam os ataques em RSSFs, a literatura ainda carece de infor-

mações descritivas e propostas para solucionar o ataque da Injeção de Falsa Informação (MARGI et al., 2009) (FERRAZ; VELLOSO; DUARTE, 2014).

As soluções existentes encarecem a rede por utilizar aplicações externas como o *Global Positioning System (GPS)* ou a implantação de dispositivos *tamper proof* (KAZI-ENKO et al., 2015). Outras não levam em consideração as limitações de hardware dos sensores, ignorando requisitos críticos para o bom funcionamento de uma *RSSF*, afetando drasticamente o seu tempo de vida útil (WANG et al., 2014).

1.1 Objetivos

Visando contribuir para a futura solução da citada lacuna na literatura, esse trabalho tem como objetivo geral propor e implementar um mecanismo capaz de detectar nós que praticam a Injeção de Falsa Informação, utilizando a comparação entre agrupamentos em *RSSFs*.

Nesse sentido, os Objetivos Específicos do trabalho são:

- (I) Levantar o estado da arte com relação ao problema de pesquisa estudado;
- (II) Propor um mecanismo capaz de detectar nós Injetores de Falsa Informação, utilizando a comparação entre agrupamentos em *RSSFs*;
- (III) Implementar o mecanismo proposto;
- (IV) Avaliar o mecanismo implementado, através da comparação entre os mecanismos da literatura.

1.2 Problema de Pesquisa

O problema de pesquisa estudado consiste na “detecção de nós Injetores de Falsa Informação, considerando a possibilidade do alto grau de comprometimento do agrupamento em *RSSFs*”.

1.3 Organização do Trabalho

O restante do trabalho está organizado de acordo com a sequência de Capítulos descritos a seguir:

- O [Capítulo 2](#) apresenta a fundamentação teórica do trabalho, descrevendo os conceitos relacionados aos sensores, às *RSSFs* e ao ataque abordado.
- No [Capítulo 3](#) é descrita a revisão da literatura e o estado da arte do ataque da Injeção de Falsa Informação em *RSSFs*.

-
- O método utilizado é descrito no [Capítulo 4](#), apresentando a imagem do processo de pesquisa e a explicação de cada etapa.
 - O desenvolvimento do trabalho é detalhadamente explicado no [Capítulo 5](#).
 - O [Capítulo 6](#) apresenta os resultados obtidos no trabalho, bem como a discussão dos mesmos em comparação com a literatura.
 - Finalmente no [Capítulo 7](#), encontram-se as conclusões e os possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

O presente Capítulo tem como objetivo apresentar ao leitor os conceitos essenciais para a compreensão do presente trabalho. Na [Seção 2.1](#) são abordadas as Redes de Sensores Sem Fio (RSSFs), apresentando os sensores sem fio, os exemplos de aplicações das RSSFs e os seus tipos de arquitetura. Na [Seção 2.2](#), são descritos os ataques mais conhecidos, dando destaque ao ataque da Injeção de Falsa Informação.

2.1 Redes de Sensores sem Fio (RSSFs)

As Redes de Sensores Sem Fio (RSSFs), do inglês *Wireless Sensor Networks (WSNs)*, conectam dispositivos, mais especificamente sensores, sem a necessidade de fios. A RSSF é uma especialização das redes *Mobile Ad Hoc Networks (MANETs)*, herdando diversas características, como: a auto-organização, a ausência de infraestrutura, a possibilidade da mobilidade entre os nós, e a arquitetura flexível (LOUREIRO et al., 2003).

Nesse sentido, uma RSSF é comumente composta por um número massivo de nós sensores. Esses nós são capazes de cooperar, capturar, interpretar e enviar dados sobre os mais diversos aspectos, como por exemplo, temperatura, pressão, movimento, sons, entre outros. Os dados coletados são transmitidos através da rede até a central, que é responsável por processar as informações e realizar a intermediação entre a rede e outras entidades, como por exemplo a *Internet of Things (IoT)* ou a *Internet* convencional. Após receber e processar os dados, a central toma as decisões necessárias (ZHANG; YANG; CHEN, 2009) (GRANJAL; MONTEIRO; SILVA, 2015).

2.1.1 Sensores sem Fio

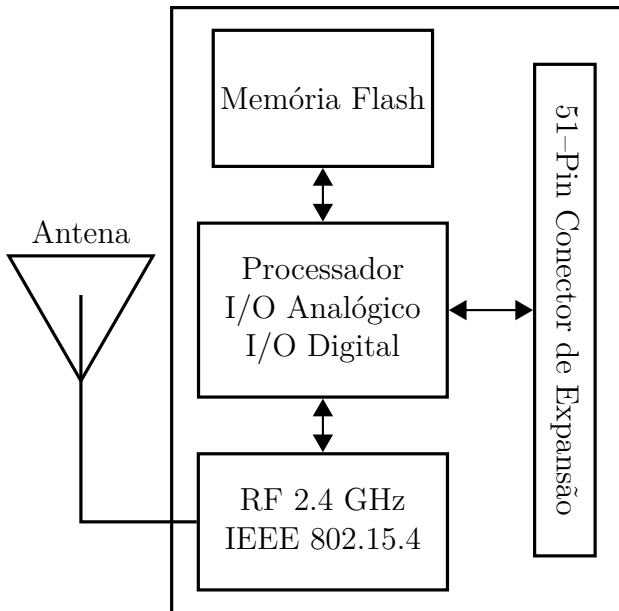
O avanço da comunicação Sem Fio, do *hardware* e da miniaturização desse, possibilitou a criação de sensores diminutos, compostos simplesmente por processador, memória, transmissor de rádio frequência, uma pequena bateria e a unidade de sensoriamento. Esses sensores são agrupados em redes complexas, formando às RSSFs, e permitem o monitoramento de vastas áreas, operando em modo autônomo e não infraestruturado (LOUREIRO et al., 2003) (KAZIENKO, 2013).

Na [Figura 1](#), é possível visualizar a representação esquemática de um dos sensores mais comuns do mercado, o MICAz MPR 2400, e na [Figura 2](#) uma imagem real do mesmo sensor.

2.1.2 Exemplos da Aplicação das RSSFs

Às RSSFs podem ser aplicadas a diversos contextos, sejam eles o monitoramento, rastreamento, coordenação, processamento e etc. Por exemplo, pode-se interconectar sensores para realizar o monitoramento e controle das condições climáticas em uma região

Figura 1: Representação esquemática geral de um nó sensor MICAz MPR 2400.



Fonte: Adaptado de MEMSIC (2010).

Figura 2: Imagem real de um nó sensor MICAz MPR 2400.



Fonte: MEMSIC (2015).

inóspita, como em uma floresta, no oceano e até mesmo fora do planeta terra. Um exemplo deste último caso é o projeto Sistema de Gerenciamento da Amazônia Azul (SisGAAz), para vigilância integrada da amazônia azul brasileira (JUNIOR, 2013).

Outro bom exemplo do dinamismo e aplicabilidade das RSSFs é o trabalho de Vieira et al. (2010), que realizou um levantamento sobre o estado da arte das Redes de Sensores Aquáticas, as quais são uma especialidade das RSSFs tradicionais. Tais redes possuem a capacidade de monitorar o meio ambiente marinho, realizar previsões sísmicas, a detecção de poluentes e substâncias contaminantes, além do monitoramento de campos de gás e petróleo.

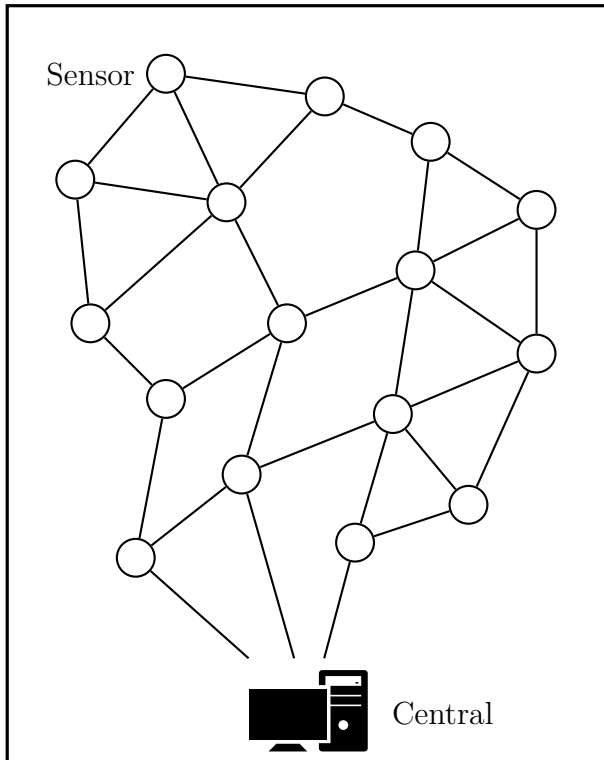
2.1.3 Tipos de Arquiteturas em RSSFs

A comunicação entre os nós sensores e a central em uma RSSF pode ocorrer em um ou múltiplos saltos. A comunicação em um salto, do inglês *single-hop*, permite que o dado coletado seja enviado diretamente, sem a necessidade de intermediários. Entretanto, exige que a distância máxima entre um nó e a central seja de apenas um salto, aumentando o consumo de recursos em redes que cobrem longas distância (RUIZ et al., 2004) (MOSCHITTA; NERI, 2014).

Por outro lado, a comunicação em múltiplos saltos, do inglês *multi-hop*, permite que a informação flua entre os nós vizinhos até o destino final, diminuindo a distância entre os saltos e aumentando a área de cobertura da rede. Isso possibilita maior economia de re-

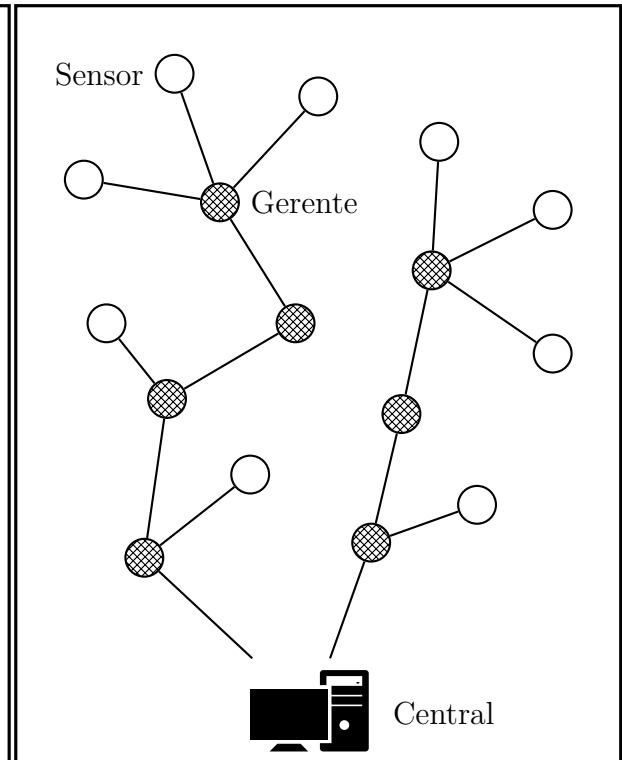
curso, principalmente o energético, melhorando a vida útil da rede (MOSCHITTA; NERI, 2014). Ao utilizar comunicação de múltiplos saltos, a arquitetura possui papel fundamental, podendo ser dividida em duas categorias: Arquitetura Descentralizada (apresentada na Figura 3) e Arquitetura Hierárquica (apresentada na Figura 4).

Figura 3: Representação da arquitetura descentralizada.



Fonte: Adaptado de Moschitta e Neri (2014).

Figura 4: Representação da arquitetura hierárquica.



Fonte: Adaptado de Moschitta e Neri (2014).

Na Arquitetura Descentralizada, cada nó sensor exerce o papel de coletar informações e repassar os pacotes recebidos, o que pode gerar um maior consumo de recursos devido ao alto número de comunicações. Já na Arquitetura Hierárquica, os nós são organizados em agrupamentos, onde um dos membros é eleito o gerente, em inglês chamado de *cluster head*, o qual é responsável por repassar as informações aos demais gerentes em direção a central (RUIZ et al., 2004) (MOSCHITTA; NERI, 2014).

A eleição do nó gerente é comumente baseada em vários critérios, como: distância entre os demais membros do agrupamento, qualidade do *link* de comunicação entre outros gerentes, disponibilidade do recurso energético (bateria), entre outros. Geralmente, a eleição e esses critérios são controlados automaticamente pelo protocolo de roteamento utilizado (RUIZ et al., 2004).

É de conhecimento na literatura que as abordagens que utilizam a Arquitetura Hierárquica permitem às RSSFs obterem um melhor aproveitamento de seus recursos, principalmente o energético (JEBAS; PARAMASIVAN, 2013) (MOSCHITTA; NERI, 2014).

2.2 Ataques em RSSFs

Garantir a segurança nas Redes de Sensores Sem Fio (RSSFs) é primordial, assim como assegurar a eficiência e resiliência das mesmas nos ambientes onde estão inseridas. Existem diversos tipos de ataques que podem afetar às RSSFs, sendo que os mais conhecidos e diagnosticados na literatura são: Buraco negro, Repasse seletivo e Personificação (LOUREIRO et al., 2003) (MARGI et al., 2009). Outro ataque comum nas RSSFs, mas que não é tão frequentemente abordado na literatura, é a Injeção de Falsa Informação. Os ataques citados acima são descritos a seguir.

- Buraco Negro

O ataque de Buraco Negro, do inglês *Black Hole*, consiste em atrair o maior tráfego possível para o nó malicioso, visando descartar ou modificar os pacotes, comprometendo severamente o funcionamento da rede. O ataque ocorre quando o nó malicioso, controlado pelo atacante, introduz na rede a informação que as melhores rotas passam por ele, atraindo assim um alto número de pacotes (WANG; ATTEBURY; RAMAMURTHY, 2006) (MARGI et al., 2009).

- Repasse Seletivo

No ataque de Repasse Seletivo, ou encaminhamento seletivo de mensagens, do inglês *Selective Forwarding Attacks*; o nó malicioso descarta pacotes de forma seletiva, isto é, ao contrário do Buraco Negro, que descarta todos os pacotes que passam por ele, o Repasse Seletivo realiza uma pre-seleção e descarta unicamente os pacotes previamente especificados (BYSANI; TURUK, 2011).

- Personificação

O ataque de Personificação, do inglês *Sybil*, consiste em forjar a identidade de vários nós (fabricando ou roubando), com o intuito de interferir nos sistemas de segurança da rede, como, por exemplo, manipular os algoritmos de confiança, comprometer o protocolo de roteamento, e extrair conhecimento das informações da rede (*Sniffing*) (WANG; ATTEBURY; RAMAMURTHY, 2006) (MARGI et al., 2009).

- Injeção de Falsa Informação

O ataque de adulteração do dado coletado por um sensor é conhecido na literatura como Injeção de Falsa Informação. Ele ocorre quando um nó malicioso produz informações falsas sobre suas coletas, omitindo o dado real, assim, comprometendo a rede. Contramedidas à Injeção de Falsa Informação são essenciais, porém não triviais (ZHANG et al., 2010) (WANG et al., 2014).

Wang, Attebury e Ramamurthy (2006) e Riecker et al. (2012) chamam a atenção para uma característica preocupante: a complexidade de detectar e controlar o ataque da Injeção de Falsa Informação, pois todos os nós infectados estão autenticados na

rede, cumprindo suas funções originais, repassando os pacotes e realizando coletas, permanecendo assim despercebido aos sistemas tradicionais de segurança.

Vale ressaltar que a Injeção de Falsa Informação pode ser iniciada após outro ataque ter deixado a rede vulnerável, por exemplo, o ataque de mascaramento, a personificação, entre outros ([WANGHAM; DOMENECH; MELLO, 2013](#)).

3 TRABALHOS RELACIONADOS

Este capítulo apresenta os critérios para seleção dos trabalhos relacionados, uma revisão da literatura partindo desses trabalhos, e o estado da arte do ataque da Injeção de Falsa Informação em [RSSFs](#). Ao final são discutidos os aspectos mais relevantes, e é apresentada uma tabela que compara os mecanismos selecionados da literatura com o mecanismo proposto e desenvolvido no presente trabalho.

3.1 Critérios para Seleção de Trabalhos Relacionados

Para a seleção dos trabalhos relacionados foram utilizados alguns critérios. O primeiro foi a definição das bases de pesquisa. Para trabalhos na língua inglesa foram utilizadas as bases: *Science Direct*, *Springer*, *Institute of Electrical and Electronics Engineers (IEEE) Xplore Digital Library* e *Association Computing Machinery (ACM) Digital Library*. Já para trabalhos em português utilizou-se o Portal de Periódicos da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior ([CAPES](#)). O *Google Scholar* também foi utilizado para encontrar trabalhos em ambas as línguas.

Em seguida definiu-se quais seriam as *strings* de busca, diferenciadas por língua. Para inglês utilizou-se: *wsn; wireless sensor network, malicious node detection, false data injection, false data injection attack; false data detection*. Já para português: *rssf; rede de sensores sem fio; injeção de falsa informação; ataque da injeção de falsa informação; detecção de nós maliciosos*.

Após a obtenção dos trabalhos em cada base, aplicou-se os seguintes critérios de inclusão e exclusão:

- **Inclusão:** artigos científicos, livros, dissertação de mestrado ou tese de doutorado que descrevem pesquisas na área da segurança em [RSSFs](#). Trabalhos publicados a partir do ano de 2003 e disponíveis para download, seja de forma gratuita ou em bases de dados que acadêmicos da Universidade Federal do Pampa ([UNIPAMPA](#)) tenham acesso.
- **Exclusão:** Trabalhos não publicados em *journals*, conferências, eventos ou não associados a universidades e instituições de pesquisa. Trabalhos que estão fora do domínio da segurança em [RSSFs](#).

Finalmente, o último refinamento dos trabalho foi aplicar uma seleção por *Qualis*, conferidos pela [CAPES](#). Esse critério foi igual a : B2, B1, A2 ou A1. Entretanto, foram aplicadas três exceções aos critérios de *Qualis*, ao selecionar trabalhos provenientes do Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais ([SBSeg](#)), que embora tenha *Qualis* B4, são fortes referências em português sobre o tema abordado.

3.2 Visão Geral Sobre a Segurança em Redes de Sensores Sem Fio (RSSFs)

Redes de Sensores Sem Fio (RSSFs) são suscetíveis a ataques, sejam eles físicos ou virtuais. Por exemplo, se uma chave criptográfica de um nó é comprometida, o atacante pode personificá-lo completamente e realizar ataques como a Injeção de Falsa Informação. Sistemas tradicionais e mecanismos de criptografia não são capazes de detectar qual é a fonte da injeção (LOUREIRO et al., 2003) (ZHANG; YU; NING, 2006) (LIN et al., 2011).

Nesse sentido, a necessidade iminente em oferecer maior segurança para as aplicações dessas redes gerou um conflito entre a minimização do consumo dos recursos de hardware e a maximização da segurança, pois a sua adoção afeta esses recursos (MARGI et al., 2009) (KAZIENKO, 2013).

Outro aspecto importante a ser pontuado é que a maioria das soluções existentes aumentam o custo financeiro da rede, porque propõem a utilização de aplicações externas como *Global Positioning System (GPS)* ou a implantação de dispositivos *tamper proof* (KAZIENKO et al., 2015). Por exemplo, a proposta de Wang et al. (2014) incorpora um dispositivo GPS a cada sensor da rede para auxiliar na detecção da falsa informação, mas não leva em consideração o quanto cada um desses dispositivos elevaria o custo econômico final da rede, além do aumento do consumo energético.

Por esses motivos, é fundamental que os mecanismos que proponham melhorar a segurança em uma RSSF não inviabilizem, sobre tudo, o aspecto econômico da rede, a pena de não apresentarem aplicabilidade em cenários reais por justamente ignorarem um dos grandes desafios nesse ambiente (MARGI et al., 2009) (GRANJAL; MONTEIRO; SILVA, 2015).

3.3 O Ataque da Injeção de Falsa Informação em RSSFs

Conforme Wang, Attebury e Ramamurthy (2006) e Wang et al. (2014), existem ataques pouco abordados na literatura, como o ataque da Injeção de Falsa Informação. Esse é um ataque relativamente recente, com seus primeiros registros realizados em meados da última década. Por esse motivo, a discussão busca descobrir e apresentar as características e os desafios na sua detecção.

Partindo desse princípio, foram selecionados e analisados diversos trabalhos conceituados da literatura, que tratam da segurança e dos ataques que ocorrem em RSSFs, mas com foco especial no problema de pesquisa abordado no presente trabalho.

Os ataques nas RSSFs podem ser classificados em três categorias: ataques contra o sigilo e a autenticação; ataques contra a disponibilidade dos serviços e resiliência da rede; e ataques furtivos contra a integridade dos serviços oferecidos pela rede. A Injeção de Falsa Informação é inserida na última categoria, cuja definição é: “ataques que tenham como objetivo primordial, enganar a rede e fazer com que ela aceite dados falsos, durante o maior tempo possível.” (WANG; ATTEBURY; RAMAMURTHY, 2006).

Existem levantamentos que reúnem o estado da arte do ataque da Injeção de Falsa Informação. Por exemplo, em [Lin et al. \(2011\)](#), são comparados diversos mecanismos e técnicas aplicadas na mitigação desse ataques, além de pontuar os desafios em aberto na área. Os autores também discutem e apresentam as principais características e os aspectos positivos ou negativos em cada proposta, chamando a atenção para a despreocupação com a utilização frequente de aplicações externas.

Já em [Zin et al. \(2015\)](#), é apresentado um levantamento sobre os protocolos de roteamento multicaminho e as motivações em sua adoção. Enumera também os ataques mais comuns em RSSFs, organizando uma classificação da segurança oferecida por cada um desses protocolos contra os ataques anteriormente pontuados. Curiosamente, embora os autores citem algumas vezes a existência do ataque de falsa informação, eles não o incluem em sua tabela como critério para classificar os protocolos, ou seja, não leva em consideração a relevância do mesmo.

No trabalho de [Ye et al. \(2004\)](#) é proposto o mecanismo *Statistical En-route Filtering (SEF)*, capaz de realizar a filtragem da falsa informação utilizando técnicas de estatísticas, durante o encaminhamento dos pacotes da rede até a central. Também faz uso da técnica de incluir o *Message Authentication Code (MAC)* em cada pacote para garantir a autenticidade dele, verificando cada novo salto pela rede. Ele foca em detectar o pacote contendo a falsa informação e descartá-lo imediatamente.

Entretanto, o mecanismo SEF não leva em consideração a comparação entre agrupamentos, durante a verificação em cada nível, além de não identificar durante a detecção qual é a fonte da Injeção de Falsa Informações na rede, permitindo assim que o ataque continue a se propagar. Outro ponto importante, é a não resiliência do SEF quando a rede contém mais de 3% de nós maliciosos, ponto no qual o mecanismo zera completamente sua detecção.

[Zhang, Yu e Ning \(2006\)](#) também alerta para o custo adicional que a utilização de aplicações externas, como o GPS, podem causar. Além disso, com o objetivo de contribuir com a mitigação da Falsa Informação, propõe um conjunto de ferramentas para filtrar e identificar a Injeção de Falsa Informação. Esse conjunto, denominado pelos autores como “*Algoritmo de Reação baseado em Alertas para a Identificação de Nós Comprometidos*”, obriga que todos os nós da rede sejam pré-carregados com uma chave simétrica, e a medida que os eventos ocorrem, os nós próximos a ele geram alertas. Esses são assinados através de um *Message Authentication Code (MAC)*, que é computado com a chave armazenada no sensor, permitindo a central, do inglês *Base Station*, autenticar o alerta. Como normalmente mais de um nó estará próximo ao evento registrado, a central receberá vários alertas, endossando a autenticidade deles.

Entretanto, os autores não avaliaram a possibilidade de o agrupamento próximo a região onde o evento foi registrado, estar fortemente comprometido e gerando falsos alertas, mesmo estando devidamente autenticados com suas chaves. Nesse caso, a central

não teria a possibilidade de identificar a falsa informação, já que existiriam diversos alertas endossando aquele falso evento.

Já o trabalho de Zhang et al. (2010) tem como foco primordial detectar, filtrar e descartar a falsa informação, denominada pelos autores como “*false report*”. Após a identificação do pacote com a falsa informação, esse é descartado para economizar energia. Entretanto eles não buscam identificar qual é origem da Injeção de Falsa Informação, ou mesmo mitigar os atacantes, excluindo-os da rede. É fundamental destacar que a característica de identificar a fonte das injeções tem um alto valor agregado, pois sem essa capacidade, os mecanismos não seriam eficientes a ponto de frear a inundação da rede com falsas informações.

Semelhante ao trabalho anterior, Zhang et al. (2010) também utiliza o MAC computado para cada relatório transmitido através da rede, e a organiza em níveis, onde o próximo salto sempre estará em um nível mais elevado, responsável por verificar a autenticidade do MAC proveniente do nível inferior.

Nesse sentido, essa proposta pode ser eficiente para detectar injeções como falsos alertas de determinados eventos, mas seria ineficaz contra as falsas coletas, que são o foco do presente trabalho. Além disso, os autores não levaram em consideração a possibilidade do agrupamento estar fortemente comprometido, o que afetaria a verificação entre níveis.

O trabalho de Jeba e Paramasivan (2013) propõe o mecanismo *Energy Efficient Multipath Data Transfer Scheme (EEMDTS)* para mitigar a Injeção de Falsa Informação em RSSF. Esse trabalho utiliza a técnica de transferir os dados através do roteamento multicaminho, a fim de prevenir que nós comprometidos tenham acesso ao roteamento e aos dados coletados que fluem por eles.

O mecanismo EEMDTS trabalha com uma central capaz de distribuir uma chave privada para cada nó da rede e administrar os dados proveniente de todos os sensores, armazenando as informações coletadas. Após a implantação da rede, ela é organizada em agrupamentos formados através de uma arquitetura hierárquica, técnica que também é utilizada no presente trabalho. Essa é conhecida na literatura como *Cluster Head*, e conforme afirmado pelos autores, é uma das forma mais eficientes de roteamento em RSSFs.

Entretanto, o EEMDTS não objetiva identificar e mitigar a origem do ataque ou excluir o nó malicioso da rede, evitando que esse continue a realizar a Injeção de Falsas Informações. Além disso, o referido mecanismo não realiza a comparação entre agrupamentos, permitindo que agrupamentos fortemente comprometidos passem despercebidos pelo mecanismo citado.

Em Wang et al. (2014) é introduzido um novo tipo de ataque envolvendo Injeção de Falsas Informações, denominado “*Collaborative False Data Injection Attack*”. Também são propostos dois mecanismos para combatê-lo.

O primeiro mecanismo, *Geographical Information based False Data Filtering Scheme*

(*GFFS*), vincula as chaves dos sensores as suas posições geográficas, obtidas através de um dispositivo *GPS*, e são utilizadas para filtrar as falsas informações. Cada agrupamento elege um sensor como *Center-of-Stimulus (CoS)*, e esse resume as coletas recebidas em um relatório, que é encaminhado para a central. Quando os nós de encaminhamento recebem o relatório, é feita uma análise das medições, verificando a legitimidade, através das posições dos sensores e o local onde as medições foram realizadas. Caso os dados sejam inconsistentes o relatório é ignorado.

O segundo mecanismo, *Neighbor-Information based False Data Filtering Scheme (NFFS)*, funciona de maneira similar, porém utiliza posições baseadas nas informações dos nós vizinhos, e não mais o *GPS*. Ordena que os sensores, após a fase de implantação, distribuam pela rede a sua lista de vizinhos, armazenando em uma lista dinâmica capaz de sofrer alterações quando um determinado nó mudar de vizinhança.

Conforme afirmado pelos próprios autores, o *GFFS* requer a implantação de um dispositivo caro em cada sensor, o *GPS*, que aumenta o custo econômico da rede, e sua taxa de detecção é zerada quando 33% dos nós são injetores de falsa informação. Já o *NFFS* requer o armazenamento de listas que pode extrapolar a limitação de memória dos sensores, e com 37% de nós maliciosos o mecanismo zera sua detecção. Além disso, ambos desconsideram a comparação entre os agrupamentos próximos para identificar os nós maliciosos.

Finalmente nos trabalhos de *Zhu, Yang e Yu (2014)* e *Zhu, Yang e Yu (2015)*, é proposto o mecanismo *Self-checking Pollution Attackers Identification Scheme (SPAIS)*, capaz de mitigar a falsa informação, denominada pelos autores como “dados corrompidos”. Classificam a Injeção de Falsa Informação como ataque de poluição, justo por sua característica de alta propagação pela rede.

O *SPAIS*, assim como a proposta de *Jeba e Paramasivan (2013)*, organiza a rede em *arquitetura hierárquica*, que viabiliza a correta formação de agrupamentos. O objetivo do *SPAIS* com os agrupamentos é possibilitar o monitoramento em modo cooperativo dos nós em níveis inferiores pelos nós sensores nos níveis superiores.

O referido mecanismo utiliza a técnica de geração de alertas para notificar a central que existe um ataque de poluição em decorrência. Entretanto, ao não prever a possibilidade de um agrupamento estar fortemente comprometido, o *SPAIS* permite que falsos alertas sejam repassados a rede, quando vários níveis da hierarquia estiverem comprometidos.

3.4 Conclusões do Capítulo

A *Tabela 1* resume a discussão desse Capítulo, comparando os mecanismos da literatura. Nas colunas da *Tabela 1* são apresentadas características relacionadas com o *problema de pesquisa*, são elas: dependências externas, arquitetura, identificação do nó comprometido e infestação máxima suportada.

Tabela 1: Comparação entre os mecanismos da literatura, onde ND significa informação Não Disponível.

Trabalho	Dependências externas	Arquitetura	Identificação do nó comprometido	Infestação máxima suportada
SEF de Ye et al. (2004)	Não	Hierárquica	Não	3%
Mecanismo de Zhang, Yu e Ning (2006)	Não	Descentralizada	Sim	15%
Mecanismo de Zhang et al. (2010)	ND	Descentralizada	Não	ND
EEMDTS de Jeba e Paramasivan (2013)	Não	Descentralizada	Não	6%
GFFS de Wang et al. (2014)	Sim	Hierárquica	Não	33%
NFFS de Wang et al. (2014)	Não	Hierárquica	Não	37%
SPAIS de Zhu, Yang e Yu (2015)	Não	Descentralizada	Sim	ND

Fonte: O próprio autor.

Pode ser observado na coluna “dependências externas” da [Tabela 1](#) que várias propostas da literatura não apresentam essas dependências, justamente por onerar economicamente a [RSSFs](#). Já a respeito da “arquitetura”, não existe uma unanimidade entre as propostas. No entanto, vale lembrar que a [arquitetura hierárquica](#) permite menor troca de mensagens entre os nós da rede, a correta formação dos agrupamentos, além de oferecer maior controle aos mecanismos de segurança ao fornecer uma hierarquia básica entre os membros.

A “identificação do nó comprometido” é um ponto frequentemente citado na literatura por sua importância e impacto positivo no combate à Injeção da Falsa Informação. Entretanto como pode ser observado na referida coluna, poucas são as propostas que realmente trabalham esse problema. Finalmente, a resiliência de cada proposta é comparada através da última coluna, que apresenta o percentual máximo de infestação de nós maliciosos na rede. Embora algumas propostas não forneçam dados sobre esse aspecto (ND), é notável que poucos mecanismos conseguem continuar protegendo a rede quando o número de nós maliciosos superam os 30% da [RSSFs](#).

Com base nos trabalhos discutidos, é possível afirmar que a literatura carece de propostas capazes de suportar a infestação de vários nós em um agrupamento, ou até mesmo quando esses estiverem fortemente comprometidos. Desta forma, a comparação entre agrupamentos próximos pode ser utilizada para permitir que a central identifique corretamente onde estão os nós maliciosos, e conseqüentemente, quais são as fontes da Injeção de Falsas Informações.

No objetivo de contribuir com os citados desafios, o presente trabalho propõe o

mecanismo denominado *Cluster-based False Data Detection (CFDD)*, o qual é descrito na [Seção 5.4](#). O CFDD pretende aumentar a segurança da rede através da detecção dos nós Injetores de Falsas Informações, mesmo quando o agrupamento esteja fortemente comprometido. Consequentemente, prover maior resiliência a rede e as aplicações que dependem dos dados provenientes dela.

4 MÉTODO

Segundo Wazlawick (2014) o método de pesquisa é composto por uma sequência lógica de passos cujos propósitos e detalhes devem ser descritos em cada etapa, visando atingir os objetivos do trabalho. Desta forma, para a realização da pesquisa desenvolveu-se um processo com base nos métodos comumente utilizados na literatura, por exemplo em Kazienko (2013). Nesse sentido, a Figura 5 apresenta as fases do processo as quais são detalhadas a seguir.

1. Definição das Hipóteses

A primeira fase do processo de pesquisa é a **Definição das Hipóteses**, utiliza como base a lacuna da literatura identificada após o levantamento detalhado do estado da arte. Cada hipótese é dividida em duas partes a fim de validar melhor a sua veracidade.

2. Verificação Formal

Na segunda fase, denominada verificação formal, busca comprovar a veracidade das hipóteses utilizando a **Distribuição Binomial**, e assim descobrir qual é a probabilidade de cada hipótese ser verdadeira ou não. Caso a probabilidade de ocorrência seja igual ou superior a 80%, o processo de pesquisa avança para a próxima fase. Caso contrário, o processo retorna para a fase 1.

3. Configuração

A terceira fase é composta por duas tarefas: Definição do Ambiente e Definição dos Cenários, ambas servem de guia para o projeto e codificação do mecanismo, bem como para a execução das simulações.

A **Definição do Ambiente** de simulação envolve a delimitação: (i) da área em metros quadrados para implantação (*deploy*) dos sensores; (ii) do número de eventos que serão utilizados no simulador; e (iii) da segurança utilizada na comunicação entre os sensores da rede. Esses parâmetros são fixos e utilizados em todas as simulações.

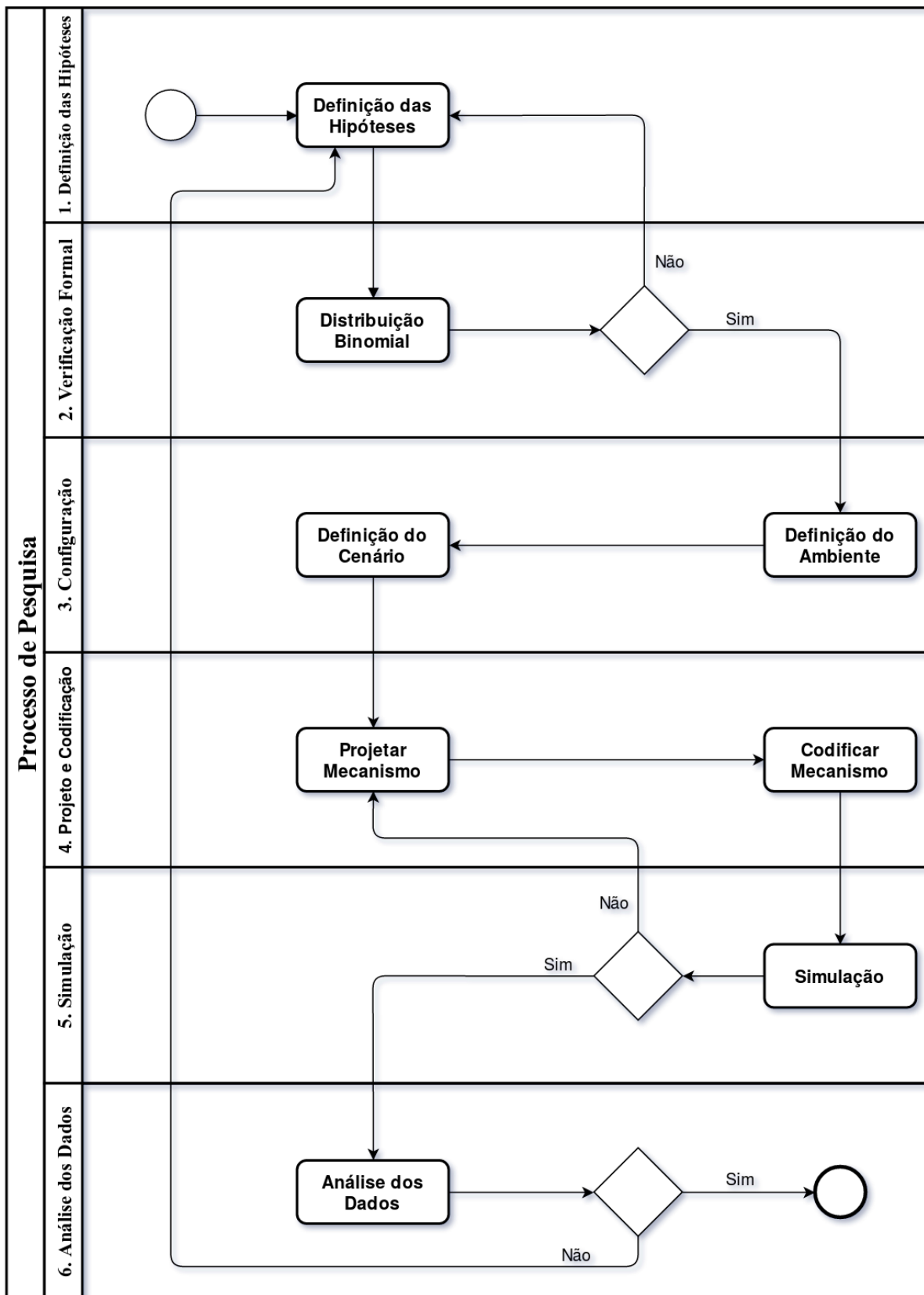
Na **Definição dos Cenários** de teste são especificados parâmetros que variam a cada simulação, como o número de nós sensores e a porcentagem de nós maliciosos. Cada par desses parâmetros implica em um cenário distinto que é simulado sobre o ambiente definido anteriormente.

4. Projeto e Codificação

Projetar e Codificar o mecanismo consiste em adequá-lo aos requisitos definidos anteriormente, para garantir a conformidade perante os objetivos da pesquisa.

O principal objetivo da fase de **projeto** é permitir um esboço do mecanismo antes mesmo de implementá-lo, definindo-se módulos com os aspectos mais importantes, isso é, o funcionamento do rastro dos pacotes na rede, os níveis de segurança a

Figura 5: Processo de pesquisa.



Fonte: O próprio autor.

serem utilizados, como seriam identificados os agrupamentos, e finalmente como os nós maliciosos seriam detectados.

Codificar o Mecanismo significa escrever o código dos algoritmos que compõem os módulos definidos anteriormente, para que esses cumpram corretamente suas funções de rastreabilidade dos pacotes, identificação dos agrupamentos e detecção dos nós injetores de falsa informação na rede.

5. Simulação

Na fase de **Simulação** são executados os diferentes cenários de testes aplicados ao ambiente definido, utilizando o mecanismo desenvolvido no trabalho. Caso a simulação seja finalizada com sucesso, o processo de pesquisa avança para a fase 6. Caso contrário é necessário reprojeter o mecanismo, considerando os resultados da simulação como referência para as alterações necessárias.

6. Análise dos Dados

A última fase do método de pesquisa é a **Análise dos Dados** obtidos a partir das simulações. Sua finalidade é confrontar os resultados perante a literatura, revelando o grau de eficiência ou não do mecanismo implementado. Caso os resultados sejam iguais ou superiores aos da literatura, os mesmos são aceitos e o processo é encerrado. Caso contrário, as hipóteses levantadas não são relevantes e, portanto, o processo é reiniciado.

5 DESENVOLVIMENTO

O presente Capítulo descreve detalhadamente como as etapas do processo de pesquisa foram desenvolvidas, ou seja, a definição das hipóteses, a validação formal, a ocorrência da etapa de configuração, o projeto e a codificação do mecanismo, e como as simulações foram executadas. As seções subsequentes seguem a ordem das atividades ilustradas na [Figura 5](#). A fase de análise dos dados é apresentada e discutida no [Capítulo 6](#).

5.1 Definição das Hipóteses

Após o levantamento do estado da arte, foram criadas hipóteses de pesquisa com o intuito de verificar quais seriam seus impactos em [RSSFs](#). Cada uma é composta por duas partes: (i) a probabilidade que o atacante tem para comprometer fortemente um agrupamento da rede; e (ii) a possibilidade de melhorar a segurança da rede contra o ataque da Injeção de Falsa Informação, sem a dependência de aplicações externas como o [GPS](#).

A primeira parte da hipótese é avaliada através de um modelo matemático. A segunda é avaliada através da análise dos resultados em comparação com a literatura. Com o propósito de definir o escopo a ser avaliado, cada hipótese levou em consideração o seguinte modelo de atacante:

- O atacante tem acesso físico e virtual ao ambiente da [RSSF](#);
- O atacante pode transitar livremente na área onde encontram-se os sensores;
- O atacante deseja comprometer a veracidade das informações transmitidas através da rede;
- O atacante fará uso da Injeção de Falsa Informação contra a rede;
- O atacante pretende capturar o maior número de sensores possível em um agrupamento, a fim de dificultar a detecção dos nós maliciosos.

Esse modelo está embasado nos contextos reais e de alta vulnerabilidade onde às [RSSFs](#) são comumente implantadas ([WANG; ATTEBURY; RAMAMURTHY, 2006](#)) ([MARGI et al., 2009](#)). Sendo assim, a seguinte hipótese de pesquisa foi selecionada para ser utilizada nesse trabalho:

- **Hipótese:** (i) o atacante tem alta probabilidade de comprometer fortemente um ou mais agrupamentos, e (ii) é possível melhorar a segurança das [RSSFs](#) utilizando a comparação entre agrupamentos contra o ataque da Injeção de Falsa Informação.

5.2 Verificação Formal

Após a definição das hipóteses de pesquisa e visando obter a probabilidade de suas ocorrências, cada uma delas foi submetida inicialmente a uma validação formal através da distribuição binomial, que permite calcular, dentro de um universo específico, qual é a probabilidade de um determinado evento ocorrer ou não. Esse passo é fundamental, pois apenas a hipótese com maior probabilidade resultante seria selecionada para continuar o processo de pesquisa.

Nesse trabalho, a distribuição binomial é utilizada para calcular a probabilidade que um dos agrupamentos de uma **RSSF** tem de sofrer um ataque, onde o inimigo captura todos ou a maior parte dos sensores pertencentes a ele. Isso significa calcular qual é a probabilidade de um agrupamento ser fortemente comprometido por um atacante. A **Equação 5.1**, proveniente do livro de **Meyer (1985)**, apresenta a definição dessa distribuição.

$$P(X = k) = \binom{n}{k} \cdot p^k \cdot q^{n-k} \quad (5.1)$$

Fonte: **Meyer (1985)**.

$$\text{Onde: } \begin{cases} n & = & \text{N}^\circ \text{ de testes} \\ k & = & \text{N}^\circ \text{ de sucessos} \\ p & = & \text{Probabilidade de sucesso} \\ q & = & \text{Probabilidade de fracasso: } q = 1 - p \end{cases}$$

Logo, considerando que um atacante deseja comprometer a veracidade das informações transmitidas através da rede, conforme o **modelo de atacante** é razoável assumir que a chance de captura de um nó sensor qualquer (s_i) é alta.

Partindo dessa suposição, pode-se considerar um agrupamento A_x com s nós sensores sujeitos a ação de um atacante, e assumindo uma probabilidade de 0,8 para a captura e adulteração de qualquer s_i , pode-se obter as probabilidades associadas à captura de cada sensor da rede, utilizando a distribuição de probabilidade binomial. Assumindo os parâmetros ponderados, a **Tabela 2** apresenta o cálculo dessa distribuição.

$$\text{Parâmetros: } \begin{cases} p & = & 0,8 \\ n & = & 10 \end{cases}$$

Agora, aplicando um somatório sobre esses resultados, é possível obter as probabilidades resultantes desse ataque. A **Equação 5.2** exibe o axioma do somatório.

$$\sum_{k=0}^n \binom{n}{k} \cdot p^k \cdot q^{n-k} = 1. \quad (5.2)$$

Tabela 2: Demonstração dos resultados do cálculo da Distribuição Binomial para cada variação de X .

X	$P(X)$
1	0,000004
2	0,000073
3	0,000786
4	0,005505
5	0,026424
6	0,088080
7	0,201326
8	0,301989
9	0,268435
10	0,107374

Fonte: O próprio autor.

Então, através de uma análise, e considerando X como o número de nós sensores capturados, constata-se que a probabilidade da captura de mais da metade dos nós sensores pertencentes a um agrupamento A_x é muito alta. Isso pode ser verificado na [Equação 5.3](#).

$$P\left(X > \frac{|N_s| \in A_x}{2}\right) \approx 0,90. \quad (5.3)$$

$$\text{Onde: } \begin{cases} X &= \text{N}^\circ \text{ de nós sensores capturados} \\ N_s &= \text{N}^\circ \text{ de nós sensores total} \\ A_x &= \text{Agrupamento } x \end{cases}$$

A probabilidade resultante é de aproximadamente 90%. Isso demonstra a elevada chance de vários nós sensores serem capturados, comprometendo todo o agrupamento e dificultando a distinção entre nós maliciosos e genuínos. Portanto, é necessário um método alternativo de detecção baseado não apenas em informações originadas no próprio agrupamento. Desse modo, a análise apresentada evidencia a relevância da comparação entre agrupamentos próximos, a fim de identificar a Injeção de Falsa Informação.

Vale ressaltar que a probabilidade de 0,8 citada, provem da hipótese aceita, pois anteriormente outras hipóteses com probabilidades menores, por exemplo 0,5, foram testadas mas não aprovadas, por não apresentarem probabilidade resultante condizente ao [modelo de atacante](#), o qual expõe a [RSSF](#) a um contexto de alta vulnerabilidade.

5.3 Configuração

Para a correta execução das simulações, e como guia para o projeto e codificação do mecanismo, é necessário realizar previamente dois passos, a Definição do Ambiente e a Definição dos Cenários.

5.3.1 Definição do Ambiente

O simulador escolhido exige a criação de um ambiente onde a **RSSF** será implantada (*deployed*), para isso é necessário delimitar os seguintes parâmetros:

- (I) A área em m^2 para implantação dos sensores;
- (II) O número de eventos que serão utilizados no simulador;
- (III) A segurança utilizada na comunicação entre os sensores da rede.

Na **Tabela 3** são apresentados os valores para os parâmetros (I) e (II), e os algoritmos de segurança adotados no parâmetro (III). Esses parâmetros são imutáveis e utilizados em todas as simulações.

Tabela 3: Parâmetros imutáveis utilizados nas simulações do mecanismo.

Parâmetro	Valores
(I) Área	300 m^2
(II) Número de eventos	100.000.000 de eventos
(III) Segurança	CBC-MAC e AES

Fonte: O próprio autor.

Conforme visto na **Tabela 3**, a área onde a rede será implantada tem uma abrangência de 300 m^2 , o que permite boa distribuição dos nós sensores. Os eventos do simulador foram definidos em 100 milhões para permitir boa quantidade de informações trocadas entre cada sensor da rede, além de fornecer milhares de trocas de pacotes entre os membros da rede por simulação.

Finalmente, é importante salientar que os níveis de segurança utilizados no trabalho englobaram a autenticidade do pacote via **CBC-MAC**, para garantir que a informação coletada pelo sensor realmente provenha de um nó autêntico, e a integridade dos dados por meio da criptografia do **AES**. Ambos são detalhados na **Seção 5.4**.

5.3.2 Definição do Cenário

Após o ambiente ser delimitado, é necessário definir os cenários de teste. Nessa tarefa são especificados parâmetros que variam a cada simulação, como o número de nós sensores e a porcentagem de nós maliciosos presentes na rede.

Para submeter o mecanismo proposto aos mais diversos cenários de vulnerabilidade e infecção, foram definidos três números de nós sensores (49, 144 e 255), e três valores para a porcentagem de nós maliciosos (20%, 40% e 75%). As três possibilidades de número de sensores visam simular redes de pequeno, médio e grande porte, respectivamente.

Cada par desses parâmetros implica em um cenário distinto que é simulado no ambiente definido anteriormente. A [Tabela 4](#) apresenta os nove cenários de teste resultantes da combinação desses parâmetros.

Tabela 4: Número de nós maliciosos por rede, baseados na porcentagem de infecção, 20%, 45% e 75%.

Rede	20%	45%	75%
49	9	22	36
144	28	64	108
255	51	114	191

Fonte: O próprio autor.

É importante destacar que cada um desses cenários foi executados 100 vezes, a fim de extrair dados consistentes das simulações. Além disso, cenários de alta infecção, como 75%, são um diferencial deste trabalho perante os demais encontrados na literatura.

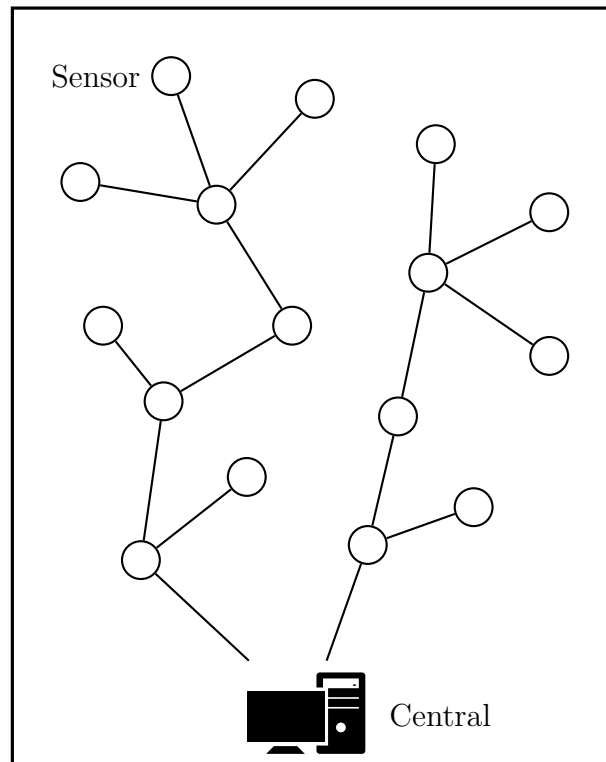
5.4 Projeto e Codificação

As fases de Projeto e Codificação são essenciais para a correta implementação do mecanismo desenvolvido neste [TCC](#), denominado *Cluster-based False Data Detection (CFDD)*. Além disso, são passos que antecedem a simulação e extração de conhecimento dos dados do mecanismo [CFDD](#).

5.4.1 Projeto do Mecanismo

O principal objetivo da fase de projeto é permitir um esboço do mecanismo antes mesmo de implementá-lo. Nessa fase foi definida a arquitetura da [RSSF](#) que seria utilizada no presente trabalho. Pode ser observado na [Figura 6](#) que a rede possui arquitetura hierárquica e é composta por um conjunto de nós sensores e uma central. Os nós sensores capturam informações do ambiente e geram pacotes que são encaminhados até a central. A central é uma máquina (computador) capaz de processar toda a gama de informações recebida da rede. A escolha de usar uma central, ao invés de fazer todo o processamento em um nó sensor da rede, deve-se ao fato dos sensores apresentarem limitações tanto de *hardware* quanto de capacidade energética.

Figura 6: Organização da **RSSF** em arquitetura hierárquica, composta por uma central e os nós sensores.



Fonte: O próprio autor.

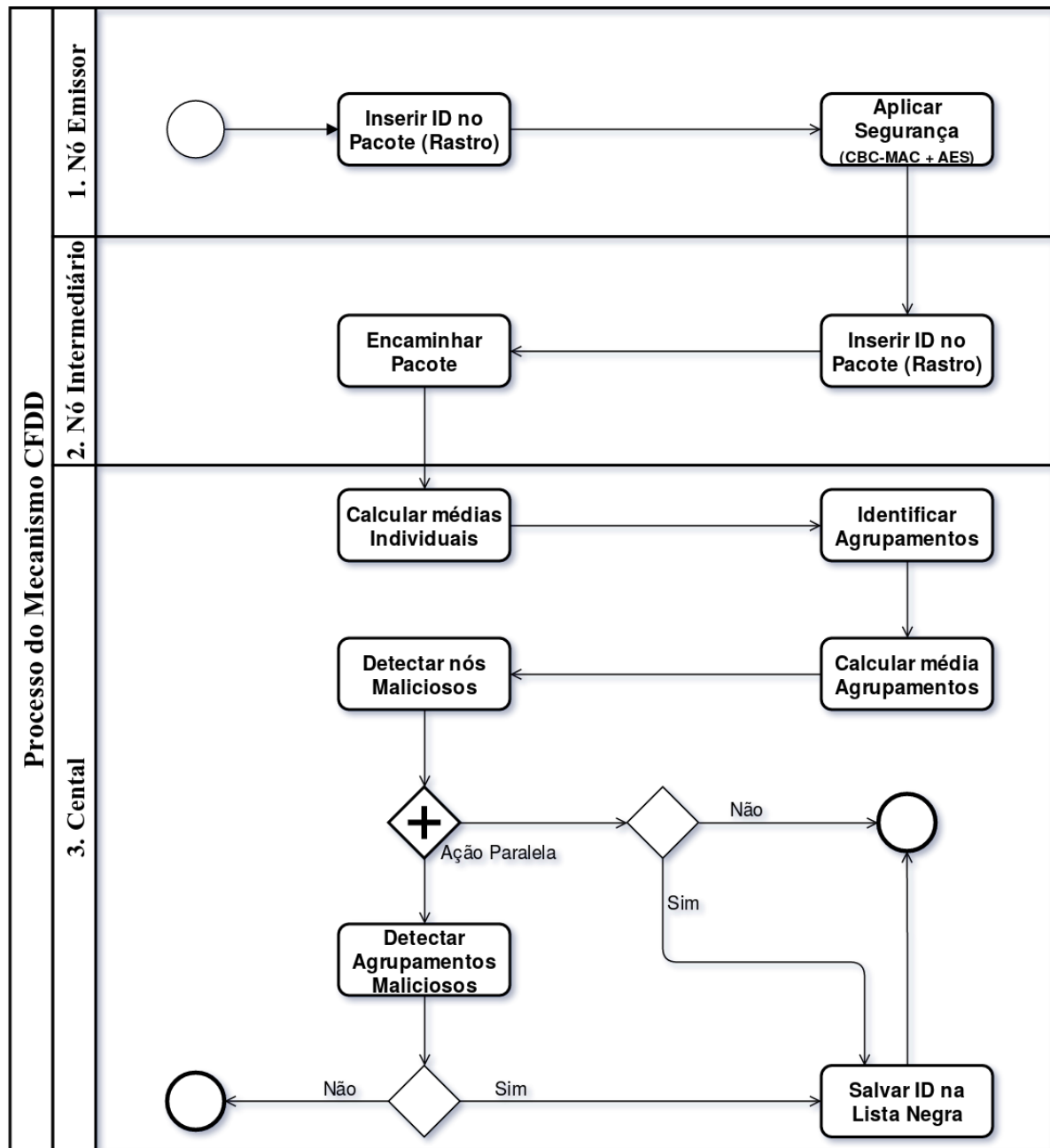
Como pode ser observado na [Figura 7](#), o mecanismo Cluster-based False Data Detection (**CFDD**) é dividido em etapas, as quais são executadas nos nós sensores e na central da rede.

Para permitir a rastreabilidade de todos os pacotes transmitidos pela rede, cada nó sensor no caminho do pacote adiciona o seu Identificador (**ID**). Além disso, o nó sensor emissor do pacote adiciona mecanismos de segurança. A segurança na comunicação sempre foi uma preocupação, tornando-se um requisito indispensável durante o projeto do mecanismo. Optou-se por utilizar o TinySec, padrão de implementação recomendado pelo próprio TinyOS para às **RSSFs**. Esse padrão necessita de autenticidade e integridade, e a sua adoção visa fornecer as camadas básicas de segurança a rede.

Dois algoritmos, que cobrem aspectos complementares na segurança da informação, foram escolhidos para serem utilizados no TinySec, o *Cypher Block Chaining Message Authentication Code* (**CBC-MAC**) para fornecer autenticidade em cada pacote trocada na rede, e o *Advanced Encryption Standard* (**AES**) para criptografar esses dados, garantindo a integridade dos mesmos.

É importante ressaltar que esses níveis de segurança não são suficientes para barrar o ataque da Injeção de Falsa Informação, pois esse não compromete a integridade ou a autenticidade dos pacotes, mas sim a veracidade da informação coletada por cada nó

Figura 7: Etapas do mecanismo CFDD. Ressalta-se que a maioria das etapas do mecanismo são executadas somente na central.



Fonte: O próprio autor.

sensor.

A maioria das etapas do mecanismo são executadas na central. A primeira delas é o cálculo das médias individuais, ou seja, a soma de todas as coletas de um determinado nó sensor da rede pela quantidade de coletas desse nó. A segunda etapa é a identificação dos agrupamentos da rede, onde a central processa os pacotes recebidos, a fim de mapear os ID e localizar cada nó sensor. Esse mapeamento é feito comparando o rastro do pacote e formando padrões (caminhos), onde vários pacotes percorreram o mesmo caminho. Esse

caminho é a trilha por cada salto que o pacote percorreu na rede, formando assim uma lista de IDs, denominada Rastro do Pacote.

Ainda na central, é iniciado o processo de detecção dos nós injetores de falsa informação. Nesse momento, é analisada cada coleta, a fim de identificar anomalias e descobrir quais são os nós maliciosos.

Com esse intuito um limiar de confiança foi estipulado, baseado na porcentagem de variação da média entre todas as detecções do nó sensor, perante a média do agrupamento ao qual ele pertence. Em seguida, os agrupamentos são comparados entre si, também através da sua média, desde que esses estejam geograficamente próximos.

Logo, utilizando os resultados de todas as comparações, o mecanismo CFDD descobre se um determinado nó sensor, bem como se o seu agrupamento, é malicioso ou não. Caso positivo, o ID é salvo na Lista Negra gerada pelo mecanismo.

5.4.2 Codificação do Mecanismo

A codificação do mecanismo consiste em implementar as etapas, permitindo a rastreabilidade dos pacotes, a garantia da segurança básica na rede, a identificação dos agrupamentos, e a detecção da Injeção de Falsa Informação. É importante destacar que o mecanismo CFDD foi implementado sobre uma base de código existente, de autoria do professor Dr. Juliano Fontoura Kazienko, que fornece uma camada básica de segurança chamada TinySec.

Conforme demonstrado na Figura 7, o CFDD é executado nos nós da rede e na central. As etapas que são executadas nos sensores foram implementadas na linguagem *network embedded systems C (nesC)*, já aquelas presentes na central, foram implementados na linguagem Python, versão 2.7.

Para implementar a etapa de rastreabilidade dos pacotes, a estrutura dele foi alterada, incluindo um novo campo do tipo vetor, chamado de *msg_trail*, para ser preenchido a medida que o pacote flui pela rede. Ou seja, a cada novo salto o ID do nó atual é inserido no final do vetor, formando assim o rastro do pacote. A Figura 8 demonstra a estrutura do pacote após a modificação.

Nesse sentido, a Figura 9 apresenta como o rastro do pacote é composto enquanto ele flui pela rede. O rastro é fundamental para todo o mecanismo, pois como se tratam de RSSFs estáticas, existe então um caminho comum por onde os pacotes de um mesmo agrupamento passam. O Apêndice B apresenta o *script* que identifica os rastros dos pacotes.

Um agrupamento em uma RSSF é composto por nós sensores que estão próximos um do outro e apresentam o mesmo caminho de comunicação entre eles e a central da rede. Por exemplo, os pacotes oriundos dos nós sensores 1 e 2 possuem os seguintes rastros: $[1, 5, 10, 12]$ e $[2, 5, 10, 12]$. Portanto, com base nos rastros, é possível inferir que os nós 1 e 2 pertencem ao mesmo agrupamento porque, a exceção do nó origem, todos os demais

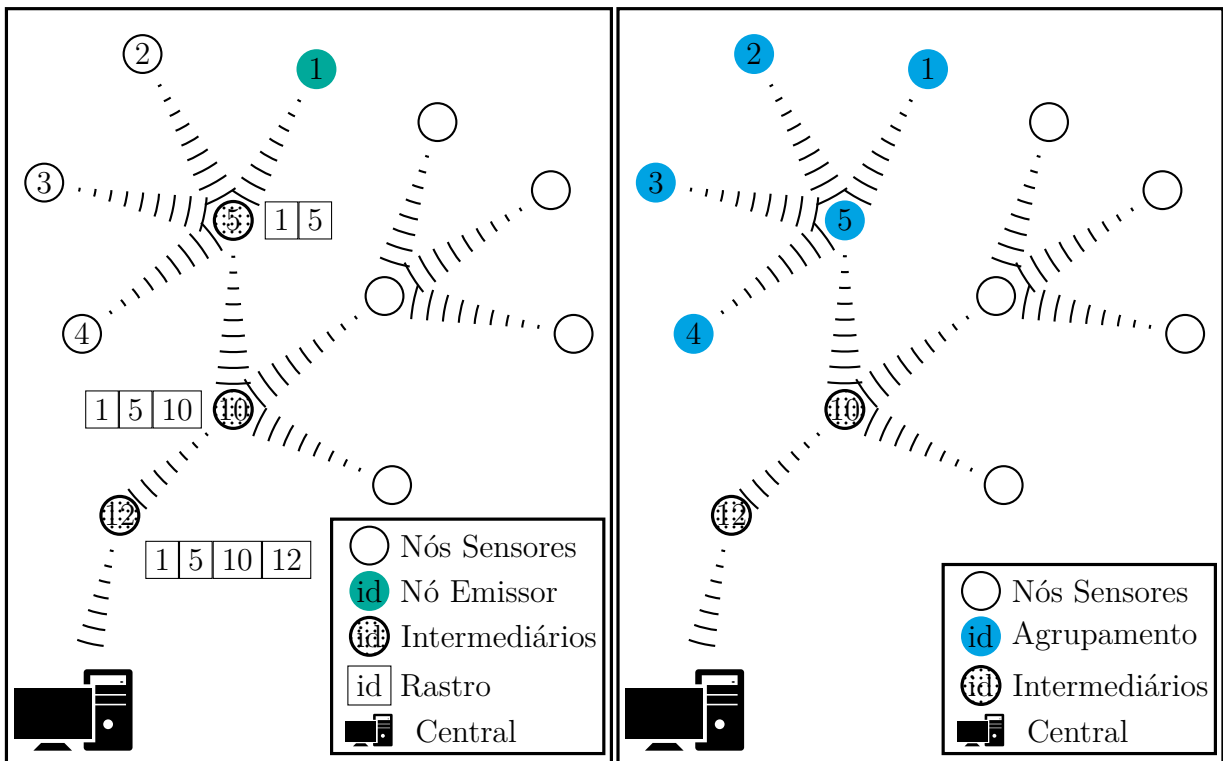
Figura 8: Inclusão do campo `msg_trail` na estrutura dos pacotes de comunicação da rede, para armazenar o rastro dos pacotes.

```
typedef nx_struct oscilloscope {
    nx_uint16_t id; // Mote ID of sending mote.
    nx_uint16_t id_msg; // Message identifier.
    nx_uint16_t readings[NREADINGS]; // Readings vector.
    nx_uint16_t hops; // The number of hops.
    nx_uint8_t msg_trail[0]; // Message trail.
    nx_uint8_t mac[]; // Message Authentication Code.
} oscilloscope_t;
```

Fonte: O próprio autor.

Figura 9: Representação da formação do rastro do pacote. O ID de cada nó intermediário é armazenado enquanto o pacote flui pela RSSF.

Figura 10: Detecção dos agrupamentos pelo CFDD. Para isso é utilizado o rastro de cada pacote proveniente do agrupamento.



Fonte: O próprio autor.

Fonte: O próprio autor.

nós presentes no rastro são comuns. O Apêndice C apresenta o *script* que identifica os agrupamentos da RSSF.

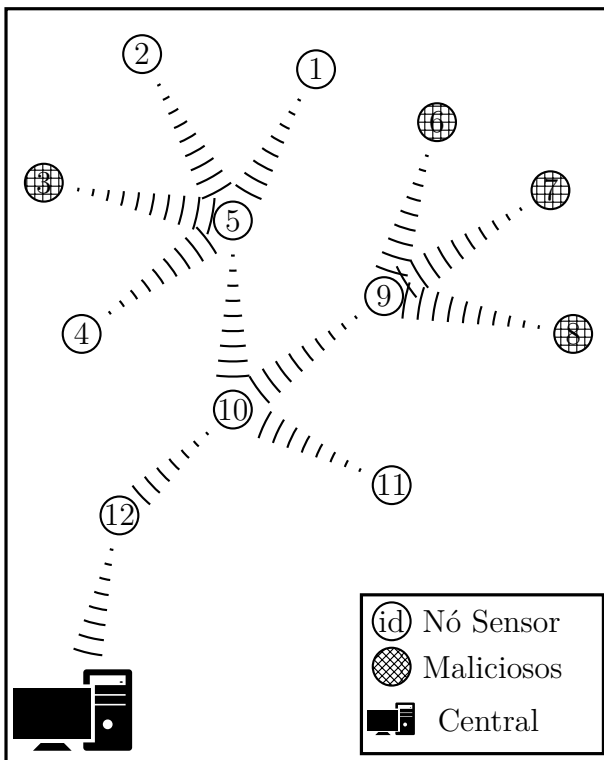
Com os agrupamentos devidamente mapeados, a central inicia a etapa para analisar cada coleta recebida da rede, a fim de detectar anomalias nos dados, o que indicaria a presença do ataque da Injeção de Falsa Informação. A Figura 10 demonstra um agru-

pamento detectado pelo CFDD.

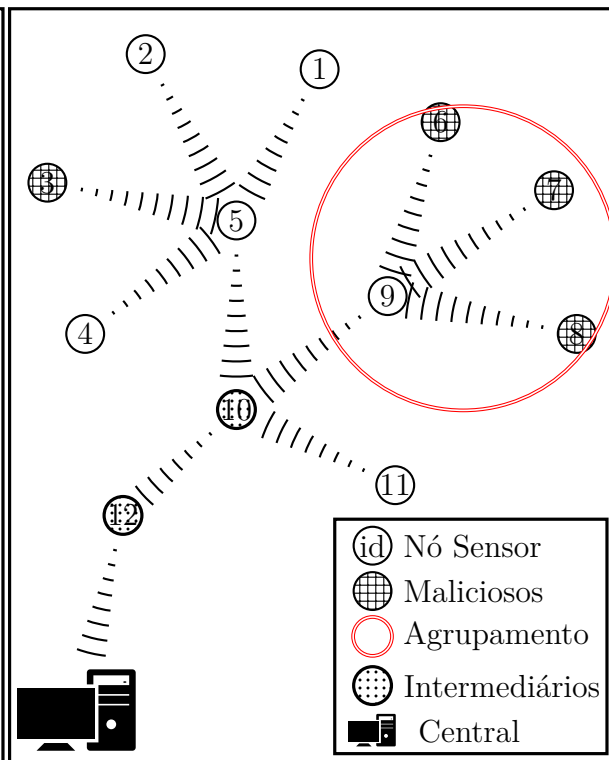
Inicialmente essa análise realiza o cálculo da média aritmética das coletas realizadas por cada nó. Posteriormente, é calculada a média aritmética de cada agrupamento, utilizando a média de cada um dos nós sensores pertencentes a ele. Em seguida, compara a média de cada nó sensor com a média do seu agrupamento, respeitando um limiar de 20% para mais ou menos. Esse limiar foi adotado como estratégia para minimizar os falsos positivos gerados pelo CFDD. Ao final, todos os nós que apresentam média que desrespeitam esse limiar são considerados nós maliciosos e, portanto, têm seus IDs salvos na Lista Negra. A Figura 11 apresenta a rede com os nós maliciosos identificados, sendo eles: 3, 6, 7 e 8. O Apêndice D apresenta o *script* que identifica os nós maliciosos presentes na RSSF.

Figura 11: Detecção dos nós maliciosos presentes na RSSF pelo mecanismo CFDD. Utilizando as médias unitárias contra a média do agrupamento ao qual cada nó pertence.

Figura 12: Detecção do agrupamento malicioso, fortemente comprometido, pelo CFDD, utilizando as médias dos agrupamentos vizinhos para comparação e detecção de anomalias.



Fonte: O próprio autor.



Fonte: O próprio autor.

Após o CFDD conhecer quais são os nós maliciosos, é iniciada a confrontação dessas informações para identificar os agrupamentos fortemente comprometidos. Para isso o mecanismo analisa quais são os agrupamentos próximos uns dos outros, utilizando o rastro dos seus pacotes. Por exemplo, na rede exibida na Figura 10, o agrupamento composto pelos nós [1, 2, 3, 4, e 5] teria sua média confrontada com o agrupamento

próximo, composto pelos nós [6, 7, 8 e 9]. Através dessa comparação, o alto grau de infestação presente nesse último agrupamento seria detectado, conforme apresentado na [Figura 12](#). Ao constatar um agrupamento malicioso, todos os IDs dos nós pertencentes a ele são salvos na Lista Negra.

5.5 Simulação

As simulações permitem submeter o mecanismo desenvolvido a cada um dos nove cenários de teste criados. Assim, ao fim de cada simulação, são exportados e salvos os dados resultantes, os quais são tratados visando a extração de conhecimento. Caso a simulação seja finalizada com sucesso, o processo de pesquisa avança para a última fase. Caso contrário, é necessário reprojeter o mecanismo, considerando os resultados da simulação como referência para as alterações necessárias.

A definição de parâmetros de simulação é uma prática utilizada na literatura para obter dados mais confiáveis, controlando exatamente da mesma forma cada uma das variáveis da simulação. Além disso, esses parâmetros permitem estabelecer um escopo claro dos testes realizados com o mecanismo proposto, o que oferece ao leitor uma compreensão holística e a possibilidade de reproduzir o trabalho, utilizando informações sólidas e precisas. A [Tabela 5](#) apresenta os parâmetros de simulação utilizados.

Tabela 5: Parâmetros utilizados nas simulações do mecanismo proposto.

Parâmetro	Padrão
Número de cenários de teste	9
Número de simulações por cenário	100
Área de implantação da RSSF	300 m ²
Número de eventos por simulação	100.000.000
Número de nós	49, 144 ou 255
Porcentagem nós maliciosos	20%, 45% ou 75%
Variável de temperatura	25, 26, 27, 28, 29 ou 30
Fonte do pacote	Nó sensor
Destino do pacote	Central
Tamanho do pacote	16 bytes
Tipo do nó sensor	MICAz

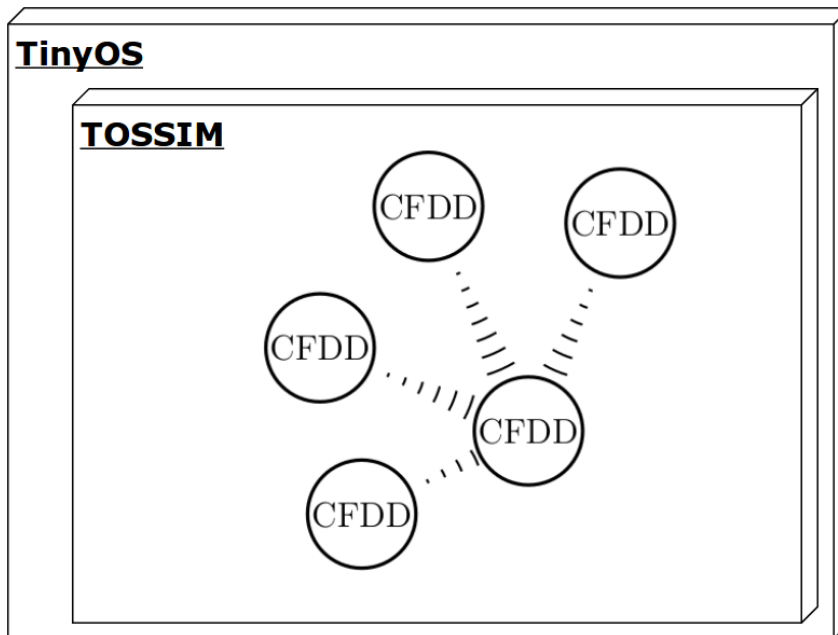
Fonte: Adaptado de [Kazienko et al. \(2015\)](#).

Na fase de implantação da rede, o simulador é responsável por eleger aleatoriamente os nós maliciosos. O número de nós depende invariavelmente da porcentagem definida em cada cenário. Além disso, o simulador armazena em uma lista o identificador (ID) de cada um dos nós maliciosos para posterior conferência da taxa de falsos positivos. Vale ressaltar que o mecanismo [CFDD](#) não tem acesso a essa lista.

Durante a simulação, cada nó sensor captura a temperatura do ambiente em intervalos definidos pelo simulador. A temperatura de cada nó sensor pode variar de modo aleatório dentro do conjunto \mathbb{N} , onde $\mathbb{N} = \{25, 26, 27, 28, 29 \text{ e } 30\}$. Já os nós maliciosos somam deliberadamente 20°C a esse valor aleatório, conferindo a característica de falsa informação na coleta original de temperatura.

O simulador **TOSSIM**, escolhido para esse trabalho, permite o controle de cada nó da rede, implantando o código do mecanismo **CFDD** em todos eles. O **TOSSIM** utilizado é o fornecido pela versão 2.2 do TinyOS. O TinyOS é um Sistema Operacional (**SO**) desenhado especificamente para **RSSFs**. A referida versão do TinyOS exige, por questão de compatibilidade, a utilização do **SO** Ubuntu versão 11.04, denominado *Natty Narwhal*. A **Figura 13** demonstra as camadas necessárias para executar as simulações.

Figura 13: Representação das camadas necessárias para simular o mecanismo **CFDD**. Imagem meramente ilustrativa.



Fonte: O próprio autor.

Cada simulação é guiada através de um *script* de automação escrito na linguagem Python versão 2.7, o qual é denominado *simula.py*, e pode ser encontrado no **Apêndice A**. O *simula.py* contém as instruções básicas para o simulador, como: número de nós presentes na rede, número de eventos da simulação e os nomes para os arquivos de *log*. Adicionalmente, ele é responsável por chamar outros *scripts* para processamento dos dados após o fim da simulação. Esses *scripts* são responsáveis, por exemplo, por salvar em arquivos separados os pacotes enviados e recebidos, eliminar pacotes repetidos (provenientes da taxa de re-entrega do simulador), calcular as médias individuais dos sensores, calcular a média dos agrupamentos, e gerar estatísticas gerais sobre a simulação. É importante ressaltar que todos os *scripts*, a exceção do *simula.py*, foram executados na central

da rede, que é uma máquina com alta capacidade computacional, muito diferente das severas limitações de hardware e bateria do sensor MICAz. Alguns desses *scripts* podem ser consultados no [Apêndice B](#), [Apêndice C](#) e [Apêndice D](#).

Nesse trabalho foram executadas 100 simulações para cada um dos nove cenários de teste previstos. Cada simulação, seguindo a norma estabelecida, utilizou 100 milhões de eventos, ou seja, foram executados ao total 90 bilhões de eventos no simulador [TOSSIM](#), totalizando cerca de 200 horas de execução.

Finalmente, após a execução de todas as simulações, fez-se necessário a cuidadosa análise do montante de dados obtidos. A extração de conhecimento proveniente dos dados das simulações permite que os resultados sejam analisados e comparados com a literatura existente, atestando o nível de qualidade da solução proposta. Os resultados dessa análise são apresentados no próximo Capítulo.

6 RESULTADOS

O presente Capítulo descreve os resultados obtidos na fase de simulação do mecanismo [CFDD](#). Inicialmente, todos os cenários são apresentados em um gráfico, o qual exibe uma visão geral da eficiência do mecanismo em relação às variações da infestação na rede por nós maliciosos. Em seguida, os mesmos dados são exibidos em um gráfico de linhas, a fim de mostrar a curva de tendência. Finalmente, esses resultados são confrontados com os presentes na literatura, revelando o grau de eficiência do mecanismo [CFDD](#).

Todos os dados foram analisados e tratados seguindo os parâmetros descritos na [Tabela 6](#).

Tabela 6: Parâmetros utilizados na análise dos resultados obtidos.

Parâmetro	Padrão
Amostra dos dados	100 simulações
Índice de confiança	95%
Tipo de média	Média aritmética

Fonte: O próprio autor.

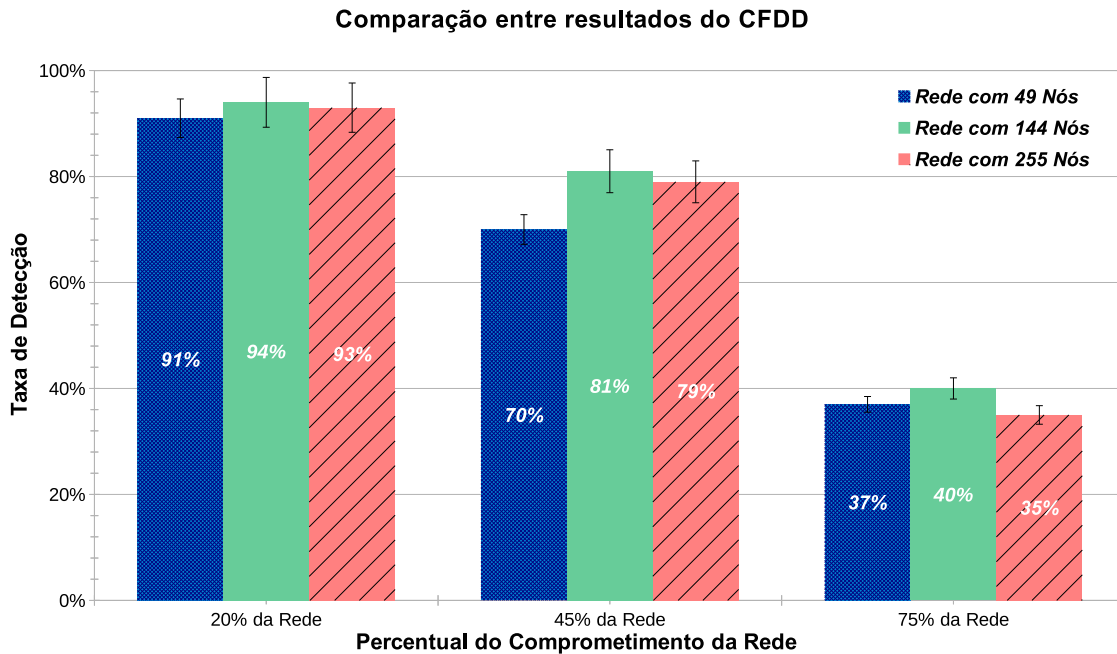
Os gráficos apresentados a seguir possuem três informações em comum: (i) a taxa da detecção (%) de nós maliciosos, representada no eixo Y ; (ii) porcentagem de comprometimento na rede, no eixo X , e (iii) o índice de confiança utilizado no trabalho, que segue a recomendação da literatura em [Wazlawick \(2014\)](#) e [Kazienko et al. \(2015\)](#).

Com o intuito de confrontar os resultados de cada cenário, o gráfico apresentado na [Figura 14](#) exibe uma visão holística da capacidade do mecanismo [CFDD](#) em detectar o ataque da Injeção de Falsa Informação, em cada variação da porcentagem de nós maliciosos na [RSSF](#).

Os resultados demonstram que, levando em consideração o índice de confiança calculado, a taxa de detecção de nós maliciosos é muito semelhante para diferentes tamanho de redes (49, 144 ou 255 nós). A exceção é o cenário com 49 nós e porcentagem de comprometimento igual a 45%, o qual apresenta uma taxa de detecção menor do que em redes com 144 e 255 nós. É importante ressaltar que, embora os dados de cada simulação tenham sido cuidadosamente reavaliados, não foram encontrados indícios de erro ou qualquer problema que tenha exercido influência sobre esse resultado.

O gráfico exibido na [Figura 15](#) apresenta os dados mostrados no gráfico da [Figura 14](#) utilizando linhas, o que permite visualizar as tendências existentes na variável “taxa de detecção” à medida que o número de nós maliciosos aumenta. Não é possível afirmar que essa tendência perduraria em outros cenários de testes. Entretanto, é interessante perceber que o comportamento do mecanismo está associado a variação do

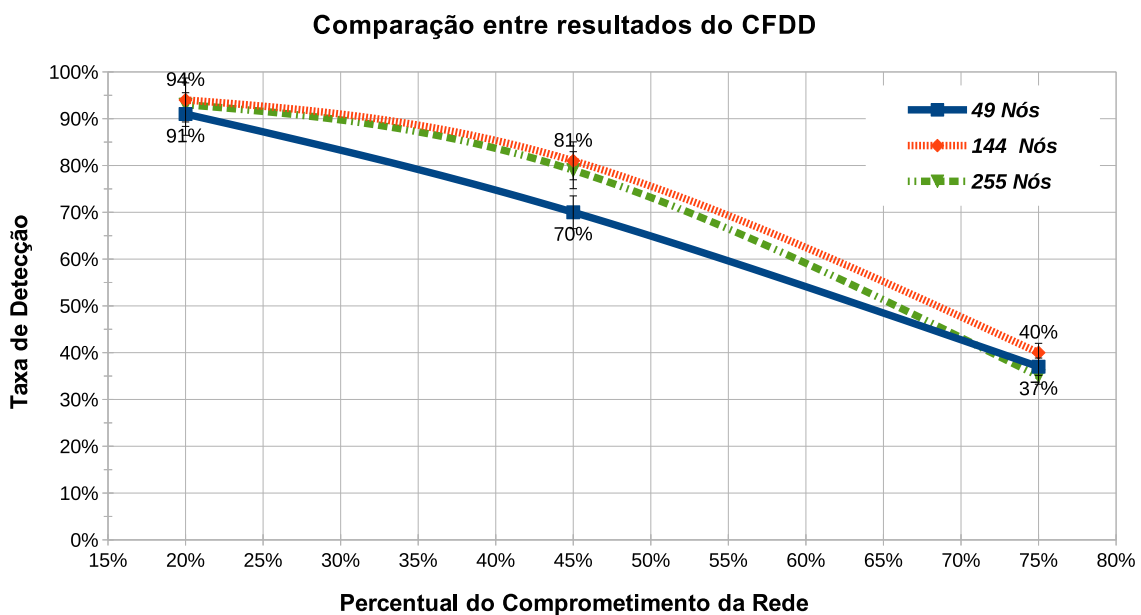
Figura 14: Comparação dos resultados da eficiência do CFDD.



Fonte: O próprio autor.

comprometimento da rede, o que demonstra a importância de identificar o quanto antes os nós Injetores de Falsa Informação, aumentando a relevância do problema abordado.

Figura 15: Comparação dos resultados do CFDD utilizando linhas para analisar suas tendências.

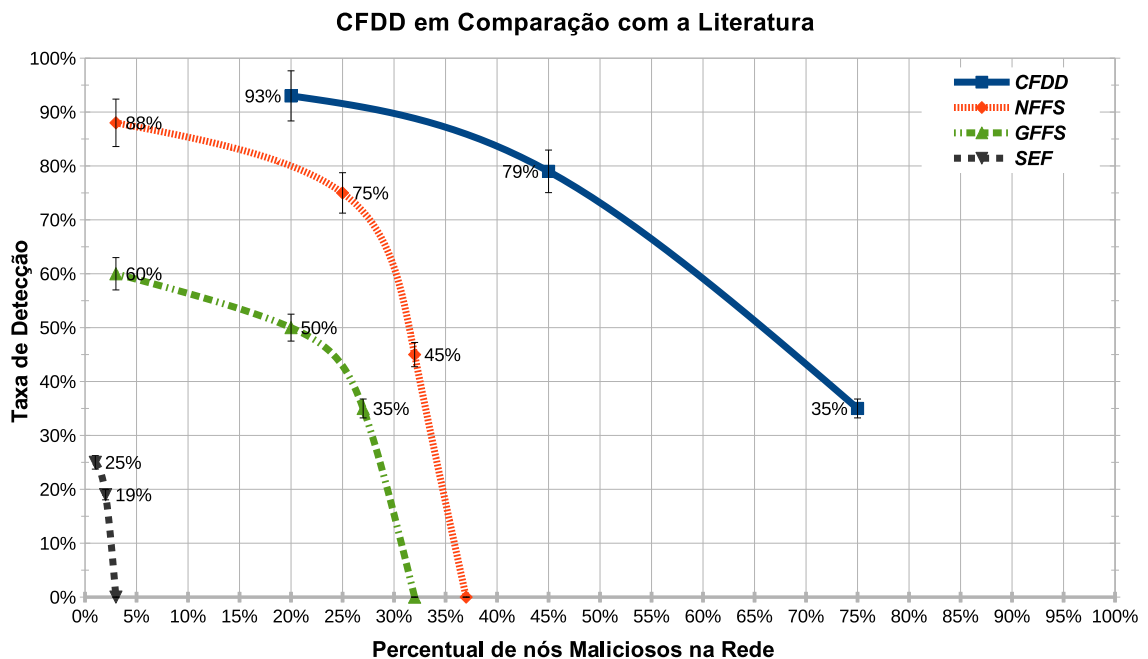


Fonte: O próprio autor.

Os resultados apresentados na [Figura 14](#) e na [Figura 15](#) demonstram que a eficiência de detecção do [CFDD](#) se mantém alta, ficando acima dos 70% mesmo quando a rede possui 45% de nós Injetores de Falsa Informação, indicando que o mecanismo é eficiente nesses cenários. Entre 45% e 60% de infestação na rede, a taxa de detecção do mecanismo descende para aproximadamente 55%, o que ainda é uma eficiência interessante, levando em consideração que mais da metade dos nós sensores já estão comprometidos pelo ataque. Por fim, entre 60% e 75% de comprometimento da rede, a taxa de detecção descende aproximadamente 35%, levando em consideração o índice de confiança calculado. Isso significa que o [CFDD](#) é capaz de identificar 35% de todos os nós maliciosos em uma cenários onde 75% da [RSSF](#) está infectada, ou seja, mesmo quando a rede está fortemente comprometida.

Para validar os resultados obtidos pelo [CFDD](#), eles foram confrontados com os apresentados na literatura. A [Figura 16](#) mostra o gráfico que compara os resultados do [CFDD](#) com mecanismos da literatura que apresentam dados que permitem tal comparação.

Figura 16: Comparação do mecanismo [CFDD](#) com mecanismos da literatura.



Fonte: O próprio autor.

Ao analisar os dados constata-se que o mecanismo *Statistical En-route Filtering (SEF)*, proveniente do trabalho de [Ye et al. \(2004\)](#), apresenta taxas de detecção de 25% quando a rede está com 1% de infestação, 19% para 2%, e zero ao atingir 3% de contaminação na rede. Já o mecanismo *Geographical Information based False Data Filtering Scheme (GFFS)*, do trabalho de [Wang et al. \(2014\)](#), consegue detectar 60% dos nós maliciosos quando a rede continha 3% de contaminação, 50% quando contem 20%, 15%

para 30%, e zera sua detecção ao atingir os 33% de nós Injetores de Falsa Informação. O terceiro mecanismo da literatura é o *Neighbor-Information based False Data Filtering Scheme (NFFS)*, também proveniente do trabalho de Wang et al. (2014), atinge 88% quando 3% da rede está infectada, 75% quando existem 25% de contaminação, 45% para 32% da rede, e com 37% de nós maliciosos o mecanismo zera sua detecção.

Perante esses dados, o mecanismo *Cluster-based False Data Detection (CFDD)* é superior aos demais trabalhos da literatura em todos os cenários analisados, principalmente quando a Rede de Sensores Sem Fio (RSSF) apresentava altas taxas de nós maliciosos. Destaca-se a resiliência do CFDD ao suportar infestações superiores a 40% da rede, sem zerar sua detecção nos cenários testados, característica não existente nos mecanismos SEF, GFFS e NFFS.

É fundamental retomar a hipótese de pesquisa do trabalho que afirma ser possível melhorar a segurança das RSSFs, utilizando a comparação entre os agrupamentos, mesmo quando essa estiver sob-ataque e fortemente comprometida. Esse trabalho demonstra que o mecanismo CFDD permite aumentar a segurança da rede em 35% no pior cenário simulado. Adicionalmente, esse resultado é expressivo quando comparado com a literatura, conforme demonstrado na Figura 16.

Unindo os resultados obtidos nas simulações com os da verificação formal, é corroborada a veracidade da hipótese de pesquisa utilizada. Demonstrando, assim, a relevância no uso das estruturas da própria RSSF como ferramenta contra ataques que envolvam a Injeção da Falsa Informação na rede.

6.1 Limitações do Trabalho

Segundo Wazlawick (2014) em um trabalho científico ou acadêmico é fundamental a indicação detalhada de suas limitações (escopo), permitindo ao leitor conhecer de antemão as fronteiras do trabalho. Além disso, auxilia na reprodutibilidade e na correta comparação dos resultados obtidos com os da literatura. Nesse sentido, a seguir são pontuadas as limitações do presente trabalho.

O mecanismo CFDD foi projetado para RSSFs:

- que coletam dados numéricos, como: temperatura, pressão, geolocalização e etc;
- geograficamente pequenas (áreas com centenas de metros quadrados e não quilômetros);
- que contenham arquitetura hierárquica;
- homogêneas, contendo um mesmo tipo de nó sensor;
- do tipo estáticas, cujos nós não apresentam perfil de mobilidade;

- que contenham sensores do tipo MicaZ ou Mica2 (as simulações contemplaram ambos, entretanto estimasse que o mecanismo possa funcionar com pequenas adaptações em outros tipos de sensores).

O mecanismo [CFDD](#) não foi desenhado para ser aplicado em [RSSFs](#):

- que coletam dados não numéricos;
- que contenham arquitetura descentralizada;
- que sejam redes heterogêneas, contendo variados tipos de nó sensor;
- que apresentem comunicação todos-para-todos (do inglês *broadcast*);
- que apresentem comportamento de mobilidade nos nós sensores.

7 CONCLUSÃO

É de conhecimento público que o avanço científico e a popularização de alta tecnologia estão revolucionando o mundo em todos os aspectos possíveis. Por exemplo, segundo [Borgia \(2014\)](#), até 2025 o número de objetos inteligentes conectados excederá 7 trilhões, o que serão cerca de 1000 dispositivos por pessoa, colocando a sociedade completamente imersa no mundo cibernético. Nesse âmbito, a Internet das Coisas, do inglês *Internet of Things (IoT)*, é a pilha de tecnologias que irá comportar esse avanço tecnológico, alcançando então entre as décadas de 2025 e 2035 o conceito conhecido como Internet de Tudo, do inglês *Internet of Everything (IoE)*.

Os principais dispositivos dentro da *IoT* e *IoE* serão os sensores sem fio, devido suas fortes características de flexibilidade e interoperabilidade. Nesse sentido, para tornar esses ambientes reais, será necessário software de excelente qualidade, para fornecer serviços de formar confiável e segura aos usuários. Segundo [Socha et al. \(2016\)](#) e [Morin, Harrand e Fleurey \(2017\)](#), a *IoT* está diretamente ligada ao futuro da Engenharia de Software, pois apenas ela poderá conduzir os processos necessários até que a sociedade esteja imersa. Já segundo [Larrucea et al. \(2017\)](#), [Spinellis \(2017\)](#) e [Zambonelli \(2017\)](#), as empresas que criarem serviços e produtos para a *IoT* e *IoE* devem utilizar a experiência com os modelos ágeis de desenvolvimento provenientes da Engenharia de Software, para construírem soluções eficientes e seguras.

Como relatado anteriormente, as *RSSFs* são suscetíveis a ataques, sejam eles físicos ou virtuais, devido ao ambiente hostil no qual elas são comumente implantadas. Dentre esses ataques, destaca-se na literatura o ataque da Injeção de Falsa Informação por sua complexidade na detecção e mitigação. Nesse sentido, o objetivo geral do presente trabalho era propor e implementar um mecanismo capaz de detectar nós que praticam a Injeção de Falsa Informação, utilizando a comparação entre agrupamentos em *RSSFs*. Para atingi-lo foi projetado e implementado o mecanismo *Cluster-based False Data Detection (CFDD)*, e sua eficiência atestada através da análise dos dados provenientes das simulações.

Dentre os objetivos específicos, o **Objetivo I** era levantar o estado da arte do ataque da Injeção de Falsa Informação. Esse objetivo foi atingido e permitiu encontrar uma lacuna na literatura para ser explorada, ou seja, utilizar a comparação entre agrupamentos para mitigar o ataque abordado.

O **Objetivo II** era propor uma mecanismo capaz de detectar a Injeção de Falsa Informação, explorando a lacuna encontrada, e o **Objetivo III** consistia em implementar tal mecanismo. Esses objetivos específicos foram atingidos através do projeto e implementação do mecanismo *CFDD*, o qual foi testado e validado através de simulações, utilizando ferramentas recomendadas pela literatura.

Finalmente, o último objetivo específico (**Objetivo IV**) era avaliar o mecanismo

proposto através da comparação com mecanismos da literatura. Na comparação realizada, o mecanismo proposto (CFDD) apresentou ser superior aos mecanismos da literatura que possuíam dados suficientes para tal comparação.

7.1 Trabalhos Futuros

Como possibilidades de trabalho futuro, o mecanismo desenvolvido pode ser evoluído das seguintes formas:

- Utilizar a lista negra gerada pelo CFDD como base para a exclusão dos nós Injetores de Falsa Informação na rede, criando então uma solução global contra esse tipo de ataque.
- Utilizar os dados estatísticos provenientes do mecanismo para distribuir informações importantes para a rede, partindo da central aos nós. Isso permitiria, por exemplo, colocar a rede em modo ativo contra o ataque da Injeção de Falsa Informação.
- Implantar o mecanismo desenvolvido em ambientes reais e avaliar seu desempenho perante situações críticas, como por exemplo, nós legítimos sendo infectados e tornando-se maliciosos.

REFERÊNCIAS

- BALLOR, C.; WILONSKY, R.; STEELE, T. **Hacking blamed for emergency sirens blaring across Dallas**. 2017. Dallasnews.com. Acessado em Abril–2017. Disponível em: <<https://www.dallasnews.com/news/dallas/2017/04/08/emergency-sirens-blare-across-dallas-county-despite-clear-weather>>. Citado na página 23.
- BORGIA, E. The internet of things vision: Key features, applications and open issues. **Computer Communications**, v. 54, p. 1–31, 2014. ISSN 0140-3664. Acesado em Março–2015. Citado 2 vezes nas páginas 23 e 65.
- BYSANI, L.; TURUK, A. A survey on selective forwarding attack in wireless sensor networks. In: IEEE. **International Conference on Devices and Communications (ICDeCom)**. [S.l.]: IEEE, 2011. p. 1–5. Citado na página 30.
- FERRAZ, L. H. G.; VELLOSO, P. B.; DUARTE, O. C. M. An accurate and precise malicious node exclusion mechanism for ad hoc networks. **Ad Hoc Networks**, v. 19, p. 142–155, 2014. ISSN 1570-8705. Citado 2 vezes nas páginas 23 e 24.
- GRANJAL, J.; MONTEIRO, E.; SILVA, J. S. Security in the integration of low-power wireless sensor networks with the internet: A survey. **Ad Hoc Networks**, v. 24, Part A, p. 264–287, 2015. ISSN 1570-8705. Citado 2 vezes nas páginas 27 e 34.
- JEBA, S. A.; PARAMASIVAN, B. Energy efficient multipath data transfer scheme to mitigate false data injection attack in wireless sensor networks. **Computers & Electrical Engineering**, v. 39, n. 6, p. 1867–1879, April 2013. ISSN 0045-7906. Special Issue on Wireless Systems: Modeling, Monitoring, Transmission, Performance Evaluation and Optimization. Citado 4 vezes nas páginas 29, 36, 37 e 38.
- JUNIOR, S. F. d. A. C. **Sistema de Gerenciamento da Amazônia Azul (SisGAAz): o passo inicial para o efetivo controle da área marítima brasileira**. [S.l.], 2013. Acesado em Novembro–2015. Disponível em: <<http://www.esg.br/images/Monografias/2013/CHAVESJUNIOR.pdf>>. Citado na página 28.
- KAZIENKO, J. F. **Armazenamento Seguro de Chaves Criptográficas em Redes de Sensores Sem Fio**. Tese (Doutorado) — Universidade Federal Fluminense, Maio 2013. Acessado em Maio–2015. Disponível em: <<http://intranet.ctism.ufsm.br/~kazienko/>>. Citado 4 vezes nas páginas 23, 27, 34 e 41.
- KAZIENKO, J. F. et al. On the performance of a secure storage mechanism for key distribution architectures in wireless sensor networks. **International Journal of Distributed Sensor Networks**, v. 11, n. 5, p. 1–14, March 2015. Citado 4 vezes nas páginas 24, 34, 55 e 59.
- KUMAR, M. **Here’s How Hacker Activated All Dallas Emergency Sirens On Friday Night**. 2017. Thehackernews.com. Acessado em Abril–2017. Disponível em: <<http://thehackernews.com/2017/04/emergency-tornado-siren-hack.html>>. Citado na página 23.
- LARRUCEA, X. et al. Software engineering for the internet of things. **IEEE Software**, v. 34, n. 1, p. 24–28, Jan 2017. ISSN 0740-7459. Citado na página 65.

LIN, J. et al. Towards effective en-route filtering against injected false data in wireless sensor networks. In: IEEE. **Global Telecommunications Conference (GLOBECOM)**. [S.l.], 2011. p. 1–5. ISSN 1930-529X. Citado 2 vezes nas páginas 34 e 35.

LOUREIRO, A. A. et al. Redes de sensores sem fio. In: XXI SBRC. **Minicursos: Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)**. [S.l.]: Sociedade Brasileira de Computação (SBC), 2003. p. 179–226. Citado 3 vezes nas páginas 27, 30 e 34.

MARGI, C. et al. Segurança em redes de sensores sem fio. In: IX SBSEG. **Minicursos: Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSEG)**. [S.l.]: Sociedade Brasileira de Computação (SBC), 2009. p. 149–194. Citado 5 vezes nas páginas 23, 24, 30, 34 e 45.

MEMSIC. **MicaZ Datasheet**. San Jose, California, 2010. Online. Acessado em Novembro–2015. Disponível em: <http://www.memsic.com/userfiles/files/Datasheets/WSN/micaz_datasheet-t.pdf>. Citado na página 28.

MEMSIC. **WSN Nodes–MPR2400CB**. San Jose, California, 2015. Online. Acessado em Novembro–2015. Disponível em: <<http://www.memsic.com/wireless-sensor-networks/MPR2400CB>>. Citado na página 28.

MEYER, P. **Probabilidade: aplicações à estatística**. 2. ed. [S.l.]: Livros Técnicos e Científicos (LTC), 1985. ISBN 9788521602941. Citado na página 46.

MORIN, B.; HARRAND, N.; FLEUREY, F. Model-based software engineering to tame the iot jungle. **IEEE Software**, v. 34, n. 1, p. 30–36, Jan 2017. ISSN 0740-7459. Citado na página 65.

MOSCHITTA, A.; NERI, I. Power consumption assessment in wireless sensor networks. In: FAGAS LUCA GAMMAITONI, D. P. G.; BERINI, G. A. (Ed.). **ICT - Energy - Concepts Towards Zero - Power Information and Communication Technology**. 1. ed. [S.l.]: InTech, 2014. online 9, p. 203–224. ISBN 978-953-51-1218-1. Citado 2 vezes nas páginas 28 e 29.

RIECKER, M. et al. On data-centric intrusion detection in wireless sensor networks. In: IEEE. **IEEE International Conference on Green Computing and Communications (GreenCom)**. [S.l.], 2012. p. 325–334. Citado 2 vezes nas páginas 23 e 30.

ROSENBERG, E.; SALAM, M. **Hacking Attack Woke Up Dallas With Emergency Sirens**. 2017. Nytimes.com. Acessado em Abril–2017. Disponível em: <<https://www.nytimes.com/2017/04/08/us/dallas-emergency-sirens-hacking.html>>. Citado na página 23.

RUIZ, L. B. et al. Arquiteturas para redes de sensores sem fio. In: XXII SBRC. **Minicursos: Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)**. [S.l.]: Sociedade Brasileira de Computação (SBC), 2004. p. 1–52. Citado 2 vezes nas páginas 28 e 29.

SOCHA, D. et al. Wide-field ethnography: Studying software engineering in 2025 and beyond. In: **2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C)**. [S.l.: s.n.], 2016. p. 797–800. Citado na página 65.

SPINELLIS, D. Software-engineering the internet of things. **IEEE Software**, v. 34, n. 1, p. 4–6, Jan 2017. ISSN 0740-7459. Citado na página 65.

VIEIRA, L. et al. Redes de sensores aquáticas. In: XXVIII SBRC. **Minicursos: Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)**. [S.l.]: Sociedade Brasileira de Computação (SBC), 2010. p. 199–240. Citado na página 28.

VINCENNELLI, A. S. **Let's Get Physical: Adding Physical Dimensions to Cyber Systems**. 2014. Online. International Conference on Cyber-Physical Systems–Internet of Everything Summit. Acessado em Junho–2015. Disponível em: <http://iccps.acm.org/drupal2/sites/default/files/ICCPS2014red.pdf>. Citado na página 23.

WANG, J. et al. Defending collaborative false data injection attacks in wireless sensor networks. **Information Sciences**, Elsevier, v. 254, p. 39–53, January 2014. Citado 7 vezes nas páginas 24, 30, 34, 36, 38, 61 e 62.

WANG, Y.; ATTEBURY, G.; RAMAMURTHY, B. A survey of security issues in wireless sensor networks. **IEEE Communications Surveys Tutorials**, v. 8, n. 2, p. 2–23, Feb 2006. ISSN 1553-877X. Citado 3 vezes nas páginas 30, 34 e 45.

WANGHAM, M.; DOMENECH, M.; MELLO, E. de. Infraestruturas de autenticação e de autorização para internet das coisas. In: XIII SBSEG. **Minicursos: Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg)**. [S.l.]: Sociedade Brasileira de Computação (SBC), 2013. p. 156–205. Citado 2 vezes nas páginas 23 e 31.

WAZLAWICK, R. **Metodologia de pesquisa para ciência da computação, 2a edição**. [S.l.]: Elsevier Brasil, 2014. v. 2. Citado 3 vezes nas páginas 41, 59 e 62.

YE, F. et al. Statistical en-route filtering of injected false data in sensor networks. In: XXIII INFOCOM. **IEEE International Conference on Computer Communications (INFOCOM)**. [S.l.]: IEEE, 2004. v. 4, p. 2446–2457 vol.4. ISSN 0743-166X. Citado 3 vezes nas páginas 35, 38 e 61.

ZAMBONELLI, F. Key abstractions for iot-oriented software engineering. **IEEE Software**, v. 34, n. 1, p. 38–45, Jan 2017. ISSN 0740-7459. Citado na página 65.

ZHANG, Q.; YU, T.; NING, P. A framework for identifying compromised nodes in sensor networks. In: IEEE. **2th International Conference on Security and Privacy in Communication Networks (Securecomm)**. [S.l.], 2006. p. 1–35. Citado 3 vezes nas páginas 34, 35 e 38.

ZHANG, Y. et al. The design and evaluation of interleaved authentication for filtering false reports in multipath routing wsns. **Wireless Networks**, Springer US, v. 16, n. 1, p. 125–140, May 2010. ISSN 1022-0038. Citado 3 vezes nas páginas 30, 36 e 38.

ZHANG, Y.; YANG, L. T.; CHEN, J. **RFID and sensor networks: architectures, protocols, security, and integrations**. 1. ed. [S.l.]: CRC Press Inc., 2009. Citado na página [27](#).

ZHU, D.; YANG, X.; YU, W. A novel self-checking pollution attackers identification scheme in wireless network coding. In: IEEE. **IEEE 11th Consumer Communications and Networking Conference (CCNC)**. [S.l.], 2014. p. 345–350. Citado na página [37](#).

ZHU, D.; YANG, X.; YU, W. Spais: A novel self-checking pollution attackers identification scheme in network coding-based wireless mesh networks. **Computer Networks**, v. 91, p. 376–389, Nov 2015. ISSN 1389-1286. Citado 2 vezes nas páginas [37](#) e [38](#).

ZIN, S. M. et al. Survey of secure multipath routing protocols for wsns. **Journal of Network and Computer Applications**, v. 55, p. 123–153, May 2015. ISSN 1084-8045. Citado na página [35](#).

Apêndices

APÊNDICE A – ARQUIVO *SIMULA.PY*, COM O *SCRIPT* QUE GUIA AS SIMULAÇÕES.

```

1 #!/usr/bin/python
2
3 import sys
4 import time
5 import subprocess
6 from TOSSIM import *
7 from os import popen
8
9 t = Tossim([])
10 r = t.radio()
11
12 # Filter and catch the 'no parameter' error
13 if (len(sys.argv) < 2):
14     print ">>> ERROR:\tMissing the number of nodes! \t<<<<"
15     sys.exit()
16 else:
17     nnodes = int(sys.argv[1])
18
19 f = open("linkgain.out", "r")
20 lines = f.readlines()
21 for line in lines:
22     s = line.split()
23     if (len(s) > 0):
24         if (s[0] == "gain"):
25             r.add(int(s[1]), int(s[2]), float(s[3]))
26
27
28 file1 = open("msg_enviadas.txt", "w")
29 t.addChannel("msg_enviadas", file1)
30
31 file2 = open("msg_recebidas.txt", "w")
32 t.addChannel("msg_recebidas", file2)
33 t.addChannel("tempo", sys.stdout)
34 t.addChannel("DBG_CRYPT0", sys.stdout)
35
36 noise = open("meyer-heavy.txt", "r")
37 lines = noise.readlines()
38 limiar = 0
39 for line in lines:
40     if (limiar >= 150):
41         break
42     else:
43         limiar = limiar + 1
44         str = line.strip()

```

```
45     if (str != ""):
46         val = int(str)
47         for i in range(1, nnodes+1):
48             t.getNode(i).addNoiseTraceReading(val)
49
50 for i in range(1, nnodes+1):
51     t.getNode(i).createNoiseModel()
52
53 t.getNode(1).bootAtTime(0)
54
55 for i in range(2, nnodes+1):
56     t.getNode(i).bootAtTime(5+i)
57
58 print " Simulating a network with ",nnodes," nodes."
59 print ' Simulation started: ', time.strftime("%H:%M:%S - %d-%m-%Y")
60 print ' Waiting simulation... '
61
62 for i in range(0, 100000000):
63     t.runNextEvent()
64
65 file1.close()
66 file2.close()
67
68 print ' Simulation ended: ', time.strftime("%H:%M:%S - %d-%m-%Y")
69 print ' Running scripts... waiting..... '
70
71 popen('python ordena_recebidas.py')
72 popen('python elimina_repetidas.py')
73 popen('python organiza_coletas.py')
74 popen('python calcula_medias_individuais.py')
75 popen('cat msg_trail.txt | uniq > msg_trail_sem_repetidas.txt')
76 popen('python identifica_agrupamentos.py')
77 popen('python calcula_media_agrupamentos.py')
78 popen('python detecta_nodos_maliciosos.py')
79 popen('sort nodes_maliciosos.txt -u -o nodes_maliciosos.txt')
80 popen('python detecta_agrupamentos_maliciosos.py')
81 popen('python gera_estatisticas.py')
82 print ' Done!'
```

**APÊNDICE B – ARQUIVO *MONTA-RASTRO.PY*, COM O *SCRIPT*
QUE DETECTA O RASTRO DOS PACOTES.**

```

1 import sys
2 from operator import itemgetter, attrgetter
3
4 arquivo_entrada = open("msg_recebidas.txt", "r")
5 arquivo_destino = open("rastros_organizados.txt", "w")
6 linhas = arquivo_entrada.readlines()
7 linha_atual = linhas[0]
8 lista_geral = []
9 lista_individual = []
10 max_msg = 0
11
12 for i in range(1, len(linhas)):
13     string_atual = linhas[i - 1].split(';')
14     if int(string_atual[1]) > max_msg:
15         max_msg = int(string_atual[1])
16
17 for cont in range(0, max_msg):
18     for i in range(1, (len(linhas))):
19         string_atual = linhas[i - 1].split(';')
20         if int(string_atual[1]) == (cont + 1):
21             lista_individual.append(string_atual[0])
22             string_prox = linhas[i].split(';')
23             if ((string_prox[0] != string_atual[0]) or (i == (len(linhas) - 1))):
24                 lista_geral.append(lista_individual)
25             lista_individual = []
26
27 for cont in range(0, max_msg):
28     arquivo_destino.write('ID_MSG: %hu; Caminho: ' % (cont+1))
29     arquivo_destino.write("%s" % ", ".join(lista_geral[cont]))
30     arquivo_destino.write("\n")
31
32 arquivo_entrada.close()
33 arquivo_destino.close()

```


**APÊNDICE C – ARQUIVO *IDENTIFICA-AGRUPAMENTOS.PY*,
COM O *SCRIPT* QUE IDENTIFICA OS AGRUPAMENTOS
PERTENCENTES A REDE.**

```

1 import sys
2
3 ##### ETAPA 1 #####
4 # Identifica a relacao entres os nodes: filhos ,pais e bisavos. Armazena as
   informacoes em um arquivo de texto.
5 arquivo_rastros = open("msg_trail_sem_repetidas.txt", "r")
6 arquivo_destino = open("parentesco_nodes.txt", "w")
7 rastros = arquivo_rastros.readlines() # rastros sera uma lista com todas
   as linhas do arquivo_rastros
8
9 lista_parentesco = [] # Cria uma lista vazia
10 break_loop = 0
11 for node in range(2, 256): # Para cada
   no na simulacao(2-255)(exceto 1)
12     for num_linha in range(0, len(rastros)): # Percorre todas as
   linhas do arquivo de rastros
13         if break_loop == 1:
14             break_loop = 0
15             break
16         linha_atual = rastros[num_linha].split(';') # Divide cada linha
   onde ha ';'
17         for item in linha_atual[1:]: # Percorre os elementos da
   linha a partir do 1 (Elem 0 = [10-2-12]);)
18             if item == str(node): # Se o elemento da linha e igual
   ao no que estamos procurando
19                 ind = linha_atual[1:].index(str(node)) # IMPORTANTE : ind =
   indice do node na lista
20                 if linha_atual[ind + 2] == '\n':
21                     # Se o elemento que procuramos e o ultimo no caminho, ele esta
   enviando para o no 1, o destino final.
22                     lista_parentesco.append(linha_atual[ind+ 1] + ';' + str(1))
23                     break_loop = 1
24                     break
25                 elif linha_atual[ind + 2] != '\n' and linha_atual[ind + 3] == '\n':
26                     # Coloca o primeiro elemento (filho) e o posterior (pai) na lista
   para ser armazenada depois
27                     lista_parentesco.append(linha_atual[ind + 1] + ';' + linha_atual[
   ind + 2] + ';' + str(1))
28                     break_loop = 1
29                     break
30                 elif linha_atual[ind + 2] != '\n' and linha_atual[ind + 3] != '\n'
   and linha_atual[ind + 4] == '\n':
31                     # Coloca filho pai e avo na lista para ser armazenada depois

```

```
32     lista_parentesco.append(linha_atual[ind+1]+';' + linha_atual[ind
+2]+';' + linha_atual[ind+3]+';' + str(1))
33     break_loop = 1
34     break
35     else:
36         # Coloca filho ,pai ,avo e bisavo na lista para ser armazenada
depois
37         lista_parentesco.append(linha_atual[ind+1]+';' + linha_atual[ind
+2]+';' + linha_atual[ind+3]+';' + linha_atual[ind+4])
38         break_loop = 1
39         break
40
41
42 for item in range(len(lista_parentesco)):           #Para cada item na lista
dos filhos e pais
43     arquivo_destino.write("%s;" % lista_parentesco[item]) #Escreve o item no
arquivo de saida
44     arquivo_destino.write("\n")
45
46 arquivo_rastros.close() # Fecha os arquivos para salvar as alteracoes
47 arquivo_destino.close()
48
49 ##### ETAPA 2 #####
50 # Identifica os agrupamentos e gera um arquivo de texto contendo as
informacoes.
51
52 arquivo_entrada = open("parentesco_nodes.txt", "r") # Abre o arquivo com
os nos filhos e pais
53 arquivo_destino = open("agrupamentos.txt", "w") # Abre o arquivo
agrupamento para receber os resultados
54
55
56 agrupamento_atual = [] # Lista vazia para armazenar agrupamentos
durante as iteracoes
57 lista_geral_agrupamentos = [] # Lista vazia para armazenar os
agrupamentos ja formados
58 agrupamentos_prontos = []
59 lista_avos = []
60 parentesco = arquivo_entrada.readlines()
61
62 for line in range(len(parentesco)):
63     linha_atual = parentesco[line].split(';')
64     node_pai = linha_atual[1]
65     if node_pai in agrupamentos_prontos:
66         continue
67     else:
68         agrupamentos_prontos.append(node_pai)
```



```

69     agrupamento_atual = []
70     numero_agrupamento = (len(agrupamentos_prontos))
71     agrupamento_atual.append(str(numero_agrupamento))
72     agrupamento_atual.append(linha_atual[1]) # Coloca o pai na lista do
agrupamento atual
73     agrupamento_atual.append(linha_atual[0])
74     for node in parentesco:
75         linha_node = node.split(';')
76         if linha_node[1] == node_pai and linha_node[0] not in
agrupamento_atual:
77             agrupamento_atual.append(linha_node[0])
78             lista_geral_agrupamentos.append(agrupamento_atual)
79
80 for item in range(len(lista_geral_agrupamentos)): #Para cada item na
lista dos filhos e pais
81     for sub_item in range(len(lista_geral_agrupamentos[item])):
82         arquivo_destino.write("%s;" % lista_geral_agrupamentos[item][sub_item])
#Escreve o item no arquivo de saida
83     arquivo_destino.write("\n")
84
85 arquivo_entrada.close()
86 arquivo_destino.close()
87 ##### ETAPA 3 #####
88 # Identifica quais agrupamentos estao proximos na rede. Armazena as
informacoes em um arquivo de texto.
89 # Agrupamentos sao proximos quando possuem o mesmo bisavo.
90 arquivo_agrupamentos = open("agrupamentos.txt", "r")
91 arquivo_parentesco = open("parentesco_nodes.txt", "r")
92 arquivo_destino = open("agrupamentos_e_bisavos.txt", "w")
93
94 lista_agrp_prox = [] # Lista para armazenar os agrupamentos que estao
proximos
95 agrupamentos = arquivo_agrupamentos.readlines()
96 parentesco = arquivo_parentesco.readlines()
97 lista_bisavos = []
98 for line in agrupamentos:
99     line_div = line.split(";")
100     node_pai = line_div[1]
101     agrupamento_atual = line_div[0]
102     for linha_parentesco in parentesco:
103         linha_p_div = linha_parentesco.split(';')
104         if linha_p_div[0] == node_pai:
105             if linha_p_div[2] == '\n':
106                 lista_bisavos.append(agrupamento_atual + ';' + str(1) + ';')
107             else:
108                 lista_bisavos.append(agrupamento_atual + ';' + linha_p_div[2] + ';'
)

```

```
109
110 for item in range(len(lista_bisavos)):
111     arquivo_destino.write("%s" % lista_bisavos[item]) #Escreve o item no
        arquivo de saida
112     arquivo_destino.write("\n")
113
114 arquivo_agrupamentos.close()
115 arquivo_parentesco.close()
116 arquivo_destino.close()
```

**APÊNDICE D – ARQUIVO *DETECTA-NODES-MALICIOSOS.PY*,
COM O *SCRIPT* QUE DETECTA OS NÓS MALICIOSOS.**

```

1 import sys
2
3 arq_nodes_maliciosos = open("nodes_maliciosos.txt", "w")
4 arq_agrp_maliciosos = open("agrupamentos_maliciosos", "w")
5 debug = open("debug.txt", "w")
6
7 tolerancia = 25 # Representa a porcentagem de tolerancia em relacao a
8                 media do agrupamento para considerar um node malicioso
9
10 tolerancia_agrp = 20 # Representa a porcentagem de tolerancia em relacao
11                      aos agrp proximos para considerar um agrupamento malicioso
12
13 lista_nodes_maliciosos = []
14 lista_agrp_maliciosos = []
15
16 #Identificando nos maliciosos#
17 for line in open('medias_individuais.txt', 'r'):
18     line = line.split(';')
19     node = line[0]
20     med_node = float(line[2])
21     for line_agrp in open('agrupamentos.txt', 'r'):
22         if node in line_agrp and node != line_agrp[0]: # Se o numero do node
23             esta presente na linha, e nao e o numero do agrupamento.
24             agrupamento = line_agrp[0]
25             for line_med_agrp in open('medias_agrupamentos.txt', 'r'):
26                 line_med_agrp = line_med_agrp.split(';')
27                 if line_med_agrp[0] == agrupamento:
28                     agrupamentos_proximos = []
29                     medias_agrp_proximos = []
30                     # Encontramos o agrupamento no arquivo das medias, precisamos
31                     agora saber quem sao os agrupamentos proximos:
32                     # Encontrando o bisavo do agrupamento atual:
33                     for line_prox in open('agrupamentos_e_bisavos.txt', 'r'):
34                         line_prox = line_prox.split(';')
35                         if agrupamento in line_prox and agrupamento != line_prox[1]:
36                             bisavo = line_prox[1]
37                             break
38                     bisavo = -1
39
40             if bisavo != -1:
41                 # Montando a lista de agrupamentos proximos:
42                 for line_prox in open('agrupamentos_e_bisavos.txt', 'r'):
43                     line_prox = line_prox.split(';')
44                     if line_prox[1] == bisavo:
45                         agrupamentos_proximos.append(line_prox[0])

```

```

41
42     if len(agrupamentos_proximos) > 0:
43         # Calculando a media dos agrupamentos proximos:
44         # Primeiro precisamos das medias dos agrupamentos proximos:
45         for line in open('medias_agrupamentos.txt', 'r'):
46             line = line.split(';')
47             if line[0] in agrupamentos_proximos:
48                 medias_agrp_proximos.append(float(line[1]))
49
50         # Com a lista das medias de cada agrupamento, podemos calcular
51 a media geral:
52         media_geral_proximos = sum(medias_agrp_proximos) / len(
53 medias_agrp_proximos)
54
55         med_agrupamento = float(line_med_agrp[1])
56         tolerancia_sup = med_agrupamento + ((med_agrupamento * tolerancia
57 ) / 100)
58         tolerancia_inf = med_agrupamento - ((med_agrupamento * tolerancia
59 ) / 100)
60         if len(agrupamentos_proximos) > 0:
61             tolerancia_sup_agrp = media_geral_proximos + ((
62 media_geral_proximos * tolerancia_agrp) / 100)
63             tolerancia_inf_agrp = media_geral_proximos - ((
64 media_geral_proximos * tolerancia_agrp) / 100)
65         if (med_node > tolerancia_sup):
66             # A media do node diverge do agrupamento, devemos checar se o
67 agrupamento e malicioso primeiro:
68             if len(agrupamentos_proximos) > 0:
69                 if (med_agrupamento > tolerancia_sup_agrp) or (
70 med_agrupamento < tolerancia_inf_agrp):
71                     if agrupamento not in lista_agrp_maliciosos:
72                         lista_agrp_maliciosos.append(agrupamento)
73                     else:
74                         lista_nodes_maliciosos.append(node)
75                 else:
76                     lista_nodes_maliciosos.append(node)
77
78
79 for item in range(0, len(lista_nodes_maliciosos)):
80     arq_nodes_maliciosos.write("%s" % lista_nodes_maliciosos[item])
81     arq_nodes_maliciosos.write("\n")
82
83 for item in range(0, len(lista_agrp_maliciosos)):
84     arq_agrp_maliciosos.write("%s" % lista_agrp_maliciosos[item])
85     arq_agrp_maliciosos.write("\n")
86
87 arq_nodes_maliciosos.close()

```

```
80 arq_agrp_maliciosos.close()  
81 debug.close()
```