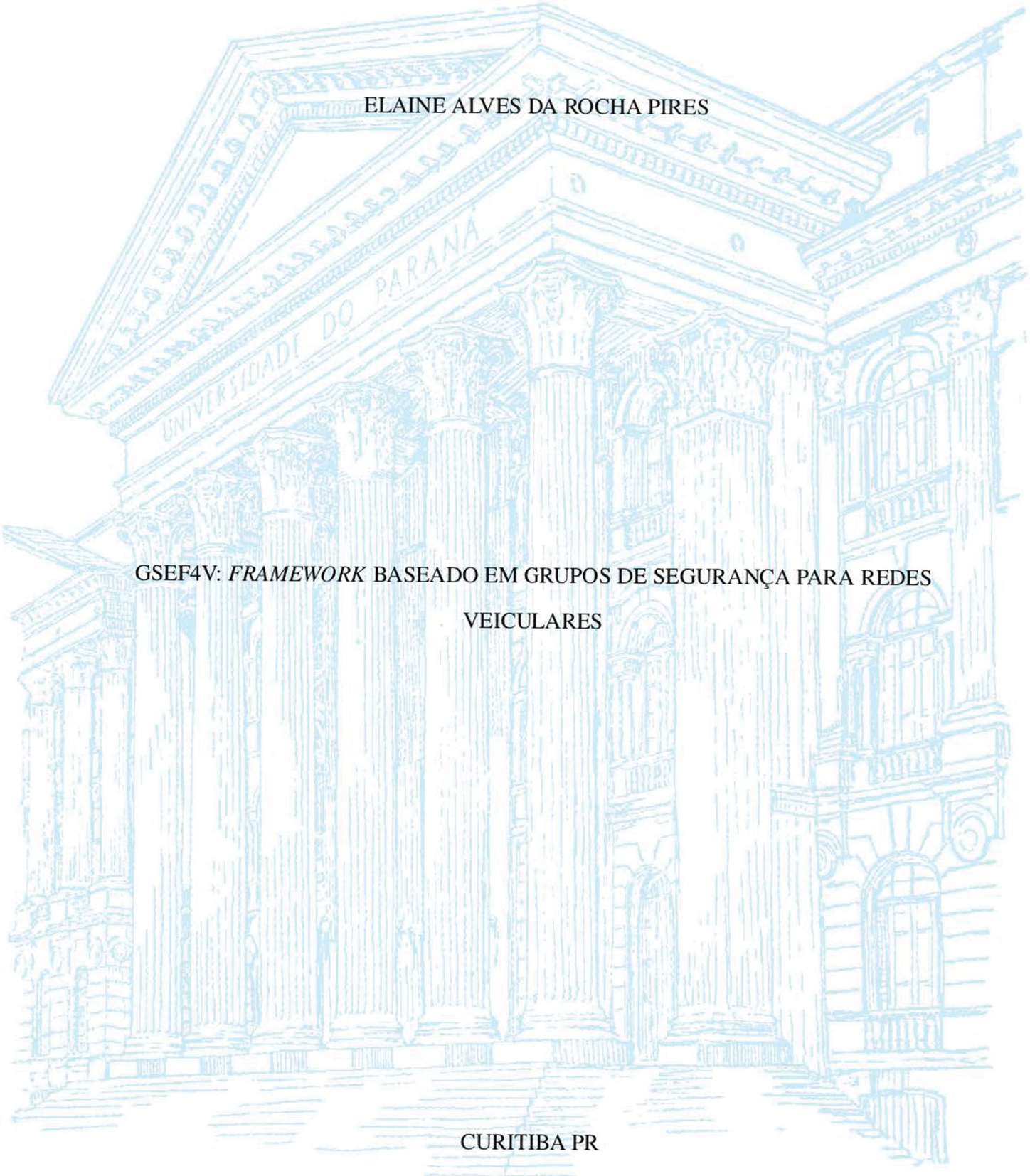
UNIVERSIDADE FEDERAL DO PARANÁ

ELAINE ALVES DA ROCHA PIRES



GSEF4V: *FRAMEWORK* BASEADO EM GRUPOS DE SEGURANÇA PARA REDES VEICULARES

CURITIBA PR

2022

ELAINE ALVES DA ROCHA PIRES

GSEF4V: *FRAMEWORK* BASEADO EM GRUPOS DE SEGURANÇA PARA REDES VEICULARES

Tese apresentada como requisito parcial à obtenção do grau de Doutor em Ciência da Computação no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Luiz Carlos Pessoa Albini.

CURITIBA PR

2022

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)
UNIVERSIDADE FEDERAL DO PARANÁ
SISTEMA DE BIBLIOTECAS – BIBLIOTECA CIÊNCIA E TECNOLOGIA

Pires, Elaine Alves da Rocha.
GSEF4V : *framework* baseado em grupos de segurança para redes veiculares. / Elaine Alves da Rocha Pires. – Curitiba, 2022.
1 recurso on-line : PDF.

Tese (Doutorado) – Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática.
Orientador: Prof. Dr. Luiz Carlos Pessoa Albini.

1. Informática. 2. *Framework* (Programa de computador). 3. Redes de computação – medidas de segurança. 4. Segurança, Sistemas de. I. Albini, Luiz Carlos. II. Universidade Federal do Paraná. Programa de Pós-Graduação em Informática. III. Título.

Bibliotecário: Nilson Carlos Vieira Júnior CRB-9/1797

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação INFORMÁTICA da Universidade Federal do Paraná foram convocados para realizar a arguição da tese de Doutorado de **ELAINE ALVES DA ROCHA PIRES** intitulada: **GSEF4V: FRAMEWORK BASEADO EM GRUPOS DE SEGURANÇA PARA REDES VEICULARES**, sob orientação do Prof. Dr. LUIZ CARLOS PESSOA ALBINI, que após terem inquirido a aluna e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de doutora está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 21 de Junho de 2022.

Assinatura Eletrônica
22/06/2022 21:43:59.0
LUIZ CARLOS PESSOA ALBINI
Presidente da Banca Examinadora

Assinatura Eletrônica
22/06/2022 13:12:00.0
CARLOS MARCELO PEDROSO
Avaliador Externo (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
22/06/2022 09:22:11.0
CARLOS ALBERTO MAZIERO
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
19/07/2022 09:32:29.0
EDUARDO DA SILVA
Avaliador Externo (INSTITUTO FEDERAL CATARINENSE)

"E disso os loucos sabem
Só os loucos sabem
Disso os loucos sabem
Só os loucos sabem"
Charlie Brown Jr.

AGRADECIMENTOS

A minha família por sempre me apoiar nas minhas escolhas, se interessar e principalmente me motivar sem se importar com a distância, amo muito vocês.

Aos meus filhos, Gabriel e Gael, todo o esforço e dedicação da mamãe é por vocês e para vocês. Mamãe ama muito vocês!

Ao meu esposo Ivan Luiz Pedroso Pires, por sempre acreditar e me incentivar nos estudos. Sem você eu não teria chegado aqui!

Aos amigos e colegas que estiveram junto comigo nesta caminhada, me dando todo o apoio e força para, principalmente, concluir as disciplinas, além de deixar os dias mais alegres, proporcionando momentos divertidos mesmo com as dificuldades normais proporcionadas pelos estudos.

Ao pessoal do IFMT - Campus Juína pela compreensão e motivação, vocês sempre me ajudaram para que eu pudesse estudar e aprimorar o meu conhecimento, mesmo sem conseguir o afastamento.

A todas as pessoas da UFPR - professores, técnicos administrativos da pós-graduação e funcionários em geral. Aos professores por sempre estarem disponíveis para sanar as dúvidas. Aos técnicos pela agilidade nos documentos e acesso as informações. E aos funcionários por proporcionarem aos alunos um ambiente agradável para a realização de pesquisas. Sem vocês nós não estaríamos aqui.

Ao professor Albini, que me orientou na realização dessa pesquisa e mesmo com tantos imprevistos, nunca me abandonou. Professor, o sucesso desse trabalho é resultado da sua paciência, crença, compreensão e sabedoria que sempre acompanharam os meus passos e tropeços na pesquisa científica - desde o Mestrado.

Muito obrigada!

RESUMO

As redes veiculares vêm crescendo como forma de comunicação *ad-hoc* entre veículos, em diferentes cenários, como autoestradas, locais remotos ou urbanos. Essa comunicação pode ser realizada por meio de estações fixas, com infraestruturas que demandam custos e viabilidade física para sua implantação. A alternativa é a comunicação descentralizada sem infraesturutura, através da criação de uma rede móvel interligando veículo a veículo, tornando-os capazes de oferecer diferentes serviços por meio de trocas de mensagens. No entanto, essas mensagens podem ser corrompidas ou acessadas indevidamente, comprometendo a confidencialidade e a integridade da comunicação. Embora existam trabalhos relacionados a essa temática publicados na literatura, eles sugerem a necessidade de uma infraestrutura direta ou indiretamente, ou, ainda, de determinados protocolos de roteamento. Este trabalho propõe um *Framework* para VANETs com comunicação veículo a veículo, sem a necessidade de infraestrutura e independente de protocolos de rsoteamento: *Group-based Security Framework for V2V-VANETs* (GSeF4V). Este *Framework* provê a confidencialidade e a integridade das informações nas comunicações entre os veículos da *Vehicular Ad-hoc Network* (VANET). Esta solução é baseada em grupos de nós, em cenários urbanos, capazes de comunicar e estabelecer uma relação de confiança entre si. O GSeF4V pontua os contatos e permanência de conectividades para estabelecer a relação de confiança, além de considerar relações sociais previamente estabelecidas. A confiança é organizada em diferentes níveis dentro de um mesmo grupo. A cada nível é atribuido um rol de serviços, como gerenciamento de nós, chaves, grupo, entre outros. Com o estabelecimento do grupo, este *Framework* faz uso de criptografia híbrida em grupo e de código de autenticação de mensagens, para garantir a confidencialidade e integridade da comunicação. O GSeF4V foi implementado para simulação e avaliação. A carga de trabalho foi gerada a partir do simulador *The Opportunistic Network Environment* (ONE). Os parâmetros da avaliação foram o número de nós, a taxa de nós maliciosos, a taxa de nós conhecidos por rede social e o intervalo de tempo para pontuar contatos e permanência. As métricas usadas foram a quantidade de nós pertencentes aos grupos, o tempo de permanência dos nós no mesmo grupo e a quantidade de grupos formados com as variações dos parâmetros. Os resultados mostraram que os pontos de confiança e permanência impactam diretamente na formação de grupo e na quantidade dos nós nos grupos. Com essa variação, foram apresentados diferentes tipos de comportamento do *Framework*, que podem atender a diferentes objetivos. A criptografia e a autenticação da mensagem foram implementadas com o criptossistema ECIES, adaptado da biblioteca MIRACL. O tempo de processamento e a quantidade de mensagens necessários para estabelecer a chave compartilhada, cifrar, decifrar e verificar a integridade das mensagens também foram reportados.

Palavras-chave: VANETs. Segurança. Grupos de Nós. Gerenciamento de Chaves e Confiança.

ABSTRACT

Vehicular networks have been growing as a way of *ad-hoc* communication of vehicles in different scenarios, such as highways, rural locations or urban. This communication can be carried out through fixed stations, with infrastructures that demand high financial cost and physical feasibility to its implementation. The alternative is a decentralized communication without infrastructure, through the creation of a mobile network to interconnecting vehicle to vehicle, and making them capable of offering different services through message exchanges. However, these messages can be corrupted or improperly accessed, compromising the confidentiality and integrity of the communication. Although there are related works to this theme published in the literature, they suggest the need for an infrastructure directly or indirectly, or, yet, on certain routing protocols. This work presents a *Framework* for VANETs with vehicle to vehicle, without the need for infrastructure and independent of routing protocols: GSeF4V. This *Framework* provides confidentiality and integrity of informations in communications between VANET vehicles. This solution is based on trust groups of nodes, in urban scenarios, capable of communicating and establishing a relationship of trust each others. The GSeF4V scores the contacts and stays of connectivity to establish a relationship of trust, in addition to considering previously established social relationships. Trust is organized at different levels within the same group. And each level is assigned a list of services, such as group key updates, group management, etc. After establishment of the group, this *Framework* makes use of hybrid group encryption and message authentication code to ensure the confidentiality and integrity of the communication. GSeF4V was implemented to simulation and evaluation. The workload was generated from the *The* ONE simulator. The evaluation parameters were number of nodes, malicious nodes rate, nodes known by social network rate and the time interval to score contacts and stays. The metrics were the number of group nodes, lenght of stays of the nodes in the same group and number of groups formed by the parameters variations. The results showed that trust and stays points directly impact on group formation and number of nodes in a group. Different types of *Framework* behavior was identified by this variation and its are presented to meet the different objectives. The message encryption and authentication was implementeds with a adapted version of ECIES cryptosystem, from the MIRACL library. The processing time and number of messages necessary to establish the shared key, encrypt, decrypt and verify the integrity of the messages were also reporteds.

Keywords: VANETs. Security. Groups of nodes. Key and Trust management.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

ACK	Acknowledgement
AES	Advanced Encryption Standard
AODV	Ad-Hoc On-Demand Distance Vector
ART	Attack-Resistant Trust Management Scheme
BMA	Bad Mouth Attack
CA	Certificate Authority
DAB	Digital Audio Broadcasting
DM	DTN Mode
DoS	Denial of Service
DSA	Digital Signature Algorithm
DSC-KM	Digital Signature Chains Key Management Scheme
DSRC	Dedicated Short Range Communication
DST	Dempster–Shafer theory of evidence
DTN	Delay Tolerant Network
DVB	Digital Video Broadcasting
EAACK	Enhanced Adaptive ACKnowledgment
ECC	Elliptic Curves Cryptography
ECDSA	Elliptic Curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption Scheme
ECMQV	Elliptic Curve Menezes-Qu-Vanstone
FAT	Fragment Authentication Tree
FEC	Forward Error-Correction
FM	Frequency Modulation
GSeF4V	Group-based Security Framework for V2V-VANETs
GPS	Global Position System
GSM	Global System for Mobile
HMAC	Hash-based Message Authentication Code
HTM	Hybrid Trust Model
ID	Identity
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
ITS	Intelligent Transportation System
KDC	Key Distribution Center
KDF	Key Derivation Function
KGC	Key Generator Center

KMS	Key Management Service
LBS	Location-Based Services
LD-AOTDV	Ad hoc On-demand Trusted-path Distance Vector - Distributes Loads
MA	Misbehavior Authority
MAC	Media Access Control
MANET	Mobile Ad-Hoc Network
MD5	Message Digest 5
MDSR	Modified Dynamic Source Routing
MitM	Man-in-the-Middle
MRA	Missbehavior Report Authentication
OBBA	Opportunistic Batch Bundle Authentication Scheme
OBU	On Board Unit
OSM	Offline Security Manager
ONE	Opportunistic Network Environment
PKG	Private Key Generator
PKI	Public Key Infrastructure
RC4	Rivest Cipher 4
RREP	Route Replay
RSA	Rivest Shamir Adleman
RSU	Road-Side Unit
S-ACK	ACK Seguro
SAODV	Secure Ad-Hoc On-Demand Distance Vector
SCAF	Store-Carry-And-Forward
SDRP	Secure and Dynamic Routing Protocol
SHA	Secure Hash Algorithm
SNVC	Social Networks for Vehicular Certification
SP	Service Provider
TA	Trusted Authority
TPD	Tamper Proof Device
UMTS	Universal Mobile Telecommunication System
V2B	Vehicle to Broadband
V2I	Vehicle to Infrastructure
V2V	Vehicle to Vehicle
V2X	Vehicle to Everything
VANET	Vehicular Ad-hoc Network
WAVE	Wireless Access for Vehicular Environment
WDMM	Working Day Movement Model
WiFi	Wireless Fidelity

SUMÁRIO

1 INTRODUÇÃO

As VANETs são redes móveis cujos nós são veículos que se comunicam entre si, sem a necessidade de uma estrutura fixa, embora sejam capazes, também, de comunicação com dispositivos fixos. Nestas redes, é possível considerar a comunicação veicular e oferecer serviços como predição de tráfego, sistema inteligente de navegação, sistema cooperativo para evitar colisões, entre outros [4, 5, 6].

A VANET [4] possui características similares às da *Mobile Ad-Hoc Network* (MANET), como *broadcast* omnidirecional, raio de transmissão curto e baixa largura de banda; além de outras características, a exemplo de topologia altamente dinâmica, desconexões frequentes, sem limite de energia, variação na densidade da rede, variação do modelo de propagação de sinal, comunicação sem fio, quantidade de nós na rede, tamanho da rede e volatilidade [7, 8].

A comunicação entre os nós nas VANETs é possível devido ao fato de os nós atuarem como remetentes, receptores e roteadores de pacotes, e poderem ser equipados com sensores sem fio e *On Board Unit*s (OBUs). OBUs são unidades com poder computacional, incluindo processamento, armazenamento e dispositivos de entrada e saída. A troca de mensagens pode ser classificada da seguinte maneira: *Vehicle to Vehicle* (V2V) - estabelece a comunicação diretamente entre os nós, *Vehicle to Infrastructure* (V2I) - requer infraestrutura para auxiliar os nós na comunicação, e *Vehicle to Everything* (V2X) - caracteriza a comunicação entre o nó e todos os elementos com ele comunicáveis, por exemplo, outro nó, estação base ou dispositivos próximos [6].

A forma de comunicação V2I requer infraestrutura com cobertura tão vasta quanto possível, para manter a conectividade entre os nós. Existem cenários em que a instalação de infraestrutura é inviável, seja pelo elevado custo monetário de implantação ou mesmo por impossibilidades físicas, como áreas rurais, regiões com pouco atrativo comercial, entre outros fatores. Estes cenários devem ser considerados nas soluções para redes veiculares, e para eles as soluções devem ser projetadas com comunicação V2V, ou seja, sem a necessidade de infraestrutura. Os cenários sem infraestrutura são o foco desta pesquisa.

Nas redes veiculares com comunicação V2V, a densidade de elementos é um fator importante e impacta diretamente na conectividade da VANET [9]. Essa densidade é altamente dinâmica devido às alterações na velocidade e na posição dos nós na rede. A variação da densidade está também relacionada com o ambiente da rede, além do horário do tráfego, mesmo considerando diferentes modelos de tráfegos, distâncias de percurso, entre outros.

Em [9], os autores mostram que a comunicação em cenário urbano pode ser estabelecida e mantida com a variação de 2 a 5 nós por quilômetro e por faixa, e a conectividade do *link* de comunicação será garantida se hovuer média e alta densidade de tráfego. A alta densidade resulta em mais obstáculos, causando diminuição da probabilidade de conectividade, devido ao impacto no canal de rádio.

O aumento da densidade resulta em mais nós transmitindo dados e, consequentemente, maior chance de existir um nó malicioso com objetivo de explorar as vulnerabilidades durante as comunicações dos nós, pois, como mostrado em [10, 11, 12, 13], a VANET é uma rede suscetível a diversos tipos de ataques. Embora haja a característica de alta dinamicidade dos nós e isto possa contribuir para que um possível atacante fique pouco tempo na rede, este tempo é suficiente para que ataques sejam realizados com sucesso.

Os ataques podem ser motivados para retirar ou agregar tráfego em uma rota, alterar o caminho de alguns alvos específicos, aumentar ou reduzir a velocidade, criar o caos no tráfego -

adulterando informações, coletar dados sensíveis de um grupo ou de um nó em específico, entre outros objetivos. Estes ataques envolvem direta ou indiretamente a adulteração e a obtenção indevida de informação.

É possível encontrar soluções na literatura para alguns desses ataques (*eavesdropping*, *collusion*, de alteração, *impersonation*, *greedy drivers*, entre outros) [14, 15, 16, 17, 18]. Entretanto, tais soluções são específicas para um determinado ataque ou cenário. Por exemplo, [14] é uma solução para a integridade do pacote, por meio da qual os autores propõem uma nova estratégia para a fragmentação do pacote. Em [17], os autores apresentam uma solução para ataques ao roteamento e propõem um novo protocolo de roteamento. De forma similar, trabalhos como [15, 16] propõem soluções para um tipo de cenário ou exigem infraestrutura.

Para que a aplicação da VANET seja efetiva e viável é necessário que aspectos de segurança sejam garantidos, como a confidencialidade e a integridade [3]. Uma forma de prover a confidencialidade e a integridade da comunicação é através da formação de grupos [19, 20, 21, 22, 23, 24]. Com um grupo é possível projetar a segurança mediante distribuição e verificação de chaves, criptografia em grupo e assinatura digital. Soluções de segurança baseadas em grupo possuem características como o uso de uma única chave compartilhada para criptografia e assinatura digital. No entanto, a formação de grupo em VANET deve ser estudada em particular, dadas as suas características únicas, como distribuir e revogar chaves no grupo mesmo com alta dinamicidade da rede, entre outros desafios. Embora haja várias propostas na literatura, *não existe uma solução independente de protocolos e infraestrutura que mantenha a mensagem íntegra e confidencial a terceiros* [6].

Este trabalho propõe a criação de um *Framework* para aspectos de segurança em redes veiculares. Em específico, objetiva prover um *Framework* capaz de garantir a confidencialidade e a integridade das informações em VANET independente de roteamento e infraestrutura. A aplicação será delimitada a cenários urbanos que utilizam a comunicação V2V para transportar as informações da origem até o destino. Este *Framework* é constituído por um sistema de pontuação que considera o contato e permanência da conectividade entre os nós, com objetivo de formar um grupo seguro de comunicação. Com a formação do grupo, o *Framework* gerencia as relações de confiança para classificação de diferentes níveis entre os membros do grupo. Por fim, diferentes serviços são relacionados aos níveis de segurança do grupo. Estes serviços abordam desde o gerenciamento de chaves, grupos, confiança até a comunicação segura entre os nós do grupo. A versão preliminar deste *Framework* foi publicada em [25].

O conceito de segurança em diferentes níveis (ou multiníveis) em grupos para o contexto de VANETs V2V é uma inovação apresentada nesta tese, cuja abordagem não foi encontrada na literatura. Este conceito foi introduzido com objetivo de permitir adequações aos diferentes serviços da VANET, a diferentes níveis de segurança ou conectividades exigidas. Desta forma, é possível estabelecer hierarquias de distintos tipos de atuação entre os nós do grupo, sem a necessidade de formar novos grupos. As métricas utilizadas para a classificação dos nós baseiam-se nos atributos que são rapidamente percebidos entre eles na rede, sem que haja a necessidade de uma entidade centralizadora e de que sejam viáveis computacionalmente no contexto de nós distribuídos. Tais atributos são: contato, permanência e confiança.

Este trabalho está organizado da seguinte forma: o Capítulo 2 aborda as VANETs e os grupos de nós; o Capítulo 3 aborda os ataques à confidencialidade e à integridade das informações e o modelo de ataque para cada ataque estudado; o Capítulo 4 aponta os serviços de segurança; o Capítulo 5, os trabalhos relacionados a esta pesquisa; o Capítulo 6 apresenta o *Framework* GSeF4V; o Capítulo 7 apresenta as simulações, a análise de desempenho e os resultados obtidos; e, por fim, o Capítulo 8 traz as considerações finais e as indicações para trabalhos futuros.

2 REDES VEICULARES

Uma rede veicular deve prover a comunicação entre os usuários que possibilite o uso do Sistema de Transporte Inteligente (*Intelligent Transportation System* (ITS)) [26]. O ITS visa prover a conectividade ubíqua nas ruas, para usuários móveis, e uma comunicação eficiente de nós para nós, cujo principal objetivo é melhorar a segurança e as condições de direção [27], independente de uma determinada topologia, número de nós e da presença ou não de uma infraestrutura de rede.

O potencial de interconexão dos nós amplia a percepção de reconhecimento de eventos e condições deles nas ruas que não podem ser detectados por sensores ou por um condutor. Além disso, condições críticas podem ser detectadas e informadas por compartilhamento de informações entre os nós próximos [28]. Para compartilhar esta informação, os nós estabelecem uma rede de comunicação espontânea, conhecida como VANET. Este capítulo introduz os conceitos fundamentais sobre redes veiculares, que servirão de base para esta tese. Eles estão organizados em: arquitetura, características, tecnologias de comunicação, padronizações e aplicações.

2.1 CARACTERÍSTICAS DA VANET

A VANET é uma classe especial de MANET para ITS, em que os nós podem se movimentar rapidamente e se comunicam entre si ou com uma estação base *Road-Side Unit* (RSU). A VANET é uma rede auto-organizada e distribuída que pode não precisar de uma infraestrutura fixa para operar a comunicação. Cada nó pode receber ou transmitir mensagens através de ligações sem fio, ainda que com alta mobilidade e movimentos livres.

Esta rede possui algumas características únicas, que são: topologia altamente dinâmica, desconexões frequentes, sem limite de energia, variação na densidade da rede e variação das características da propagação de sinal [7].

A topologia da rede veicular muda constantemente devido à alta velocidade dos nós, tornando-a *altamente dinâmica*. Além disso, os nós podem entrar ou deixar a rede por curtos períodos, em função das várias direções que um nó pode seguir. Por causa desta mobilidade, o *link* da comunicação entre os nós pode sofrer *desconexões frequentes* durante a transmissão.

Nas VANETs, os nós utilizam a própria carga da bateria e, por isso, podem ser considerados *sem limite de energia*. Esta característica permite que a VANET utilizar maior poder computacional em relação a redes com restrição energética. Embora a VANET não tenha restrição de energia, possui a *restrição de mobilidade*, pois o movimento dos nós obedece ao *layout* das ruas e às regras de trânsito. Essa característica de restrição é interessante, pois, aliada ao comportamento do motorista, permite predizer a posição futura do nó, como apresentado em [7].

A VANET possui *variação na densidade da rede* em diferentes cenários. Esta característica é perceptível quando há a variação entre poucos ou nenhum nó para muitos nós. Por exemplo, um nó pode trafegar de áreas rurais com densidade baixa para o centro urbano com densidade alta; ou mesmo apenas em cenários urbanos, porém com variações entre o horário de calmaria e o horário de pico, período em que a densidade dos nós varia de baixa para alta.

As VANETs são projetadas para operar em ambientes urbano, rural e autoestradas, e, por isso, pode haver *variação das características de propagação de sinal*. Esses ambientes são bem diferentes, por exemplo: as autoestradas possuem poucos caminhos, porém espaçados;

este ambiente pode sofrer interferências pela reflexão dos objetos localizados no decorrer das estradas; os ambientes urbanos possuem muitos prédios, árvores e vários outros objetos, por isso a propagação do sinal sofre com sombras, múltiplos caminhos e atenuação; e nos ambientes rurais pode haver florestas densas, vales, campos que devem ser considerados para não haver interferência na comunicação sem fio, como observado em [7].

Em [8] , os autores abordam também a comunicação sem fio, a segurança do motorista, a quantidade de nós na rede, o tamanho da rede e a volatilidade. A *comunicação* na rede é feita totalmente sem fio; nesse sentido, a segurança deve ser garantida, para que ela seja viável.

A *quantidade de nós na rede* depende do trânsito. Por exemplo, em horários de pico, a rede terá muitos nós, e em horários mais calmos, terá poucos. O *tamanho da rede* pode variar muito a depender do local. Ela será maior em regiões centrais em comparação a bairros distantes. A VANET possui alta *volatilidade*, a comunicação entre dois nós é estabelecida quando eles estão próximos, porém estes nós podem se afastar e perder a comunicação por um curto período, isto pode ocorrer devido à velocidade deles ou mudança de rota.

2.2 FORMAS DE COMUNICAÇÃO

Existem dois tipos de comunicação: interna e externa. A comunicação interna refere-se à comunicação dos equipamentos de um nó e pode ser classificada em passiva e ativa. A comunicação interna passiva refere-se a um conjunto de ferramentas que melhoram a segurança dos passageiros, como por exemplo, o *airbag*. A comunicação interna ativa refere-se às ferramentas de assistência ao motorista, como o controle de faixa.

A comunicação externa pode ser feita por V2V, V2I ou V2X. A V2V atua sob comunicação de curto alcance e trafega a informação por meio de saltos entre os nós do remetente até seu destino. Esta comunicação pode ser estabelecida em estradas urbanas ou rurais, e não depende de nenhuma infraestrutura.

A V2I requer infraestrutura para os nós se conectarem diretamente às estações bases dispostas ao longo do percurso, isso facilita a comunicação, pois, embora haja a mobilidade, os nós sempre estarão conectados. Entretanto, não há infraestrutura global dedicada a redes veiculares com uma cobertura substancial nas ruas e estradas, sejam elas em áreas rurais ou urbanas [29].

A V2X é caracterizada pela comunicação entre o nó com qualquer outro elemento. Esta forma de comunicação vem se destacando com o crescimento da rede $5G$, tratada por alguns autores [30, 6, 31] como um novo paradigma $5G$-V2X. O principal objetivo dessa rede é permitir a efetiva comunicação completamente conectada, incrementando o suporte para novos serviços, como a direção de nós totalmente autônomos. Porém, para a $5G$-V2X ser praticável, é requisito que haja alta densidade de nós e infraestrutura celular para dar suporte à comunicação veicular [31].

Desta forma, considerar tanto V2I quanto V2X requer cenários com investimentos em infraestrutura que ofereçam cobertura tão vasta quanto possível para manter os nós sempre conectados. Existem cenários nos quais considerar tal infraestrutura é inviável, seja por questões econômicas ou pela simples inviabilidade tecnológica. Estes cenários devem ser avaliados e soluções devem ser projetadas para redes veiculares com comunicação direta entre os nós, ou seja, comunicação V2V sem a necessidade de uma infraestrutura.

2.2.1 Pilha de protocolo

Esta seção será dividida em 3 subseções: camada física e de enlace, camada de rede e camada de transporte e aplicação.

2.2.1.1 Camadas física e de enlace

As tecnologias de comunicação para aplicações ITS podem ser divididas em relação aos dois métodos de comunicação em VANET: baseado em infraestrutura e sem infraestrutura. As comunicações baseadas em infraestrutura podem utilizar ondas de rádio de frequência (como *Frequency Modulation* (FM)), *Digital Video Broadcasting* (DVB)/*Digital Audio Broadcasting* (DAB)), para comunicação *broadcast*, além de rede celular (como *Global System for Mobile* (GSM), *Universal Mobile Telecommunication System* (UMTS)). Para comunicações sem infraestrutura, podem ser usadas redes de curto alcance, como *Zigbee* e *Bluetooth*, ou de médio alcance, como *Wireless Fidelity* (WiFi) e *Dedicated Short Range Communication* (DSRC).

A camada de enlace, também conhecida como camada *Media Access Control* (MAC), fornece os meios para a transferência dos dados da rede, que podem ser em comunicações um para um ou um para muitos. Além disso, essa camada possui mecanismos de controle de fluxo, usados entre os nós para controlar o fluxo de dados da camada de agregação. A camada de enlace é responsável pelo transporte dos pacotes de um nó para outro.

O *Institute of Electrical and Electronics Engineers* (IEEE) vem desenvolvendo o padrão 802.11p [32], chamado *Wireless Access for Vehicular Environment* (WAVE), para comunicação de médio alcance em VANET. Com base na forma de ligações de dados, vários protocolos de roteamento de rede são propostos para VANETs, com abordagens específicas, muitos voltados às topologias da rede e sua dinamicidade, além de focarem nos serviços a serem oferecidos.

2.2.1.2 Camada de rede

Na camada de rede são implementados o endereçamento e as formas de roteamento. As VANETs podem utilizar os endereçamentos de IPv6 para permitir um grande número de nós na rede. As formas de roteamento são implementadas pelos protocolos de roteamento, que devem considerar as características da rede veicular.

Os protocolos de roteamento para VANET enfrentam a alta mobilidade dos nós e rápidas trocas da topologia, com foco em entregar um pacote mesmo sob essas circunstâncias, objetivando reduzir o atraso e a perda de pacotes. Os cenários destes protocolos podem ser de alta densidade de nós separados por curtas distâncias ou com poucos nós em alta velocidade, com maior espaço entre eles. Um protocolo de comunicação eficiente impacta na melhora de muitos fatores, como a confiabilidade do sistema, aumentando a porcentagem de entrega de pacotes, também reduz os efeitos das interferências causadas por altos prédios no percurso da comunicação e visa à escalabilidade.

Devido à topologia dinâmica, vários tipos de disseminação de dados foram propostos para atender a objetivos específicos. A disseminação de dados mais comum é por *broadcast*, através do qual os nós transmitem o mesmo pacote simultaneamente para seus vizinhos. Quando o tráfego é direcionado a um grupo de usuários, ao invés de envio individual, pode ser feito por meio de *broadcast*. No entanto, existem outras técnicas para disseminar os dados, como *unicast*, *geocast*, *multicast*, *peer-to-peer* e baseadas em *cluster*, como apresentado em [39]. Estes dados são trafegados em diferentes tecnologias de comunicação.

2.2.1.3 Camadas de transporte e aplicação

Nas camadas de transporte e aplicação é possível implementar os serviços que abrangem as funcionalidades das camadas de transporte, sessão, apresentação e aplicação do modelo de referência *Open System Interconnection* (OSI) [33]. Em VANET, a comunicação fim a fim é responsável por estabelecer a conexão e sessão entre os nós comunicantes. Ela pode ser

implementada tanto sobre o protocolo *User Datagram Protocol* (UDP) quanto o *Transmission Control Protocol* (TCP), pois dependendo do objetivo da comunicação a entrega deve ser confiável ou não, e para uma comunicação que não exija a confiabilidade, a sobrecarga das conexões TCP é desnecessária. Além disso, os serviços de multimídias ou de tempo real podem também ser ofertados para as redes veiculares, e, neste contexto, considerar o UDP torna-se ainda mais vantajoso. Os trabalhos [34, 35, 36] exploram as abordagens dos protocolos de transporte em redes veiculares.

É possível implementar serviços como segurança e aplicações diversas para as VANETs. Estes serviços podem conter um protocolo próprio de comunicação como forma de padronização e apresentação dos dados diretamente entre as aplicações ou oferecer uma camada extra de segurança, como a cifragem dos dados transmitidos. Existem diversas soluções que envolvem a comunicação ao nível de camada de aplicação, alguns estudos recentes podem ser encontrados em [37, 38, 39, 40].

2.3 PADRONIZAÇÃO

Os nós na VANET possuem diversos recursos, como sensores, atuadores, dispositivos de armazenamento, microprocessadores, que interagem entre si para prover serviços aos utilizadores do nó. Os nós que possuem estes recursos podem se comunicar com os recursos de outros nós para prover as aplicações da ITS [41, 42]. Para haver a comunicação entre diferentes nós e diferentes dispositivos é necessário que haja padronização na tecnologia da comunicação entre eles, independentemente do fabricante ou tecnologia empregada nestes dispositivos.

Desde 1999, a FCC alocou o espectro de 75MHz na banda de 5.9GHz para comunicação dedicada de curto alcance (DSRC). Este espectro foi planejado para oferecer 7 canais com largura de banda de 10 MHz cada, dos quais um canal está reservado para controle, dois canais das bordas do espectro são reservados para propósitos específicos e o restante para os diferentes serviços [43]. A partir dessa iniciativa, o IEEE vem desenvolvendo o padrão 802.11p [32], também conhecido como WAVE, para comunicação de médio alcance em VANET. Trata-se de uma comunicação sem fio, uni ou bidirecional, projetada para o ITS. O modelo WAVE também utiliza outras especificações nas demais camadas, além da 802.11p para as camadas física e enlace.

A camada física do padrão WAVE utiliza os 7 canais de 10 MHz para cada banda. O espectro é alocado nas frequências de 5.860 a 5.920GHz. Os canais das bordas foram direcionados para proteção contra acidentes de vida (Canal 5.860GHz) e poder de longo alcance (5.920GHz). Os demais são os canais de serviço e o canal de controle, como ilustrado na figura 2.1. Os diferentes canais não podem ser usados simultaneamente, pois este padrão de comunicação deve permitir unidades de recebimento únicas ou múltiplas. Cada estação continuamente alterna entre o controle de canais e um dos canais de serviço ou canal de segurança.

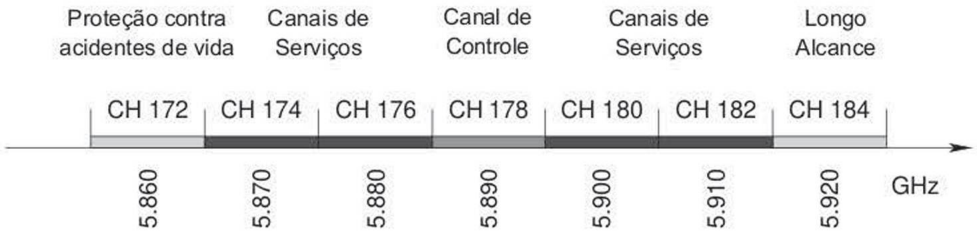


Figura 2.1: Organização dos canais do padrão de comunicação WAVE. Adaptado de [1].

Na camada de enlace do WAVE, as mensagens de aplicação são categorizadas conforme prioridades e utilizam esta categorização para organizar os pacotes em fila. Há uma fila para cada prioridade, e estas filas disputam o meio externo de comunicação usando seus parâmetros de contenção [1] para enviar os pacotes. O *Carrier Sense Multiple Access/Collision Avoidance* (CSMA/CA) é utilizado para garantir que os nós compartilhem o meio de transmissão corretamente, da seguinte forma:

- O nó remetente envia uma mensagem curta (*Ready To Send* - (RTS)) ao nó de destino.
- O nó de destino responde com outra mensagem curta (*Clear To Send* - (CTS)).
- Todos os nós ao alcance do nó remetente percebem o RTS.
- Todos os nós ao alcance do nó destinatário percebem o CTS.
- Os demais nós sabem da ocorrência de transmissão de pacotes entre os nós remetente e destinatário.

Entretanto, há casos em que mais de um nó pode sentir o meio simultaneamente, causando colisão desta breve transmissão. Nesta situação, é realizado um *backoff* exponencial, como no *Carrier Sense Multiple Access/Collision Detection* (CSMA/CD), no qual cada nó interrompe a sua transmissão e escolhe um tempo aleatório para tentar a retransmissão.

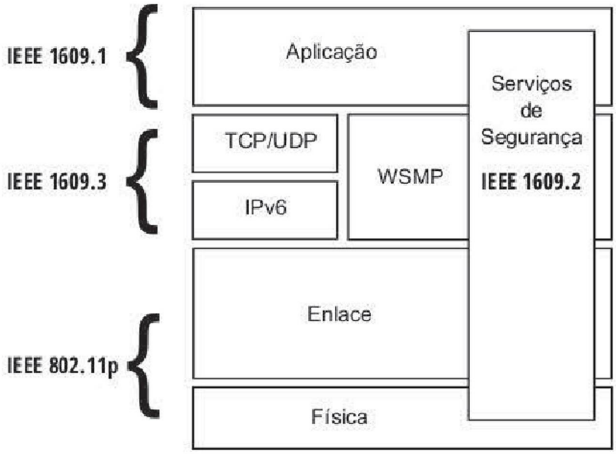


Figura 2.2: Modelo WAVE relacionado com os documentos de especificações que o compõem. Adaptado de [2].

Além do padrão 802.11p, os documentos IEEE P1609.1 [44], IEEE P1609.2 [45], IEEE P1609.3 [46] e IEEE P1609.4 [47] também são empregados na VANET [2], constituindo a arquitetura WAVE. O documento IEEE 1609.1 especifica a comunicação entre as OBUs e os aplicativos remotos. Ele atua como um gerente de recursos que multiplexa a comunicação de várias aplicações com várias OBUs. O padrão IEEE P1609.2 especifica o serviço de segurança para aplicações e mensagens de gerenciamento, nele são definidos os formatos das mensagens de segurança no padrão WAVE. O padrão P1609.3 especifica os serviços da camada de Controle de Link Lógico (LLC), camada de transporte e de rede. Ele permite utilizar IPV6, TCP/UDP ou mensagens curtas do padrão WAVE (WSMP - *WAVE Short Message Protocol*) mais adequadas para o ambiente veicular, para reduzir a baixa sobrecarga. O padrão IEEE P1609.4 especifica as operações de múltiplos canais, em que os dispositivos monitoram o canal de controle para obter os números de canais de serviço. A Figura 2.2 ilustra a arquitetura WAVE.

Outras tecnologias de comunicação também são empregadas nas VANETs. Em [43, 48], os autores também abordam as VANETs com uso do padrão 802.16 para aplicações multimídia na rede, enquanto o IEEE 802.11p fica voltado para comunicação V2V e V2I entre os nós. Além disso, a rede 5G está sendo explorada como possibilidade de comunicação veicular com dispositivos da *Internet of Things* (IoT) [48, 49].

2.4 APLICAÇÕES

Nas VANETs, os nós podem ser equipados com vários sensores que os permitem coletar, processar e distribuir informações. Estas informações podem ser usadas para várias aplicações que beneficiam os motoristas dos nós e a própria rede na qual estão inseridos.

Segundo [5], estas aplicações são classificadas como aplicações relacionadas à segurança ou aplicações não relacionadas à segurança. As relacionadas à segurança são: detecção de pré-colisão, velocidade de curva, mudança de faixa, violação de sinal de trânsito, luz de freio eletrônico de emergência, alerta cooperativo de colisão frontal, sinal de parada de movimento e assistente de virar à esquerda.

As aplicações não relacionadas à segurança são chamadas de comerciais ou de conforto. Essas aplicações visam melhorar a eficiência do tráfego, o conforto dos passageiros e as plataformas comerciais, como publicidade ou cobrança de pedágio. As aplicações são: disponibilidade de informações meteorológicas, intensidade de tráfego e capacidade de localizar pontos de interesse como estacionamentos, postos de combustível, *shoppings*, hotéis, restaurantes, entre outros [5].

Em [50], os autores apresentam algumas aplicações, como navegação ITS para automóvel *ad-hoc*, controle de congestionamento, arquitetura de serviço *ad-hoc*, rastreamento de nós, aplicações de conveniência e entretenimento, distribuição de dados geográficos e melhoria da infraestrutura, descritas a seguir:

- Navegação ITS para automóvel ad-hoc: esta aplicação permite otimizar a direção, tornando-a mais segura, como, por exemplo, na ação de mudança de faixa e/ou saída da estrada. Quando o nó precisar mudar de faixa, transmitirá a sua intenção e os demais nós abrirão espaço para ele realizar a mudança com segurança. Ação semelhante ocorre quando o nó pretende sair da estrada, pois para isso é necessário que ele informe aos demais sobre essa intenção, para que diminuam a velocidade, evitando colisões.

- Controle de congestionamento: em ruas movimentadas ou em horário de pico, há um grande fluxo de nós gerando lentidão; ou, também, em casos de acidente, quando o trânsito fica parado e os nós se acumulam com o passar do tempo. Para evitar isso, a aplicação de controle de congestionamento envia uma mensagem para os nós, informando sobre o problema. Após o nó recebê-la, pode buscar um caminho melhor e mais rápido.

- Arquitetura de serviço *ad-hoc*: esta aplicação fornece aos motoristas informações sobre o comércio local, usando a rede VANET como interface. Para isso, é necessário que os donos das lojas instalem um dispositivo de infraestrutura que encaminhe informações sobre o seu estabelecimento comercial para a rede VANET. Após isso, estas informações serão transmitidas para os nós que passarem perto do estabelecimento, detalhando a sua localização e os serviços ali oferecidos.

- Rastreamento de nós: esta aplicação pode ser útil para localizar nós roubados. Para isso, devem ser usadas as próprias identificações que o nó fornece ao se conectar na rede

VANET. Através desta aplicação também é possível o rastreamento de nós pertencentes a criminosos, na tentativa de capturá-los, a exemplo dos envolvidos em sequestro.

- Aplicações de conveniência e entretenimento: através destas aplicações, um usuário remoto pode verificar se o usuário atual do nó está realizando atividades de manutenção de rotina, como, por exemplo, uso do cinto de segurança, controle de velocidade, de combustível, entre outros. Outra função é a conexão da rede VANET à internet em locais remotos, possibilitando ao usuário o acesso à internet, para, por exemplo, fornecer retransmissão de vídeo em tempo real enquanto viaja.

- Distribuição de dados geográficos: esta aplicação usa as próprias informações de rotas dos nós da rede VANET para criar um mapa detalhado da área local, tanto com rotas quanto com os comércios locais. Em casos de congestionamento, ela poderia apontar um caminho alternativo. Também pode ser usada para detectar informações ao redor dos nós, além das estradas, como, por exemplo, vagas de estacionamento e condições perigosas, como buracos na estrada ou neve.

- Melhoria da infraestrutura: esta aplicação poderia modernizar os sistemas de pedágio, tornando-os mais automáticos, criando um sistema integrado. Nesse sentido, quando um nó passasse pelo posto de cobrança seria registrado o número de identificação emitido por ele, tal número estaria vinculado a um banco de dados detentor das informações relativas à conta para o pagamento do pedágio. Esta aplicação evitaria a necessidade de instalação de um *hardware* nos nós, facilitando a adesão ao sistema e diminuindo o congestionamento nas praças de pedágio.

As aplicações podem reduzir o número de acidentes, coletar informações e auxiliar o motorista em relação ao trânsito ou ao comércio próximo. Desta forma, beneficiam tanto o motorista quanto a rede.

2.5 FORMAÇÃO DE GRUPOS

Nas VANETs, quando vários nós se deslocam próximos uns aos outros, podem formar grupos. Existem várias formas de formação de grupos [51, 52, 53, 54, 55, 56, 19, 57]. Estes trabalhos serão abordados na subseção 5.1.

A formação de grupos é abordada com o objetivo de obter alto rendimento e menor sobrecarga da rede [58, 59], realizar comunicação segura e eficiente dentro do grupo [60], evitar colisões de dados, diminuindo o número de conexões trocadas entre os veículos [61], promover a privacidade de localização dos nós [62], entre outros.

Existem várias formas de adquirir segurança e confiança nos grupos. A segurança pode ser obtida através de autenticação baseada na assinatura de grupos [53], de sistema de gerenciamento e revogação de confiança baseado em grupo [57], de sistema de gerenciamento de chaves [21], de esquema de autenticação em lote e de um mecanismo de acordo de chave secreta de grupo [63].

A confiança pode ser adquirida através de modelo de confiança com comunicação baseado em líder de grupo [22], de esquema de gerenciamento de confiança com reconhecimento de contexto [64], de cooperação centralizada e distribuída entre os nós [65] ou através do uso do perfil de rede social do motorista do nó. A próxima seção aborda a confiança baseada em rede social.

2.6 CONFIANÇA BASEADA EM REDE SOCIAL

As redes sociais online (*Online Social Network - OSN*) [66] são plataformas que permitem a seus usuários interagir por meio de publicações de conteúdo, contatos diretos e reações aos conteúdos publicados, estabelecendo uma relação de proximidade, podendo, também, refletir suas relações sociais físicas. Estas relações podem representar um modelo de confiança social entre seus usuários. Neste contexto, o modelo de confiança social pode ser entendido pelos aspectos de risco e interdependência [67]. O risco representa uma incerteza sobre a intenção da outra parte. A interdependência significa que a obtenção do almejado pelas duas partes depende de ambos os usuários, ou seja, nenhum deles pode alcançar seu próprio objetivo sem o outro. Dessa forma, qualquer alteração desses aspectos, seja positiva ou negativa, impacta diretamente na relação de confiança entre as partes.

As OSNs podem também servir como base para estabelecer, de certa forma, uma relação de confiança entre seus usuários. Essas abordagens podem ser encontradas no contexto das VANETs, nas redes sociais veiculares. As redes sociais veiculares - Vehicular Social Network (VSN) [68] são sistemas de comunicação móvel considerados como uma classe emergente das VANETs, em que os nós podem se comunicar explorando atributos sociais das OSNs. A VSN não provê mecanismos de segurança na comunicação, pois o objetivo é propor uma forma de relação de confiança entre os nós da rede, considerando características únicas, como mobilidade da rede, conforme apresentado em [69, 70, 71]. Embora as VSNs sejam uma área das VANETs baseadas na teoria das redes sociais, elas fogem ao escopo desta pesquisa. No entanto, os modelos de seguranças e formas como utilizam a confiança são interessantes para este estudo.

Algumas pesquisas [72, 73, 74] utilizam as OSNs ou parte das VSNs em suas soluções de confiança nas VANETs. Em [72], os autores apresentam uma arquitetura para VANETs chamada de Social Internet of Vehicules (SIoV) para conectar as OSNs com a Internet dos Veículos (IoV). O objetivo é estimar a honestidade dos motoristas e passageiros com base em seus perfis da rede social. A identificação é realizada por meio de ferramentas intermediárias, como leitor de digital, leitor de íris ou identificação de usuário para autenticação na rede social. As informações são obtidas por meio de TAs e, em sua ausência, a conexão acontece através da utilização de *smartphones* e rede de telefonia celular. As principais informações do perfil social consideradas para estabelecer determinado grau de honestidade são: descrição inicial do perfil; escolha da imagem ou avatar; interações, comentários e discussões; e julgamento dos demais usuários. Estas informações são combinadas com informações obtidas do comportamento do veículo na rede. Além disso, este trabalho também utiliza as métricas *Advogato* e *Honesty Humam Factor* [75]. Todo o sistema é baseado no uso de RSU como autoridade confiável. Esta solução não funciona para ambientes sem infraestrutura e limita-se a buscar informações que as OSNs oferecem. Embora esta solução não seja definitiva, ela oferece informações complementares, principalmente para motoristas de aplicativos. Além disso, essas informações podem ser compartilhadas entre os demais veículos da VANET que já possuem alguma relação de confiança preestabelecida.

Em [73], os autores propõem um modelo de gerenciamento sensível ao contexto para avaliar a veracidade das mensagens recebidas de veículos, a fim de garantir que uma informação falsa não influenciará o processo de tomada de decisão do motorista. Para alcançar este objetivo, os autores combinam os dados trafegados entre os veículos com os perfis sociais dos motoristas, beneficiando-se do envolvimento social online como reflexo dos relacionamentos físicos entre os indivíduos. Estes dados da rede social servem como base para um modelo de aprendizado por reforço de cada veículo, com objetivo de ajustar a estratégia de avaliação em diferentes cenários. Em [74], os autores apresentam uma proposta para VSN que reduz os veículos *sybil* na rede. Esta

abordagem usa as OSNs para obtenção de informações dos motoristas e gráficos de proximidades entre os veículos na VANET, além de estabelecer uma lista de *ranking*.

2.7 CONSIDERAÇÕES SOBRE O CAPÍTULO

Este capítulo apresentou as redes veiculares *ad-hoc* e a formação de grupos em redes veiculares. As VANETs são redes formadas por nós que se comunicam entre si ou com estações-base, e podem ocorrer em qualquer tipo de estrada.

Vários trabalhos utilizam a formação de grupos para evitar mensagens redundantes, fornecer alto rendimento, diminuir o congestionamento da rede, evitar a propagação de informações irrelevantes e reduzir a sobrecarga de processamento. Porém, a formação de grupos lida com problemas de constante mudança dos nós pertencentes ao grupo e de formação de grupos que duram pouco tempo.

Existem vários tipos de comunicação: intraveicular, V2V, V2X e V2I. As comunicações que precisam de infraestrutura se tornam impossíveis de implementar, dado o custo de instalação dessas infraestruturas. Por causa disso, esta pesquisa se concentrará apenas nas comunicações V2V.

3 ATAQUES A REDES VEICULARES

Os ataques em redes veiculares procuram obter informações, afetar o comportamento da rede ou prejudicar o desempenho do sistema. Stallings [3] classifica os serviços de segurança como medidas de confidencialidade, integridade, disponibilidade, autenticidade e não repúdio, definidas a seguir:

- Confidencialidade: garante que a informação não será conhecida por pessoas que não estejam autorizadas; ou seja, ninguém além do destinatário tem acesso ao conteúdo da informação.
- Integridade: garante que a informação armazenada ou transferida será apresentada corretamente a quem a consulta. Em outras palavras, não permite a alteração da informação.
- Disponibilidade: garante que a informação possa ser obtida sempre que for necessário; isto é, que esteja disponível para quem precisar. Além disso, garante que o sistema permaneça disponível.
- Autenticidade: não permite alterações de identidade.
- Não repúdio: garante que o nó emissor ou receptor não negue uma mensagem transmitida; dessa forma, quando uma mensagem é enviada, o receptor pode provar quem a emitiu e o emissor pode provar que ela foi recebida.

Os ataques à confidencialidade e à integridade da informação são mais sensíveis para as redes veiculares, pois sem haver formas de oferecer estes serviços de segurança todo o sistema torna-se inviável [10, 11, 12, 13]. Este capítulo apresenta uma revisão da literatura sobre ataques à confidencialidade e à integridade da comunicação em redes veiculares com comunicação V2V.

Os referidos ataques podem ser ativos ou passivos. Os ataques ativos agem diretamente no sistema, causando modificações para atingir seus objetivos. Os ataques passivos não alteram a mensagem nem interferem no sistema. Estes são mais difíceis de se detectar e a contramedida mais eficaz é a prevenção. Tanto os ativos quanto os passivos são realizados por diferentes tipos de atacantes, classificados em [11, 76] como internos ou externos, maliciosos ou racionais, ativos ou passivos, locais ou estendidos, e independentes ou em coluio, conforme a natureza ou tipo do ataque.

- Interno x externo: o atacante interno é um nó autenticado na rede; enquanto o externo é um componente fora da rede, com recursos limitados, mas que consegue prejudicar a comunicação.
- Malicioso x racional: o atacante malicioso não visa a obter privilégio, somente a interferir no sistema; diferentemente do racional, que realiza o ataque em benefício de si próprio.
- Ativo x passivo: o atacante ativo possui acesso à rede e interfere diretamente no sistema; já o passivo age em sigilo, para coletar informações.

- Local x estendido: o atacante é considerado local se o acesso dele for limitado, controlando veículos ou estações base da rede. O estendido possui acesso através de vários membros espalhados na rede.

- Independente x coluio: o atacante é considerado independente se ele agir sozinho para a troca de informações e considerado em coluio se estiver cooperando com outros, a fim de realizar ataques eficazes.

Um ataque à confidencialidade pode ser bem-sucedido através de um ataque passivo. Ele pode ser realizado por um atacante interno, racional e passivo, com objetivo de analisar o tráfego das mensagens entre os diferentes nós por meio de coleta e filtragem dos dados. A coleta da informação no ataque à confidencialidade pode ser suficiente para encontrar uma informação confidencial ou que seja sensível ao funcionamento correto do sistema.

A informação obtida indevidamente pode ser enviada ao atacante para a obtenção de privilégios sobre o sistema ou sobre os usuários, ou, ainda, servir de base para outros tipos de ataque. Os principais ataques à confidencialidade em redes veiculares com comunicação V2V são *Eavesdropping*, *Collusion*, *Impersonation* e *IN-Transit Tampering* [76, 77, 78, 79, 80, 81].

O ataque à integridade ocorre através de um ataque ativo. O atacante pode ser do tipo interno, malicioso, racional ou ativo. Neste ataque, ele altera intencionalmente os dados trafegados durante uma comunicação, de forma não autorizada. Um serviço de segurança que garanta a integridade deve proteger a alteração intencional ou acidental da informação. Com a violação da integridade, o atacante pode induzir um nó ao erro, obter privilégios indevidos, modificar o comportamento do sistema e ser utilizado como ponto de partida para outros tipos de ataques. Os principais ataques à integridade da informação são os ataques de alteração, *IN-Transit Tampering*, falsificação de mensagem, ilusão e motorista ganancioso. As mensagens a serem adulteradas podem ser recebidas diretamente pelo atacante ou capturadas por um ataque do tipo *man-in-the-middle* [76, 82, 83, 84, 85].

3.1 *EAVESDROPPING*

O ataque *eavesdropping* é visto como um dos mais potentes ataques à confidencialidade da informação em redes veiculares [11, 86]. O seu objetivo é coletar informações confidenciais de outros nós secretamente. O ataque de análise de tráfego [87] possui uma abordagem bem similar, na qual o atacante intercepta o tráfego de pacotes para coletar dados. Nesta pesquisa, todos os ataques com essa abordagem serão considerados como *eavesdropping*.

O atacante pode ser um nó em movimento ou estacionado. Deve-se considerar as duas situações. O atacante em movimento pode ser pertencente a uma rede veicular que coleta os dados somente dos seus vizinhos ao longo do trajeto, no entanto, o estacionado pode agir como uma unidade estacionária e coletar os dados de todos os nós que passam por ele, como se fosse um ataque de uma unidade V2I.

O ataque *eavesdropping* com um nó estacionado pode ser entendido como um ataque de uma unidade estacionária, ainda que o atacante possa obter vantagem da mobilidade do nó para sempre estacionar em pontos onde haja maior fluxo, onde a rede esteja vulnerável ou o local seja mais estratégico, como nós específicos que possuam informações que se almeja obter indevidamente. A Figura 3.1 ilustra este ataque.

Nesta ação, o atacante pode ser interno ou externo. Os atacantes externos são os que não pertencem à rede veicular. Embora seja mais difícil executar este ataque, ele é possível. O atacante pode espionar o canal de comunicação da rede veicular em busca de informações de

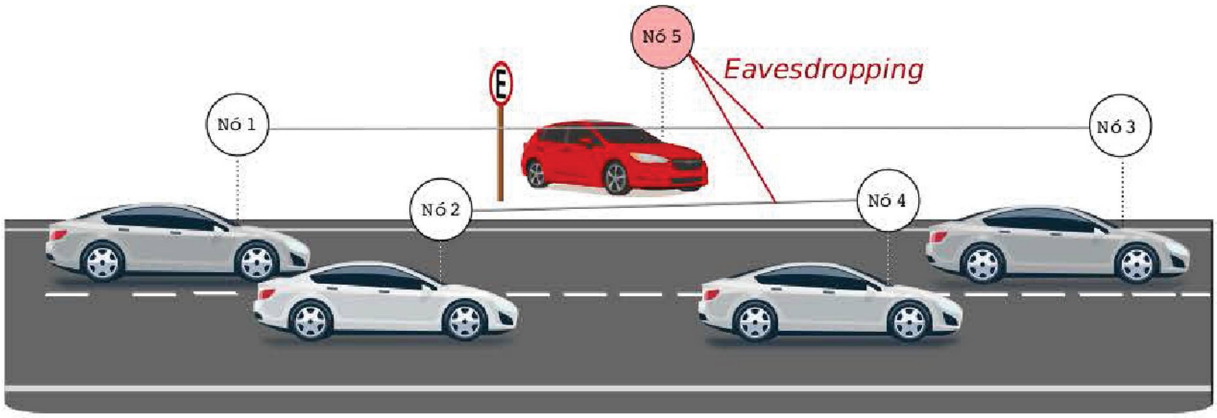


Figura 3.1: Ataque *eavesdropping*.

usuários sem que estes percebam e usar tais informações futuramente, de forma que gere alguma vantagem para si ou malefício ao nó atacado.

Os atacantes internos podem coletar dados com consentimento do usuário durante uma troca de informação, e isto não representa um ataque. Nesta ação, é possível que um nó interno na rede, sem notificar os outros usuários, colete informações para propósitos indevidos ou para ter vantagem sobre os demais, como ilustrado em [88]; situação em que o atacante pode ser um nó contratado por uma empresa para coletar dados de dentro da rede e repassá-los ao contratante para potencializar seu produto de mercado.

Tanto de forma interna quanto externa, o atacante pode usar o *eavesdropping* como ponto de partida de novos tipos de ataque. Por exemplo, ele pode ser usado por um nó malicioso para violar a privacidade, ao se obter a identidade real de um nó e usá-la indevidamente [89]. O atacante monitora o tráfego da rede até encontrar uma informação que deseja capturar. Ele pode examinar as mensagens trafegadas para reunir informações sobre os nós e os padrões de comunicação, e, assim, realizar ataques ativos.

3.2 *COLLUSION*

O ataque *collusion* (coluio) é utilizado nas redes veiculares de forma direta ou combinado com o ataque de *Man-in-the-Middle* (MitM) [90]. O *collusion* consiste no acordo secreto entre dois ou mais nós da rede veicular para coletar informações mesmo sem determinado nível de acesso ou permissão [78]. O ataque *Man-in-the-Middle*, também conhecido como homem-do-meio, é uma interceptação da informação para modificá-la de forma seletiva, disfarçando-se o atacante como nó da rede envolvido na comunicação [3].

No ataque direto, o nó atacante infiltra-se como um terceiro na comunicação entre nós. Sem ser percebido, ele intercepta a mensagem, obtém seu conteúdo e repassa a informação a outro nó atacante em coluio e vice-versa. No ataque combinado com o MitM, o nó malicioso observa a comunicação estabelecida entre dois outros nós, para fingir ser um deles e responder ao outro, com objetivo de capturar uma informação ou mesmo inserir informações falsas entre eles. A Figura 3.2 ilustra o ataque *collusion*, em que o nó 3 está inserido na rede como um nó convencional, mas está em coluio com o atacante, representado pelo nó 5, fornecendo informações dos demais nós.

O *collusion* pode ser combinado com outros tipos de ataque. Em [91], os autores mostram esse ataque combinado com outro chamado *bogus*. Nele, o atacante entra em coluio com o maior número de atacantes que conseguir, para criar informações falsas e beneficiar-se ou beneficiar um determinado grupo de atacantes, como, por exemplo, ao forjar a informação de

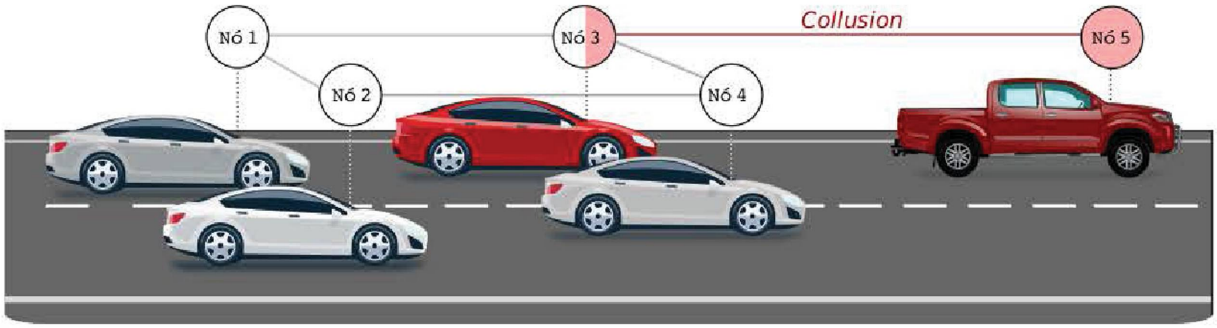


Figura 3.2: Ataque *collusion*.

um congestionamento em uma determinada rodovia para obter acesso livre de trânsito em seu caminho.

3.3 ATAQUES DE ALTERAÇÃO

Os ataques de alteração adulteram as mensagens e as enviam para outros usuários, com objetivos de obter privilégios, beneficiar-se, prejudicar o sistema ou os elementos que o compõem. A Figura 3.3 mostra um exemplo simples desses tipos de ataque, no qual o atacante recebe uma mensagem de aviso de freada brusca, adultera-a e envia para os outros nós a mensagem modificada informando que a rodovia está livre.

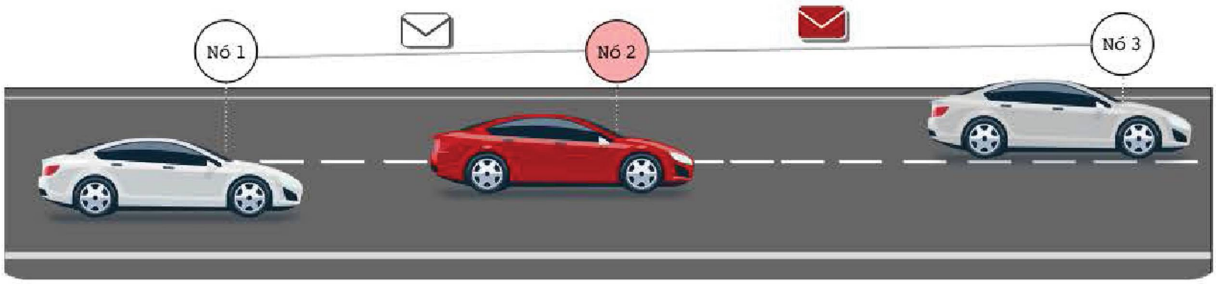


Figura 3.3: Ataque de alteração.

O ataque de alteração de mensagem pode ser combinado com outro tipo de ataque, como o *replay*. Neste ataque, o atacante deixa de encaminhar um aviso de congestionamento para usá-lo posteriormente, forçando os nós a aguardarem no tráfego, provocando obstrução de trânsito. A mensagem também pode ser adulterada para burlar possíveis técnicas de detecção, como número de sequência. O ataque de alteração atinge a integridade da mensagem, e é feito por um atacante interno e malicioso.

Este ataque também pode ser entendido como ataque de *falsificação da mensagem*, pois o atacante modifica as mensagens, falsificando-as para beneficiar-se. O ataque *Forgery* também utiliza esta abordagem. Nele, um atacante único causa a falsificação para a transmissão de mensagens de aviso de perigo falsas (por exemplo, informação de más condições da estrada), levando grandes porções da rede a serem contaminadas com informações incorretas. As várias dimensões e o comportamento do atacante na estrada em VANETs são discutidos em [92].

Um ataque similar ao ataque de falsificação é o *greedy drivers* (motorista ganancioso), por meio do qual o atacante afeta a rede para benefício pessoal. Os nós vizinhos são convencidos pelo atacante de que há congestionamento à frente na estrada, de modo que possam escolher uma rota alternativa, proporcionando ao atacante um caminho livre adiante. Este ataque interfere

na integridade da mensagem e pode ser feito por um atacante interno, racional e ativo. Além disso, o ataque *greedy drivers* ocorre, geralmente, com o ataque de falsificação de mensagem, para alterar as mensagens, e com o de atraso de mensagem, no qual as mensagens críticas não são transmitidas. A Figura 3.4 mostra um exemplo deste ataque, em que o atacante (nó 1) envia uma mensagem de alerta aos vizinhos para obter o acesso livre à rodovia em sua frente.

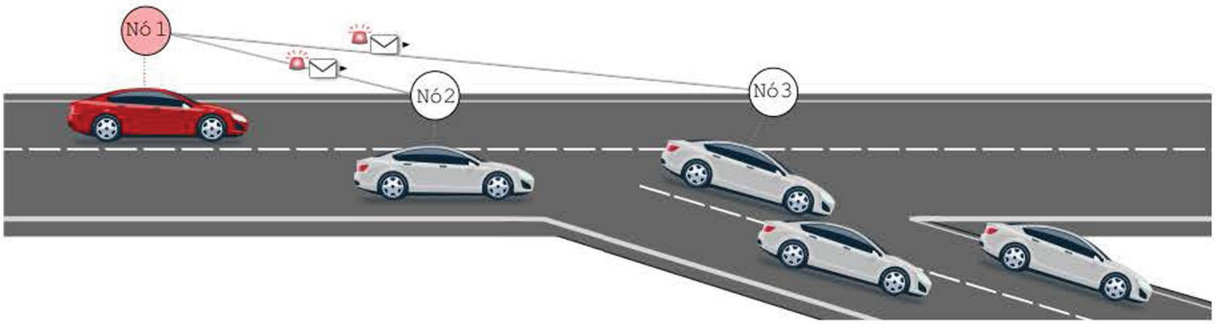


Figura 3.4: Ataque *greedy drivers*.

O ataque de *ilusão* tem objetivo e modo de operar muito similares aos do ataque do *motorista ganancioso*. Nele, o atacante visa afetar o comportamento dos outros nós, distribuindo informações falsas sobre o cenário real da estrada. Para isso, ele altera a velocidade percebida, a posição, a direção, entre outros fatores, de modo a conseguir alguma vantagem com isso. O atacante pode falsificar e enganar os sensores do nó, transmitindo sinais de alerta de tráfego não corretos. Essa transmissão errada pode levar a acidentes, engarrafamentos, avisos de más condições na estrada, além de causar perda de desempenho da rede.

Este ataque afeta a integridade da mensagem e pode ser feito por um atacante classificado como interno e malicioso. A Figura 3.4 pode também servir para exemplificar o ataque de ilusão, pois o atacante pode perceber que os nós da frente estão se movendo lentamente e gerar uma mensagem falsa de acidente, fazendo com que os demais motoristas acreditem no acidente e procurem uma rota alternativa.

Uma classificação de ataque semelhante ao ataque de alteração é o ataque *IN-transit tampering* (adulteração de tráfego em trânsito), em que o nó que atua como retransmissor pode reproduzir, modificar, eliminar, corromper as mensagens ou interromper a comunicação. Desta forma, o atacante pode manipular as mensagens de segurança ou as notificações de tráfego [76]. Este ataque atinge a confidencialidade e a integridade das mensagens, e pode ser feito pelo atacante classificado como interno e ativo.

3.4 *IMPERSONATION*

No ataque *impersonation*, o atacante tenta conseguir as informações pessoais sobre as entidades da rede, na tentativa de adquirir as identidades de outros nós, para obter benefícios. O atacante também pode associar a identidade dos nós às mensagens enviadas. Neste ataque, o nó pode fingir que possui uma ou mais identidades. O problema desse ataque é que os demais nós da rede não conseguem diferenciar se a informação recebida é de origem de apenas um nó ou de vários nós. Dessa forma, o atacante consegue moldar a rede conforme a sua necessidade e os seus objetivos [93, 94].

O *impersonation* ataca a confidencialidade das mensagens. O atacante é classificado como externo, malicioso e passivo. A Figura 3.5 mostra um exemplo deste ataque. Neste exemplo, o atacante altera a sua identidade para enviar informações incorretas. O nó 1 as envia ao atacante que está envolvido em um acidente e encontra-se parado na estrada. O atacante toma posse da

identidade do nó 1, altera a mensagem para parecer um nó em movimento, sem danos, e a envia aos seus vizinhos, gerando informações incorretas referentes às condições da estrada [95].

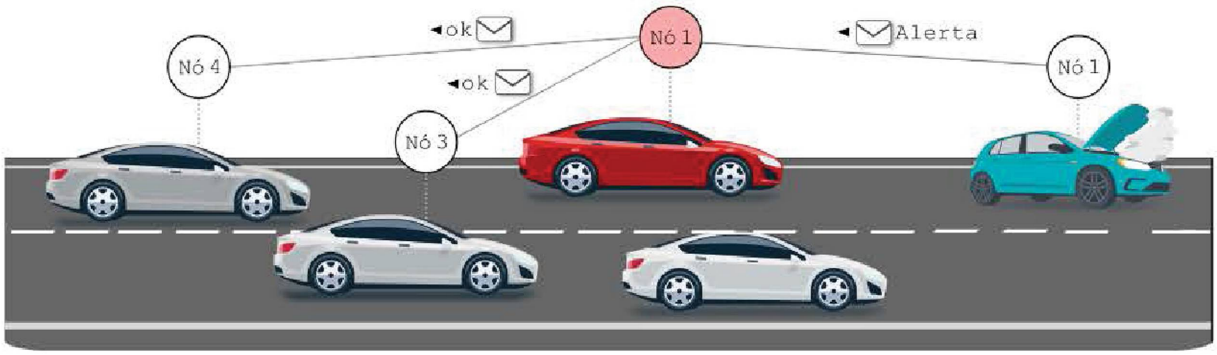


Figura 3.5: Ataque *impersonation*.

3.5 CONSIDERAÇÕES SOBRE O CAPÍTULO

Em redes veiculares, alguns tipos de pacotes enviados pela rede devem ser protegidos contra possíveis ataques. Este capítulo apresentou os ataques à confidencialidade e à integridade da informação que podem atuar nas redes veiculares com comunicação nó a nó, ou seja, sem infraestrutura. Foram mostrados a descrição desses ataques e os cenários que os exemplificam. A Tabela 3.1 apresenta os ataques, os serviços de segurança aos quais estão relacionados e a classificação dos atacantes.

Tabela 3.1: Classificação dos ataques.

		Serviços de segurança		Classificação dos atacantes									
Ataques	Referência	Confidencialidade	Integridade	Interno	Externo	Malicioso	Racional	Ativo	Passivo	Local	Estendido	Independente	Coluio
Eavesdropping	[77]	X		X	X		X		X	X		X	
Collusion	[78]	X		X			X	X			X		X
Alteration	[76]	X	X	X		X		X		X		X	
Falsificação	[82]		X	X			X	X		X		X	
IN-Transit Tampering	[76]	X	X	X			X	X		X		X	
Impersonation	[80]	X			X	X			X	X		X	
Motorista Ganancioso	[84]		X	X			X	X		X		X	
Ilusão	[83]		X	X		X		X		X		X	

Entre os vastos desafios para garantir a segurança contra ataques em redes veiculares [85] encontra-se a manutenção da confidencialidade e da integridade da informação, temática que é foco deste trabalho, assim como de outras pesquisas [10]. O método mais eficaz para prevenir o ataque à confidencialidade é o uso da criptografia de dados sensíveis. Uma abordagem desafiadora ao se considerar as restrições inerentes às VANETs. Além disso, a proteção à integridade pode envolver diferentes tipos de soluções que devem ser estudadas. O próximo capítulo (4) aborda os aspectos de segurança para redes veiculares.

4 SEGURANÇA PARA VANETS

Este capítulo aborda os mecanismos de segurança que visam garantir a *confidencialidade* e a *integridade* da informação. Nos mecanismos de segurança existem vários serviços que podem promover esta garantia. Stallings [3] classifica esses mecanismos como técnicas para implementar os serviços de segurança. Essas técnicas podem ser implementadas de forma individual ou em conjunto para fornecer um serviço específico.

A confidencialidade pode ser garantida, principalmente, por meio da cifragem. A integridade pode, também, ser garantida por meio da cifragem e da assinatura digital. Embora a principal solução para evitar o ataque à confidencialidade seja a cifragem [3], os nós que compõem uma rede veicular são naturalmente dinâmicos e não permitem o uso tradicional de uma única chave compartilhada entre toda a rede. O uso de algoritmos de criptografia assimétricos é uma solução viável para sistemas nos quais os nós não são conhecidos, mas precisam se comunicar de forma segura, ou ainda utilizar formas alternativas como formação de grupos. No entanto, há desafios a serem vencidos, como a forma de distribuição das chaves públicas e a verificação dessas chaves, dado que a mobilidade e disrupção devem ser consideradas.

A formação de grupos dinâmicos nessas redes (ver Seção 5.1) pode ser uma forte aliada no processo de garantir a confidencialidade na comunicação da VANET, pois é possível adaptar uma chave simétrica efêmera para o contexto de grupos e oferecer uma solução com melhor desempenho do que a cifragem assimétrica [96]. Os grupos podem, também, estabelecer uma relação de confiança entre os nós e, com isso, agregar funções que auxiliem na criação e na manutenção do grupo. Com os grupos formados e as chaves corretamente distribuídas e validadas, é possível utilizar a assinatura digital para, também, garantir a integridade da informação, porém com um certo custo computacional.

Com foco nessas abordagens, este capítulo apresenta os conceitos e fundamentos sobre aspectos de segurança para VANETs e uma revisão de trabalhos que abordam a utilização de grupos para segurança em VANETs, o gerenciamento e distribuição dos dados, o gerenciamento de confiança e o uso de cifragem em grupo para redes veiculares. Esses trabalhos foram selecionados, dentre os presentes na literatura, pela viabilidade e possibilidade de compor ou servir de ponto de partida para esta pesquisa.

4.1 CRIPTOGRAFIA

A criptografia, ou cifragem, é uma técnica que garante a confidencialidade através do ocultamento de informações, usando uma chave secreta, transformando um texto claro em um cifrado e, posteriormente, decifrando-o com a chave [97]. Para a compreensão dos termos relacionados à cifragem, algumas definições devem ser esclarecidas:

- Texto claro: informação a ser cifrada.
- Algoritmo de cifra: algoritmo que cifrará o texto claro, transformando a informação em texto cifrado.
- Texto cifrado: texto claro cifrado pelo algoritmo de cifra.
- Chave: informação necessária para cifrar e decifrar.

- Algoritmo de decifra: algoritmo que decifrará o texto cifrado, transformando-o em texto claro de saída.
- Texto decifrado: resultado do algoritmo de decifra, ou seja, o texto claro recuperado.

Os sistemas de cifragem podem ser divididos em simétricos e assimétricos. O sistema simétrico usa apenas uma chave compartilhada entre o remetente e o destinatário para cifrar e decifrar. O sistema assimétrico, também conhecido como cifragem de chave pública, usa duas chaves: pública e privada. A chave pública é fornecida livremente aos remetentes para a cifragem do texto claro, enquanto a chave privada é secreta e usada somente pelo destinatário para decifrar o texto cifrado.

4.1.1 Cifragem simétrica

A cifragem simétrica, ou cifragem de chave única, possui uma única chave para transformar o texto claro em texto cifrado e fazer o processo inverso. A cifragem simétrica possui texto claro, algoritmo de cifragem, chave secreta, texto cifrado e o algoritmo de decifragem, como ilustrado na Figura 4.1. Para que esse sistema seja considerado seguro, deve-se atentar aos seguintes requisitos:

- Um oponente que conheça o algoritmo de cifragem e tenha acesso a um ou mais textos cifrados não deve conseguir decifrar o texto ou descobrir a chave.
- O emissor e o receptor devem manter as suas cópias da chave compartilhada em segredo e segurança.

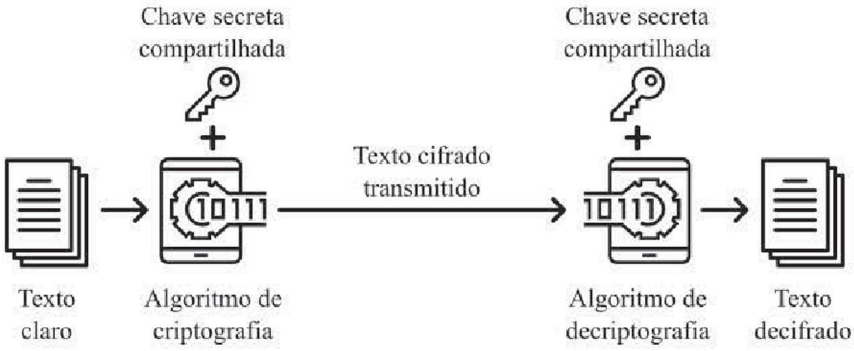


Figura 4.1: Cifragem simétrica.

Na cifragem simétrica, é importante manter em sigilo apenas a chave secreta. O algoritmo de cifragem pode ser conhecido. Neste tipo de cifragem, o principal problema de segurança consiste em manter em sigilo a chave secreta. Porém, mesmo sem a chave, é possível que um terceiro indivíduo realize um ataque utilizando criptoanálise ou ataque por força bruta, na intenção de obter a chave secreta:

- Criptoanálise: este ataque explora características comuns de um texto cifrado para tentar obter o texto claro.
- Ataque por força bruta: executa o algoritmo diversas vezes, alterando a chave secreta até resultar em um texto legível.

Se, usando uma destas duas formas, o atacante obtiver a chave secreta, conseguirá acessar todas as mensagens que foram ou serão cifradas por meio desta chave. Desta forma, no desenvolvimento de um algoritmo de cifragem, devem ser considerados os seguintes critérios: o custo para quebrar a cifra deve ultrapassar o valor da informação cifrada; e o tempo para quebrar o texto cifrado deve superar o tempo de vida útil da informação.

Tabela 4.1: Tempo de busca de uma chave por força bruta [3].

Tamanho da chave (bits)	Número de chaves alternativas	Tempo necessário para 1 decifragem / μs	Tempo necessário para 10^6 decifragem / μs
32	$2^{32} = 4,3 \times 10^9$	35 minutos	2,15 milissegundos
56 (ex.: DES)	$2^{56} = 7,2 \times 10^{16}$	1142 anos	10 horas
128 (ex.: AES)	$2^{128} = 3,4 \times 10^{38}$	5×10^{24} anos	5×10^{18} anos
168 (ex.: 3DES)	$2^{168} = 3,7 \times 10^{50}$	5×10^{36} anos	6×10^{30} anos
26 caracteres	$26! = 4 \times 10^{26}$	6×10^{12} anos	6×10^6 anos

Os algoritmos de cifragem simétricos, como o *Advanced Encryption Standard* (AES) [98], podem ser considerados viáveis devido ao tempo necessário de computação para decifrar uma mensagem apenas com base no texto cifrado e no algoritmo de cifragem. O tamanho da chave é fundamental para evitar ataques por força bruta. Como apresentado na Tabela 4.1, ele impacta diretamente na segurança da cifra [3]. Dessa forma, o algoritmo pode ser de domínio público, o texto cifrado pode ser conhecido e apenas a chave precisa ser secreta.

4.1.2 Cifragem assimétrica

A cifragem assimétrica é o método para cifrar dados com uso de chaves públicas e decifrá-los com chaves privadas. Cada nó comunicante possui o seu próprio par de chave. A chave privada deve ser mantida em segredo e tanto a chave pública quanto o algoritmo de cifragem são de conhecimento público. Desta forma, para um nó a enviar uma mensagem cifrada ao nó b, deve cifrar a mensagem com a chave pública do nó b. Ao receber a mensagem cifrada, o nó b utiliza a sua chave privada para recuperar a mensagem original.

Um esquema de cifragem assimétrica possui: texto claro, algoritmo de cifragem, chave pública, chave privada, texto cifrado e o algoritmo de decifragem. As chaves assimétricas podem ser usadas em sistemas de cifragem, em assinatura digital e em trocas de chaves simétricas.

- Cifragem/decifragem: o emissor cifra a mensagem com a chave pública do destinatário.
- Assinatura digital: o emissor assina uma mensagem com a sua chave privada.
- Troca de chave: dois lados cooperam para trocar uma chave de sessão.

As principais abordagens que utilizam as chaves assimétricas são: Diffie-Hellman[99] para a troca de chaves, *Rivest Shamir Adleman* (RSA) [100] e cifragem de curvas elípticas (*Elliptic Curves Cryptography* (ECC)) [101] para a cifragem e a decifragem. A Tabela 4.2 compara as três principais abordagens que utilizam chaves assimétricas.

Uma vantagem dos algoritmos assimétricos é o uso de uma chave para cifrar e outra para decifrar, tornando computacionalmente inviável determinar a chave de decifragem, mesmo conhecendo o algoritmo de cifragem e a chave de cifragem.

A cifragem assimétrica possui as seguintes etapas:

Tabela 4.2: Aplicações para criptossistemas de chaves públicas. Adaptado de [3].

Algoritmo	Cifragem/Decifragem	Assinatura digital	Troca de chaves
RSA	Sim	Sim	Sim
Curva elíptica	Sim	Sim	Sim
Diffie-Hellman	Não	Não	Sim

- Cada usuário gera um par de chaves a ser usado para cifragem e decifragem das mensagens.
- Cada usuário disponibiliza publicamente a sua chave pública. A outra chave deve ser mantida em sigilo. Além disso, cada usuário deve manter/conhecer as chaves públicas de outros nós.
- Se *a* quer enviar uma mensagem confidencial para *b*, deve cifrá-la usando a chave pública de *b*.
- Quando *b* recebe a mensagem cifrada, a decifra usando a sua chave privada. Ninguém além de *b* pode decifrar a mensagem porque somente *b* conhece a sua chave privada. A Figura 4.3 ilustra estas etapas.

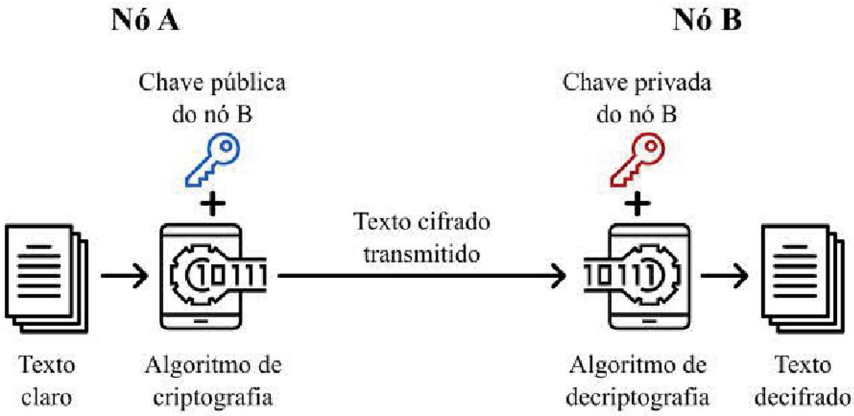


Figura 4.2: Cifragem com chaves assimétricas.

Além de garantir a confidencialidade por meio da cifragem e decifragem do texto, a cifragem de chave pública também permite autenticar a mensagem e prover aspectos como autenticidade e não repúdio [3]. Os passos para a autenticação são similares aos da cifragem e decifragem, porém utiliza-se a chave privada para assinar a mensagem e a chave pública para verificar a autenticidade. A Figura 4.3 ilustra esta autenticação.

A cifragem assimétrica ofereceu uma mudança radical nos sistemas de cifragem, que eram apenas simétricos, pois é baseada em funções matemáticas, ao invés de substituições e permutações, como ocorre na maioria dos sistemas simétricos de cifragem [97]. Uma importante questão no uso da cifragem assimétrica é o modo como as chaves públicas são disponibilizadas no ambiente, garantindo a autenticidade e a irretratabilidade. Existem acordos e gerenciamentos de chaves que vão desde a distribuição simples da chave até sofisticadas unidades de autoridades certificadoras.

Um sistema de chave assimétrico não é mais seguro contra ataques de criptoanálise (um ataque de criptoanálise pode ser encontrado em [102]) do que um sistema simétrico ou

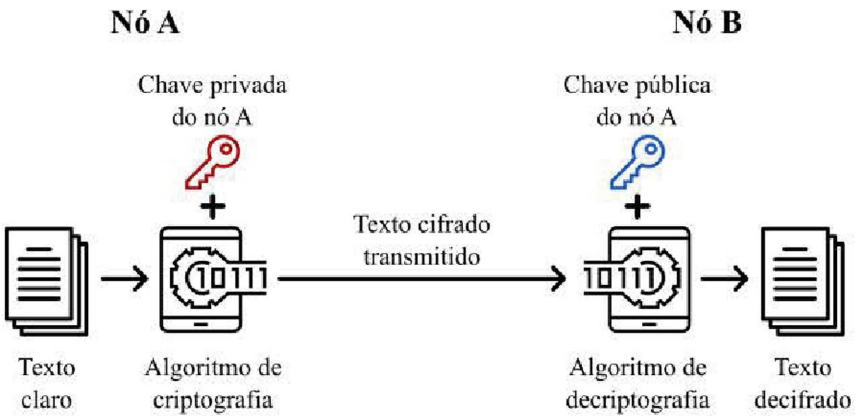


Figura 4.3: Autenticação com chaves assimétricas.

vice-versa, pois a segurança de qualquer sistema de cifragem depende do tamanho da chave e do trabalho computacional envolvido para a quebra da cifra [97]. A grande vantagem da cifragem assimétrica é a não necessidade de uma chave compartilhada, pois a transmissão dessa chave é um problema em ambientes não confiáveis, como a internet. Além disso, com a cifragem assimétrica é possível realizar a assinatura digital e garantir a autenticidade da mensagem [97].

Entre os sistemas de cifragem assimétricos, a ECC (Criptografia de Curvas Elípticas [103], [104]) destaca-se por permitir o uso de chaves menores do que outros sistemas de cifragem com nível equivalente de segurança e, por consequência, necessitar de menor espaço de armazenamento, gasto energético e processamento. Para exemplificar essa proporção: uma chave ECC de 160 bits possui segurança equivalente a 1024 bits do RSA, e 224 bits do ECC equivalem a 2048 bits do RSA [105].

Nenhum sistema de cifragem deve ser considerado inquebrável, imune às técnicas e algoritmos como a força bruta (como o ataque ao *Rivest Cipher 4* (RC4) apresentado em [106]), e sim a quebra deve ser considerada impraticável para a computação moderna. Assim, a segurança dos sistemas de cifragem modernos está no nível de dificuldade em resolver um problema matemático difícil, como a fatoração de inteiros [107], o problema do logaritmo discreto [108] e logaritmo discreto para curvas elípticas [109]. Esses problemas requerem demasiado tempo para encontrar a chave ou para decifrar o texto sem ela, o que torna este ataque impraticável e garante a segurança desses sistemas.

A cifragem de chave pública é vulnerável ao ataque de força bruta e a contramedida é o uso de chaves grandes. Porém, o tamanho desta chave deve ser grande o suficiente para tornar o ataque de força bruta impraticável, mas pequeno para que a cifragem e decifragem sejam viáveis.

4.1.3 Cifragem Híbrida

O uso de cifragem assimétrica permite uma série de vantagens sobre a cifragem simétrica, no entanto o desempenho não é uma delas. O *Elliptic Curve Integrated Encryption Scheme* (ECIES) combina as vantagens de cada sistema, integrando o criptossistema assimétrico ECC e o simétrico AES. Com isso, há a possibilidade de usar o bom desempenho para cifrar e decifrar com um criptossistema simétrico, mas sem necessitar de uma chave secreta compartilhada; ao invés disso, utiliza-se a chave pública e um acordo de chave. Além disso, não há a necessidade de mapear as mensagens para pontos na curva, como em um sistema puramente ECC. Em síntese, as chaves públicas e privadas são usadas para criar uma chave efêmera compartilhada somente para uma determinada comunicação e para combinar o melhor dos dois criptossistemas.

O ECIES é um sistema de cifragem que combina trocas de chaves para gerar uma chave efêmera, ou de sessão, e utilizá-la como chave secreta compartilhada em uma cifra simétrica. O ECIES pode utilizar a *Elliptic Curve Diffie-Hellman* (ECDH) [110], uma variação do acordo de chaves Diffie-Hellman com curvas elípticas, ou o *Elliptic Curve Menezes-Qu-Vanstone* (ECMQV), proposto por Menezes, Qu e Vanstone, para troca de chave sob curvas elípticas. O ECMQV possui a vantagem de ser resistente ao ataque de homem do meio [111]. A integridade da mensagem é garantida com uso de *hash*, implementado com o algoritmo *Secure Hash Algorithm* (SHA), em suas versões 1, 2 ou 3, no criptograma.

A cifra simétrica utilizada é o AES, com tamanhos de 128, 192 ou 256 *bits*. O nível de segurança promovido pelos 128 *bits* é considerado suficientemente seguro para a comunicação e possui um bom desempenho, principalmente quando se trata de ambientes com recursos computacionais restritos.

As funções do ECIES permitem que o usuário A envie uma mensagem cifrada para o usuário B. No entanto, como este sistema é baseado em curvas elípticas, primeiramente a curva elíptica deve ser estabelecida e acordada entre todos; após isso, o ponto gerador deve ser selecionado de modo que cada nó consiga gerar suas chaves, como apresentado em [110]. Após o estabelecimento da fase inicial da curva elíptica, as funções do ECIES podem ser aplicadas na comunicação segura. Em síntese, elas são:

- Acordo de chaves: função usada pelas duas partes para gerar uma chave secreta.
- Gerador de chaves: responsável por gerar as chaves efêmeras para a comunicação em sessão.
- Derivação de chave: mecanismo que produz duas chaves, uma para o criptossistema simétrico e outra para gerar a etiqueta.
- Cifragem: algoritmo de cifra e de decifra simétricas, para garantir a integridade.
- Código de autenticação de mensagem (*Media Access Control* (MAC)): gera um *hash* com o resumo da mensagem, chamado de etiqueta para verificação da integridade.

A Figura 4.4 apresenta os passos das funções ECIES. O primeiro passo é cada nó gerar um par de chaves efêmeras: privada (u) e pública (U). Este passo deve ser realizado em momento pré-comunicação. O segundo passo é a realização do acordo de chaves (KA) entre os nós. Ele utiliza a chave privada efêmera do nó remetente e a chave pública (V) do nó destinatário; a saída é a chave secreta compartilhada ($u*V$). O terceiro passo é executar a função derivadora de chaves. Ela irá gerar uma chave para a cifragem simétrica (k_{ENC}) e outra para a autenticação de mensagem (k_{MAC}). O quarto passo é a cifragem do texto plano em texto cifrado, a partir de um algoritmo de cifragem simétrica (ENC), por meio da chave de criptografia compartilhada. O quinto passo é a geração de resumo da mensagem cifrada com uso da chave de autenticação de mensagem. Ela produzirá uma etiqueta que, com a mensagem cifrada e a chave efêmera pública, formará o criptograma a ser enviado.

Com a entrada do texto plano e a chave K_{ENC}, a mensagem é cifrada com um algoritmo de criptografia (recomendável o AES 128 *bits*, denotado por ENC). Este algoritmo produz uma mensagem cifrada - c, que com a chave k_{MAC} produzirá uma etiqueta para validação da integridade. A chave pública efêmera - U, a etiqueta e a mensagem cifrada formam o criptograma a ser enviado para o destinatário. Ainda é possível inserir parâmetros adicionais nas funções KDF e MAC, para aumentar a segurança ou alguma forma de controle na implementação do

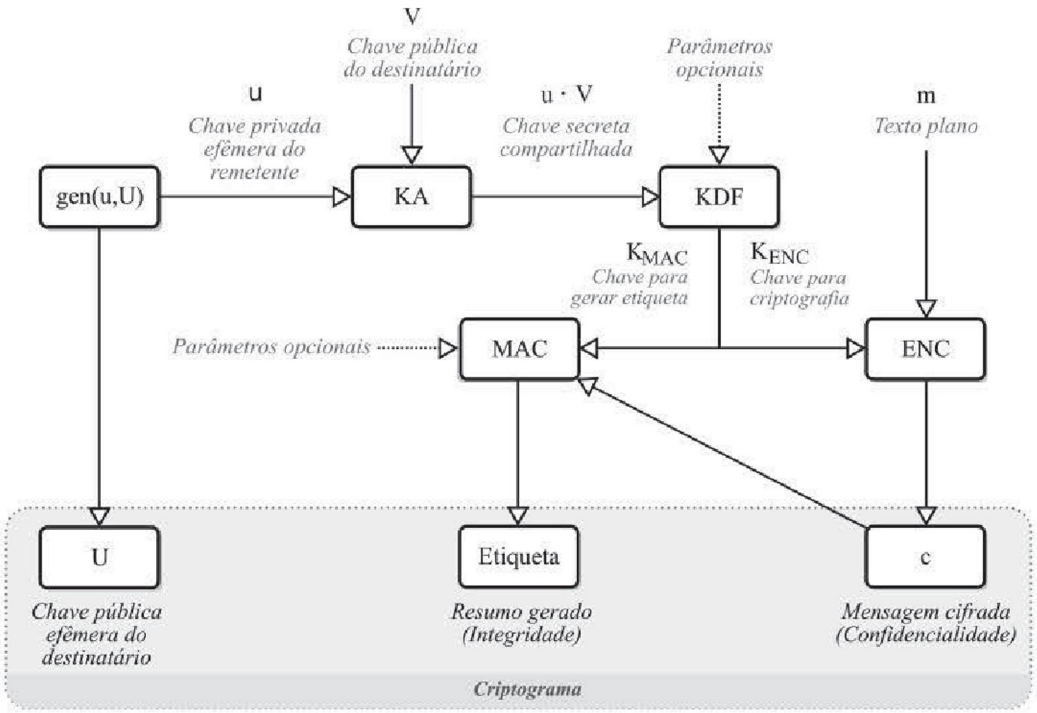


Figura 4.4: Passos do ECIES.

criptossistema ECIES. O processo de decifra é similar, porém nele será usado o algoritmo de decifragem e a etiqueta produzida será comparada com a etiqueta recebida no criptograma.

A Tabela 4.3 mostra os passos para cifrar com ECIES, enquanto a Tabela 4.4 mostra os passos para a decifra. Para estes passos, será usado o exemplo de comunicação de Alice para Bob. Estes mesmos passos estão ilustrados na Figura 4.4.

Tabela 4.3: Etapas da cifragem com ECIES.

Passo	Descrição
1	Alice deve criar um par de chaves efêmeras, consistindo em um elemento de corpo finito e o ponto gerador da curva elíptica. O par de chave deve ser gerado, exclusivamente, de forma pseudoaleatória.
2	Alice usará uma função de acordo de chave para criar um valor secreto compartilhado com o Bob, o qual é resultado da multiplicação escalar entre o elemento do corpo finito e a chave pública do Bob.
3	Então, Alice deve obter a chave secreta e usá-la, opcionalmente com outros parâmetros, como entrada para a função de derivação de chave (KDF). A saída dessa função é a concatenação da chave de cifra simétrica com uma chave MAC.
4	Com a chave compartilhada e a mensagem clara, Alice usará o AES para produzir a mensagem cifrada.
5	Com o texto cifrado e parâmetros opcionais, Alice deve usar a função MAC selecionada para produzir uma etiqueta.
6	Finalmente, Alice enviará um criptograma contendo a sua chave pública efêmera, a etiqueta, e o texto cifrado para o Bob.

Tabela 4.4: Etapas da decifragem com ECIES.

Passo	Descrição
1	Após receber o criptograma da Alice, Bob deve recuperar a chave pública efêmera, a etiqueta, e a mensagem cifrada.
2	Usando a chave pública efêmera recuperada e sua própria chave privada, Bob irá multiplicar ambos os elementos, a fim de produzir o valor secreto compartilhado.
3	Tomando como entrada a chave compartilhada e os parâmetros opcionais usados por Alice, Bob deve produzir a mesma cifra e chave MAC por meio da função KDF.
4	Com a chave MAC, a mensagem cifrada e os mesmos parâmetros opcionais usados por Alice, Bob irá computar a etiqueta e, então, irá comparar o seu valor com a etiqueta recebida. Se o valor diferir, Bob deve rejeitar o criptograma, devido à falha na verificação MAC.
5	Se o valor da etiqueta gerada por Bob for correto, ele irá continuar o processo para decifrar a mensagem cifrada - c, usando o AES e a chave simétrica compartilhada. Com isso, Bob recupera o texto plano que Alice lhe enviou.

4.1.4 Cifragem em grupo

As chaves pública e privada permitem a confidencialidade da informação. Elas são cruciais para promover a segurança proposta neste *Framework*. Em uma comunicação direta, ou seja, nó a nó, o uso dessas chaves é suficiente para garantir a confidencialidade da informação. Comunicações por *broadcast* tendem a ser mais onerosas, pois cada par comunicante deve estabelecer a confidencialidade, utilizando um algoritmo de criptografia assimétrico para chaves públicas e privadas. É possível atenuar custos da comunicação com a cifragem em grupo, pois, nesse modelo, uma única chave compartilhada pode ser usada com algoritmo de criptografia simétrica para todos os membros do grupo.

A cifragem em grupo é possível pela formação de grupos e, consequentemente, com o estabelecimento de uma chave de grupo. Esta é uma técnica que compartilha a chave criptográfica entre os nós do grupo; assim, todos com acesso à chave podem usá-la para cifrar e decifrar todas as mensagens transmitidas entre eles por meio de um canal inseguro [96]. A chave de grupo é classificada em centralizada, descentralizada e distribuída. A centralizada é caracterizada por ter uma única entidade responsável por gerenciar o grupo; a descentralizada possui múltiplos nós responsáveis por gerenciar todo o grupo ou os subgrupos; na distribuída não há um nó central nem uma abordagem descentralizada, nela, os membros dos grupos são responsáveis pelo gerenciamento. Para que a comunicação direta com o uso das chaves pública e privada ou a comunicação em grupo com a cifragem de grupo funcione, é necessário gerenciar as chaves, ou seja, distribuir, validar e revogá-las, além de estabelecer chaves de grupo sempre que necessário. A distribuição de chaves pode ser realizada diretamente pelos membros do grupo de forma segura, através de um adequado acordo de chaves. A validação de chave pode ser realizada de forma distribuída por nós confiáveis no grupo, que possuem informações de identificação.

A atribuição das chaves compartilhadas no grupo é um item fundamental para cifragem em grupo. Esta atribuição pode acontecer de diversas formas, seja com uma unidade centralizadora para geração de chaves, por um líder atribuído ou eleito, por algum membro do grupo, entre outras maneiras. É possível classificá-la como de forma isolada ou distribuída. É de forma isolada

quando qualquer membro do grupo pode definir uma nova chave compartilhada e atualizar os demais membros; e de forma distribuída quando os membros do grupo participam da composição da chave compartilhada. Podem participar todos os membros ou parte deles, mas todos devem conhecer a chave gerada.

A revogação da chave pode ser utilizada na exclusão do nó no grupo. Ao perceber que um nó deve ser excluído, os demais estabelecem uma nova chave de grupo que deve ser conhecida entre os membros atuais; ou seja, os membros excluídos não conhecerão a nova chave. Eles podem identificar que foram excluídos por não receberem as mensagens de resposta ou por não receberem uma pequena mensagem de manutenção, como um *beacon*. Com a atualização da nova chave de grupo é possível identificar os nós excluídos, para que os demais nós, opcionalmente, removam as informações dos nós excluídos. É possível, também, utilizar uma unidade confiável e central para registrar um certificado de cada nó. Enquanto o certificado for válido, o nó será membro do grupo; dessa forma, se a unidade revogar o certificado, ele não será mais parte do grupo.

4.2 MODELOS BASEADOS EM CONFIANÇA

Nas VANETs, as abordagens criptográficas e não criptográficas têm sido usadas na tentativa de mitigar questões referentes à privacidade e à segurança. Porém, alguns objetivos de segurança, como, por exemplo, a confiança e a reputação, ainda são difíceis de atingir por meio de abordagens criptográficas tradicionais. Por isso, alguns autores propuseram calcular o valor de confiança tanto para os nós geradores de conteúdo quanto para o próprio conteúdo, resultando nos seguintes modelos de confiança: *baseado em contatos*, *baseado em redes sociais*, *baseado em comportamento*, *baseado em reputação* e *híbridos* [112, 113, 114].

Os modelos de confiança funcionam em arquiteturas centralizadas ou descentralizadas. Nas arquiteturas centralizadas, existe um nó central responsável por monitorar todas as comunicações, analisar os dados e, através destas informações, avaliar o nível de confiança para cada nó. Nas arquiteturas descentralizadas, cada nó pode avaliar o nível de confiança de qualquer outro nó.

O modelo de confiança baseado em contatos estabelece o grau de confiança através dos dados recebidos de nós vizinhos; ou seja, quando o nó recebe uma informação, deve verificar o grau de confiança desta informação. Para isso, em [115], os autores apresentaram um modelo que avalia a confiabilidade considerando a opinião do nó vizinho que esteja próximo ao evento com maior peso.

No modelo de confiança baseado em Redes Sociais Online (OSNs), a confiança pode ser classificada por meio de três fases: coleta de informações, avaliação de confiança e disseminação de informações. Para saber o quão honesto e confiável é o dono de um determinado perfil, a confiança social é baseada em uma estimativa através das informações do perfil do usuário, como a sua identidade e as interações com outros usuários. Após a observação das informações, esta estimativa de confiança será fornecida aos usuários finais, podendo ser usada de várias formas [72, 116].

No modelo de confiança baseado em comportamento, a confiança pode ser calculada de três modos diferentes: cálculo direto, indireto ou híbrido. No cálculo direto, devem existir parâmetros predefinidos na comunicação entre dois nós. O cálculo indireto é baseado na opinião dos nós vizinhos. No híbrido, é feita a junção entre o cálculo direto e o indireto. Neste modelo, também é possível detectar mau comportamento. Esta detecção pode ser feita pelo próprio nó, com base nas informações coletadas, como, por exemplo, *beacons* e mensagens de alerta, ou pode ser feita mediante utilização de uma *Trusted Authority* (TA) [112].

O modelo de confiança baseado em reputação possui o objetivo de determinar o nível de confiança de cada nó da VANET. O nível de confiança pode ser calculado através de uma avaliação direta ou por uma avaliação indireta [117, 115].

Em [118], os autores classificam as métricas para sistemas de reputação em quatro tipos: métricas baseadas no tempo - medem o tempo ativo de rádio cumulativo; métricas baseadas em tráfego - medem o número cumulativo de pacotes transmitidos por um nó; métricas baseadas em tarefas - medem o número cumulativo de tarefas executadas; e métricas baseadas em autenticação - se baseiam em comunicações de rede seguras. Os autores tentam sanar o ataque de *collusion* usando as seguintes métricas: número de vezes e quantidade de tempo que o rádio está ligado, número de retransmissões de pacotes, número de pacotes recebidos, número de *Acknowledge* (ACKs) e *no Acknowledge* (nACKs) enviados, número de pacotes transmitidos, número de pacotes transmitidos com sucesso, número de pacotes negativos transmitidos, frequência de transmissões para uma determinada tarefa, frequência das transmissões por um vizinho, número de transmissões de pacotes infrequentes, número de tarefas executadas, número de tarefas postadas no sistema, número de falhas de conclusão de tarefas, número de nenhum reconhecimento, número de pacotes duplicados, capacidade de substituir pacotes de dados, tipo de método de autenticação usado e método de autenticação fornecido alterado.

Em [112], os autores propuseram o *Hybrid Trust Model* (HTM), que combina o modelo de confiança baseado em entidade e o baseado em dados para avaliar o comportamento dos nós e estimar seus valores de métrica de confiança. O HTM usa um sistema de monitoramento baseado na cooperação entre grupos de veículos e a validade dos seus dados transportados. Este monitoramento é contínuo e dinâmico, e os valores são alterados a cada monitoramento recebido.

Este modelo híbrido é composto por um sistema *back-end* e pelos grupos de nós. O sistema *back-end* é responsável pelo processamento e verificação de certificados, pela comunicação com os nós e pela detecção e revogação de mau comportamento. Os grupos de nós se comunicam entre si, com o líder do grupo e com esse sistema por meio das RSUs. Através das RSUs, os nós recebem as informações do sistema *back-end*, como, por exemplo, listas de revogação de certificados e atualizações de tráfego, e o líder do grupo pode enviar para o sistema *back-end* as informações recebidas dos nós do seu grupo sobre comportamentos inadequados. Os nós transmitem, periodicamente, aos seus vizinhos, sinais contendo um certificado de curto prazo e a sua assinatura digital. Inicialmente, o primeiro nó que se autenticar no sistema será eleito como líder e o segundo como líder de *backup*. Posteriormente, o líder será escolhido com base nas métricas de confiança. Quando o líder ficar fora de cobertura, o líder de *backup* se torna líder, e os nós que estiverem fora do alcance deste novo líder começam um novo processo de formação de grupo.

O líder do grupo é responsável por definir um *Identity* (ID) de grupo e transmiti-lo para os nós vizinhos. Os nós são obrigados a usar chaves de grupo para se comunicar com um grupo. Para o processo de geração de chaves, o líder gera sua própria chave privada, a chave pública do grupo e a chave simétrica do grupo. A chave privada é utilizada para emitir certificados de associação para os membros do grupo; a chave pública do grupo é usada para identificar os membros; e a chave simétrica é usada para cifrar os dados confidenciais entre os membros do grupo. Os autores não informam o processo de geração de chaves pública e privada de cada nó.

Após a formação do grupo, o líder transmite para todos os nós desse grupo a chave simétrica de grupo cifrada com cada chave pública de nó e a chave pública do grupo cifrada com a chave simétrica de grupo e assinada com a chave privada do líder. Quando um novo nó quer entrar no grupo, o líder fornece-lhe a sua chave secreta de assinatura, a chave pública do grupo e a chave simétrica do grupo. Quando um membro deixa o grupo, o líder atualiza os demais com uma nova chave pública de grupo e uma nova chave simétrica de grupo. Além disso, qualquer nó

com um par de chaves pública e privada pode aplicar o algoritmo de geração de assinatura, para produzir uma assinatura digital em alguma mensagem.

Para cada nó, as métricas de confiança são baseadas em cálculos diretos e indiretos, transmitidos ao líder do grupo, que repassa essas métricas de confiança ao sistema *back-end*. No sistema *back-end*, a *Certificate Authority* (CA) é responsável por calcular uma métrica de confiança global para cada nó participante. O nó é considerado confiável quando essa métrica alcança um determinado valor. Os nós que não alcançarem este valor são submetidos a um conjunto de regras, na tentativa de encontrar nós maliciosos. A *Misbehavior Authority* (MA) (pertencente ao sistema *back-end*) é responsável por desativar os nós considerados maliciosos. Este esquema possui como desvantagens a sobrecarga de armazenamento e computação.

4.3 FUNÇÃO *HASH*

Uma função *hash*, ou resumo da mensagem, é um algoritmo que mapeia dados de comprimento variável para dados de comprimento fixo, sendo utilizado pelo remetente para garantir a integridade da informação; assim, o receptor pode comprovar que não houve alteração no pacote transmitido [3].

Existem vários tipos de funções hash: MD5, SHA-1, SHA-256, SHA-384, SHA-512 e RIPEMD-160. A função *hash* é caracterizada por saída de tamanho fixo independentemente do valor de entrada; dessa forma, as saídas possuem a mesma quantidade de letras e números.

Um exemplo de uso de *hash* é o *Hash-based Message Authentication Code* (HMAC) [119], um tipo de MAC que mescla uma função *hash* e uma chave criptográfica secreta. Qualquer função *hash* pode ser usada no cálculo do HMAC. Esse autenticador de mensagem possui o objetivo de verificar a integridade e a autenticidade dos pacotes transmitidos.

O HMAC não criptografa a mensagem. A mensagem criptografada deve ser enviada juntamente com o HMAC. Os receptores, com a chave secreta, irão calcular novamente o *hash* da mensagem e, se for igual, significa que o pacote está íntegro [120, 121].

4.4 CONSIDERAÇÕES SOBRE O CAPÍTULO

Este capítulo apresentou os mecanismos de segurança aplicados à confidencialidade e à integridade que podem ser aplicados no contexto da VANET. Foram apresentados os conceitos fundamentais sobre criptografia simétrica, assimétrica e híbrida, focando em curvas elípticas e ECIES. A maior vantagem em utilizar uma cifra simétrica é o desempenho, enquanto na utilização da cifra assimétrica é a não necessidade de uma única chave secreta compartilhada conhecida previamente. Para utilizar o melhor dos dois sistemas, este capítulo explorou a cifragem híbrida, em especial o uso do ECIES e de modelos baseados em confiança para promover a formação de grupo em que uma cifragem híbrida possa ser aplicada.

Como o objetivo deste trabalho é fazer uma abordagem de segurança baseada em grupo, este capítulo também apresentou a cifragem em grupo. Embora existam diferentes formas de fazer a cifragem em grupo, exploramos a possibilidade de usar uma cifra híbrida como o ECIES para estabelecer uma chave compartilhada não apenas entre os dois nós comunicantes, mas com todos os nós de um grupo; além de abordarmos brevemente, aspectos como distribuição de chave, atribuição de chave compartilhada e revogação das chaves.

A criptografia em grupo exige a formação de um grupo com nós confiáveis, assim, este capítulo também apresentou os modelos baseados em confiança para formação de grupo. Em síntese, os modelos de confiança podem ser baseados em contato, redes sociais, comportamento, reputação ou ser híbridos. Cada modelo tem suas abordagens particulares e pode ser usado com ou sem o auxílio de infraestrutura. A maioria dos trabalhos pesquisados utiliza modelos de confiança com infraestrutura, porém é possível adaptar estes modelos para uma proposta totalmente distribuída.

5 TRABALHOS RELACIONADOS

Este capítulo apresenta os trabalhos que utilizam soluções de segurança para VANETs baseados em grupo. Os trabalhos foram selecionados por apresentarem abordagens de segurança da informação no contexto das VANETs e por utilizarem grupos como base de suas soluções. O capítulo está organizado em três seções. A primeira foca em soluções de segurança com grupos, a segunda expõe as formas que os trabalhos utilizam para distribuir os dados de modo seguro no grupo, e a terceira apresenta como os trabalhos gerenciam a confiança no grupo.

5.1 GRUPOS PARA SEGURANÇA EM VANETS

O conceito básico de grupos para o contexto das VANETs é a formação de um conjunto de nós que se deslocam próximos uns dos outros por um determinado tempo. Através do grupo de nós, é possível reduzir as mensagens trafegadas na rede, evitar o envio de mensagens redundantes ou pacotes duplicados e aumentar a taxa de transferência da rede. Desta forma, formar grupos de nós fornece alto rendimento e menor sobrecarga da rede [58, 59]. Ao agrupar os nós, é possível coordená-los nas transmissões e retransmissões de mensagens, e controlar o alcance e a direção da propagação da mensagem, tornando rápidos a comunicação dos nós dentro de um mesmo grupo e o envio das mensagens para fora do grupo [122].

Os grupos também podem ser usados para transmitir apenas as mensagens consideradas importantes. Para isso, é eleito um líder responsável por retransmitir as mensagens dos outros grupos para os membros do seu grupo, se considerar esta mensagem importante, e repassar as mensagens trafegadas no seu grupo para fora, caso julgue necessário. Esta abordagem enfrenta muitos desafios, principalmente o de eleger um líder para tomar as decisões [123].

Alguns autores propõem a utilização de grupos para reduzir a transmissão de pacotes na rede. Em [124], foi proposto um esquema de transferência de dados baseado em agrupamento, usando posições e direção de movimento de nós. Cada grupo possui um líder responsável por enviar pacotes para os outros grupos baseados nas informações do nó, que consistem em sua própria posição, na posição do nó de destino e na direção de movimento do grupo; para isso, cada nó deve possuir um *Global Position System* (GPS). Com base nestas informações, se for decidido encaminhar o pacote, é utilizado o encaminhamento epidêmico de um grupo para outro. Apenas o líder do grupo armazena os pacotes de dados no *buffer* de pacotes. Este esquema possui o objetivo de reduzir a sobrecarga da rede e diminuir o tempo de transferência de dados.

Em [51], os autores propuseram o agrupamento baseado em contexto para colaboração eficiente em VANETs, no qual apenas informações relevantes são compartilhadas entre os nós de um grupo. Os grupos são formados com base na localização e na direção de um nó. Para isso, é assumido que todos possuem GPS, sendo divididos em até quatro subgrupos, com base em seus interesses: principal - nó líder; cauda – o último nó; revezamento - vários nós que permitem a cobertura total da transmissão; e normal - não possui nenhuma das funções acima. Somente o nó principal, os da cauda e os de revezamento podem conversar com os nós de dentro e de fora do grupo. Todas as mensagens são enviadas para os nós do grupo que avaliam o conteúdo e adicionam a sua identificação e a relevância da mensagem. Esta avaliação é baseada em parâmetros como posição, velocidade, direção, tempo de vida, interesse, entre outros. Este mecanismo possui o objetivo de reduzir os fluxos de informações irrelevantes ou redundantes e o uso geral do tráfego de rede.

Os grupos de nós nas redes veiculares também são usados para adicionar algum tipo de segurança na troca de mensagens ou para detectar nós maliciosos. Em [52], os autores afirmam que a detecção dos participantes mal-intencionados em uma rede VANET é um problema em aberto, e que estes nós devem ser excluídos do grupo. Além disso, abordam que as operações em grupos também são um problema em aberto nas VANETs e que devem ser usadas para fornecer uma estrutura mais flexível nestas redes. Existem vários autores que propõem trabalhos com aspectos de segurança na literatura ([125, 126, 127, 128, 129]), dentre os quais os trabalhos [53, 54, 55, 56, 19, 57] são relacionados a esta pesquisa.

Em [53], foi apresentado um esquema que usa autenticação baseada na assinatura de grupos para preservação de privacidade em redes VANETs. Em uma rede, primeiro é feita a divisão dos limites em vários domínios e dentro de cada domínio são utilizadas as RSUs para a distribuição de chaves privadas de grupo e para o gerenciamento dos nós. Os autores utilizaram o HMAC [120] para autenticar a origem de uma mensagem e sua integridade, anexando um MAC à mensagem, o que é realizado por uma função *hash* criptografada. A autenticação é feita de forma cooperativa.

Para esse esquema, são usados: TA, RSUs fixas no lado da estrada e nós móveis equipados com OBUs. A TA é o centro de gerenciamento da rede, que fornece o registro e a certificação para as RSUs e OBUs quando eles ingressam na rede, e é responsável por dividir todo o ambiente em vários domínios. As RSUs gerenciam os nós dentro de sua faixa de domínio, sendo responsáveis por emitir as chaves de assinatura de grupo, contendo uma chave pública de grupo, chaves privadas de grupo e uma chave de rastreamento para um domínio.

Em seguida, o nó e a RSU podem realizar autenticação mútua usando os materiais pré-armazenados e os parâmetros do sistema publicados. Após o nó ser verificado pela RSU do seu domínio, passará a receber as mensagens de atualização de chave de grupo regularmente das RSUs. Todos os nós são equipados com OBU, sendo responsáveis por transmitir, periodicamente, informações de *status* relacionadas ao tráfego.

Em [54], foi proposto um esquema de entrega de chave de grupo baseado na comunicação V2V e considera-se a comunicação V2I fora do intervalo de alcance dos nós. Para isso, é necessário que pelo menos um nó esteja no intervalo de alcance de uma RSU e atualize a sua chave de grupo via comunicação V2I. Após, este nó repassa a chave para os demais nós do grupo, via transmissão V2V; neste momento, todos os nós podem estar fora do alcance de qualquer RSU, assim, o esquema permite que os nós de um mesmo grupo que estão fora do alcance da RSU recebam a chave de grupo de outros nós pertencentes ao grupo.

O esquema possui dois protocolos baseados em Diffie-Hellman: verificação de nós – o nó portador da chave de grupo consegue verificar se o nó que tenta recebê-la é um membro de grupo; e verificação de chave de grupo - o receptor da chave de grupo pode verificar a integridade da chave, para isso, é adicionado o *Key Distribution Center* (KDC), um procedimento de geração de chave de grupo que ajuda os nós a verificarem a integridade da chave do grupo por si mesmos. O KDC é responsável por gerar a chave de grupo, registrar os nós no grupo, gerar as chaves privada e pública dos nós e gerar a chave de verificação dos nós.

Este esquema considera que o remetente não consegue ver as informações do destinatário, que são criptografadas. As informações são verificadas sem a necessidade de decriptografar. Os autores consideram que a geração de chaves de grupo é totalmente confiável. Toda vez que a associação do grupo é alterada, a chave do grupo deve ser atualizada e transmitida para os nós. Assim, as RSUs verificam se o nó é membro do grupo se comunicando com o KDC, após a verificação, é enviada ao nó a chave do grupo criptografada com a chave pública do nó.

Em [55], foi proposto um esquema de autenticação de grupo entre domínios para redes veiculares. O esquema possui simplificação da geração de chaves de sessão através do uso

de criptografia de curva elíptica, que realiza a autenticação de grupo de domínio cruzado. O objetivo do esquema é verificar a integridade das mensagens e resistir ao ataque de negação de serviço. O esquema é dividido em três entidades: OBU – cada nó é equipado com uma OBU usada para enviar informações de tráfego, armazenamento e manipulação de operação básica e algoritmo básico de criptografia; RSU; e TA - entidade totalmente confiável que possui poder de computação e espaço de armazenamento. A TA é responsável pela RSU e pelo registro de nós, além de verificar todas as informações da rede, como as informações de congestionamento de transmissão, entre outras.

O esquema é dividido em inicialização do sistema, fase de autenticação de grupo e fase de concordância de chave entre OBU e RSU. Na inicialização do sistema, as instituições confiáveis geram uma chave-mestra. Para obter a identidade secreta da chave correspondente, as RSUs e as OBUs devem usar sua identidade na TA para registro. Cada nó tem um ID de identidade de fabricação. Na fase de autenticação de grupo, para enviar pacotes para fora do grupo, os nós têm que acessar as RSUs e, então, acessam a TA, que possui acesso à internet. Quando vários nós tentam acessar a RSU ao mesmo tempo, eles precisam eleger um líder, que os representará, e os demais nós serão responsáveis pela autenticação mútua com a RSU. Após a confirmação da identidade dos nós de um mesmo grupo, será gerada uma chave de sessão. Isso serve para garantir a segurança na comunicação entre a OBU e RSU. O líder escolhido enviará o seu MAC para a RSU e, após a confirmação da identidade dos nós de um mesmo grupo, será gerada uma chave de sessão, para garantir a segurança na comunicação entre o líder do grupo e a RSU. Na fase de acordo de chave de OBU e RSU, cada RSU deve realizar a comunicação com os nós e a coleta e detecção de informações de tráfego dentro da sua área. Neste processo, é importante o acordo de chaves entre RSU e OBU, para evitar a falsificação das informações do nó.

Em [56], os autores propõem um esquema para comunicação V2V baseado em grupo, que utiliza autenticação única para o grupo e chave simétrica para a comunicação. O grupo de nós é formado por meio da utilização da localização e da velocidade dos nós, sendo selecionado um líder para cada grupo. Para isso, os autores consideram o uso de TA – responsável por emitir certificado, chaves pública e privada para cada nó após o registro; atribuir um Pseudo ID (usado para fornecer anonimato) e chave simétrica do grupo a cada nó após a formação do grupo; pela formação do grupo e pela escolha do líder. No caso de atividade maliciosa, a revogação de certificados também é feita pelo TA. Consideram, também, o uso de RSUs – possuem uma capacidade de cálculo muito elevada, estão diretamente conectadas ao TA e atuam como mediadoras entre TA e nós. E, ainda, consideram o uso de nós – cada nó é equipado com uma OBU com alta capacidade de computação e um *Tamper Proof Device* (TPD), que armazena os materiais criptográficos, sendo considerado um dispositivo seguro.

Para este esquema, são utilizados: ECC para a geração das chaves públicas e privadas; *Elliptic Curve Digital Signature Algorithm* (ECDSA) para a geração e verificação de assinatura de 56 *bytes*; AES para criptografar e decriptografar a mensagem de 16 *bytes*; *Message Digest 5* (MD5) para calcular o *hash* de pseudo IDs; e MAC com o tamanho de 20 *bytes*. A TA é responsável por gerar os parâmetros do sistema e os pré-carregar no TPD de cada nó. Após, a TA gera os pares de chaves pública e privada para o nó, usando ECC, e atribui o certificado ao nó. A TA também é responsável pela formação do grupo e escolha do líder do grupo, com base na velocidade e na localização de cada nó. Após a formação do grupo, um nó envia uma mensagem indicando que ele é o líder, e espera receber confirmações dos demais nós do grupo com os seus certificados. A confirmação é feita através do uso da chave pública da TA.

Posteriormente, o líder encaminha todas as mensagens para a TA, que decriptografa e verifica todos os certificados e as assinaturas usando a chave pública do nó. Em seguida, a

TA autentica todos os nós do grupo e gera a chave simétrica e o pseudo ID para cada nó. Além disso, a TA calcula o *hash* dos pseudo IDs dos nós do grupo separadamente e os armazena em sua tabela de *hash* de grupo. A TA envia as informações para as RSUs ou para o líder do grupo, que devem repassar as informações para os demais nós do grupo. Os nós armazenam os *hashs* dos demais nós em sua tabela de *hash* do grupo, e a chave simétrica do grupo e seu pseudo ID em seu TPD. Após, todos os nós transmitem mensagens de segurança em intervalos de tempo específicos.

Quando um nó recebe uma mensagem, calcula o *hash* do *Public Safety Identification* (PSID) anexado a ela e o verifica na sua tabela de *hash* de grupo. Se ele localizar o *hash* na tabela, aceitará a mensagem; caso contrário, a rejeitará. O nó consegue ler a mensagem decriptografando-a com a chave simétrica do grupo. Para análise de desempenho, os autores consideraram apenas o custo computacional para gerar o *hash*, criptografar e decriptografar as mensagens, assinar e verificar a mensagem, calcular o MAC e verificar o certificado. Além disso, os autores consideram um grupo com 10 nós.

Em [19], foi proposto um *Framework* de segurança para comunicação V2V que utiliza autenticação e comunicação baseadas em grupo. Para isso, é formado um grupo veicular e são distribuídas as chaves para a criptografia e a assinatura por grupo. Os grupos formados são baseados na localização atual e na velocidade do nó na estrada. Neste trabalho, são considerados os usos de RSU, CA, nós equipados com OBU e TPD com recursos de processamento de criptografia, e um EDR (*Event Data Recorder*) para armazenar os eventos, os movimentos, as posições e o tempo dos nós. Este *Framework* foi projetado para estradas com apenas uma direção e o grupo é formado em uma área geográfica de 300 metros em torno dos nós. Cada grupo de nós possui um líder, responsável por gerar uma chave simétrica e as chaves públicas e privadas para a assinatura digital do grupo. O primeiro nó que se autenticar na RSU se torna o líder do grupo. O líder altera essas chaves quando um novo membro ingressa ou deixa o grupo. Para manter contato, os nós transmitem mensagens periodicamente para atualizar sua adesão e posição no grupo.

Em um grupo, a autenticação mútua dos nós é feita utilizando a RSU, responsável por identificar os nós novos que entraram na área, a velocidade, a placa e a posição geográfica. Essas informações são usadas para fazer a autenticação deste novo nó dentro no grupo. Após, o líder do grupo autentica o novo nó via RSU e CA, e gera uma nova chave simétrica e chaves públicas e privadas *offline* para a assinatura digital do grupo. Para essa assinatura, é utilizado o ECDSA. Um nó pode pertencer a mais de um grupo, e terá uma chave de criptografia simétrica e assinatura digital *offline* para cada grupo do qual é membro. O uso da assinatura digital do grupo *offline* serve para evitar o retorno à CA para verificar a chave pública, ou seja, a autenticação no grupo é feita apenas com o líder do grupo. O trabalho não aborda a distribuição de chaves no grupo, e sim adota o uso de soluções existentes.

Em [57], os autores propõem um novo sistema de gerenciamento e de revogação de confiança baseado em grupo, que avalia a confiabilidade dos nós participantes com base no comportamento em seus grupos, para comunicações V2V em VANETs. Com base na avaliação de confiança, os algoritmos de detecção de mau comportamento dos nós, os líderes de grupo e a infraestrutura acionam uma revogação hierárquica para excluir entidades maliciosas em uma área específica. Esse sistema é baseado em *Public Key Infrastructure* (PKI). Este trabalho considera o uso de RSUs, infraestruturas (CA) e grupos veiculares. As RSUs transmitem informações entre grupos veiculares e a infraestrutura, e vice-versa. A infraestrutura fornece uma PKI modular e segura, e possui as seguintes funcionalidades: policiamento no Sistema de Gerenciamento de Credenciais de Segurança (SCMS), processamento de certificado, detecção/revogação de mau comportamento e comunicação com os nós. Cada grupo veicular possui um líder e nós membros.

Tabela 5.1: Comparação dos trabalhos estudados, referente a grupos para segurança em VANETs.

	[53]	[54]	[55]	[56]	[19]	[57]
Comunicação	V2I	V2I	V2I	V2I	V2V, V2I	V2V, V2I
Infraestrutura	✓	✓	✓	✓	✓	✓
Confidencialidade	—	—	—	—	✓	—
Integridade	✓	✓	✓	✓	✓	—
Múltiplos grupos	—	—	—	—	✓	—
Comunicação entre grupos	—	—	—	—	✓	—

A formação do grupo é baseada em localização atual, direção e velocidade dos nós na estrada. O líder é eleito e a associação ao grupo é gerenciada dinamicamente. Após, o líder gera um par de chaves de grupo assimétrico e o transmite para todos os membros do grupo. O par de chaves é criptografado com a chave simétrica de grupo. A CA atribui um ID a cada membro do grupo. A comunicação multi pulo é feita pelos nós que pertencem a mais de um grupo. Um certificado é válido por 5 minutos, sendo descartado após seu uso. É usado o HTM baseado em grupo para avaliar a confiabilidade dos nós de toda a rede, com base em seus comportamentos dentro de seus respectivos grupos. A avaliação do comportamento dos nós é baseada em parâmetros relacionados à comunicação, transmissão/recepção de um nó, GPS ou sensores e cálculo de variáveis. É feita uma combinação entre o cálculo de confiança direta e o *feedback* recebido de nós vizinhos, para fazer a avaliação. Quando a pontuação de confiança exceder um limite estipulado, o nó será considerado confiável; caso contrário, são utilizados algoritmos de detecção de mau comportamento nos nós, no líder e na infraestrutura, com a intenção de filtrar o mau comportamento e executar ações específicas. Quando um novo nó estiver entrando na área geográfica, será autenticado através da RSU mais próxima; posteriormente, obterá seu certificado e o valor de confiança inicial, que será modificado após averiguação do seu comportamento na estrada. A infraestrutura mantém um banco de dados local contendo seis tabelas: referente à confiança global para cada nó, detalhes de comportamento inadequado, certificados revogados, nós honestos, nós intermediários e maliciosos.

De forma resumida, a formação de grupos em redes veiculares possui vantagens e desvantagens. As vantagens são reduzir as mensagens redundantes, fornecer alto rendimento, menor congestionamento da rede, evitar a propagação de informações irrelevantes, reduzir a sobrecarga de processamento, adicionar segurança na troca de informações no grupo, entre outras. As desvantagens são constantes mudanças dos nós pertencentes ao grupo e a formação de grupos que duram pouco tempo.

A Tabela 5.1 apresenta as comparações entre os trabalhos [53, 54, 55, 56, 19, 57] sobre grupos para VANETs. Os parâmetros utilizados para esta comparação são: a comunicação utilizada, necessidade de infraestrutura, o oferecimento de serviços de confidencialidade e integridade, suporte a múltiplos grupos, e a existência ou não de comunicação entre múltiplos grupos. Vale ressaltar que apenas as garantias de integridade e confidencialidade da mensagem foram consideradas. Abordagens que usam mecanismos de segurança para outras finalidades que não sejam garantir a integridade e a confidencialidade da mensagem serão desconsideradas.

Todos os trabalhos utilizam a comunicação V2I no grupo e isto torna a infraestrutura necessária. Apenas os trabalhos [19, 57] usam a comunicação V2V além da comunicação V2I, no entanto, também necessitam de infraestrutura. No trabalho [19], a infraestrutura é utilizada para autenticar os nós móveis. Após isso, os nós podem se comunicar no grupo utilizando,

também, a comunicação V2V. Os autores de [19] publicaram um novo trabalho [57], em que acrescentam o uso de PKI para distribuição de chaves, mas as abordagens de comunicação V2V e a necessidade de infraestrutura permaneceram.

Em [53, 54, 55, 56], as RSUs, TAs ou CAs são responsáveis por todo o funcionamento da rede, como, por exemplo, pelo registro dos nós, emissão de chaves, emissão de certificados, gerenciamento dos nós e gerenciamento da rede. Em [19, 57], o líder do grupo é responsável pela geração de chaves e pela sua distribuição. Em ambos os trabalhos, o nó líder pode ser qualquer nó. Em [19], o líder será o primeiro que se autenticar na RSU e, em [57], o líder é eleito. Além disso, os nós podem estar em vários grupos e a comunicação entre os grupos é possível através destes nós.

Em [54], a geração de chaves é considerada totalmente confiável; em [55], a TA é considerada totalmente confiável; em [56], os autores consideram o uso de um dispositivo à prova de violação nos nós, para armazenar materiais criptográficos; e em [19], todos os nós têm um dispositivo à prova de violação. Somente [19] atende à confidencialidade e à integridade, além de possuir múltiplos grupos e comunicação entre estes múltiplos grupos, porém necessita da infraestrutura para gerar as identidades e os certificados de cada nó. De forma geral, estes trabalhos apresentam soluções voltadas à segurança com uso de grupos; contudo, todas as abordagens são projetadas para infraestrutura, por isso não são adequadas para atingir o objetivo deste trabalho.

5.2 GERENCIAMENTO E DISTRIBUIÇÃO DE DADOS

Em [20], os autores propõem um esquema de *Location-Based Services* (LBS), usando emparelhamento bilinear. O sistema consiste em um *Key Generator Center* (KGC) confiável que gera as chaves privadas para os nós e para os fornecedores LBS, e emite chaves secretas de membros para os nós; RSUs distribuídas semi-confiáveis, conectadas aos provedores LBS por rede cabeada; nós em movimento equipados com OBUs e uma variedade de recursos LBS.

Nesse sistema, as RSUs e LBS são baseadas em ID e o nó precisa apenas de uma chave de membro do grupo. Um nó produz assinaturas de grupo de revogação local verificado com sua chave de membro do grupo. Os fornecedores LBS verificam essas assinaturas, sem violar a privacidade dos nós. Um nó pode obter um serviço de maneira anônima, tendo a sua privacidade protegida; porém, se o nó usar o serviço com intuito malicioso, o centro de geração de chaves pode rastrear a identidade do nó e revogá-lo.

Em [130], os autores propõem um esquema que faz a junção de assinatura de grupo e assinatura baseada em identidade, utilizando o algoritmo de assinatura digital de curva elíptica (ECDSA) para autenticar os nós maliciosos, com base em posição, velocidade, registro de data e hora, entre outros. Para isso, são necessários uma CA confiável, responsável por gerar e armazenar segredos e credenciais de assinatura do nó móvel, da autoridade local (LA) e RSUs; uma autoridade local (nó móvel escolhido com base no tempo e experiência na rede), que mantém as credenciais exclusivas de todos os nós móveis mais próximos; e os nós móveis.

A CA armazena todas as informações da rede e é responsável pelos certificados de todos os nós móveis. Este processo de certificação é feito quando o nó entra na rede. A LA inicia a autenticação do nó quando um nó tenta acessar a VANET. Após a autenticação bem-sucedida, o nó pode obter um certificado da CA. Posteriormente, a mensagem deste nó é autenticada e verificada através da LA, sem o envolvimento da CA.

Em [131], os autores propõem um esquema de gerenciamento de chaves para autenticação baseada em assinatura de grupo. Para isso, é necessária uma TA - totalmente segura, que registra e gera os certificados para os nós. Também, são necessárias RSUs - um grupo de RSUs divididas

em RSUs membros (M-RSU) e RSUs líderes (L-RSU); as L-RSUs são totalmente confiáveis e com a TA geram as chaves privadas e públicas do grupo para os nós, também são responsáveis por gerenciar e manter o banco de dados das chaves de grupo. Quando as L-RSUs detectam um nó mal intencionado, entram em contato com a TA para identificar o nó malicioso. As M-RSUs ajudam os nós a obter as chaves de grupo da L-RSU, sendo consideradas semi confiáveis. Além disso, também são necessários ao esquema proposto pelos autores, os nós veiculares - são equipados com OBUs, GPS e uma interface para interagir com os motoristas. Os nós podem se comunicar entre si e com as RSUs. Durante qualquer comunicação, eles são obrigados a usar as chaves de grupo e seus próprios pares de chaves pública e privada, para autenticação e criptografia / decriptografia. Os nós obtêm as chaves públicas quando são registrados pela TA e as atualizam sempre. Estas chaves são armazenadas em um dispositivo à prova de violação.

Para entregar as chaves com segurança aos nós, é utilizada uma chave simétrica compartilhada entre um nó e uma RSU, por meio do protocolo de trocas de chaves Diffie-Hellman. A carga de cálculos é distribuída a partir da RSU líder e da TA. As RSUs são agrupadas para formar um domínio. Cada domínio possui RSUs líderes e RSUs membros. A primeira chave simétrica compartilhada é estabelecida entre o membro RSU e o nó. Posteriormente, as chaves de grupo (par de chaves pública / privada) da RSU líder em nome da TA são fornecidas com segurança ao nó. O nó pode usar essas chaves de grupo para se comunicar com as RSUs no domínio em questão.

Em [21], os autores propõem um sistema de gerenciamento de chaves responsável por gerar, distribuir e atualizar as chaves na VANET, baseado na PKI, no qual cada nó possui uma chave pública e uma privada, e permite que a rede autentique um novo nó antes de trocar as chaves. No sistema proposto, a rede é dividida em grupos dinâmicos gerenciados por um *Key Management Service* (KMS), que distribui a chave do grupo em seu grupo e a atualiza quando um nó entra ou sai do grupo. Existem dois tipos de chaves distribuídas em cada grupo: uma chave secreta compartilhada entre um nó e o KMS; e uma chave de grupo compartilhada entre o KMS e todos os nós do grupo. Este trabalho necessita de uma CA, um KMS, RSU e nós equipados com uma OBU.

A CA é uma autoridade confiável fora da VANET e é responsável apenas por emitir o certificado de chave pública para todos os KMSs e OBUs dessa rede. O KMS é um servidor regional da VANET que é conectado aos nós através das RSUs. Cada KMS mantém um grupo de nós em movimento, que entram ou saem da sua região. O KMS é responsável por gerar, distribuir, revogar e atualizar as chaves em sua região. As RSUs são entidades fixas, à beira da estrada, responsáveis pela comunicação entre as OBUs e o KMS. A conexão da RSU com o KMS é feita por conexão cabeada de alta velocidade e segura; e a conexão da RSU com as OBUs é feita por conexão sem fio.

Quando um novo nó quer ingressar na rede, deve, primeiramente, ter o seu certificado de chave pública emitido pela CA; em seguida, quando a OBU do nó entrar no intervalo de uma RSU, receberá uma mensagem de boas-vindas do KMS da região. O nó deve enviar uma solicitação de ingresso na rede ao KMS; para isso, envia-lhe seu ID e certificado de chave pública. O KMS verifica a validade do certificado, a assinatura do emissor e a assinatura da OBU, se tudo estiver certo, seleciona uma chave secreta, criptografa usando a chave pública da OBU e envia a OBU.

A chave de grupo é mantida pelo KMS da região, sendo usada para criptografar as mensagens transmitidas no grupo. Os autores se baseiam no *Chinese Remainder Theorem* (CRT) ([132]) para computar e distribuir a chave no grupo. Para isso, o KMS seleciona um número primo grande, a fim de definir o grupo multiplicativo e calcular a chave de grupo. Após, são armazenados os valores correspondentes a cada nó e, por fim, através da junção destes valores,

serão computadas várias chaves secretas. Para fazer a distribuição da chave secreta do grupo, o KMS seleciona, aleatoriamente, uma chave de grupo, multiplica por um valor aleatório e envia este número para todos que estão em sua região. Através deste valor enviado, as OBUs conseguem calcular a chave de grupo, usando a sua chave secreta.

A atualização da chave do grupo é feita pelo KMS, quando um nó entra ou sai de sua região; para isso, ao invés de fazer todos os cálculos novamente, o KMS usa os valores já pré-calculados. Ele busca a chave e decrementa os valores do nó que saiu da região ou adiciona os novos valores do nó que entrou na região. Com base nestes valores, seleciona um novo grupo de chaves e calcula um valor. Este é enviado para os demais nós da região, e, através dele, os nós conseguem calcular a nova chave de grupo.

Em [63], os autores propõem um esquema de autenticação em lote e um mecanismo de acordo de chave secreta de grupo entre os nós móveis, RSUs e TA, para VANETs. Quando um nó móvel entra na rede, se registra em uma TA. A TA autentica o nó e o atribui à identificação de um grupo. A RSU calcula a chave de grupo para os nós em sua cobertura, conforme a identificação do grupo. Quando os nós estão na mesma área de cobertura de uma RSU e com a mesma identificação de grupo, é iniciada a negociação de chave de grupo, para a comunicação entre os nós do grupo. Devido às características da RSU (localização fixa, ampla cobertura e poder de processamento), ela é selecionada como gerente do grupo para concluir a autenticação em lote de assinatura dos nós, além de calcular e distribuir a chave do grupo.

O esquema possui sete módulos: inicialização de parâmetro - a TA gera alguns parâmetros iniciais do sistema apenas uma vez para todo o sistema (ela é responsável por atualizar, periodicamente, a chave mestre do sistema); registro de nó e RSU - os nós e as RSUs são registrados na TA (ela é responsável por atribuir as informações de registro correspondente a eles); assinatura de nó - a RSU autentica os nós para se preparem para o contrato de chave de grupo; verificação de RSU - permite que as RSUs verifiquem as assinaturas dos nós, esta verificação pode ser feita como verificação única (apenas um nó) ou verificação em lote (vários nós); geração de chave de grupo - depois que os nós são autenticados, a RSU gera a chave de grupo para eles; junção de membro de grupo - quando um novo nó quer entrar no grupo, o grupo já possui uma chave que precisa ser alterada, então, o novo nó seleciona um número aleatório e gera uma pseudoidentidade. Através destas informações, calcula a sua assinatura e envia para a RSU, que a recebe, decriptografa e verifica se o tempo dela é atual; caso seja, verifica se ela é válida e, se for, permite que ele entre no grupo. Além disso, a RSU seleciona um novo número aleatório e recalcula a chave de grupo, incluindo o novo nó, em seguida calcula a sua assinatura e retransmite para todos os nós do grupo. Quando os nós recebem a assinatura da RSU, a verificam e, se for válida, calculam a nova chave de grupo. Por fim, o sétimo módulo: saída de membro do grupo - toda vez que um nó sai do grupo, a chave de grupo deve ser atualizada; para isso, a RSU seleciona um número aleatório e calcula a chave de grupo. Posteriormente, a RSU calcula a nova assinatura e a transmite para os nós que estão no grupo. Quando os nós recebem esta assinatura, verificam se é válida, em caso afirmativo, calculam a nova chave de grupo.

Em [133], os autores propõem um mecanismo usando ECC e o protocolo de troca de chaves Diffie-Hellman com o mecanismo de mapa bilinear para VANET. Para isso, é considerado o uso de TA, RSU e dispositivos veiculares. A TA é responsável por manter o gerenciamento de chaves e estabelecer a comunicação hierárquica. A RSU é uma unidade fixa na estrada, que está presente na região do nó para executar o encaminhamento de dados.

A estrutura da rede hierárquica é criada com a TA (classificada como uma estação central), as RSUs e os dispositivos de nós. A TA é responsável pela inicialização do sistema. Ela cria um mapa bilinear individual para cada usuário e para a comunicação em grupo. Todos os nós móveis e as RSUs entram em contato direto com a TA para se registrarem. Em seguida, ela

Tabela 5.2: Comparação dos trabalhos estudados, referente a gerenciamento e distribuição de dados.

	[20]	[130]	[131]	[21]	[63]	[133]	[134]
Comunicação	V2I	V2I	V2I	V2I	V2I	V2V, V2I	V2I
Infraestrutura	✓	✓	✓	✓	✓	✓	✓
Confidencialidade	✓	—	—	✓	—	—	—
Integridade	—	—	—	—	—	—	—
Geração das chaves	KGC	CA	RSU	KMS	TA	TA	CA
Distribuição das chaves	KGC	CA	RSU	KMS	TA	TA	CA

fornece uma chave secreta para a comunicação V2V e V2I, utilizando ECC e *hash*. As RSUs e os nós recebem e armazenam essa chave como parâmetro do sistema. Os dispositivos veiculares selecionam a RSU mais próxima e geram uma chave compartilhada usando essa chave secreta. A RSU verifica o parâmetro do sistema dos nós, fornecido pela TA, para validar a confiabilidade deles. Esse processo é repetido para todos os nós antes de se estabelecer a comunicação. Este mesmo processo de verificação de parâmetro do sistema é feito pelos dispositivos do nó para a transmissão V2V. A comunicação V2I é estabelecida quando o nó encaminha dados para a RSU e a RSU encaminha estes dados para a TA. A comunicação V2V é gerenciada através do processo de descoberta de rota. Durante a descoberta de rota e o processo de transmissão de dados, o processo de comunicação é anônimo, ocultando-se a identidade dos dispositivos veiculares.

Em [134], os autores propõem um esquema de autenticação de assinatura de grupo para VANETs, que usa certificado e *primer* de sincronização para autenticar os membros do grupo. Para isso, são utilizados as RSUs, a TA, a CA, o *Service Provider* (SP) e a OBU. TA - responsável pela emissão de certificados de usuário do grupo para as OBUs; é utilizado emparelhamento bilinear para fazer as chaves do grupo. CA - responsável pela emissão das chaves pública e privada para as OBUs. SP - responsável por fornecer vários serviços às OBUs na rede e por autenticar a identidade delas. RSU – unidade fixa que se comunica com a TA por rede cabeada e com os nós através de rede sem fio. OBUs - cada nó móvel possui uma OBU que transmite, periodicamente, mensagens referentes à velocidade, direção da viagem, entre outras, para as demais OBUs e RSUs, e pode usar os serviços fornecidos pelo SP para trocar dados entre RSU e SP.

Quando um novo nó entra no grupo, a TA gera um novo certificado para si mesma (chave privada de grupo). Quando a TA for empregar um novo serviço no grupo, enviará para o SP a sua chave pública, um número de sincronização principal e a sua assinatura. O SP verificará a assinatura, recuperará o número de sincronização principal, utilizando a chave pública da TA, e, se for verdade, o SP adicionará as informações da TA na sua própria lista. A TA adicionará o SP na sua lista também. Toda vez que um nó móvel entrar no grupo, já deve possuir as suas chaves pública e privada. Este nó envia para a TA as suas informações, sendo elas um valor aleatório, chave pública, assinatura gerada a partir do valor aleatório e da sua chave privada e um número aleatório. A TA verifica a assinatura do nó móvel e, se for verdadeira, envia as suas informações para ele, atualiza o seu certificado e adiciona esse nó na lista de nós do grupo. Em seguida, ela atualiza os outros membros do grupo e o SP, enviando o novo número de sincronização. Para revogar um nó que não pertence mais ao grupo, a TA computa um novo número de sincronização para todos os SP e para todos os membros do grupo.

A Tabela 5.2 apresenta as comparações entre os trabalhos [20, 130, 131, 21, 63, 133, 134], sobre gerenciamento e distribuição de dados para VANETs. Os parâmetros usados para esta

comparação são: a comunicação utilizada; necessidade de infraestrutura; ofertas de serviços de confidencialidade e integridade; entidade responsável pela geração de chaves e pela distribuição das chaves.

Todos os trabalhos apresentados na Tabela 5.2 utilizam RSUs para realizar o cálculo da chave [131, 63] ou na comunicação entre os nós e a TA [20, 130, 21, 133, 134]; com isso, fica também ilustrada a necessidade de infraestrutura para funcionarem. Desses trabalhos, apenas os [20, 21] abordam confidencialidade com uso de criptografia baseada em grupos. Em [20], os autores oferecem o serviço de confidencialidade com o uso de criptografia de chave simétrica, em que o KGC é responsável por todo o procedimento. Em [21], os autores oferecem o serviço de confidencialidade com o uso de criptografia de chave pública, no qual o KMS é responsável por todo o procedimento.

Em [131, 134], a TA é responsável pela emissão dos certificados e, em [130, 21], a CA é responsável por esse tipo de emissão. Em [20, 21, 134], os autores usam conexão cabeada para conectar RSUs e LBS/KMS/TA. Em [130], a LA é um nó móvel escolhido com base no tempo e experiência na rede. Desta forma, estes trabalhos necessitam de uma unidade central para oferecer o serviço de segurança.

5.3 GERENCIAMENTO DE CONFIANÇA

Em [22], os autores propõem uma técnica de detecção de padrões de mobilidade e mau comportamento em que as informações de localização e tempo sobre os nós podem ser usadas para detectar comportamentos maliciosos. Para isso, as informações sobre a localização dos nós são armazenadas em uma tabela e, a cada intervalo de tempo, os nós obtêm a localização atual, via GPS, dos demais nós de um mesmo grupo. Os nós enviam essas tabelas para os demais nós do grupo. Após seu recebimento, é comparada a hora e a localização do nó de transmissão com a hora e localização do nó através do GPS. Se estas informações diferirem, o nó transmissor é identificado como mau comportamento.

Em [23], os autores propõem um modelo de confiança com comunicação, baseado em líder de grupo, para redes VANETs, em que se classifica os nós com base em sua confiabilidade e eles são eleitos como potenciais líderes de grupo. Na formação de um grupo, o líder deve ser um nó confiável. Ele é o servidor central de todos os nós que entram no grupo e é responsável por gerenciar a formação do grupo, gerar e distribuir as chaves de grupo. Para saber a confiabilidade de um nó, o modelo de confiança calcula valores através das métricas de confiança. Essas métricas são baseadas na validade dos dados transmitidos e na cooperação dos nós, líderes, RSUs e MA. Através desses valores calculados, o nó julga a confiabilidade e, em caso de comportamento malicioso, reporta à MA, para que o nó específico seja desativado.

Em [64], os autores propõem um esquema de gerenciamento de confiança com reconhecimento de contexto, que permite que os nós avaliem a confiabilidade dos eventos recebidos, considerando a reputação da entidade dos remetentes. O esquema é dividido em duas partes: *A Secure Linkability Scheme* (ALRS) e *Aware Trust Management Scheme* (ATMS). O ALRS permite que os nós reconheçam identidades ou níveis de confiança de outros nós. Essas informações são mantidas em sigilo contra os nós internos ou externos não autorizados, por meio da técnica de criptografia. O ATMS permite que os nós avaliem a confiabilidade dos eventos recebidos, considerando a reputação da entidade dos remetentes. Para calcular o valor de confiança, são utilizadas as informações de vinculação do ALRS. Um nó consegue avaliar os dados do evento baseado em sua distância: se estiverem próximos, o evento poderá ser verificado observando-se se a resposta do movimento do originador de dados é condizente com ele; se estiverem longe, são coletados diferentes relatórios de eventos e agregados às suas relações de confiança com os

originadores dos eventos; e, após, é projetada uma árvore de decisão que estima a confiança da entidade de forma adaptável às informações de vinculação disponíveis.

Em [65], os autores propõem um modelo de confiança para VANETs que mescla a cooperação centralizada e distribuída entre os nós e a infraestrutura, para obter o nó mais confiável como líder do grupo. Este líder é a referência para todas as comunicações entre os membros de seu grupo. Para escolher o líder, é utilizado um sistema de monitoramento de processamento baseado na cooperação dos nós como comunicação, na transmissão/recepção de um nó, nas informações fornecidas pelo GPS ou sensores e na validade dos dados transmitidos. Cada nó calcula a métrica de confiança de seus vizinhos de um salto baseado no comportamento deles. Após calcular, os nós enviam os resultados para o líder do grupo, que envia todas as métricas de confiança de todos os nós para a RSU, que mesclará e atualizará estas métricas, obtendo como resultado uma métrica de confiança global para cada nó. A validade das métricas de confiança é de 200 ms.

O nó é considerado confiável quando o limite da métrica de confiança é excedido. O que possuir maior valor de métrica de confiança será um potencial líder de grupo. Os nós são classificados em três tipos: honestos - nós com bom comportamento; intermediários - nós com comportamento duvidoso (nós desse tipo serão monitorados por um determinado período - entre 300 ms e 5 minutos) - se o mau comportamento continuar após o tempo expirar, será considerado malicioso); e maliciosos - nós com mau comportamento. Os mesmos autores publicaram um novo trabalho [135], adicionando uma MA que verifica se um nó é ou não malicioso.

Em [24], foi proposto um esquema de gerenciamento de confiança que utiliza o HMAC, a assinatura digital e um valor de confiança do nó. Os autores consideram o uso de RSU, OBU, agente da autoridade confiável (responsável por reunir as credenciais dos nós registrados na rede) e autoridade de confiança (responsável pelo registro de todos na rede e por emitir as chaves públicas e privadas). A RSU é considerada confiável e é responsável por realizar as avaliações de confiança para cada nó baseadas no valor de confiança dos nós vizinhos e pontos de recompensa para mensagens de alerta de segurança. O esquema possui duas fases: registro *offline* e avaliação e atualização do valor de confiança.

Na primeira fase (registro *offline*), os nós, as RSUs e os agentes da autoridade confiável se registram na autoridade de confiança e são gerados o par de chaves para todos os envolvidos na rede. Posteriormente, os agentes da autoridade confiável carregam as credenciais do sistema dentro da sua área de cobertura. Na segunda fase (avaliação e atualização do valor de confiança), as RSUs recebem as mensagens de segurança de diferentes nós e calculam o valor de confiança do nó baseadas nos pontos de valor de confiança e recompensa dos nós vizinhos.

Em [136], foi proposto um esquema de gerenciamento de confiança híbrido, que classifica as mensagens trocadas entre os nós de um grupo, identifica e exclui os nós maliciosos. Todos os nós atribuem um valor de confiança para os seus vizinhos de um salto considerando o comportamento deles. Se as informações compartilhadas pelo nó vizinho forem legítimas, o nó será considerado confiável e terá o valor de confiança 1; caso contrário, se as informações repassadas não forem legítimas, será atribuído o valor de confiança 0. O nó com maior valor de confiança é escolhido para ser o líder e é responsável por todas as solicitações de dados dos demais nós do grupo.

A Tabela 5.3 apresenta as comparações entre os trabalhos [22, 23, 64, 135, 65, 24, 136], sobre gerenciamento da confiança. Os parâmetros utilizados para esta comparação são: comunicação; necessidade de infraestrutura; oferecimento de serviços de confidencialidade e integridade; serviço de confiança baseado em tabela; a avaliação dos níveis de confiança de cada nó; baseado em localização; e entidade responsável pela geração das chaves.

Somente em [136], é utilizada apenas a comunicação V2V, sem a necessidade de qualquer infraestrutura; porém os autores abordam apenas o gerenciamento de confiança entre os nós. A avaliação da confiança é feita com base nas informações compartilhadas por um nó. Se estas informações forem legítimas, o nó se torna confiável. Os autores não abordam geração e distribuição de chaves no grupo. Em [23, 64, 135], os autores abordam a confidencialidade para proteger os dados dos nós, como, por exemplo, a tabela de confiança, mas não abordam a confidencialidade para a proteção dos pacotes trafegados.

Em [22, 23, 135, 24], os autores abordam a integridade com o uso de autenticação ou assinatura digital. Em [22], usam autenticação de chave de grupo para prover a integridade dos pacotes trafegados. Para isso, é necessário o uso de RSU para auxiliar neste processo. Em [23], para prover o serviço de integridade, os autores usam autenticação mútua com a RSU. Em [135], usam assinatura digital para prover a integridade, mas dependem da CA para verificar esta assinatura. Em [24], o remetente adiciona o HMAC nas mensagens, para garantir a integridade no lado do receptor. Este processo depende apenas dos nós, porém, para a rede funcionar, é necessário o uso de TA, para o registro dos nós na rede.

Em [64], é possível avaliar o valor de confiança baseado na distância. Se o nó estiver perto, é verificado se as informações são consistentes, e, se estiver longe, é feita a verificação através de vários relatórios. Em [24], os nós avaliam a confiança dos nós vizinhos e enviam para a RSU, responsável por realizar as avaliações de confiança. Além disso, os autores incrementaram pontos de recompensa para mensagens de alerta de segurança.

Em [23, 65, 135, 24], os autores usam a infraestrutura para fazer a avaliação de confiança de cada nó. Em [22, 64, 136], os próprios nós fazem a avaliação da confiança. Em [22], os nós mantêm um mapa com informações sobre a estrada, e, através do comportamento de um nó nessa estrada, ele pode ser detectado como de mau comportamento. Em [64], o nó calcula a confiança do vizinho baseado nas interações passadas. Apenas ele tem acesso ao valor deste cálculo. Se detectar que o nó é mau, ele o bloqueia e encerra qualquer tipo de interação com ele. Em [136], a avaliação da confiança é feita com base nas informações compartilhadas por um nó.

Em [22, 135, 65], os autores usam o GPS para salvar as informações de localização dos nós móveis. Em [22], estas informações de localização servem para detectar nós maliciosos, através da comparação da hora e localização do nó transmissor com o recebido. Se estes dados diferirem, o nó é identificado com mau comportamento, e, desta forma, o GPS é utilizado para promover a confiança baseada em localização.

Em [23, 65, 135, 136], os autores elegem um nó como líder do grupo para realizar a formação do grupo, gerar e distribuir as chaves [23] e atuar como referência para todas as comunicações e solicitações de dados dos membros do grupo [65, 135, 136]. Em [23, 65, 135, 136], a escolha do líder é feita com base na sua confiabilidade e na cooperação dos nós. Em [65, 135], é feita a classificação dos nós para escolher o nó mais honesto como líder.

5.4 CONSIDERAÇÕES SOBRE O CAPÍTULO

Este capítulo apresentou a formação de grupos em redes veiculares e os trabalhos relacionados ao tema que mais se aproximam desta pesquisa. Estes trabalhos, em síntese, abordam a formação de grupos, o gerenciamento de chaves e o gerenciamento de confiança. O grupo é formado, dinamicamente, por dois ou mais nós próximos e confiáveis.

Vários trabalhos utilizam a formação de grupos para evitar mensagens redundantes, fornecer alto rendimento, diminuir o congestionamento da rede, evitar a propagação de informações irrelevantes, reduzir a sobrecarga de processamento e para garantir algum tipo de segurança. Um problema encontrado na formação de grupos é a constante mudança dos nós pertencentes a um

Tabela 5.3: Comparação dos trabalhos estudados, referente a gerenciamento de confiança.

	[22]	[23]	[64]	[65]	[135]	[24]	[136]
Comunicação	V2I	V2I	V2I	V2I	V2I	V2I	V2V
Infraestrutura	✓	✓	✓	✓	✓	✓	—
Confidencialidade	—	—	—	—	—	—	—
Integridade	✓	✓	—	—	✓	✓	—
Tabela de confiança	✓	—	✓	✓	✓	✓	—
Avaliação	Nós	MA	Nós	RSU	RSU	RSU	Nós
Localização	✓	—	✓	✓	✓	—	—
Geração das chaves	TA	Líder	PKG	CA	Líder	TA	—

grupo, além da formação de grupos que duram pouco tempo. A maioria dos trabalhos elege um líder no grupo para ser responsável pela comunicação com as RSUs e pela geração das chaves de cada nó. Porém, esta abordagem enfrenta muitos desafios, como, por exemplo, eleger um líder para tomar as decisões.

O gerenciamento e a distribuição de chaves são importantes para manter a confidencialidade da informação. Para esta pesquisa, os trabalhos apresentados na seção 5.2 serviram como base para esta proposta. O gerenciamento de confiança é importante para os nós conhecerem os seus vizinhos e saberem se podem enviar uma mensagem ou não para eles.

Com o gerenciamento de confiança, é possível remover os nós maliciosos no processo de formação de grupo. Mesmo que um nó uma vez autenticado volte a transmitir e, por algum motivo, torne-se não confiável, deve conquistar a confiança novamente. Com este gerenciamento de confiança, o sistema pode estar apto a prevenir comunicação com nós não confiáveis. O gerenciamento de chave permite, de certa forma, que os usuários na VANET conheçam e validem as chaves públicas, e, dessa forma, tenham o direito de usar os serviços e aplicações da rede.

Os trabalhos relacionados, apresentados neste capítulo, abordam a utilização de RSUs, TA ou CA para auxiliar no processo de autenticação, certificação e diagnosticar os nós mal-intencionados; ou seja, as RSUs, TA ou CA são responsáveis por quase todo o gerenciamento da rede. Se por algum motivo, perder a conexão com estas centrais, a rede irá perder o seu funcionamento. Além disso, se estas centrais (RSUs, TA ou CA) se tornarem mal-intencionadas, a rede não será mais confiável, prejudicando o desempenho da VANET.

As Tabelas 5.1, 5.2 e 5.3 apresentam a comparação dos trabalhos estudados, referente à confidencialidade e à integridade, e mostram que não há um trabalho que ofereça serviços de segurança para a confidencialidade e a integridade que consiga atuar unicamente em comunicação V2V para redes veiculares. Assim, esta pesquisa irá se basear nos trabalhos apresentados nas seções 5.3 e 5.2. O capítulo 6 apresenta a proposta de pesquisa.

6 *FRAMEWORK* BASEADO EM GRUPO PARA CONFIDENCIALIDADE E INTEGRIDADE DA INFORMAÇÃO EM REDES VEICULARES

Este capítulo apresenta o *GSeF4V - Group-based Security Framework for V2V-VANETs*, um *Framework* de segurança baseado em grupo para redes veiculares sem infraestrutura. A segurança do GSeF4V foca na confidencialidade e na integridade da comunicação, aproveitando os frequentes contatos de nós na rede para formar grupos dinâmicos e estabelecer uma relação entre confiança e tempo de permanência, a fim de atribuir diferentes níveis de atuação dentro de um mesmo grupo. O objetivo do GSeF4V é garantir a confidencialidade e a integridade da comunicação V2V de forma independente de protocolos, considerando a formação de grupos multiníveis confiáveis.

O GSeF4V é formado por quatro módulos: gerenciamento de confiança, gerenciamento do grupo, gerenciamento das chaves e comunicação. Cada módulo é composto por vários serviços que atendem ao seu propósito. Embora estes módulos sejam categorizados de forma independente entre si, todos eles estão entrelaçados de tal maneira que um depende do outro para o seu correto funcionamento.

Neste trabalho, inserimos o conceito de multinível em um grupo de nós na VANET V2V. Em um mesmo grupo de nós haverá mais de um nível, variando as métricas e os serviços para cada nível. O principal objetivo é possibilitar hierarquias de diferentes categorias de atuação entre os nós do grupo, sem que haja a necessidade de dividir ou formar novos grupos, dada a variação dos limites de pontuação exigidos no processo de adição ou remoção de membros, e adequar essa pontuação às exigências necessárias de cada serviço.

As métricas utilizadas para a classificação dos nós nos níveis são os seus conceitos fundamentais: contato, permanência e confiança. Estas métricas são necessárias para a formação dos grupos multiníveis. Embora possam ser ajustadas ou modificadas pelo implementador do *Framework*, elas são adequadas para alcançar o objetivo. As próximas seções abordam as delimitações e pressupostos, a relação de contato, permanência, confiança, serviços e níveis de segurança.

6.1 DELIMITAÇÕES E PRESSUPOSTOS

O correto funcionamento do GSeF4V demanda requisitos preliminares à sua implementação. Embora os requisitos resultem na diminuição do escopo de atuação, são fundamentais para garantir a existência de um ponto de partida e de condições mínimas para esse funcionamento. Este trabalho pressupõe que os nós possuem capacidades plenas de atender a todos os requisitos. A viabilidade do GSeF4V depende destes pressupostos.

O primeiro requisito exige que os nós possuam poder de processamento, de armazenamento e energia suficientes para computar as chaves, cifrar e decifrar com criptossistema simétrico ou assimétrico. Cada nó deve possuir um endereço único, um par de chaves pública e privada, atribuído de forma confiável e confidencial. Além disso, todos os nós devem ser seguros a ataques físicos.

A comunicação entre os nós deve ocorrer sem perdas de mensagens. Embora existam possibilidades de retransmissão ao usar protocolos de comunicação fim a fim, este não é o foco deste trabalho. Além disso, as garantias de entrega, retransmissão, ordenação e confirmação de recebimento não são tratadas neste *Framework* e utilizamos dessa premissa para a comunicação sem falhas entre os nós.

Os nós no GSeF4V devem conseguir armazenar as informações necessárias para estabelecer a comunicação. Eles possuem uma tabela local armazenada de forma adequada e segura para conter as informações de confiança e pontuação dos demais nós da rede. Cada nó deve prover a segurança das informações de suas tabelas locais.

O GSeF4V foi projetado para cenários urbanos com comunicação V2V. Embora esta delimitação possa diminuir o campo de atuação, esses cenários não necessitam de infraestrutura, ou seja, neles a comunicação pode ser realizada nó a nó na VANET, desde que os nós utilizem uma tecnologia de intercomunicação igual ou similar a IEEE 802.11 [137]. Ao considerar um cenário urbano, pressupõe-se que os veículos se locomovem a uma velocidade compatível ao perímetro urbano (de 20 a 60 km/h). Além disso, o *Framework* também considera que os condutores dos veículos possam se conhecer previamente para permitir um contato baseado em rede social.

O termo *consenso* neste trabalho não refere-se ao termo *consensus* comumente utilizado para denotar uma série de abordagens de sistemas distribuídos. Neste trabalho, *consenso* deve ser entendido como uma simples consulta de uma quantidade específica e arbitrária de nós que concordam entre si para a realização de uma tarefa.

6.2 CONTATO E PERMANÊNCIA

O contato é o encontro de um nó com outro nó na VANET. O *GSeF4V* considera o termo *contato* como a relação da quantidade de vezes que um nó encontra o outro com objetivo de formarem ou serem partes de um mesmo grupo. Assim, ele é responsável por identificar, inicialmente, os nós aptos a agruparem-se, visto que os nós não conhecem a reputação dos demais nós e nem analisaram seus comportamentos, ou mesmo os conhecem previamente, ainda que tenham se encontrado várias vezes.

O encontro pode ser considerado como uma forma preliminar de um nó conhecer o outro e estabelecer incipiente relação de confiança. Ele é considerado uma métrica para formação de grupos na literatura da área [138, 139]. Esta métrica permite a formação muito rápida de grupos, o que a torna interessante para um passo inicial. No entanto, ela é impactada pela necessidade de um limite superior.

O limite superior de encontros deve ser estabelecido em relação ao número de contatos necessários para iniciar o processo de agrupamento. Este valor pode resultar em diferentes comportamentos. Se o valor for muito baixo, próximo a 1, resultará em muitos nós dentro de um mesmo grupo, porém esses nós terão uma baixa relação de confiança entre eles. Caso contrário, com um limite maior, a relação de confiança será também maior, porém com número menor de nós em um mesmo grupo.

A formação de um grupo somente baseado em encontros incorre em alguns problemas. Um nó pode encontrar rapidamente com outro nó por várias vezes sem proximidade suficiente para estabelecer uma comunicação. Para exemplificar esta situação: considere nós que se cruzam ao longo do dia por estarem constantemente no trânsito; ou mesmo nós que sempre se encontram diariamente em determinados momentos. Estes exemplos podem resultar em um certo nível de confiança pelos encontros, mas não há possibilidade de estabelecer uma conectividade minimamente duradoura entre eles. Sem a comunicação não há meios de utilizar vários serviços da VANET, tornando este grupo inviável para o alcance do objetivo deste *Framework*.

Outro aspecto a considerar é o número de encontros dos nós ser suficiente para torná-los conhecidos, mas não suficiente para classificá-los como confiáveis. Neste *Framework*, o termo *conhecido* significa que ambos os nós conhecem seus dados de identificação, como chave pública, números de identificação únicos, entre outros atributos. O pouco tempo que os nós permanecem

próximos durante o contato não permite analisar o comportamento dos demais nós. Entretanto, seria possível buscar a reputação do nó por sua identificação, mas se o nó estiver em contato esporádico também com os demais nós, o mesmo problema aconteceria, dado que os seus vizinhos teriam apenas contatos breves com este nó.

Para mitigar esta questão, a confiança pode ser atribuída por conhecimento prévio via rede social. Nesta abordagem, o condutor de um veículo conhece previamente o outro condutor através de rede social. Esta informação deve estar contida no momento inicial da utilização do *Framework*. Assim, um dos atributos de identificação servirá também para identificar o contato social entre os condutores dos nós. No entanto, contar apenas com essa métrica pode inviabilizar a presença de diversos nós na rede, tendo em vista que em um cenário urbano pode haver mais nós com condutores desconhecidos do que com condutores que se conhecem previamente por rede social.

Contudo, se os nós permanecerem próximos entre si, será possível considerar o tempo de permanência como métrica substancial ao estabelecimento de confiança entre os nós na VANET. Neste *Framework*, o termo *permanência* diz respeito ao tempo em que um nó permanece em contínuo contato com os demais nós para haver a possibilidade de comunicação e análise do seu comportamento. Vale observar que, ao considerar o cenário urbano, espera-se que os nós mantenham um determinado tempo de permanência durante o trânsito, seja por estarem parados em semáforos, em congestionamento, em vias com baixo limite de velocidade, ou movendo por mesmas vias para destinos próximos.

O tempo de permanência impacta na relação de confiança que pode ser estabelecida entre os nós do grupo. Com maior permanência, o nó torna-se mais conhecido pelos demais nós e fica mais tempo em análise comportamental por eles. Isso aumenta a chance do nó ser considerado confiável ou mal intencionado, conforme o seu comportamento. Com menor permanência há menos chances de um nó ser classificado pelos demais. Esta métrica pode ser utilizada para quantificar a aptidão de um nó para ingressar ou permanecer em um grupo, ou mesmo para elevar seu nível dentro de um grupo já estabelecido.

Assim, a métrica de permanência resulta em nós conhecidos que permanecem conectados por um tempo igual ou superior ao limite preestabelecido. Este limite é fundamental para classificar o tempo de permanência necessário para identificar se um nó pode ser considerado confiável e o nível de confiança que pode ser atribuído a ele. Esta métrica é estratégica para realizar serviços no grupo que demandam conectividade por longo tempo.

O número de contatos aliado à permanência traz um importante passo para a confiança, pois o fato de os nós já serem conhecidos e permanecerem conectados permite ao *Framework* aplicar outras métricas de confiança. A relação entre encontros e permanência para obtenção da confiança e para a formação de grupos aplicada a este *Framework* foi explorada e publicada na literatura em [25].

6.3 CONFIANÇA

A confiança é o conceito positivo que se tem a respeito de algo, e é reforçada em função de atos ou fatos. Ela é uma métrica fundamental na abordagem multinível, pois serve de requisito para os níveis acima do inicial. Ao se classificar um nó como confiável, uma série de atividades podem ser atribuídas a ele, inclusive atividades relacionadas ao funcionamento do próprio *Framework*.

Neste *Framework* a confiança é um atributo binário, ou seja, ou o nó é confiável ou não, não há *ranking* de confiança e, consequentemente, não há nós mais ou menos confiáveis do que outros nós. Contudo, há pontuações para analisar os nós e classificá-los como confiáveis

ou não. As pontuações de atribuição de confiança no GSeF4V estão baseadas em análise de comportamento, consulta da reputação dos nós com seus vizinhos e no perfil social do condutor responsável pelo nó que é carregado quando ele liga o veículo. O módulo de gerenciamento de confiança é responsável por implementar este conceito no GSeF4V.

6.4 GRUPOS MULTINÍVEIS

O GSeF4V é baseado em grupos multiníveis. O conceito de grupo multinível é entendido, neste *Framework*, como um conjunto de nós que formam um todo, onde cada nó é identificado por um nível. O nível é uma posição bem definida dentro de uma hierarquia. Sendo assim, o grupo multinível é formado por nós conhecidos, organizados em níveis que seguem uma classificação hierárquica segundo determinados critérios.

Os níveis são fundamentais para organizar os serviços que compõem e mantêm o GSeF4V, ou seja, cada serviço pode ser atribuído a determinado nível, dependendo da fase em que se encontra o grupo. Além disso, deve haver uma quantidade mínima de nós presentes no grupo para haver plenas condições de estabelecer uma relação de hierarquia. Esta quantidade está representada nas fases.

6.4.1 Fases do grupo

O grupo no GSeF4V está organizado em 3 fases: inicial, intermediária e completa. Cada fase representa um momento de cada grupo, desde a formação até o pleno estabelecimento. Estas fases são necessárias para identificar as diferentes etapas no processo de formação de grupo e para atribuir a relação de serviços, de acordo com os níveis em cada uma destas etapas. As fases são classificadas conforme o contato e o tempo de permanência dos nós no grupo, e a existência ou não de uma relação de confiança estabelecida entre eles. Elas podem ser classificadas como:

- *Fase 1 - inicial*: esta é a primeira fase. É composta apenas por nós recentes, ou seja, com valor de contato e permanência próximos ao mínimo exigido para formação de grupos. Esta fase é composta por nós que ainda não possuem uma relação forte de confiança, embora sejam conhecidos e classificados em um mesmo nível. Nesta fase, todos os serviços básicos são executados por todos os nós. Após o aumento no tempo de permanência entre nós do grupo, é realizada a mudança para a fase 2.

- *Fase 2 - intermediária*: é composta por nós com tempo de permanência no grupo suficiente para terem seu comportamento avaliado pelos demais nós, ao ponto de ser possível atribuir diferentes níveis para distingui-los. Após o aumento de permanência e o estabelecimento de confiança entre os nós é realizada a mudança para a fase 3.

- *Fase 3 - completa*: é composta também por nós que possuem um valor de contato e permanência maior que nas fases anteriores, e uma relação forte de confiança. Nesta fase, todos os níveis podem ser aplicados e seus serviços distribuídos adequadamente entre os níveis.

6.4.2 Níveis dos nós

O *Framework* possui três níveis dentro de um mesmo grupo: nível 1, 2 e 3. A cada nível será atribuído um rol de serviços, considerando as etapas de cada grupo. A separação em níveis deve estar bem definida para atender a diferentes requisitos exigidos pelos diferentes tipos de serviços. Os requisitos para atribuição de níveis consideram o contato, a permanência

e a confiança. Eles são representados pelas variáveis $trustTime$, $trust$, $pointsL1$, $pointsL2$ e $pointsL3$. Essas variáveis são entendidas da seguinte forma:

- $trustTime$: representa o somatório entre o número de contato e o tempo de permanência de um nó com os demais nós do grupo;
- $trust$: é um valor booleano que representa se o nó é classificado como confiável ou não, pela análise do comportamento;
- $socialTrust$: é um valor booleano que representa se o nó é classificado como confiável ou não, por conhecimento prévio dos seus condutores via rede social;
- $pointsL1$: representa o limite mínimo de contato e de permanência para classificar um nó em um nível. Ele é um valor arbitrário;
- $pointsL2$: representa um valor a ser comparado com o tempo de permanência de um nó no grupo. Ele é um valor arbitrário;
- $pointsL3$: representa um valor superior de permanência de um nó no grupo. Este valor deve ser superior ao valor de $pointsL2$. Ele é utilizado para diferenciar os nós que possuem ainda mais tempo de permanência no grupo.

Tais variáveis são utilizadas como expressões algébricas em funções para denotar os requisitos dos níveis. Esses requisitos são as métricas de cada nível. Os nós no GSeF4V incrementam e decrementam valores relacionados ao número de contato, tempo de permanência, análise do comportamento e conhecimento prévio via rede social, durante a interação com os demais nós na rede.

Algoritmo 1 Classificação dos níveis dos nós no grupo.

```
1: level : Integer
2: contactPermanence : Integer
3: trustTime : Integer
4: pointsL1 : Integer
5: pointsL2 : Integer
6: pointsL3 : Integer
7: trust : Boolean
8: socialTrust : Boolean
9: level ← 0
10: if contactPermanence ≥ pointsL1 then
11:     level ⇐ 1
12: else
13:     if ((trustTime ≥ pointsL1 + pointsL2) and trust) or ((trustTime ≥
        pointsL1) and socialTrust) then
14:         level ← 2
15:     else
16:         if (pointsL3 > pointsL2) and ((trustTime ≥ (pointsL1 + pointsL3)) and trust) then
17:             level ← 3
18:         end if
19:     end if
20: end if
```

Os valores obtidos pelos nós compõem os parâmetros dinâmicos que serão constantemente mensurados pelo *Framework* para atingir as métricas e atribuir seus respectivos níveis (observando a fase de cada grupo). O Algoritmo 1 apresenta as regras para atribuição de cada nível aos nós.

A relação dos níveis com os serviços varia de acordo com a fase atual do grupo. Isso acontece devido às exigências de cada fase. Para exemplificar: na fase 1, todos os nós são recentes, não analisaram o comportamento entre si, pois não possuem tempo de permanência suficiente para tal, assim são obrigatoriamente atribuídos ao nível 1. Contudo, estes nós necessitam de realizar os serviços para a manutenção e desenvolvimento do grupo.

À medida que o grupo se desenvolve, algumas funções devem ser realizadas por nós confiáveis, por demandar maior segurança. Este é um procedimento esperado na evolução dos grupos, porém é uma fase de transição de um grupo inicial até se estabelecer. Alguns serviços devem ser atribuídos a níveis superiores conforme eles forem existindo nos grupos. O objetivo é que, com a evolução do grupo, apenas os nós do nível 2 executem os serviços que demandam confiança, e os nós no nível 3 executem os serviços que exijam confiança e permanência do nó no grupo.

Inicialmente, os grupos são formados apenas por nós no nível 1, ou seja, os primeiros nós que atingiram $trustTime \geq pointsL1$. Após o grupo já formado, os nós analisarão o comportamento e atribuirão valores de reputação a seus nós vizinhos. Os nós classificados como confiáveis ($trust$ **ou** $socialTrust$) poderão ser elevados para o nível 2 se atenderem à métrica exigida.

Os nós do nível 2 são conhecidos e classificados como confiáveis. Eles podem se manter no grupo por um tempo prolongado ou serem breves. Para diferenciá-los, o próximo nível adiciona um critério que valoriza os nós confiáveis com períodos maiores no mesmo grupo. Dessa forma, se os nós se mantiverem confiáveis e permanecerem por mais tempo no grupo eles serão classificados como nível 3. A Figura 6.1 ilustra a relação dos nós nos níveis, onde $L1$, $L2$ e $L3$ representam os níveis 1, 2 e 3, respectivamente.

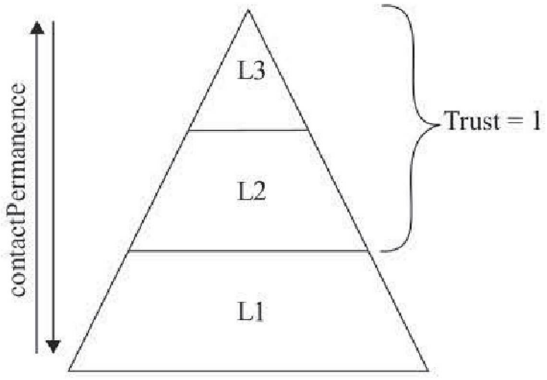


Figura 6.1: Níveis do *Framework*.

A quantidade de níveis e os serviços de cada nível podem também ser modificados pelo implementador conforme os objetivos e requisitos de cada projeto, bem como os valores a serem atribuídos às variáveis de limite para cada nível. Vale observar que o objetivo do GSeF4V é propor uma solução para formar grupos seguros para VANETs com integridade e confidencialidade na comunicação V2V. A flexibilidade desses parâmetros não impacta no objetivo, ao contrário, torna-o versátil para atender a diferentes projetos de implementação.

6.4.3 Fases e níveis

A relação de fase do grupo com os níveis dos nós é estabelecida dinamicamente na interação entre os nós do grupo. No processo de evolução natural, um grupo parte da fase 1, passa pela fase 2 e se estabelece na fase 3. A fase 1 contém os nós no nível 1; a fase 2, os nós nos níveis 1 e 2; e, por fim, a fase 3 detém os nós nos níveis 1, 2 e 3. A Figura 6.2 representa visualmente a relação das fases com os níveis e suas transições.

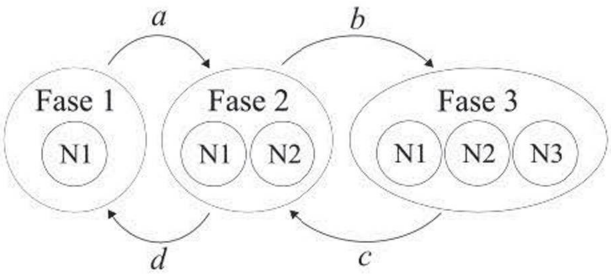


Figura 6.2: Fases do grupo.

As transições entre as fases são caracterizadas pelas mudanças dos níveis dos nós que compõem o grupo. Elas estão organizadas em 4 categorias, representadas na Figura 6.2 por a, b, c e d:

- Transição a: esta transição representa a mudança da fase 1 para a fase 2. Ela acontece quando um ou mais membros possuem a confiança dos demais, seja por bom comportamento ou confiança prévia por rede social, além de um tempo mínimo de permanência no grupo. Por exemplo, em um grupo recém formado há apenas nós no nível 1, porém se alguns nós permanecem próximos o tempo suficiente para adquirir a confiança por bom comportamento, eles irão para o nível 2 e, automaticamente, o grupo estará na fase 2.

- Transição b: esta transição representa a mudança da fase 2 para a fase 3. Ela acontece quando um ou mais membros que estão no nível 2 se elevam para o nível 3. Por exemplo, se um nó se mantém conectado constantemente com os demais nós do grupo, ele será elevado para o nível 3 e, automaticamente, o grupo estará na fase 3.

- Transição c: esta transição representa a mudança da fase 3 para a fase 2. Ela acontece quando não há mais a presença de algum membro no nível 3. Vale observar que o valor de $trustTime$ diminui gradativamente quando não há a permanência do nó no grupo, ou seja, quando o nó não estabelece mais contato ou proximidade com os demais membros do grupo. Assim, não há a migração direta do nível 3 para o nível 1, ela acontece de forma gradual. Por exemplo, quando os membros que estão no nível 3 perdem a conexão com os demais membros do grupo ao ponto de não existir mais nenhum membro no nível 3, este grupo será rebaixado para a fase 2.

- Transição d: esta transição representa a mudança da fase 2 para a fase 1. Ela acontece quando não há mais nós no nível 2. Os nós podem ser rebaixados para o nível 1 se perderem a confiança ou diminuírem seu valor de $trustTime$ para menos do que o limite inferior ($pointsL1$). Por exemplo, se um grupo recém formado possui dois membros que se conhecem via rede social, este grupo estará na fase 2; mas se ambos seguem por diferente rota e se desconectam do grupo, não haverá mais membros no nível 2 e o grupo será rebaixado para a fase 1.

É possível observar não haver transição da fase 1 para a fase 3, nem da fase 3 para a fase 1. Isto se deve às métricas para elevação de nível dos nós, ou seja, um nó deve aumentar seu $trustTime$ e possuir confiança para elevar do nível 1 para o 2, e aumentar mais ainda o seu valor de $trustTime$ para alcançar o nível 3. Assim, a mudança gradual dos níveis reflete também na mudança das fases do grupo.

Há um item a ser observado: o mau comportamento ou perda da confiança. Se em qualquer momento o valor de $trust$ de um nó for alterado para $falso$, esse nó pode ser removido do grupo mediante consenso (detalhado na seção 6.6). Isto acontece tanto nas fases 2 ou 3, pois elas exigem a confiança dos membros do grupo. Desta forma, não há a transição da fase 3 para a fase 1 por motivo de perda de confiança.

6.5 GERENCIAMENTO DE CONFIANÇA

O gerenciamento de confiança é responsável pelos serviços que implementam a confiança no *Framework*. Os principais serviços deste gerenciamento são análise de comportamento, reputação do nó e confiança por perfil social. O serviço de análise de comportamento e reputação estão diretamente interligados na definição de confiança, enquanto o perfil social é abordado de forma independente.

6.5.1 Análise de comportamento

O comportamento é um termo que caracteriza toda e qualquer ação ou reação de um elemento, considerando-se o meio em que está inserido. Neste trabalho, o comportamento é entendido como as ações e as reações realizadas pelos nós na rede antes e durante o processo de formação ou admissão a um grupo de nós confiáveis.

A análise por comportamento acontece de forma descentralizada, ou seja, cada nó analisa os demais nós na rede, seguindo um conjunto de métricas de avaliação de comportamento. As métricas para análise de comportamento foram baseadas e adaptadas de [118], elas são:

- Baseadas em tráfego: considera o número de pacotes recebidos, transmitidos e retransmitidos com sucesso e falhas, além de *Acknowledge* (ACKs) e *No Acknowledge* (NACKs) enviados. Esta métrica avalia se um nó envia apenas pacotes corretos ou se também envia pacotes indevidos com objetivo de prejudicar o sistema. Além disso, se um nó apresentar ações que prejudiquem a comunicação por qualquer motivo, ao ponto de impactar no desempenho ou funcionalidade de serviços do grupo, isso também será considerado um mau comportamento.

- Baseadas em tarefas: considera se um nó executa corretamente as tarefas que são partes do funcionamento do *Framework* ou relevantes para o bom funcionamento da VANET. São as métricas para o número de tarefas executadas por um nó ou por seus vizinhos e a quantidade de tarefas executadas com falhas. Para exemplificar: se um nó não encaminha propositalmente partes das informações de pedido para adição de novos membros no grupo com objetivo de prejudicar algum nó específico, ele será considerado como execução de tarefa com falha. Isto pode ser medido mediante informação sobre a tarefa a ser executada e observações das ações por seus vizinhos.

A definição das métricas para análise de comportamento possibilita classificar os nós quanto à confiança. Essa classificação é implementada no momento em que um nó observa os demais nós próximos a ele. Quando um nó detecta o comportamento do seu vizinho, ele o

observa e pontua positivamente se a ação realizada corresponde à esperada, ou negativamente caso ocorra o contrário.

A análise de comportamento é realizada de forma distribuída, assim cada nó forma uma visão própria de confiança sobre os nós vizinhos analisados, baseada na pontuação atribuída a cada um deles. Esta pontuação é registrada na tabela local. O objetivo é pontuar o comportamento dos nós até atingir um determinado valor limite.

O GSeF4V possui um limite arbitrário de pontuação do comportamento, conhecido previamente para classificar o nó como confiável. Se o nó se comportar de maneira adequada suficientemente para alcançar este limite, ele será considerado confiável por análise de comportamento, caso contrário ele será considerado não confiável. Vale observar que o comportamento dos nós é constantemente analisado, e que um nó considerado confiável pode perder sua confiança se não cumprir as métricas estabelecidas para a confiança. Além disso, a relação de confiança é simétrica, ou seja, se o nó A confia em B então B confia em A.

A classificação do nó como confiável por comportamento ocorre por consenso. O consenso neste *Framework* é entendido como a decisão dos membros do grupo em resposta a uma solicitação de um membro direcionada ao grupo, e acontece somente quando existe um grupo formado por dois ou mais nós, ou seja, no processo de formação do grupo, ambos os nós são considerados confiáveis e irão realizar o consenso a partir da adição do terceiro nó. Cada nó do grupo possui a mesma importância no processo de decisão, respeitando-se os níveis necessários para cada consenso.

Se o nó a classifica o nó b como confiável, ele consultará os membros do seu nível (observando as fases da formação do grupo) sobre a reputação do nó b; se houver consenso, o nó b será convidado a subir de nível. Se o nó a detectar comportamentos maliciosos por parte do nó b pertencente ao mesmo nível, solicitará manifestação dos demais membros para retirar a confiança do membro b e, consequentemente, removê-lo do grupo. A junção dos valores de comportamento recebidos dos vizinhos será o valor de reputação do nó, ou seja, a reputação de um nó é resultado das análises de comportamento que os vizinhos têm sobre ele, otimizando o tempo necessário para um nó ser considerado confiável.

6.5.2 Reputação

A reputação é o conjunto de informações relacionadas ao comportamento de um determinado nó, às quais os demais têm acesso. Esta reputação é consultada mediante uma pesquisa dentre os nós do grupo, nos momentos de solicitação para elevação de nível e atribuição de confiança. Vale observar que os nós podem ter reputações diferentes sobre um mesmo nó, e isto é considerado no processo de consenso.

A análise de comportamento e de reputação devem ser realizadas nos níveis 1 e 2, observando as fases do grupo. Esta análise é feita no nível 1 apenas na fase 1 do grupo, quando este ainda não possui os demais níveis. Isto é importante para permitir a evolução do grupo para a fase 2. Após a transição para a fase 2, apenas os membros do nível 2 farão esta análise.

6.5.3 Confiança por perfil social

Um nó pode ser considerado confiável também através de perfis de rede social dos condutores dos veículos. A identificação do perfil da rede social será armazenada com as informações de identificação de cada nó. Ao realizar o contato e os nós pertencerem ao mesmo grupo, podem estabelecer a confiança como reflexo da confiança estabelecida entre os condutores nas relações sociais, como apresentado na seção 4.2. Dessa forma, se o condutor responsável pelo nó a for confiável na rede social do usuário responsável pelo nó b, os nós serão considerados

confiáveis entre si e poderão ser convidados a subir de nível, se for consenso entre os membros do nível.

A análise de perfil social é realizada apenas no nível 2, pois se os motoristas dos nós forem conhecidos, eles poderão entrar no nível 2 mediante consenso. Exemplo: o nó *a*, membro de nível 2, tem a intenção de adicionar o nó *b* como membro do grupo diretamente no nível 2, por perceber que ele atingiu ($trustTime \geq pointsL1$ **and** $socialTrust$); então, pergunta aos demais membros se pode adicionar o nó *b*, se ele receber 50% + 1 das respostas positivas (este valor pode ser alterado pelo implementador), envia o convite para o nó *b*; se ele aceitar, o nó *a* o adiciona em seu grupo, já no nível 2.

6.6 GERENCIAMENTO DE GRUPO

O gerenciamento de grupo é responsável pelos serviços de adicionar, remover e aceitar membros. A adição do nó no grupo é realizada a partir dos contatos entre os nós, da seguinte forma: a cada encontro, um valor de pontuação é acrescentado na tabela local de ambos os nós até alcançar o valor estabelecido para formação de grupo. Este módulo utiliza o mesmo $trustTime$ dos grupos multiníveis como parâmetro, a mesma métrica do nível 1 para admissão e remoção, ou seja, o somatório entre número de contato e tempo de permanência será medido por $trustTime \geq pointsL1$.

Inicialmente, as interações começam com o valor 0, e a cada encontro com o mesmo nó é acrescentado o valor de 1. Este valor pode ser acrescido também pelo tempo de permanência que os nós estão em contato. Quando os nós alcançarem o valor de $pointsL1$, poderão ser parte de um mesmo grupo, seja formando um grupo novo, seja por meio da adição de um nó como membro em um grupo existente ou da mesclagem de grupos distintos em um único.

A quantidade de pontos para considerar um nó como membro do grupo no nível 1 (valor de $pointsL1$) é arbitrária e pode ser modificada a cada implementação do *Framework*. Todo o processo de adição de membros não é imperativo e sim realizado através de convites, ou seja, um nó pode alcançar o ponto de confiança necessário para fazer parte de um grupo e recusar-se por quaisquer motivos.

Para cada nó adicionado na tabela local, são registrados a data, a hora e o valor de encontro e permanência. Para os nós que já estão na tabela, os valores são alterados para a data e a hora do último encontro, acrescentando-se o valor de 1 do novo encontro. No caso de um nó ser adicionado a um grupo já estabelecido, a admissão se dará da seguinte forma: o nó que inseriu o novo membro de grupo deve exportar sua tabela local de contatos para o novo nó e o contato do novo nó para todos os membros do grupo por *broadcast* de mensagens. Os nós considerados já confiáveis por um vizinho *b* pertencente ao grupo do nó *a* também pertencerão ao mesmo grupo, mediante um consenso dos membros, observando seus níveis. Ainda que alguns nós não sejam conhecidos por *a*, independentemente do nível que estejam, os nós trazidos de outros grupos irão para o nível 1. A partir deste momento é gerada uma nova chave de grupo (Seção 6.10).

Quando os nós *a* e *b* não forem membros de grupos e o nó *a* verificar que o *b* atingiu $trustTime \geq pointsL1$, o convidará para formarem um grupo. Se o nó *b* aceitar, os dois formarão um novo grupo. Se o nó *a* já possuir um grupo, quando ele perceber que o nó *b* atingiu o valor de confiança *n*, perguntará para os demais membros do grupo se eles concordam em adicionar o nó *b*; se, por consenso, um determinado número de nós concordar com esta adição, o nó *a* irá convidar o *b* para participar do grupo; se ele aceitar, o *a* irá fazer os procedimentos para adicioná-lo ao grupo. Caso o nó *b* seja membro de outro grupo, os nós irão se mesclar em um único grupo, mediante aceitação por consenso dos nós dos grupos.

Os novos nós só terão acesso às mensagens a partir deste momento, não será possível abrir as mensagens anteriores com a chave atual de grupo. Da mesma forma, quando um nó sair do grupo, a chave de grupo será atualizada e o nó que saiu não conseguirá mais ler as mensagens.

Qualquer nó pertencente ao grupo pode ser excluído. A exclusão ocorre de duas formas: por tempo ou por comportamento. A exclusão por tempo está relacionada ao tempo em que o nó não informa a sua existência no grupo. A cada intervalo de tempo sem comunicar a sua permanência no grupo, o nó perde 1 ponto no nível de confiança; e ao ficar abaixo do valor determinado pelo utilizador, ele será excluído do grupo de forma arbitrária. Este intervalo de tempo é fixo e escolhido pelo utilizador. A exclusão por comportamento acontece de forma similar, o nó perde 1 ponto a cada comportamento inadequado percebido pelos membros do grupo. Ao atingir o valor inferior ao determinado pelo utilizador, ele será excluído do grupo. Tanto por tempo quanto por comportamento, a exclusão do grupo acontece por revogação da chave compartilhada. O nó excluído poderá, posteriormente, voltar ao grupo se atingir novamente o nível de confiança determinado pelo utilizador do *Framework*.

O consenso está relacionado diretamente com o serviço de aceitar membros. Este serviço serve para que um nó não consiga tomar decisões sozinho, pois, se isto fosse possível, um nó malicioso poderia adicionar mais nós maliciosos, comprometendo o funcionamento do *Framework*. O consenso funciona da seguinte forma: quando o nó a tem a intenção de adicionar um nó b no grupo, dispara uma mensagem com esta intenção e informações do nó b a todos os nós do grupo que tem o nível adequado para realizar o consenso (serviço "participar de decisões"na subseção 6.9). O nó a precisa do aceite de x outros nós, evitando que apenas um nó tome esta decisão. O valor de x deve ser decidido pelo implementador. Caso o nó a seja malicioso e tente, mesmo sem o consenso, adicionar o nó b, os nós que participaram do consenso perceberão o mau comportamento e poderão definir o nó a como malicioso, excluí-lo do grupo e não permitir a entrada do nó b neste momento.

Os serviços de adicionar e remover membros devem ser feitos pelo nível mais confiável - nível 3, porém, na fase inicial, a rede possui apenas o nível 1, e, por estes dois serviços serem básicos, nesta fase, o nível 1 é responsável por eles. À medida que a rede se desenvolve, estes serviços passam a ser desenvolvidos pelo nível mais confiável. O serviço de aceitar novos membros faz parte de "participar de decisões", pelo fato de juntar a resposta de vários nós e, com base nessas respostas, tomar a decisão. Os nós do nível 2 são responsáveis por participar dessas decisões. No início da rede, na fase 1, o nível 1 é responsável por este serviço, mas à medida que a rede se desenvolve, ele será feito pelo nível 2.

6.7 GERENCIAMENTO DE CHAVES

O gerenciamento de chaves é composto pelos serviços de validar informações, calcular nova chave compartilhada e distribuir as chaves. Estes serviços são oferecidos pelo nível mais confiável, contudo são essenciais para a formação de grupos e devem sempre existir, mesmo quando o grupo for recém-formado. Neste sentido, podem ser oferecidos por nós no nível 1, somente na fase inicial do grupo. Conforme haja aumento de nós no grupo e os demais níveis sejam estabelecidos, este serviço deve ser oferecido por nós no nível mais confiável.

A validação da informação é um serviço de consulta de informações ou da veracidade delas. Se um nó precisar obter ou confirmar uma informação sobre outro nó, ele irá recorrer aos nós de níveis 2 e 3. Este serviço pode ser usado em combinação com outros para auxiliar na reputação dos nós, pois pode ser útil para detectar um mau comportamento. Além disso, informações como identificação e chaves públicas podem ser obtidas ou confirmadas, se necessário.

O cálculo da chave de grupo compartilhada é realizado com um sistema de criptografia híbrida. Como mencionado em 4.1.3, a cifragem híbrida é uma forte aliada neste processo. Em particular, o sistema ECIES é viável para utilização do cálculo da chave de grupo. Como ilustrado na Figura 4.4, durante o estabelecimento de uma chave de sessão entre os nós, uma chave secreta compartilhada é gerada e derivada em chave para cifragem e chave para MAC, com objetivo de garantir a confidencialidade e integridade, consecutivamente.

Quando os nós estabelecem a chave simétrica compartilhada, ela é propagada entre os membros da nova formação do grupo. Vale observar que não é necessário todos os nós participarem do processo de geração da chave, visto que já existe uma relação de confiança em diferentes níveis entre os membros do grupo. Para cada nível haverá uma chave compartilhada de forma adicional, ou seja, um nó no nível mais alto possuirá todas as chaves simétricas compartilhadas existentes no grupo. Este serviço pode ser realizado por diferentes nós, observando a fase e nível do *Framework* (6.1).

A distribuição da chave acontece sempre que uma nova chave for estabelecida, e se dá de duas formas:

- Adição de membros: por *multicast* assegurada pelas chaves de grupo anteriores. Para exemplificar, considere o caso em que dois grupos se mesclarão por intermédio dos nós *a* e *b* de grupos distintos. A distribuição acontece da seguinte forma: quando um nó *a* estabelece a nova chave de grupo com o nó *b* eles informarão esta nova chave aos seus contatos, ou seja, o nó *a* usará a chave anterior do grupo de *a* para cifrar a mensagem com a nova chave, enquanto o nó *b* realizará o mesmo procedimento com seu grupo. Quando todos os nós souberem a mesma chave, o grupo estará estabelecido.

- Remoção de membros: por *multicast* assegurada pela nova chave de grupo gerada sem o nó removido.

6.8 COMUNICAÇÃO

A comunicação é responsável pelos serviços de informações sobre o trânsito, informações de risco, pedidos de ajuda ou mesmo de propagar informações de propósitos gerais. Estes serviços são oferecidos por diferentes níveis. Os serviços de informações de trânsito, pedidos de ajuda e propagar informações são oferecidos apenas pelos nós no nível 1. Os serviços de informações de risco e decisões de trânsito são oferecidos apenas no nível 3, pois estes requerem nós confiáveis para seu fornecimento. Vale ressaltar que um nó no nível 3 também faz parte dos níveis 2 e 1.

O serviço de informações de trânsito serve para informar os demais nós sobre como está o trânsito, por exemplo, em relação a congestionamento, acidente, pistas bloqueadas, entre outros. O serviço de pedido de ajuda serve para caso um nó incorra em acidente, como batida, pneu furado, entre outros; assim, poderá notificar o ocorrido e solicitar ajuda a outros nós. O serviço de informações de risco refere-se à notificação de sinistros que podem gerar risco aos demais nós na rede. As informações que não exigem confiança podem ser realizadas por nós no nível 1, porém se a mensagem representar risco, deve ser gerada por um nó com nível 3. Um nó possivelmente atacante no nível 1 pode enviar mensagens de ataque, mas como ele não pertence aos níveis 2 ou 3 (que exigem confiança) essa mensagem será ignorada. Caso um nó situado em níveis de confiança torne-se malicioso, ele será rapidamente identificado por seu mau comportamento e será removido do grupo.

O serviço de propagar informações serve para divulgar mensagens para todos os nós do grupo. Como este serviço tende a ser utilizado constantemente, deve ser realizado por nós

do nível 1, por demandar maior quantidade de nós propagadores. A mensagem será cifrada observando as chaves compartilhadas de cada nível, de modo que a confiança não é um item obrigatório, pois a integridade e a confidencialidade estarão garantidas. Contudo, se um nó do nível 1 comportar-se de forma maliciosa, seu comportamento será percebido e ele será removido do grupo.

Com as informações propagadas corretamente, bem como informações de riscos, os nós em nível 3 podem oferecer o serviço de decisões de trânsito. Este serviço pode ser usado para melhorar o fluxo de veículos ou desviar os nós de rotas que estejam possivelmente bloqueadas por acidente, avarias ou manutenção. Neste caso, os nós mais confiáveis podem decidir por melhores caminhos e recomendá-los aos demais nós, para que cada um possa tomar sua decisão de trânsito. Vale observar que esta mensagem é uma recomendação que cada nó pode ponderar.

O implementador pode classificar os serviços e atribuí-los aos níveis da melhor forma para a rede proposta, bem como atribuir novos serviços e suas adequações nos níveis. O foco deste *Framework* não é explorar os serviços, e sim criar um ambiente em que estes serviços possam ser realizados de forma segura em um grupo na VANET, ou seja, possibilitar as classificações e níveis. Os valores de $pointsL1$, $pointsL2$ e $pointsL3$ devem ser determinados pelo implementador, pois ele quantificará os limites de cada nível. A quantidade de níveis também pode ser alterada conforme a necessidade.

6.9 RELAÇÃO DE SERVIÇOS, NÍVEIS E FASES

Os serviços são as ações realizadas pelos nós. Eles estão organizados nos módulos de gerenciamento de confiança, grupo, chave e comunicação, e são demanda externa e interna. As demandas externas estão relacionadas com o uso da VANET para prover ou trafegar informações de aplicações, a exemplo de aplicações para entretenimento, coleta de dados sobre condições das estradas ou velocidade dos veículos; as demandas internas são todas as ações que formam o GSeF4V.

Os serviços podem ser classificados de acordo com suas exigências de confiança e permanência. Os serviços que exigem confiança devem ser executados por nós que estejam, no mínimo, no nível 2, pois para entrar nesse nível eles devem ser considerados confiáveis. Tais serviços são: atribuir reputação, participar de decisões que apenas demandem confiança, analisar comportamento, analisar reputação e analisar perfil social.

Contudo, alguns serviços exigem, além da confiança, que o nó mantenha-se conectado no grupo por mais tempo, ou seja, tenha um tempo maior de permanência no grupo. Estes serviços requerem que sejam executados por um nó no nível 3. São eles: decisões de trânsito, adição de membros, remoção de membros, validação de informações, cálculo de nova chave, distribuição da chave de grupo, percepção e divulgação de informações de risco.

Há serviços, também, que não exigem segurança ou que o nó tenha alta permanência no grupo. Esses serviços podem ser executados por qualquer nó no grupo, desde o nível 1. Eles são: informações de trânsito, pedidos de ajuda e propagação de informações. A principal vantagem de serviços dependerem apenas de nós do nível 1 é a quantidade de nós existentes nesse nível, pois todos podem executá-los e, assim, haverá maior quantidade de nós móveis para esta finalidade. Porém esses serviços não poderão contar com segurança, nem exigir permanência dos nós no grupo.

Os grupos podem não possuir membros em todos os níveis a todo momento. Por esse motivo, a atribuição dos serviços aos níveis dos nós deve, primeiramente, atentar-se à fase em que o grupo se encontra. Os serviços que implementam os módulos deste *Framework* são fundamentais para seu funcionamento e devem ser realizados ainda que não haja nós nos níveis

Tabela 6.1: Serviços oferecidos em cada nível do GSeF4V.

Serviços	Níveis a cada fase		
	Fase 1	**Fase 2**	**Fase 3**
a) Decisões de trânsito	-	-	N3
b) Atribuir reputação	N1	N2	N2
c) Adicionar membro	N1	N2	N3
d) Remover membro	N1	N2	N3
e) Participar de decisões	N1	N2	N2
f) Analisar comportamento	N1	N2	N2
g) Analisar reputação	N1	N2	N2
h) Analisar perfil social	-	N2	N2
i) Validar informações	N1	N2	N3
j) Calcular nova chave	N1	N2	N3
k) Distribuição da chave de grupo	N1	N2	N3
l) Informações de trânsito	N1	N1	N1
m) Informações de risco	-	-	N3
n) Pedidos de ajuda	N1	N1	N1
o) Propagar informações	N1	N1	N1

exigidos. Dessa forma, as atribuições dos serviços aos nós mudam conforme a fase em que o grupo se encontra e a sua transição. A Tabela 6.1 apresenta a relação de serviços, níveis dos nós e fases do grupo.

Esta relação de serviços, níveis e fases é um passo inicial para organização deste *Framework*. Novos serviços podem ser adicionados, bem como os serviços atuais podem ser alterados, para melhor adaptação ao objetivo da implantação. Esta relação de serviços e níveis deve ser entendida como uma opção padrão para o seu funcionamento. A possibilidade de mudanças e adaptações torna este *Framework* ainda mais versátil e dinâmico. É importante ressaltar que o objetivo do *Framework* é criar um ambiente baseado em grupo multinível, que ofereça a comunicação íntegra e confidencial, e que essa organização de serviço é parte deste processo.

6.10 GARANTIA DA CONFIDENCIALIDADE E INTEGRIDADE

A garantia da confidencialidade ocorre através de cifragem e acontece da seguinte forma: após formar um grupo, todos os seus membros conhecem a chave compartilhada de cada nível e estão aptos a se comunicar; a cifragem simétrica será usada para proteger a comunicação contra ataques à confidencialidade realizados por nós externos ao grupo. O algoritmo de cifragem é uma decisão de implementação do utilizador do *Framework*. Exemplos de sistemas que podem ser escolhidos são AES [140], RC4 [106], *Triple Data Encryption Standard* (3DES) [141], entre outros; no entanto, ao considerar uma abordagem de cifra híbrida, em específico o ECIES, a cifra simétrica AES torna-se uma escolha mais adequada.

A chave compartilhada de grupo destina-se a garantir a confidencialidade da comunicação por difusão, seja *broadcast* ou *multicast*, pois todos os nós de mesmo nível do grupo conhecem a mesma chave. A comunicação direta entre dois nós terá a confidencialidade garantida através de um algoritmo de criptografia assimétrico. Vale ressaltar que neste momento os nós pertencentes ao grupo conhecem a chave pública de todos os outros, pois, no momento de associação do grupo, a tabela local de identificação dos nós, contendo a chave pública de todos, é informada ou atualizada. De forma similar à comunicação de grupo, o algoritmo de criptografia para a comunicação direta também é uma decisão de projeto, porém, para a escolha de tal algoritmo, deve ser considerado o nível de segurança desejado em relação ao desempenho que se espera na rede. Exemplos de algoritmo de criptografia assimétrica que podem ser utilizados são RSA [102], ECC [142], entre outros.

A integridade da comunicação será garantida por *hash*, provido por algoritmos de MAC. O nó remetente calcula o *hash* da mensagem antes de enviar, gerando um resumo que será enviado com a mensagem. Esse cálculo pode ser implementado com diferentes algoritmos de *hash*, como o SHA-224 [143], SHA-256 [144], MD5 [145], entre outros, a serem escolhidos conforme a exigência do projeto em relação a desempenho e segurança. Uma comparação entre estes algoritmos pode ser encontrada em [146, 147, 148]. Vale ressaltar que o algoritmo de *hash* utiliza uma chave criptográfica em sua operação, momento no qual o uso de cifra híbrida, como o ECIES, torna-se conveniente, pois, como ilustrado na Figura 4.4, uma chave para MAC pode ser derivada da chave simétrica compartilhada para gerar e conferir o *hash* no envio e recebimento, respectivamente; dessa forma, os nós conseguirão garantir a integridade da mensagem.

6.11 CONCLUSÃO DO CAPÍTULO

Este capítulo apresentou o GSeF4V, um *Framework* para segurança multinível, baseado em grupo. O GSeF4V foi projetado e utilizado a partir dos conceitos de contato e permanência, confiança, níveis, serviços e garantia da confidencialidade e integridade. Cada parte deste *Framework* é fundamental para as demais partes, de tal forma que estão interdependentes para alcançar seu objetivo.

O contato e permanência são utilizados para conhecer e classificar os nós, além de permitirem a análise de confiança. Estas métricas são importantes para a entrada dos nós em um grupo. Após essa entrada, a confiança é observada, seja por comportamento, reputação ou perfil social. Essas métricas juntas formam os critérios para estabelecer os níveis dentro de um mesmo grupo, e podem oferecer serviços correspondentes a cada nível.

Os serviços foram mostrados para ilustrar como os níveis podem ser utilizados. Estes serviços variam conforme a fase do grupo. Eles demandam níveis diferentes para serem executados, e consideram desde nós recém adicionados, para difusão de mensagens, até nós confiáveis e com maior permanência no grupo, para serviços mais sensíveis, como remoção de confiança por mau comportamento.

O gerenciamento das chaves é um serviço também realizado por nós com níveis adequados. Este serviço é fundamental para estabelecer um grupo e permitir a garantia de confidencialidade e integridade. Com o grupo formado e com as chaves estabelecidas, são utilizados algoritmos de cifragem e MAC para confidencialidade e integridade das mensagens trafegadas na rede, ainda que em um canal inseguro e sem a necessidade de uma unidade centralizadora.

Dessa forma, este *Framework* atende ao objetivo do trabalho, pois garante a confidencialidade e integridade da comunicação de redes veiculares, sem a necessidade de infraestrutura e independente de protocolos de comunicação. No entanto, o cenário de ataque, a cobertura da proteção em relação aos ataques conhecidos na literatura, a mensuração do desempenho e a definição dos custos computacionais e de tempo devem ser analisados para verificar a viabilidade deste *Framework*.

Como objetivo de trabalhos futuros no âmbito desta pesquisa, destaca-se a possibilidade de aumentar as métricas de redes, como, por exemplo, *jitter*, o número de pulos para chegar nos nós, a intensidade do sinal, entre outras. O capítulo 7 apresenta o modelo de ataque, cobertura de proteção do *Framework*, bem como a avaliação de desempenho e seus custos.

7 AVALIAÇÕES DO *FRAMEWORK*

Este capítulo apresenta a avaliação do *framework* proposto e está organizado da seguinte forma: ambiente de simulação, carga de trabalho, simulador do *Framework*, avaliações do *Framework* e considerações do capítulo. As avaliações estão organizadas em 4 partes: avaliação de nós maliciosos, avaliação dos limites para $pointsL1$, avaliação dos limites para permanência, avaliação dos limites para confiança e avaliação da segurança. O objetivo da escolha destas avaliações está em perceber o impacto da variação dos parâmetros no comportamento gerado pelo *Framework*, seja por consequência da carga de trabalho ou para auxiliar o implementador na tomada de decisão na atribuição de valores arbitrários; além de apresentar os custos relacionados à segurança.

7.1 AMBIENTE DE SIMULAÇÃO

As simulações foram realizadas em um computador pessoal cujos parâmetros de *hardware* estão apresentados na Tabela 7.1. Foram utilizados dois *softwares* para a simulação, o simulador *The* ONE [149] e o GSeF4V-Simulator, um *software* construído especificamente para testar este *Framework*. Ambos os *softwares* foram escritos na linguagem de programação Java, sendo executados sob a máquina virtual *OpenJDK Runtime Environment* (build 11.0.13+8-Ubuntu-0ubuntu1.20.04).

Tabela 7.1: Parâmetros de hardware.

Parâmetro	**Valor**
Processador	Intel Core i5-9300H CPU @ 2.40GHz
Arquitetura	X86-64
Memória	16 GB DDR4 2666 MHz
Armazenamento	SSD-NVME 128 GB
Sistema Operacional	Ubuntu 20.04.3 LTS (64 bits)
Versão do Kernel	5.11.0-41-generic

7.2 CARGA DE TRABALHO

O *The* ONE foi escolhido para simular o padrão de mobilidade dos nós e gerar traços para entrada da simulação do GSeF4V. Ele é um simulador projetado para auxiliar na avaliação de soluções de redes veiculares e permite aos usuários criar cenários baseados em diferentes modelos sintéticos e traços reais. As principais funções do *The* ONE são: modelagem de movimento do nó, contato entre os nós, verificação da conectividade, além de manipulação das mensagens e roteamento. As funções de modelagem de movimento, contato e conexão foram o ponto-chave para escolha deste simulador.

O *The* ONE foi executado com o modelo de movimento de um dia de trabalho, extraído a partir do trânsito da cidade de Helsinque na Finlândia. Este traço de entrada é oferecido pelos

próprios autores desse simulador, é bem aceito na comunidade científica e adequado para esta avaliação de desempenho. O volume da carga foi definido para representar o trânsito de 1 semana com a densidade de veículos variando entre 150, 300 e 450. A Tabela 7.2 apresenta todos os parâmetros utilizados na execução do *The* ONE.

Tabela 7.2: Parâmetros da simulação.

Parâmetro	**Valor**
Tempo de simulação	1 semana
Camada MAC	WiFi 802.11ax
PHY	HE-OFDMA
Modulação	MIMO-OFDM
Taxa de dados	1147 Mbps
Alcance	120 metros
Quantidade de nós	150, 300 e 450 veículos
Velocidade dos nós	10 até 50 km/h
Modelo de movimento	*Working Day Movement Model*
Mapa mundial	Helsinque - Finlândia
Tamanho do mundo	4500 x 3400 metros

Nesta avaliação, o *The* ONE foi utilizado para gerar a carga de trabalho a partir de dados gerados no relatório de conectividade. Este relatório é o resultado de todos os contatos e conectividade entre os veículos a partir do modelo de movimento. A Figura 7.1 apresenta o modelo deste relatório, no qual o *timestamp* marca o tempo em que ocorreu a percepção da conectividade, o *idA* é a identificação do nó que percebeu a conectividade, o *idB* é a identificação do nó percebido e o *status* apresenta a situação da conexão neste momento, que retorna *UP* se os nós estão conectados e *DOWN* caso contrário.

```
<timestamp> <id A> <id B> <status>
```

Figura 7.1: Formato da carga de trabalho.

Os traços contêm um comportamento de rede de uma semana e o objetivo de aproximar o movimento simulado do mundo real, considerando o modelo de conduta dos motoristas. O MAC *Layer* WiFi 802.11ax (phy HE-OFDMA, frequência 2.4 Ghz, largura de banda de 20 Mhz e MU-MIMO, taxa de dados de 1147 Mbps e alcance externo de aproximadamente 120 metros) foi escolhido para fornecer uma taxa de dados mais alta para alcance externo. As contagens de *host* de 150, 300 e 450 foram escolhidas para variar a densidade da rede. A velocidade do veículo de 10 a 50 km/h se aproxima da velocidade real na maioria das cidades, considerando ruas e avenidas pequenas. O modelo de movimento do simulador *The* ONE, o mapa mundial e o tamanho do mundo fornecem parâmetros viáveis para recuperar dados de rastreamento para estrutura da simulação do GSeF4V.

7.3 SIMULADOR

O GSeF4V-Simulator é o simulador que implementa este *Framework*, e está dividido em 3 pacotes: *Main*, *FileHander* e *Framework*. A execução começa com as classes do pacote *Main*, responsáveis pela atribuição dos parâmetros iniciais do *Framework* e pela inicialização do *software*. Os parâmetros da simulação são: *number_nodes*, o número de nós na simulação (obtido a partir dos traços do *The* ONE); *malicious_rate*, taxa de nós maliciosos; *social_rate*, taxa de nós conhecidos por rede social; *trust_time*, intervalo de tempo exigido para atribuição de pontuação por permanência em contato; *beta_points*, para registrar o comportamento dos nós; e os parâmetros do GSeF4V $pointsL1$, $pointsL2$, $pointsL3$. O código fonte desta implementação está disponível no Apêndice A.

Após a configuração inicial, atribuição e verificação dos parâmetros, as classes do pacote *Filehandler* são invocadas para manipular os arquivos com os traços e entrada. A API java.nio [150] é utilizada para trabalhar com operações de entrada e saída, com objetivo de carregar todo o conteúdo da carga de trabalho na memória principal e organizar os dados em pequenos trechos (*tokens*) de *string*. Esses *tokens* são verificados, carregados em um modelo de dados abstrato e armazenados em um conjunto de dados, para posterior utilização.

Após carregar e verificar toda a carga de trabalho, as classes do pacote *Framework* são instanciadas para executar a implementação deste *Framework*. As principais classes deste pacote são: *Framework*, *Vehicle*, *Group* e *LocalTrust*. Elas estão ilustradas na Figura 7.2. Essas classes são interdependentes. A classe *Framework* realiza instâncias da classe *Vehicle*. A classe *Vehicle* possui instâncias da *LocalTrust* e da classe *Group*. A Classe *Group* possui um conjunto de instância da classe *Vehicle*, chamado Membros, em que cada uma possui uma tabela local de confiança (*LocalTrust*) relacionada a ele. Embora o simulador possua outras classes nos pacotes, essas são as principais, que implementam as regras e interações do GSeF4V.

7.3.1 Framework

A classe *Framework* implementa os métodos run, $getVehicle$ e $setLog$. O método *run* executa a simulação de forma geral. Ele é responsável por invocar os métodos das instâncias das demais classes para executar as regras e interações do *Framework*, enviar os parâmetros e registrar os resultados. O método *setLog* é responsável por criar os arquivos com os *logs* do sistema, contendo os resultados obtidos da simulação. Os resultados são: o número máximo de nós que alcançaram os níveis em um grupo (observe que, como a presença dos nós nos grupos é muito dinâmica, este resultado registra apenas o valor máximo); de forma similar, também é registrado o número máximo de veículos que alcançaram cada nível, ou seja, do universo total de nós quantos alcançaram o nível 1, 2 e 3; o número máximo de nós em um grupo; o tempo máximo de permanência de um nó em um grupo; e a quantidade de nós criados durante toda a simulação.

O traço de entrada fornece todas as movimentações de cada nó. Inicialmente, as primeiras ocorrências servem para iniciar a instância do nó e atribuir os primeiros contatos, as ocorrências seguintes reportam os comportamentos dos nós já instanciados. O método *getVehicle* é responsável por gerenciar essas instâncias, além disso, no momento da instanciação, este método expõe se um nó será malicioso ou não, ou se ele é confiável previamente por relação de redes sociais, conforme os parâmetros de *malicious_rate* e *social_rate*.

O *Framework* trabalha com uma abordagem descentralizada, assim todas as regras que o implementam são realizadas pelas instâncias dos nós, ou seja, cada nó tem sua própria visão do sistema. Os nós de um grupo devem conhecer os membros e seus níveis para realizar operações como consenso de adição ou de remoção de nós no grupo. Quando houver necessidade de que um

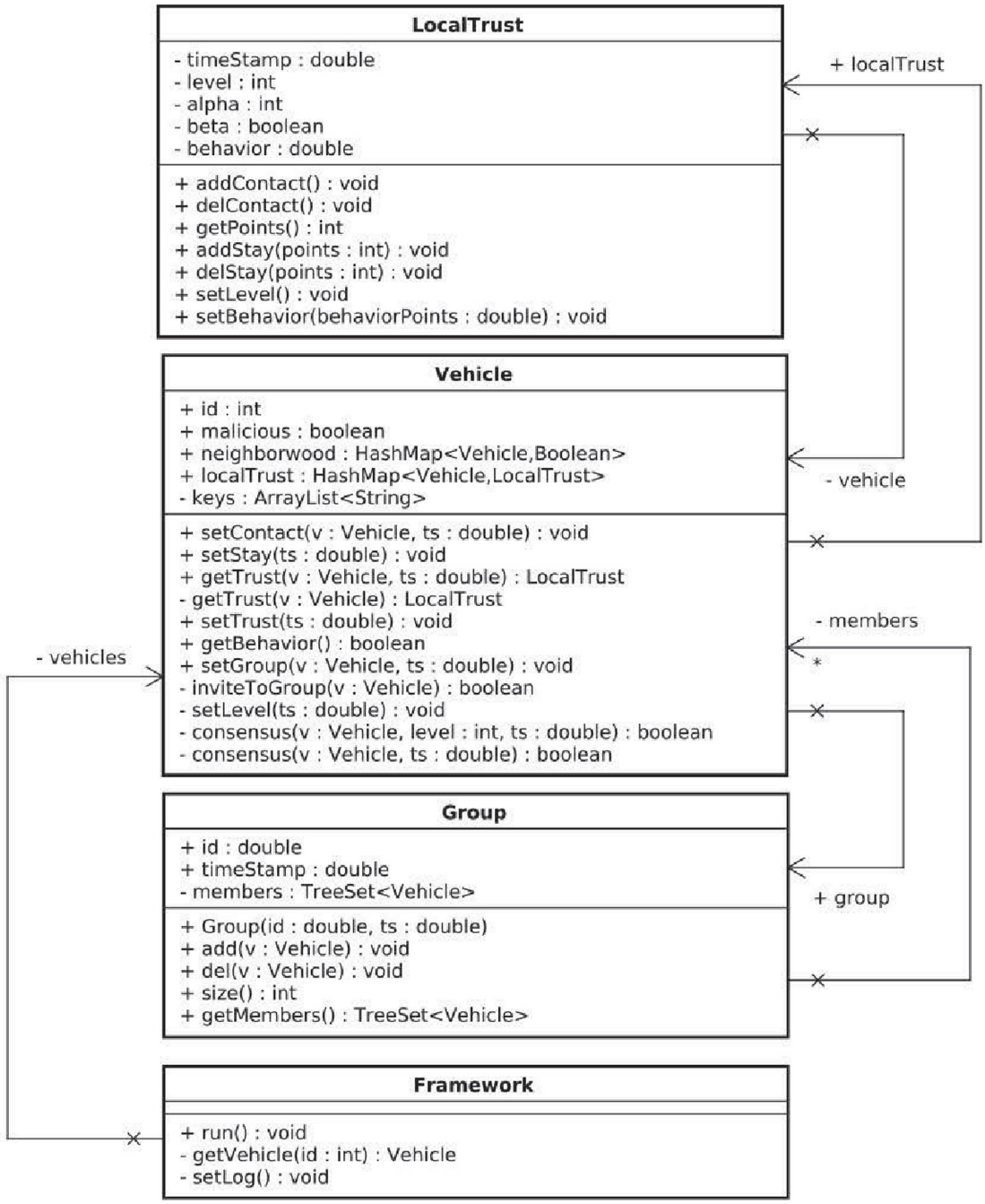


Figura 7.2: Diagrama de classe do GSeF4V-Simulator.

conjunto de nós tenha uma mesma visão, ele o fará por mensagens *broadcast*. Este procedimento foi implementado neste *Framework* por troca de mensagens nos parâmetros, pois o uso de *threads* não foi implementado. Vale observar que o tempo de processamento não foi considerado para avaliar as regras de grupos e níveis desta estrutura, apenas os resultados produzidos pelas regras a partir da carga de trabalho.

7.3.2 Veículo

A classe *Vehicle* implementa o modelo de veículos, que, essencialmente possui uma identificação única (atributo *id*), uma vizinhança conhecida (atributo *neighboorwood*), a habilidade de identificar se o veículo é essencialmente malicioso ou não (atributo *malicious*),

uma tabela de confiança local contendo dados de confiança de todos os nós conhecidos (atributo *localTrust*) e um conjunto de dados para armazenar todas as suas chaves (atributo *keys*). As principais funções do *Framework* são: *setContact*, *setStay*, *setTrust*, *getBehavior*, *setLevel*, *setGroup*, *inviteToGoup*, e *consensus*. O método *setContact* é responsável por pontuar todo contato realizado pelos nós em sua vizinhança; positivamente, se conectados, e negativamente, se desconectado. A vizinhança é o conjunto de informações dos veículos próximos armazenados no atributo *neighboorwood*. O método *setStay* utiliza o parâmetro de entrada *trust_time* para pontuar o tempo que os nós permanecem conectados ou desconectados. O método *setTrust* é responsável por analisar o comportamento dos demais nós. Ele invoca o método *getBehavior* para verificar se o comportamento realizado é malicioso ou não, e registrar em sua tabela de confiança local. O método *getBehavior* simula o comportamento dos nós, se o nó for confiável, todo o comportamento será positivo, porém se o nó for malicioso, este método retorna aleatoriamente um comportamento positivo ou negativo, pois o nó malicioso pode agir de forma correta ou não. Todas as pseudoaleatoridades foram implementadas com classes nativas, por meio da utilização de sementes de 48 bits e o *timestamp* como informação inicial para auxiliar na variação da semente.

O método *setLevel* verifica o nível dos membros do grupo, buscando por alteração de nível. Se o nó percebe que houve alteração, seja para elevar ou diminuir de nível, ele realiza um consenso para efetivar ou não esta mudança, por meio do método *consensus*. Se houver consenso, todos os membros do grupo atribuem o mesmo nível de confiança em suas tabelas locais de confiança. O método *setGroup* é responsável por criar o grupo, adicionar e remover membro. Se um nó percebe que seu vizinho já alcançou os pontos estabelecidos nos parâmetros iniciais, eles podem formar um novo grupo, se ambos não estiverem em nenhum outro. Em todo processo de adição de membro no grupo é necessária a aceitação de um convite, implementado pelo método *inviteToGroup*. Caso algum dos nós já possua um grupo, será realizado, também, um consenso, por meio do método *consensus*, para permitir a entrada do novo nó no grupo. Se ambos já possuírem grupos, todos os membros dos dois grupos serão mesclados em um único, mediante consenso entre os membros de cada grupo. A remoção de um nó ocorre de forma similar, também sob consenso.

Como já mencionado, o método *consensus* implementa o consenso tanto para atribuição de nível quanto para gerenciamento do grupo. O consenso é um escrutínio realizado apenas por membros não maliciosos, de nível competente ao serviço requerido. Se o nó a ser consentido for conhecidamente malicioso por qualquer membro votante, o consenso será negado, sem a realização de escrutínio. Para a atribuição de nível, a maioria dos nós deve ter a pontuação e confiança que permitam a classificação do nível requerida. Se houver o consenso, todos os membros devem atualizar sua tabela de confiança local sobre o nó escrutinado para o nível consentido e iniciar o processo de nova chave compartilhada de grupo para os membros deste nível. Vale ressaltar que os membros de nível superior são um subgrupo dos níveis inferiores, ou seja, quando houver mudança na chave compartilhada nos níveis mais baixos, os membros dos níveis mais altos também serão atualizados com a mudança.

O consenso da adição no grupo ocorre de forma similar. Os demais membros verificam em suas tabelas locais se o nó se comporta de forma maliciosa e votam pela entrada ou não dele. Se não houver consenso, os grupos permanecem como estão. Se houver, todos os nós do grupo adicionam as informações de confiança sobre o novo nó em suas tabelas locais de confiança; se não houver uma informação prévia, adicionam o novo nó em sua visão local de grupo e inicializam o processo de nova chave compartilhada de grupo. O consenso da remoção de membro acontece de forma mais simples, basta que nenhum membro seja contra a remoção. A decisão de ser contra a remoção é tomada a partir da pontuação que um membro tem do nó

em votação, ou seja, se algum nó tiver pontos suficientes para sua manutenção, ele será contra a remoção.

7.3.3 Confiança Local

A classe *LocalTrust* implementa o modelo de confiança do *Framework*. Ela é uma estrutura de dados abstrata que registra todos os valores de confiança que um nó tem sobre os outros. Note que a tabela de confiança local na classe veículo é um conjunto de várias instâncias dessa classe (um para cada nó) e possui, principalmente, os atributos $timeStamp$, $level$, $alpha$, $behavior$ e $beta$. O atributo $timeStamp$ é utilizado para marcar o tempo de cada alteração da confiança local e para auxiliar na detecção de atualizações; o atributo $level$ registra o nível que o nó tem em relação ao outro; o atributo $alpha$ armazena a computação dos pontos resultantes dos contatos e do tempo de permanência; o atributo $behavior$ armazena a computação da análise de comportamento; e o atributo $beta$ representa se o nó é confiável ou não.

Os principais métodos dessa classe são: $addContact$, $delContact$, $getPoints$, $addStay$, $delStay$, $setLevel$ e $setBehavior$. Os métodos $addContact$ e $delContact$ são responsáveis por adicionar e remover os pontos por contato; de forma similar, os métodos $addStay$ e $delStay$ são responsáveis por adicionar e remover os pontos por tempo de permanência; o método $getPoints$ é responsável por retornar o valor de $alpha$, computado pelos contatos e tempo de permanência; o método $setLevel$ é responsável por atribuir o nível de cada nó, a partir das regras de nível (Tabela 1); o método $setBehavior$ recebe a pontuação obtida pela análise do comportamento, computa e verifica se o nó alcançou os pontos para se tornar confiável, considerando os parâmetros iniciais, então a confiança poderá ser atribuída ou retirada, ou seja, conferirá o valor para o atributo $beta$.

7.3.4 Grupo e cifragem

A Classe grupo possui, principalmente, os atributos id, como identificador único; $timeStamp$, para registrar o tempo de criação e atualização; e $members$, contendo um conjunto dos nós membros do grupo. Os principais métodos são: add, para adicionar um nó no grupo; del, para remover um nó do grupo; $size$, para retornar à quantidade de membros existentes no grupo; e $getMembers$, que retorna a referência para cada membro do grupo. Vale ressaltar que cada nó possui sua própria instância de grupo, e que todas as instâncias são sincronizadas nos momentos de alteração e consenso por troca de mensagens.

Em cada momento de adição ou remoção do grupo, o mecanismo de geração de chaves é invocado. O foco deste simulador está em perceber o comportamento das regras do processo de formação de grupos e níveis. Quando o grupo e os diferentes níveis estão estabelecidos, é possível utilizar implementações já conhecidas para processar o sistema de cifragem ECIES, escolhido previamente. Dessa forma, este simulador utilizou uma implementação adaptada do ECIES com a curva elíptica secp256r1 e 128 *bits* AES, disponibilizada pela biblioteca *Multiprecision Integer and Rational Arithmetic Cryptographic Library* (MIRACL), disponibilizado por [151]. A implementação desta biblioteca permite calcular o tempo de processamento para estabelecer uma chave de sessão com o ECIES, para cifrar e decifrar com o AES, para realizar uma etiqueta (*hash*) e para sua verificação.

7.4 AVALIAÇÕES

As avaliações visam medir o comportamento do *Framework* pelas variações de parâmetros. O objetivo é analisar o sistema em relação a diferentes quantidades de nós maliciosos; testar

diferentes limites para alcançar a métrica $pointsL1$ (pontos necessários para fazer parte de um grupo ou formar um, como nó em nível 1); os limites para pontuar a permanência dos nós no grupo; os limites de pontos necessários para considerar um nó confiável; e, por fim, avaliar a segurança. Os parâmetros são:

- *Quantidade de nós*: representa a quantidade de nós inseridos na simulação. As variações escolhidas para este parâmetro foram 150, 300 e 450 nós. Estes valores foram escolhidos para representar diferentes densidades de nós na simulação e, com isto, obter diferentes comportamentos.

- *Quantidade de nós maliciosos*: representa o número de nós maliciosos injetados na rede veicular. Este parâmetro é útil para o *Framework* classificar pseudoaleatoriamente a quantidade de nós com maus comportamentos. O valor de 10% representa a menor porcentagem possível, resultando em pelo menos 1 nó malicioso, considerando as variações da quantidade de nós na rede.

- *Quantidade de nós confiáveis por rede social*: representa a quantidade de nós cujos motoristas se conhecem previamente via rede social. O valor de 20% é fixo nas simulações, sendo escolhido por ser suficiente para ilustrar o funcionamento da característica de confiança por rede social incluída neste *Framework*.

- *Limite para confiança*: representa a quantidade de pontos que o nó deve obter para ser considerado confiável. O valor de 1000 pontos foi definido como o valor mínimo, mediante observação das simulações. A variação deste valor é explorada na seção 7.4.4.

- *Limite para permanência*: representa o contato e o tempo em segundos que um nó deve permanecer com os demais para incrementar pontos que serão medidos com as métricas $pointsL1$, $pointsL2$ e $pointsL3$. Este limite também é usado para decrementar a pontuação. O tempo de 10 segundos foi escolhido considerando a dinamicidade dos nós nas redes veiculares e o tempo mínimo que poderia ser utilizado, observando o comportamento nas simulações. A seção 7.4.3 explora a variação deste parâmetro.

- *Limite para* $pointsL1$: representa a quantidade de pontos de contato e permanência exigida para que o nó possa ser convidado para ser membro de um grupo ou formar um novo, além de ser caracterizado como nó de nível 1. Este valor impacta diretamente na quantidade de nós em um grupo ou mesmo na quantidade de grupos que possam se formar; quanto menor o valor mais rapidamente um nó pode ser membro de um grupo. A seção 7.4.2 explora a variação deste parâmetro.

- *Limite para* $pointsL2$: representa o valor limite de pontos extras, adquiridos a partir da permanência, que um nó deve possuir para ser elevado ao nível 2. Vale lembrar que parte do requisito para ingressar no nível 2 é $contactPermanence \geq pointsL1 + pointsL2 \wedge trust$. O valor de 5 pontos foi escolhido por ser o mesmo limite mínimo de $pointsL1$; ao somá-los, tem-se o dobro de tempo de permanência exigido do nível 1 para o nó ingressar no nível 2 (sem considerar a variação do limite para $pointsL1$).

- *Limite para* $pointsL3$: é o valor limite de pontos maior que o $pointsL2$ para representar ainda mais tempo de permanência de um nó no grupo. Este parâmetro é parte do requisito para o nó ingressar no nível 3. O valor de 10 pontos foi escolhido por ser o mesmo limite mínimo de $pointsL2$; ao somá-lo com $pointsL1$, tem-se o triplo de tempo de permanência exigido do nível 1 para o nó ingressar no nível 3 (também sem considerar a variação do limite para $pointsL1$).

7.4.1 Avaliação de nós maliciosos

O objetivo desta avaliação é verificar qual é o impacto em cada nível, considerando a variação da quantidade de nós maliciosos. Estes nós são considerados maliciosos desde o início até o final da simulação, ou seja, eles não mudam o comportamento. O resultado auxiliará o implementador a decidir quais serviços podem ser atribuídos aos níveis, dependendo da variação de nós maliciosos. A Tabela 7.3 mostra os parâmetros usados nesta avaliação.

Tabela 7.3: Variação da quantidade de nós maliciosos.

Parâmetro	**Valor**
Quantidade de nós	150, 300 e 450
Quantidade de nós maliciosos	10%, 20%, 30% e 40%
Quantidade de nós confiáveis por rede social	20%
Limite para confiança	1000 pontos
Limite para permanência	10 segundos
Limite para $pointsL1$	5 pontos
Limite para $pointsL2$	5 pontos
Limite para $pointsL3$	10 pontos

A avaliação de nós maliciosos varia a porcentagem de 10%, 20%, 30% e 40% desses nós nos grupos com 150, 300 e 450 nós. As Figuras 7.3, 7.4 e 7.5 mostram os resultados das avaliações. A Figura 7.3 possui os gráficos *a*, que representa os testes com 150 nós; *b*, que representa os testes com 300 nós; e *c*, que representa os testes com 450 nós. Essa variação ocorre nas demais figuras desta avaliação. Independentemente da variação de números de nós, é possível notar que conforme aumenta a porcentagem de nós maliciosos, diminui a quantidade de nós nos níveis em cada grupo.

O comportamento obtido nesta avaliação ocorre porque quando um nó é classificado como malicioso, é excluído do grupo. O nível 1 não sofre este impacto porque ele não requer confiança para a entrada do nó no nível 1, somente contato e permanência. Isto é esperado, pois ao nível 1 se atribui apenas serviços que não requerem alto nível de segurança, como distribuição de mensagens. Vale ressaltar que, embora a métrica de confiança não seja analisada para o nó entrar no nível 1, se o nó se comportar de forma maliciosa, ele será excluído do grupo, independentemente do seu nível.

Os resultados apresentados na Figura 7.3 apresentam a quantidade máxima de nós por grupo, alcançada em algum momento durante a simulação. Estes resultados são restritos a existência dos grupos, considerando sua alta dinamicidade, ou seja, os grupos são criados e desfeitos com muita frequência ao longo do tempo.

Neste gráfico, é possível observar que a quantidade máxima de nós no nível 1 por grupo não sofreu muito impacto com o aumento da porcentagem de nós maliciosos para 150 veículos, devido à baixa densidade de nós na rede. Porém, este impacto aconteceu de forma mais acentuada nos testes com 300 e 450 veículos. É possível observar que com maior densidade, maior será o número de nós maliciosos e, com isso, menos nós comporão os grupos; consequentemente, haverá menos nós em cada nível.

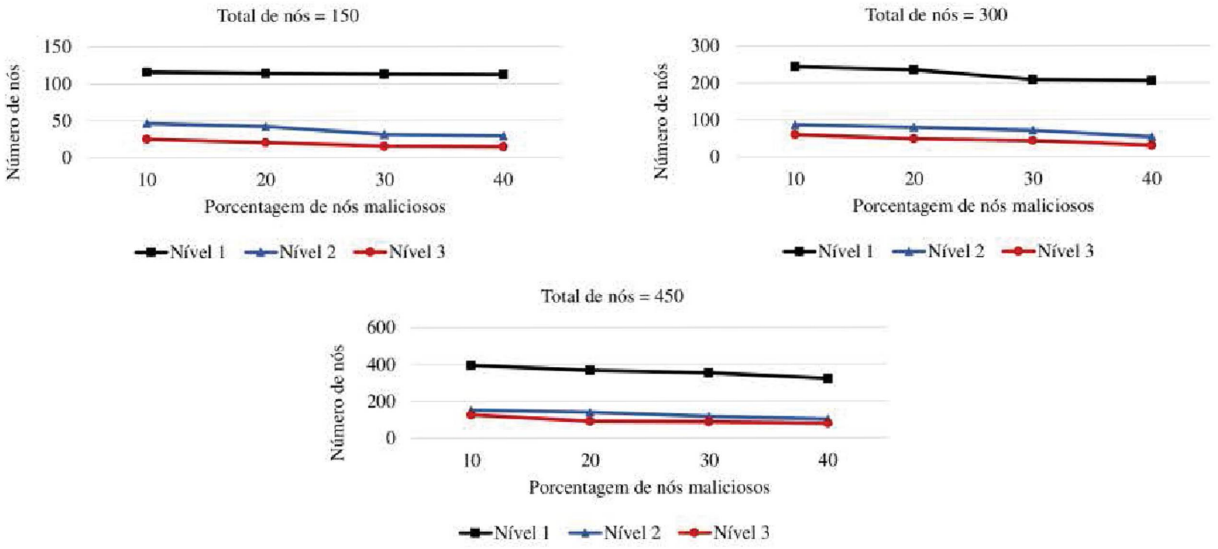


Figura 7.3: Impacto da variação de nós maliciosos nos grupos multiníveis.

A Figura 7.4 apresenta o impacto da variação dos nós maliciosos, sem a restrição de grupo, em todo universo de nós; ou seja, o quanto a presença dos nós maliciosos pode afetar a elevação de níveis dos nós, independentemente de grupos. Este resultado mostra que todos os nós fizeram parte de grupo e alcançaram o nível 1 em algum momento da simulação. Dessa forma, é possível deduzir que este *Framework* alcançou uma cobertura de 100% dos veículos em algum momento.

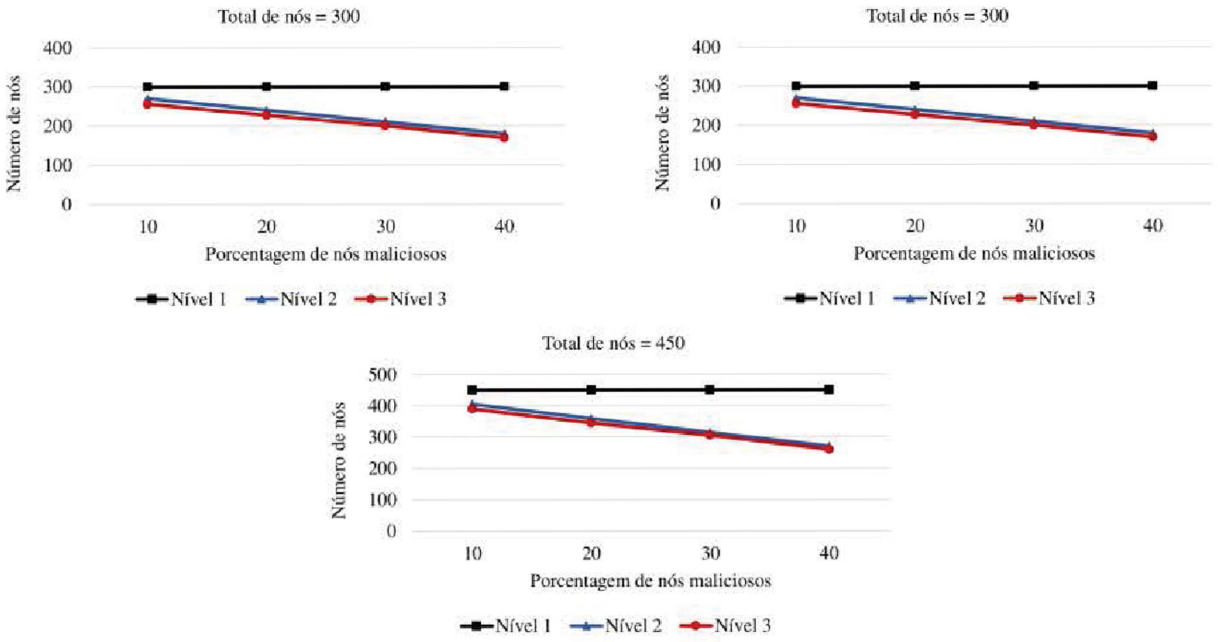


Figura 7.4: Impacto da variação de nós maliciosos no total de nós.

Vale observar que o nó malicioso entrará em um grupo como nível 1, porém ele será removido assim que for identificado como malicioso pelos demais membros do grupo; não fará parte dos níveis superiores. Nesta Figura também é possível observar que conforme aumenta a quantidade de nós maliciosos na rede, menor é o número de nós nos níveis 2 e 3, pois os

nós maliciosos pertencentes ao nível 1 serão excluídos do grupo tão breve quanto possível; consequentemente, haverá menos nós para elevação de nível.

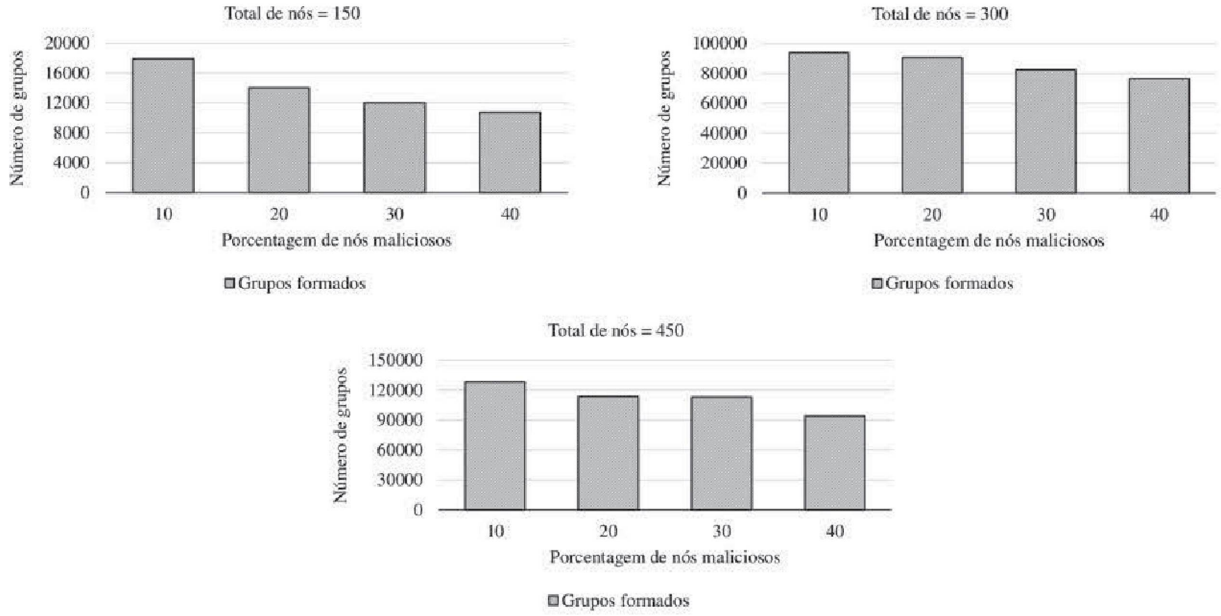


Figura 7.5: Impacto da variação de nós maliciosos na formação de grupos.

A Figura 7.5 mostra a variação da quantidade de grupos formados, alternando a porcentagem dos nós maliciosos. Neste *Framework*, cada alteração realizada no grupo será considerada como um grupo novo, ou seja, toda vez que um novo membro for adicionado ou um membro atual removido será realizada uma nova chave compartilhada de grupo, e este será considerado como um novo grupo. Assim, os valores obtidos nessa simulação podem ser também considerados como a quantidade de alterações que os grupos sofrem ao longo da simulação.

Nesta figura, é possível notar que conforme aumenta a porcentagem de nós maliciosos, diminui a quantidade de grupos formados na rede. Isto acontece porque mesmo que inicialmente um nó malicioso forme um grupo como nó de nível 1, ele será rapidamente identificado por seu mau comportamento, será removido do grupo e rotulado como malicioso pelos demais membros em suas tabelas locais. Assim, nas próximas interações, ele não fará mais parte de grupos com os nós que já o conhecem, impactando o surgimento de novos grupos ou mesmo nas alterações dos grupos existentes por sua admissão e, consequentemente, na geração de novo grupo, visto que a mudança da chave compartilhada resulta em novo grupo.

Por este comportamento, nota-se que novos grupos deixam de ser formados, dado que um nó não malicioso não forma grupo com nós maliciosos. A permanência não foi considerada nestes testes, pois a variação dos nós maliciosos não impacta no tempo em que cada nó permanece conectado com outros nós. Este resultado confirma o correto funcionamento da formação de grupos na presença dos nós maliciosos.

7.4.2 Avaliação dos limites para $pointsL1$

Esta avaliação visa a mensurar os impactos da variação da métrica $pointsL1$ nas quantidades de nós nos níveis e grupos. As métricas para subir de nível foram apresentadas na Tabela 1. Esta variação está relacionada ao limite necessário para os nós serem convidados a subir de nível, dado que o $pointsL1$ é adicionado ao $pointsL2$ ou $pointsL3$ para elevação ao nível 2 e 3, respectivamente, considerando as fases do grupo. Com a elevação do valor de

$pointsL1$, maior será a pontuação de $contactPermanence$ que o nó deve alcançar. A Tabela 7.4 mostra os parâmetros usados nesta avaliação.

Tabela 7.4: Variação dos valores limites para $pointsL1$.

Parâmetro	**Valor**
Quantidade de nós	150, 300 e 450
Quantidade de nós maliciosos	10%
Quantidade de nós confiáveis por rede social	20%
Limite para confiança	1000 pontos
Limite para permanência	10 segundos
Limite para $pointsL1$	2, 4, 6, 8 e 10 pontos
Limite para $pointsL2$	5 pontos
Limite para $pointsL3$	10 pontos

Vale relembrar que na fase inicial do grupo não há nós nos níveis superiores, eles são atribuídos pelos demais nós por seu bom comportamento e tempo de permanência, mediante consenso. Com a evolução do grupo, novos membros surgem e todos os níveis serão completos, com isso, os serviços do *Framework* são melhor distribuídos entre os níveis, como apresentado na Tabela 6.1.

É importante ressaltar que o $contactPermanence$ é o resultado da computação dos contatos e permanência entre os nós de um mesmo grupo, ou seja, quanto mais contatos e tempo conectados maior será o valor de $contactPermanence$, e o contrário também ocorre. O contato é contabilizado a partir dos traços da carga de trabalho. Ele representa uma atividade simulada da vida real e não há como alterar este parâmetro nesta simulação.

A variação do $pointsL1$ é reportada nas Figuras 7.6, 7.7, 7.8, 7.9, 7.10. O objetivo desta variação é verificar o impacto da escolha do $pointsL1$ em cada nível, para decisão dos pontos limites. Este resultado auxiliará o implementador a decidir o melhor valor para atribuir a $pointsL1$, ou seja, demonstrar como a variação do valor de $pointsL1$ influencia na quantidade de nós em cada nível.

A Figura 7.6 mostra a variação do $pointsL1$ por nível no grupo. É possível perceber que conforme aumenta o valor de $pointsL1$ diminui a quantidade de nós nos níveis do grupo. No entanto, o aumento no valor do $pointsL1$ significa que os nós que formam grupos possuem mais contatos de conexão e tempo de conectividade. Se o objetivo for formar grupos com maior número de nós possíveis, independentemente das demandas do serviço por conectividade e contato, um valor baixo de $pointsL1$ pode ser o suficiente.

A Figura 7.7 mostra a variação do limite de $pointsL1$ por nível, independentemente de grupo. É possível observar que conforme aumenta o valor de $pointsL1$, o nível 1 permanece sem alteração, independentemente da quantidade de nós; o nível 2 diminui com 150 nós e permanece igual com 300 e 450 nós; e o nível 3 diminui com 150 e 300 nós, mas permanece igual com 450 nós. Isto acontece porque o nível 1 é facilmente alcançado com essa pontuação, de tal forma que todos os nós alcançam o nível 1, mesmo com a variação do $pointsL1$.

No entanto, nos casos com 150 e 300 nós, o acréscimo de $pointsL2$ e de $pointsL3$ ao valor de $pointsL1$ impactou na quantidade de nós. Com 150 nós, o impacto foi mais

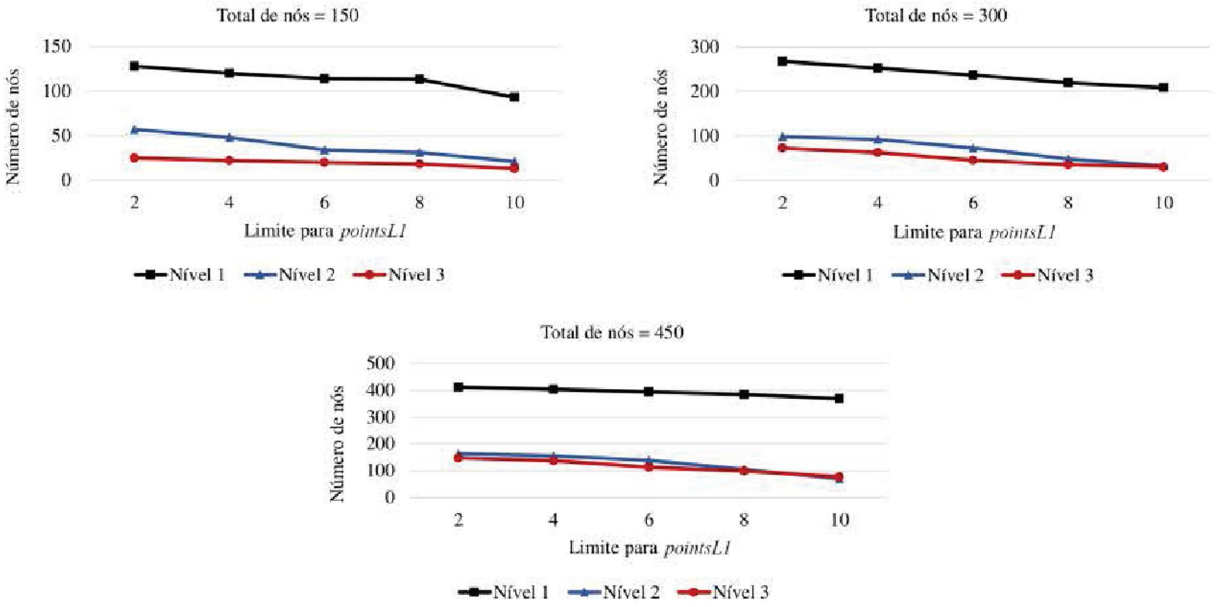


Figura 7.6: Impacto da variação do $pointsL1$ nos grupos multiníveis.

acentuado, porém, conforme aumenta a densidade dos veículos, este impacto é mitigado. Isto fica evidenciado no nível 2 com 300 nós e com 450 nós. Neste último caso, todos os níveis tiveram a mesma quantidade de nós. Todos os 450 nós, em algum momento, estiveram no nível 1 e apenas 405 nós alcançaram o nível 2 e 3, ou seja, excetuando os 10% (45 nós) maliciosos injetados neste teste, os demais alcançaram os níveis 2 e 3. Vale lembrar que o nível 1 não excetua os nós maliciosos como critério de entrada.

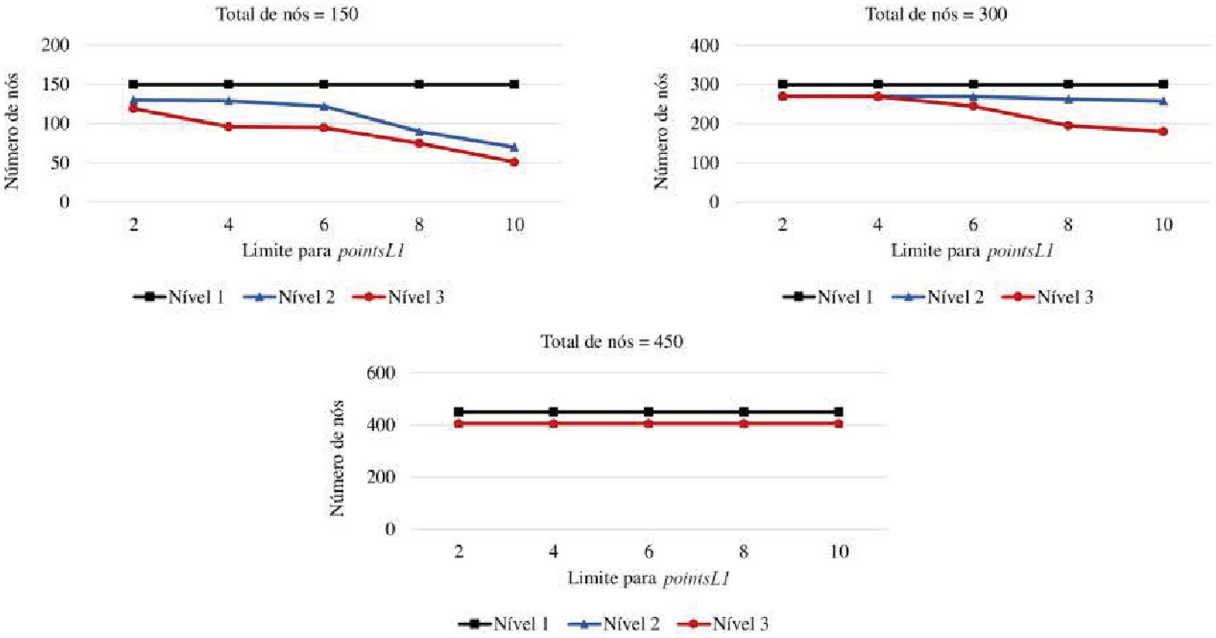


Figura 7.7: Impacto da variação do $pointsL1$ no total de nós.

A Figura 7.8 mostra a variação da quantidade de grupos formados, com variação do $pointsL1$ em 2, 4, 6, 8 e 10. Através desta figura, é possível notar que conforme aumenta o valor de $pointsL1$, aumenta a quantidade de grupos formados, independentemente da quantidade de

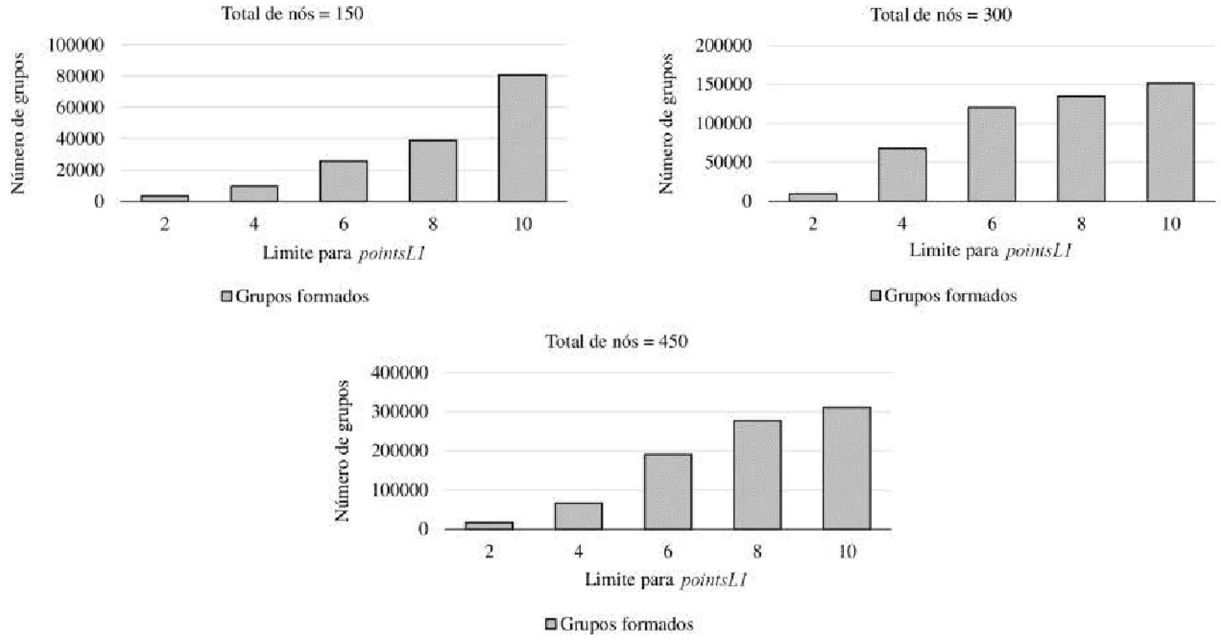


Figura 7.8: Impacto da variação do $pointsL1$ na formação de grupos.

nós na rede. Este comportamento acontece porque os nós precisam de mais pontos para alcançar o valor de $contactPermanence$ exigido, por isso, formam-se mais grupos com menos nós, porém, esses nós permanecem por mais tempo em um mesmo grupo. Este tempo de permanência pode ser observado na Figura 7.9. Nela está evidente como os nós ficam juntos por mais tempo em um mesmo grupo, devido à pontuação requerida.

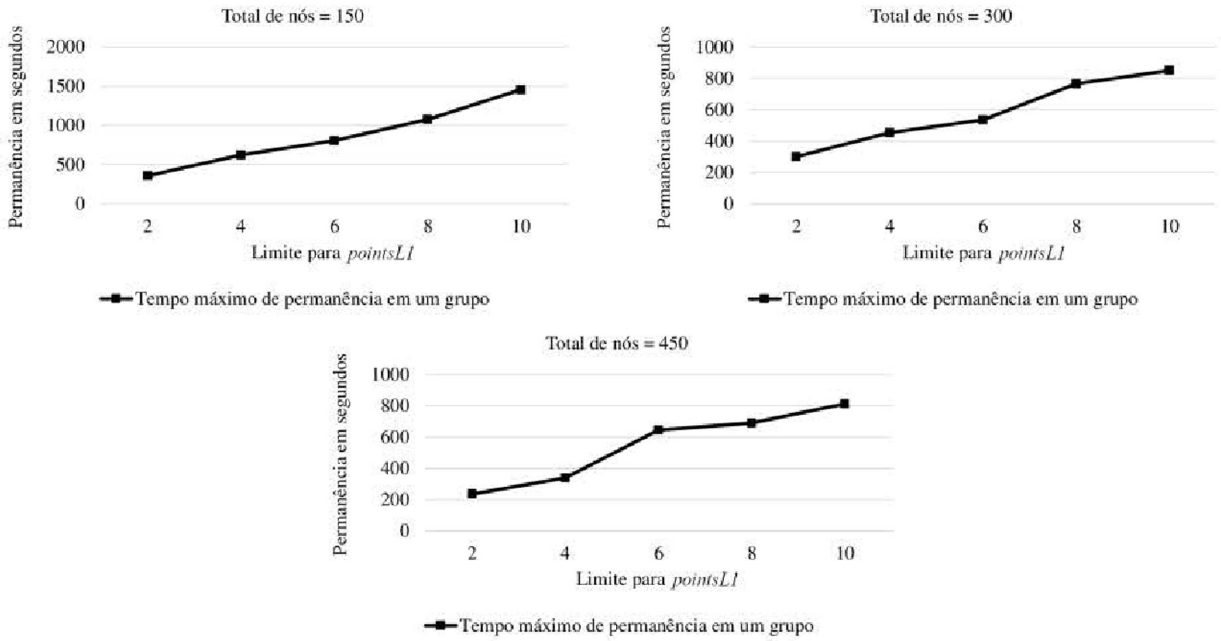


Figura 7.9: Impacto da variação do $pointsL1$ na permanência dos nós no grupo.

A Figura 7.10 apresenta a variação da quantidade de nós dentro de um mesmo grupo, em relação à variação do valor limite para $pointsL1$, somente do nível 1. É possível observar que conforme aumenta o valor para atingir o limite de $pointsL1$, diminui a quantidade de nós em um grupo. Note que este parâmetro é o requisito inicial para que um nó participe do grupo no nível 1. Dentre os limites estabelecidos, ele é o menor e, com isso, resulta sempre em um grande número de nós por grupo. A partir dele, os outros limites, como permanência e confiança, aumentarão o critério e afunilarão a quantidade de nós nos demais níveis.

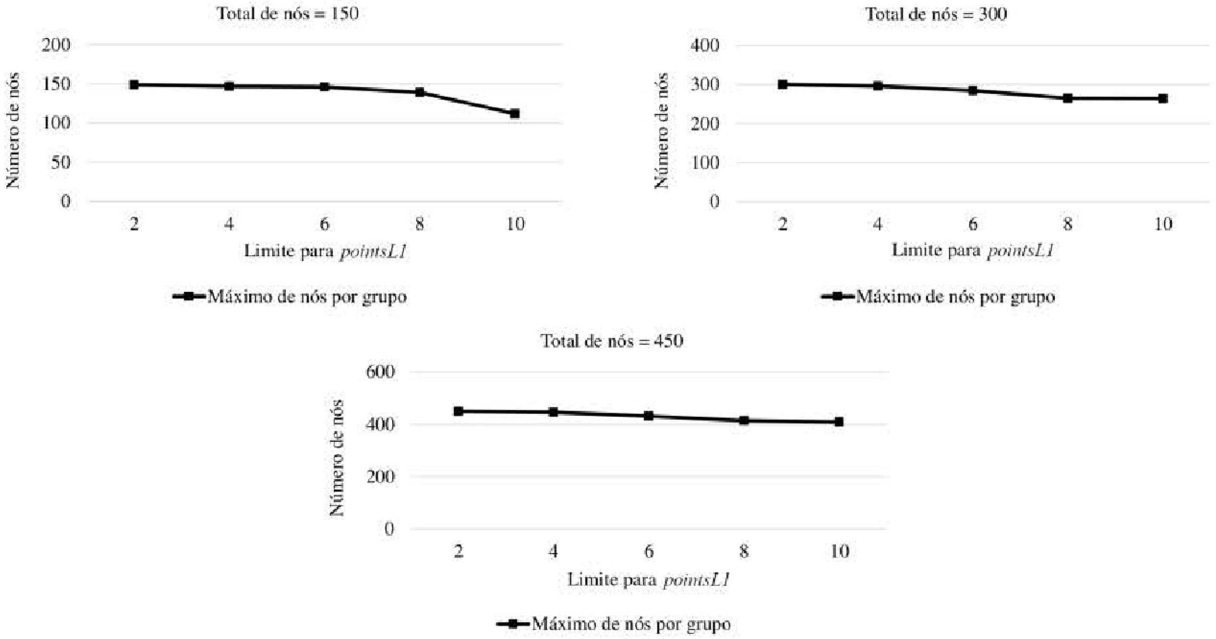


Figura 7.10: Impacto da variação do $pointsL1$ na quantidade de nós no grupo.

7.4.3 Avaliação dos limites para permanência

Ao contrário da pontuação apenas por contato, a pontuação por permanência pode ter seu limite variado. O implementador deve atribuir um limite de tempo necessário para pontuar a permanência no valor de $contactPermanence$, seja para incrementar, se conectado, ou decrementar, se desconectado. A variação do parâmetro limite de permanência é demonstrada nas Figuras 7.11, 7.12, 7.13, 7.14 e 7.16. A Tabela 7.5 mostra os parâmetros usados nesta avaliação.

As Figuras 7.11, 7.12, 7.13, 7.14, 7.15 apresentam a variação do tempo em segundos, para incrementar ou decrementar a pontuação dos nós. Esta variação foi testada em 10, 20, 30, 40, 50 e 60 segundos. Nessa simulação, este parâmetro pode ser entendido como tempo de permanência, apenas permanência ou por *trustTime*, nome dado na implementação da simulação.

A Figura 7.11 apresenta a quantidade de nós nos níveis, em um grupo variando os segundos para os ambientes com 150, 300 e 450 nós. Nestas imagens, é possível verificar que conforme aumenta o *trustTime*, diminui a quantidade de nós nos grupos. Os nós nos níveis 2 e 3 apresentaram menor número de nós no nível em relação ao nível 1, devido ao tempo de permanência exigido.

Tabela 7.5: Variação do tempo limite para obtenção de pontos por permanência.

Parâmetro	**Valor**
Quantidade de nós	150, 300 e 450
Quantidade de nós maliciosos	10%
Quantidade de nós confiáveis por rede social	20%
Limite para confiança	1000 pontos
Limite para permanência	10, 20, 30, 40, 50 e 60 segundos
Limite para $pointsL1$	5 pontos
Limite para $pointsL2$	5 pontos
Limite para $pointsL3$	10 pontos

Para exemplificar: a variação do parâmetro tempo para 60 segundos significa que o nó irá obter um ponto a cada 60 segundos; considerando as métricas de elevação de nível e os parâmetros dessa simulação (nível 1 exige 5 pontos, o nível 2 exige 10 pontos e o nível 3 exige 15 pontos), o nó confiável deverá permanecer no mínimo 15 minutos em um mesmo grupo para atingir o nível 3. Como a formação do grupo é muito dinâmica e a quantidade de veículos está limitada a 450, são poucos os casos reportados na simulação de veículos que alcançam esta pontuação. Isto mostra que o *Framework* atua de forma correta ao filtrar a permanência para elevação de nível, mesmo que isto ocasione a existência de poucos nós neste nível.

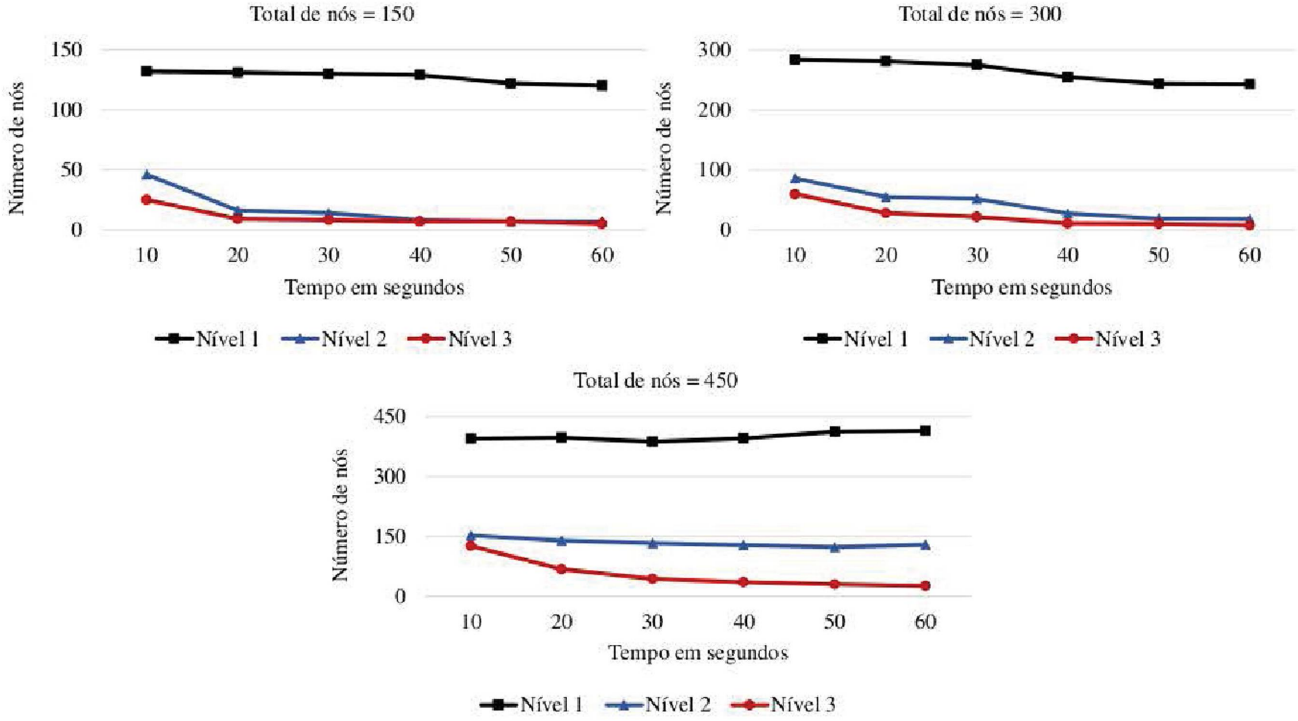


Figura 7.11: Impacto da variação dos segundos nos grupos multiníveis.

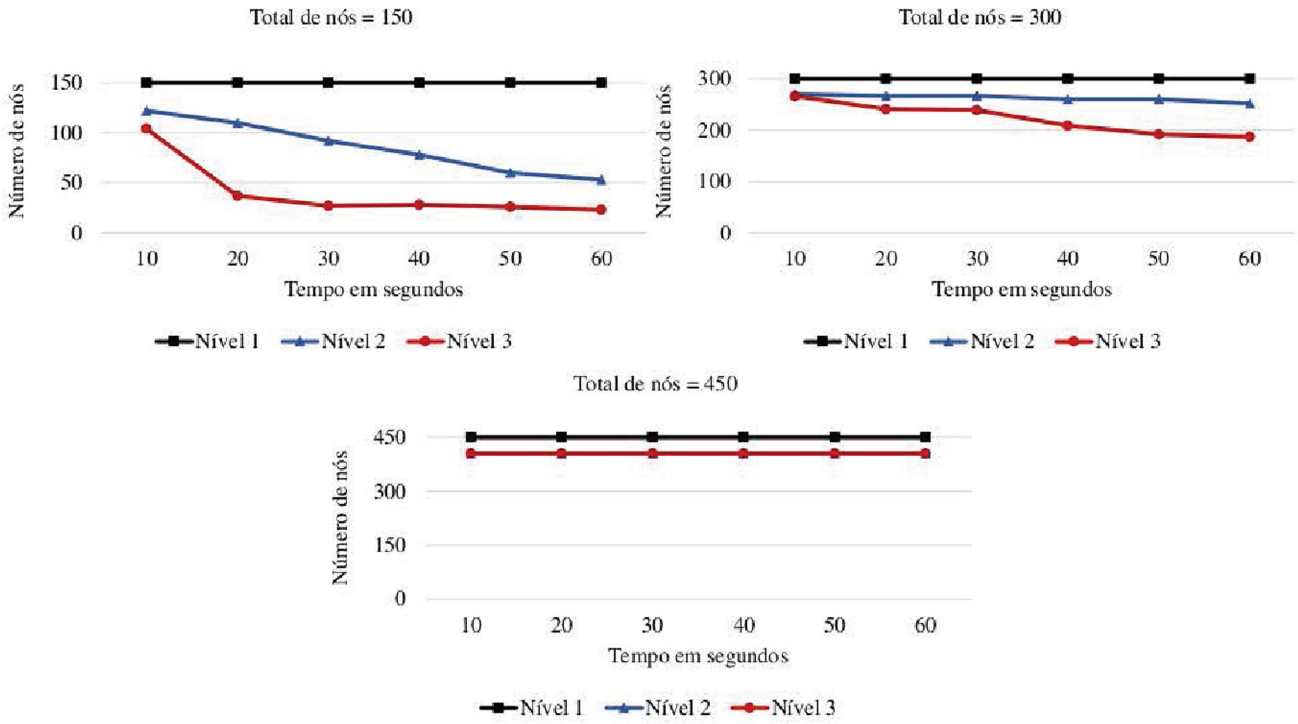


Figura 7.12: Impacto da variação dos segundos no total de nós.

A Figura 7.12 refere-se apenas à quantidade de nós sem a restrição de pertencer a grupos. Percebe-se que a variação do tempo no nível 1 foi indiferente; no nível 2 e no nível 3, proporcionou uma diminuição na quantidade de nós, considerando 150 e 300; e foi indiferente no ambiente de 450. Isto acontece devido ao impacto da densidade dos nós nos níveis do *Framework*, pois se há muitos veículos no sistema, esta variação de pontos se torna efetiva. Quando a densidade de nós é elevada, há muitos contatos de conexão, os nós tendem a permanecerem mais tempo conectados, resultando em fácil alcance dos pontos exigidos; então todos os veículos alcançam os níveis, exceto os maliciosos.

Esta Figura mostra a variação dos segundos na visão dos nós. Nela, é possível ver que no nível 1 não há diferença nos segundos, devido à pontuação do $pointsL1$; porém nos níveis superiores, em que o $pointsL1$ é somado com o $pointsL2$ e $pointsL3$, gerando um valor maior, a variação dos segundos influencia na quantidade de nós. Nesta figura, é possível observar que todos os nós alcançam o nível 1; isto ocorre porque para o ingresso no nível 1 não é verificado o comportamento do nó. Porém, quando o nó está no nível 1 e o grupo está na fase 1, é feita a verificação do comportamento do nó, e se ele agir com mau comportamento, será excluído do grupo.

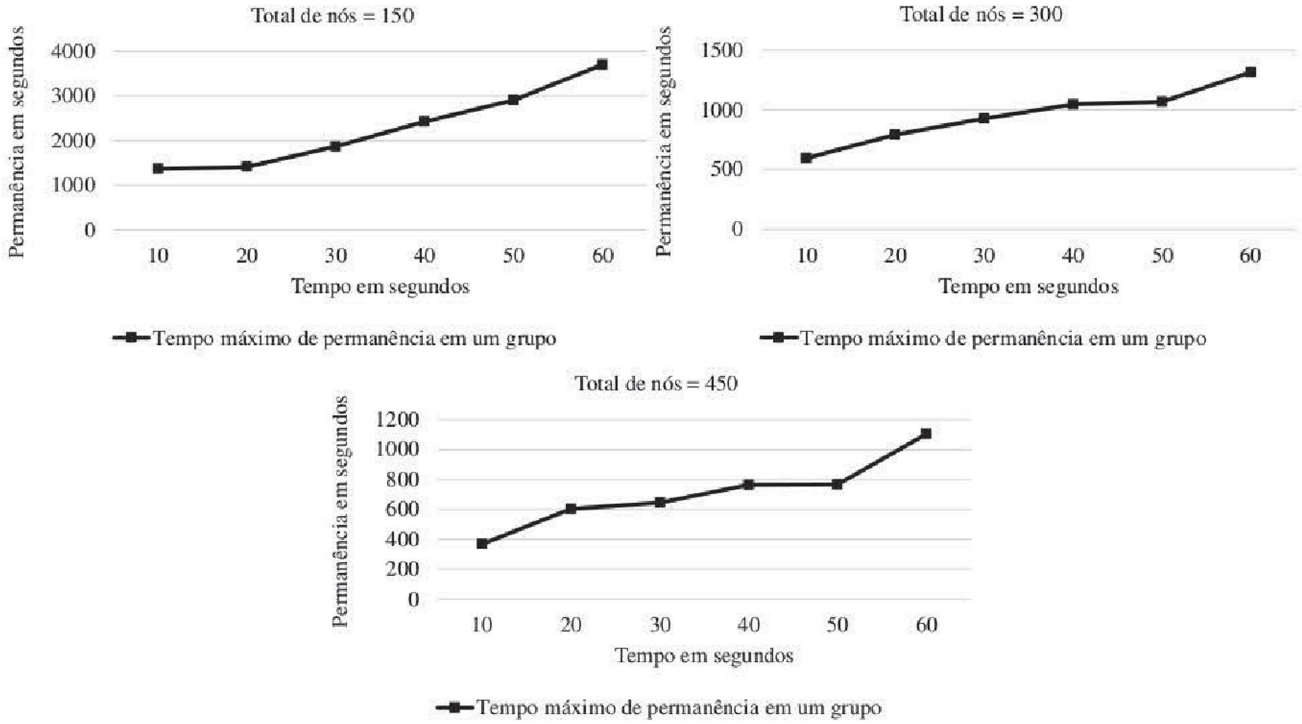


Figura 7.13: Impacto da variação dos segundos na permanência em um grupo.

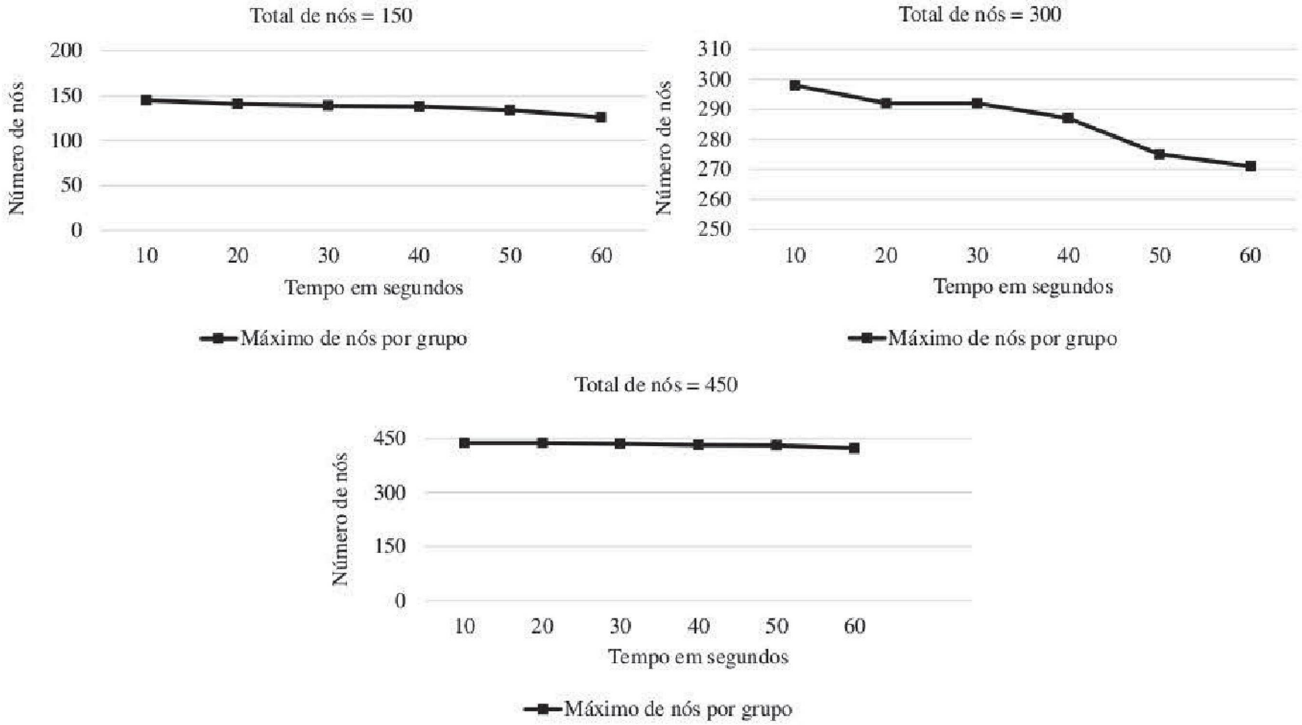


Figura 7.14: Impacto da variação dos segundos na formação de grupos.

A Figura 7.13 mostra a permanência dos nós no grupo, variando os segundos. Através desta simulação, é perceptível que à medida que aumentam os segundos, aumenta a permanência dos nós no grupo. A Figura 7.14 mostra que quando aumentam os segundos, diminui a quantidade de nós no grupo; e a Figura 7.15 mostra que, sob as mesmas circunstâncias, diminui a quantidade de grupos, a quantidade de nós em um grupo, e aumenta a permanência destes nós no grupo.

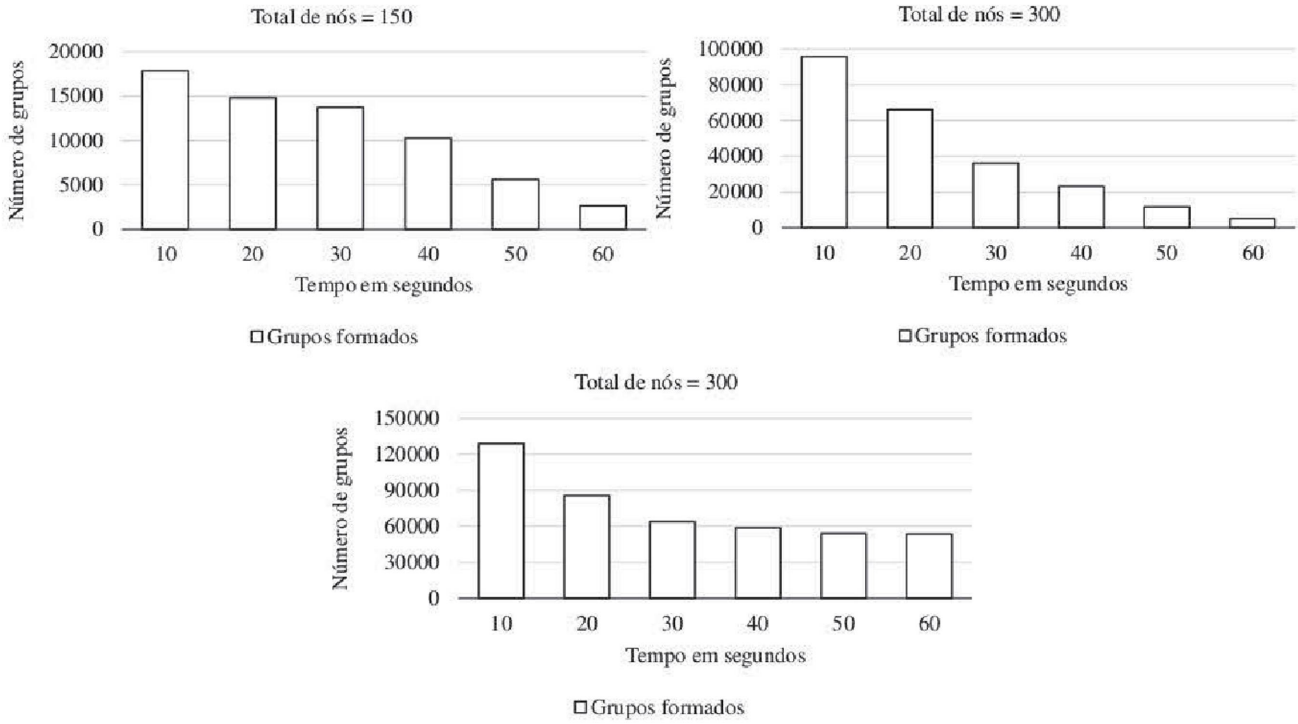


Figura 7.15: Impacto da variação dos segundos na quantidade de nós no grupo.

7.4.4 Avaliação dos limites para confiança

O objetivo desta avaliação é demonstrar o comportamento do *Framework* variando as configurações de nível de confiança (pontos de confiança). Nesta avaliação, foi analisado o máximo de nós em um grupo e o máximo de nós sem a restrição do grupo, para traços de entrada com 150, 300 e 450 nós. Os valores dos resultados são obtidos por meio da variação do ponto de confiança de 1000, 5000, 10000, 15000 e 20000.

Tabela 7.6: Variação dos pontos limites para obtenção da confiança.

Parâmetro	**Valor**
Quantidade de nós	150, 300 e 450
Quantidade de nós maliciosos	10%
Quantidade de nós confiáveis por rede social	20%
Limite para confiança	1000, 5000, 10000, 15000 e 20000 pontos
Limite para permanência	10 segundos
Limite para $pointsL1$	5 pontos
Limite para $pointsL2$	5 pontos
Limite para $pointsL3$	10 pontos

Vale observar que a confiança é representada pela variável booleana *trust*. Ela indica se um nó é confiável ou não. Os pontos de confiança são os parâmetros usados para alcançar o *trust*; em outras palavras, os pontos necessários para que o nó seja considerado confiável. A Tabela 7.6 mostra os parâmetros usados nesta avaliação. As Figuras 7.16 e 7.17 mostram os resultados. O título do eixo *x* destes gráficos refere-se aos pontos necessários para alcançarem o *trust*.

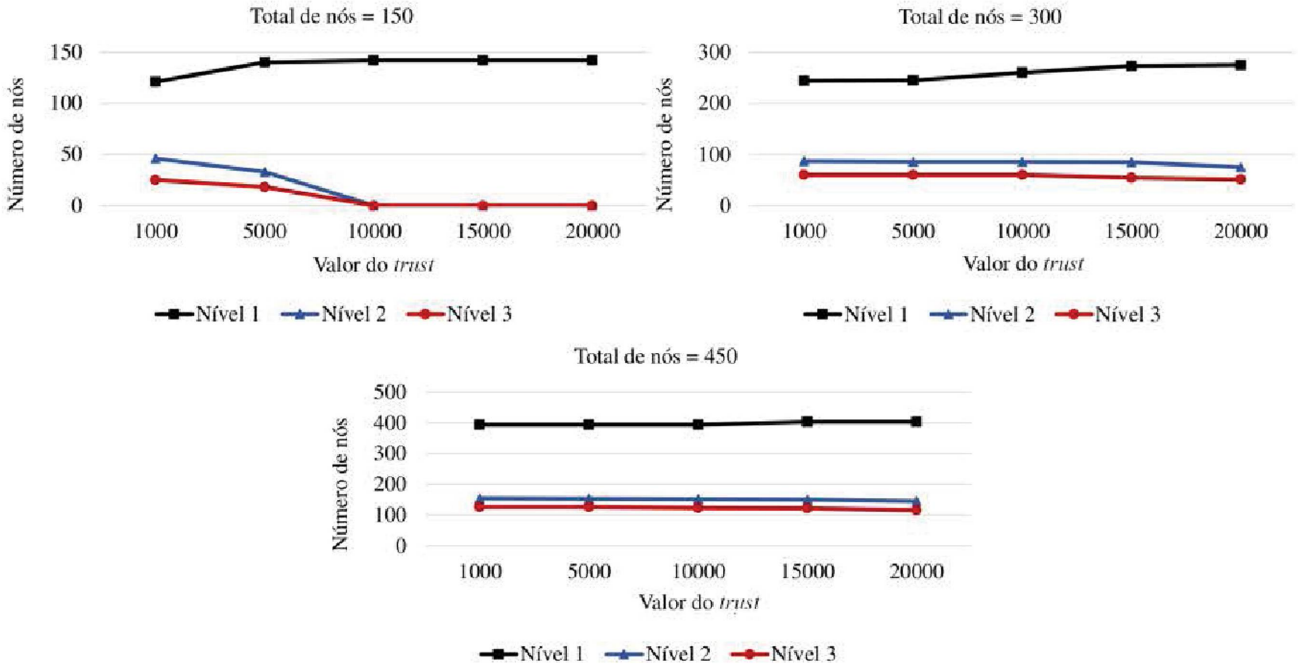


Figura 7.16: Impacto da variação do *trust* nos grupos multiníveis.

Na Figura 7.16, percebe-se que quando aumenta o valor de *trust*, tem-se um leve aumento de nós no grupo no nível 1, pois não há exigência da métrica confiança, e que os níveis 2 e 3 possuem uma queda na quantidade de nós nos ambientes com 150 e 300 nós. Este valor de *trust* não influenciou na quantidade de nós no grupo considerando a simulação com 450 nós. Isso se justifica pelo fato de que em um ambiente com muitos nós, cuja maioria tem bom comportamento, o valor de confiança é incrementado rapidamente e se mantém, o que influencia em não haver variações significativas na quantidade de níveis dos nós de um grupo.

Na Figura 7.17, é possível notar que não houve diferença no nível 1, isso é totalmente esperado porque o nível 1 não requer confiança para o nó entrar no grupo. Nos níveis 2 e 3, para os ambientes com 150 e 300 nós, na medida em que aumenta o valor de *trust*, há uma diminuição na quantidade de nós no grupo. No ambiente com 450 nós não houve diferença na quantidade de nós no grupo. Observa-se, então, que para um mesmo ambiente, com poucos nós, houve um grande impacto na quantidade de nós no grupo, porém, à medida que a quantidade de nós no ambiente aumentou, este impacto diminuiu, chegando ao ponto de ser irrelevante.

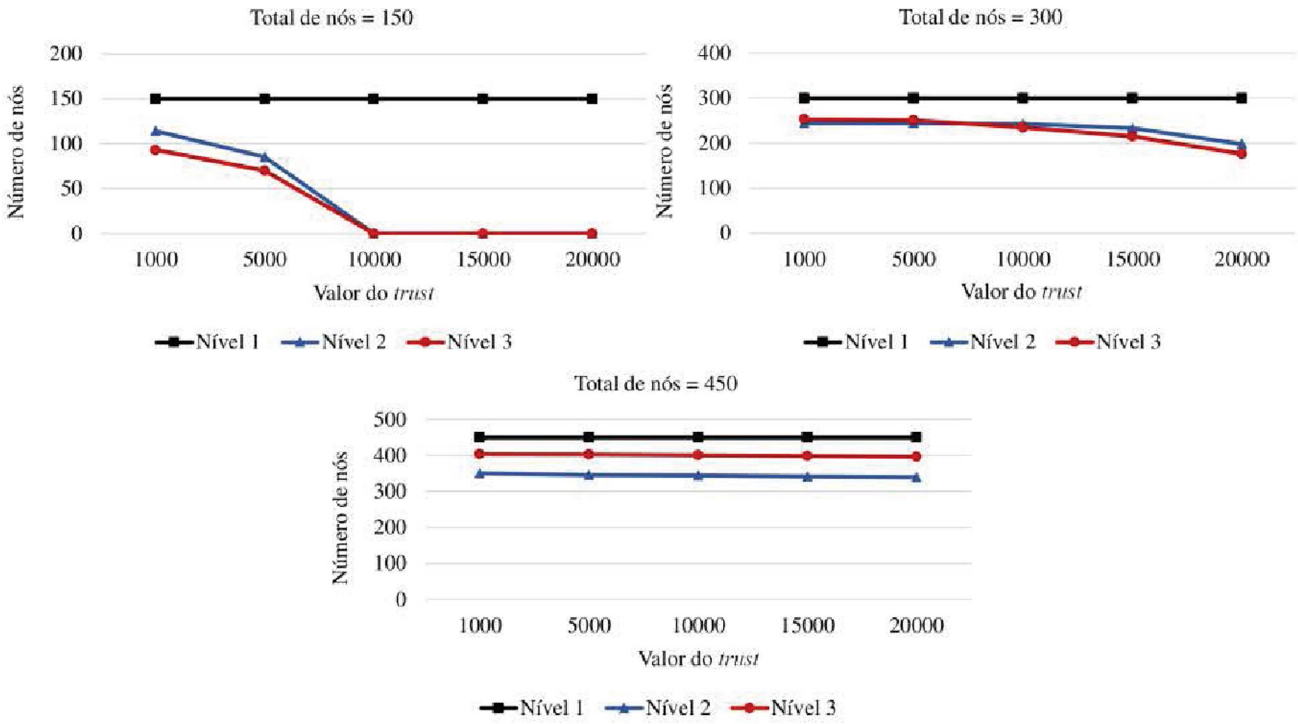


Figura 7.17: Impacto da variação do *trust* no total de nós.

7.4.5 Avaliação dos limites de confiança por rede social

O objetivo desta avaliação é demonstrar o comportamento do *Framework*, variando as configurações de confiança por rede social. Nesta avaliação foi analisado o máximo de nós em um grupo, o máximo de nós sem a restrição do grupo, o máximo de veículos por grupo, o máximo de permanência por grupo e o número máximo de grupos para traços de entrada com 150, 300 e 450 nós. Os valores dos resultados são obtidos variando a porcentagem de rede social em 0, 5, 10, 15 e 20%.

Tabela 7.7: Variação da % de confiança obtida por rede social.

Parâmetro	**Valor**
Quantidade de nós	150, 300 e 450
Quantidade de nós maliciosos	10%
Quantidade de nós confiáveis por rede social	0, 5, 10, 15 e 20%
Limite para confiança	1000 pontos
Limite para permanência	10 segundos
Limite para $pointsL1$	5 pontos
Limite para $pointsL2$	5 pontos
Limite para $pointsL3$	10 pontos

Os resultados para todos os testes com 150 nós, variando a porcentagem de confiança por rede social, não demonstraram diferença, ou seja, foram idênticos. Isso é esperado, pois

se o nó não conhece o outro via perfil de rede social, ele obtém a confiança via pontuação. A confiança por rede social é apenas uma alternativa para atingir a confiança de forma mais rápida. O mesmo aconteceu nos demais testes, com 300 e 450 nós.

Vale ressaltar que o foco da avaliação é a quantidade dos nós nos níveis e não a velocidade em que os nós atingem estes níveis. Dessa forma, a rede social deve ser vista como um atalho para os nós avançarem do nível 1 para o nível 2, não impactando em seus totais.

7.4.6 Avaliação da segurança

A avaliação de segurança visa mensurar o número de mensagens necessárias para estabelecer uma chave compartilhada entre os nós de um grupo, considerando os diferentes níveis e densidade; e medir o tempo de processamento necessário para cifrar, decifrar, gerar e verificar a etiqueta do criptograma. A implementação foi realizada instrumentando os arquivos fontes originais da biblioteca MIRACL, e por meio da adição de marcas do tempo de início e fim para a execução de cada método avaliado. Para as trocas de mensagens, foram utilizadas passagens de valor por parâmetros nas chamadas das funções.

7.4.6.1 Chave compartilhada

A geração da chave simétrica compartilhada foi implementada com o criptossistema ECIES, curva elíptica $secp256r1$ e cifra simétrica AES-CBC de 128 *bits*. O custo para estabelecer essa chave simétrica entre os nós é a quantidade de mensagens necessárias trocadas entre eles e o tempo de processamento das funções que implementam a geração da chave. A Figura 4.4 mostra os passos do ECIES, dos quais os principais métodos implementados para esta avaliação são:

- GEN(u,U): esta função implementa a geração das chaves efêmeras pública e privada. Ela é executada em momento prévio ao estabelecimento de chave compartilhada de grupo. Cada nó gera seu par de chaves efêmeras assim que possível.
- KA(u,V): esta função implementa o acordo de chave entre dois nós, em que o nó remetente realiza uma multiplicação escalar sob aritméticas de corpos finitos com sua chave efêmera privada e a chave pública do destinatário (as chaves são pontos da curva elíptica), resultando na chave secreta a ser compartilhada no grupo. Neste ponto, os nós já conhecem suas chaves públicas, obtidas nos primeiros contatos. Vale ressaltar que as trocas de chaves podem ser realizadas de forma segura com o ECMQV.
- KDF: esta função implementa a derivação de chave secreta compartilhada em outras duas chaves, para cifrar/decifrar e gerar etiqueta. Cada nó está apto a realizar este método; no entanto, há a possibilidade de implementar a distribuição no grupo das duas chaves geradas, ao invés de uma única chave a ser consumida pela KDF, evitando o mesmo processamento, porém com o ônus de uma mensagem maior.
- ENC/DEC: estas funções foram implementadas com o AES. Elas realizam a cifragem e decifragem da mensagem a partir do texto plano e da chave de criptografia gerada pela KDF.
- MAC: esta função é implementada com o algoritmo de *Secure-Hash* de 256 *bits* (SHA-256). O retorno dela é o resumo gerado a partir da mensagem cifrada.

Para calcular o custo de processamento, foram realizadas 10 rodadas de testes, variando-se a semente das funções geradoras de chave que contêm pseudoaleatoriedade com mensagens de

1 B a 32 KB. O tempo de processamento médio foi obtido pela média geométrica das rodadas de teste. O custo da cifragem por meio do sistema ECIES (funções KA + KDF + ENC + MAC) foi de $3,150ms$, enquanto o custo da decifragem (KA + KDF + DEC + MAC) foi de $2,322ms$. Após a chave compartilhada ser estabelecida no grupo, serão usadas apenas as funções de cifragem e decifragem do AES (AES + MAC). De forma similar, o custo da cifragem e decifragem com o AES também foi mensurado. A média de custo para a cifragem foi $1,538ms$ e para a decifragem foi $1,524ms$. O objetivo desta análise não é otimizar ou avaliar o desempenho de sistemas de criptografia, apenas apresentar o custo para sua utilização.

O custo do tempo para transmitir uma mensagem é variável, seja por comunicação *unicast* ou *broadcast*, pois depende da quantidade de nós comunicantes no grupo, da posição dos nós (diâmetro da rede), do protocolo de roteamento e de diversas outras variáveis, sendo altamente afetado por qualquer alteração dessas variáveis. Por este motivo, esta avaliação apresentará a quantidade de mensagens necessárias para estabelecer a chave compartilhada de grupo, com diferentes cenários. As mensagens estão associadas ao tempo de processamento e os resultados totais serão apresentados em 3 diferentes cenários de atribuição de chave. Em todos os cenários, o nó A convida o nó B para integrarem um grupo; o nó B aceita o convite e existe o consenso favorável nos cenários 2 e 3. O mesmo custo de atribuição de chave compartilhada para formação de grupos se aplica ao conceito de níveis, pois é usado o mesmo criptossistema e a chave compartilhada é gerada da mesma forma, bem como a realização do consenso. Assim, os resultados para estes 3 cenários da formação de grupo refletem nos custos para atribuição de níveis.

No primeiro cenário, tanto o nó A quanto o nó B não possuem grupos, eles formarão um novo grupo entre eles. O número de mensagens e funções necessárias para este cenário são:

- Nó A: 1 mensagem para convite de admissão.
- Nó A: 1 cifra ECIES referente à mensagem de convite.
- Nó B: 1 decifra ECIES para o recebimento do convite de admissão.
- Nó B: 1 mensagem de resposta ao convite de admissão.
- Nó B: 1 cifra AES referente à resposta ao convite de admissão.
- Nó A: 1 decifra AES.

No primeiro cenário, o custo são 2 mensagens *unicast*, pois quando o nó B envia a resposta de aceite, ambos nós já possuem a mesma chave compartilhada e a mesma visão de grupo. Observe que o GSeF4V obtém vantagens do criptossistema ECIES ao usar a chave compartilhada simétrica para comunicação em grupo. O custo de processamento total é 8, 534 ms, desconsiderando o tempo das trocas de mensagens. Vale ressaltar que a comunicação pode ser realizada com abordagens de retransmissão de pacotes não recebidos, variando conforme o protocolo utilizado, e isto foge ao escopo deste *Framework*.

No segundo cenário, o nó A é membro de um grupo e convida o nó B para fazer parte deste grupo. Neste cenário, o número de mensagens e funções necessárias são:

- Nó A: 1 mensagem ao nó B para convite de admissão .
- Nó A: 1 cifra ECIES referente à mensagem de convite.
- Nó B: 1 decifra ECIES para o recebimento do convite de admissão.

- Nó B: 1 mensagem de resposta ao convite de admissão.
- Nó B: 1 cifra AES referente à resposta ao convite de admissão.
- Nó A: 1 decifra AES.
- Nó A: 1 cifra AES, usando a chave de grupo atual. Esta mensagem cifrada contém a solicitação de consenso dos membros do grupo, a nova chave de grupo e as informações do nó B.
- Nó A: 1 *broadcast* para $n - 1$ nós, onde n é o número de nós do grupo atual. Este *broadcast* é a solicitação de consenso.
- Nós membros atuais: $n - 1$ decifras AES.
- Nós membros atuais: $n - 1$ cifras AES contendo o voto.
- Nós membros atuais: $n - 1$ mensagens para o nó A, informando seu voto.
- Nó A: $n - 1$ decifras AES, para obter o resultado do consenso.
- Nó A: 1 cifra AES com a chave compartilhada atual com o resultado do consenso e o estabelecimento da nova chave como atual.
- Nó A: 1 *broadcast* para $n - 1$ nós, referente ao resultado do consenso.
- Nós membros atuais: $n - 1$ decifras AES.
- Nó A: 1 cifra AES com informações dos nós atuais do grupo.
- Nó A: 1 mensagem para o nó B com as informações dos membros e sua aceitação no grupo com chave já conhecida.
- Nó B: 1 decifra AES com informações recebidas sobre os membros do grupo.

Neste panorama, os custos são $3 * n$ mensagens, dos quais são 2 *broadcasts*, envio a todos os membros atuais, exceto a si mesmo, e $n + 2$ mensagens *unicast*. Para o cálculo do custo de processamento, foi considerado que os processamentos resultantes de mensagem *broadcast* foram realizados em paralelo. O custo de processamento total para estabelecer a chave compartilhada no grupo neste cenário, com admissão do nó B, foi de $17,725$ ms $+ n - 1$ decifras AES. É possível observar que houve um aumento considerável de tempo de processamento em relação ao cenário 1, devido à necessidade do consenso, tanto para a solicitação quanto para a informação do resultado. Este consenso é necessário para que todos os nós membros tenham a mesma visão do grupo.

O terceiro e último cenário apresenta a hipótese de o nó A possuir um grupo e o nó B possuir outro. Os nós mesclam os membros, formando um novo grupo entre todos. O número de mensagens e funções necessárias para este cenário são:

- Nó A: 1 mensagem ao nó B para convite de admissão.
- Nó A: 1 cifra ECIES referente à mensagem de convite.
- Nó B: 1 decifra ECIES para o recebimento do convite de admissão.

- Nó B: 1 mensagem de resposta ao convite de admissão, contendo informações de todos os membros atuais do grupo de B.
- Nó B: 1 cifra AES referente à resposta ao convite de admissão e informações.
- Nó A: 1 decifra AES.
- Nó A: 1 cifra AES, usando a chave de grupo atual. Esta mensagem cifrada contém a solicitação de consenso dos membros do grupo, a nova chave de grupo e as informações do nó B e dos membros do seu grupo atual.
- Nó A: 1 *broadcast* para $n - 1$ nós, em que n é o número de nós do grupo atual do nó. Este *broadcast* é a solicitação de consenso.
- Nós membros atuais do grupo de A: $n - 1$ decifras AES.
- Nós membros atuais do grupo de A: $n - 1$ cifras AES, contendo o voto.
- Nós membros atuais do grupo de A: $n - 1$ mensagens para o nó A, informando seu voto.
- Nó A: $n - 1$ decifras AES, para obter o resultado do consenso.
- Nó A: 1 cifra AES com a chave compartilhada atual com o resultado do consenso e o estabelecimento da nova chave como atual.
- Nó A: 1 *broadcast* para $n - 1$ nós, referente ao resultado do consenso.
- Nós membros atuais do grupo de A: $n - 1$ decifras AES.
- Nó A: 1 cifra AES com informações dos nós atuais do grupo de A.
- Nó A: 1 mensagem para o nó B com as informações dos membros e sua aceitação no grupo com chave já conhecida.
- Nó B: 1 decifra AES com informações recebidas sobre os membros do grupo.
- Nó B: 1 cifra AES, usando a chave de grupo atual. Esta mensagem cifrada contém o resultado do consenso para a sua admissão e dos membros do seu grupo, a nova chave de grupo e as informações de todos os membros de A.
- Nó B: 1 *broadcast* para seus membros atuais com as novas informações do novo grupo.
- Nós atuais do grupo de B: decifra das novas informações de B.

No último cenário, a quantidade de mensagens totais são: $4 * N - 1$, pois houve a necessidade de 1 *broadcast* adicional em relação ao segundo cenário. Este *broadcast* é responsável por atualizar os membros do grupo de B, para que todos tenham a mesma visão de grupo, com a mesma chave compartilhada no novo grupo. O custo de processamento foi $20,788$ ms $+ n - 1$ decifras AES, custo também mais elevado do que o do cenário anterior, pelo mesmo motivo do aumento das mensagens, para que todos os membros do grupo de B pudessem ter acesso às novas informações, e os membros do grupo de A e os do grupo de B a mesma visão e chave compartilhada de grupo. Vale ressaltar que, nesta implementação, o nó B decidiu pelo aceite em nome de todos os membros.

Além da adição de membro, a remoção de um nó do nível ou grupo também impacta o sistema quanto ao número de mensagens e tempo de processamento. Estes custos estão ligados diretamente à geração de uma nova chave compartilhada no grupo e informação aos demais nós do grupo. Para ilustrar essa remoção, considera-se que o nó A deseja remover o nó B do grupo ou nível e que há o consenso a favor da remoção. A quantidade de mensagens e funções necessárias são:

- Nó A: $n-2$ mensagens de consenso. Apenas o nó A e B não receberão essas mensagem.
- Nó A: $n-2$ cifras AES da mensagem de solicitação de consenso para remover o nó B.
- $n-2$ decifras AES para recuperar a mensagem de solicitação de consenso.
- $n-2$ cifras AES com o voto de cada membro consultado.
- $n-2$ mensagens para o nó A, contendo o voto de consenso.
- Nó A: $n-2$ decifras AES para recuperar os votos.
- Nó A: 1 mensagem para o nó B, informando que ele está removido.
- Nó A: 1 cifra AES para o nó B sobre a exclusão.
- Nó B: 1 decifra AES do no B sobre sua exclusão.
- Nó A: geração de uma nova chave compartilhada do grupo com a chave pública do nó com maior pontuação de confiança, pertencente ao mesmo nível e grupo, que possui em sua tabela local (funções $GEN + KA + KDF$).
- Nó A: cifra AES com a chave atual da mensagem contendo o resultado do consenso (exclusão com sucesso) e a nova chave compartilhada no grupo.
- Nó A: $n-2$ mensagens (todos os membros atuais, exceto a si mesmo e ao nó excluído) com o resultado do consenso e a nova chave compartilhada estabelecida.
- $n-2$ decifras AES e atualização das novas informações.

Para a remoção do nó e definição de uma nova chave de grupo, o total de mensagens necessárias são $3*n-5$, considerando as trocas de mensagens para o consenso, informação do nó excluído e atribuição da nova chave. O tempo de processamento necessário foi $2*n-2$ cifras AES, $3*n-5$ decifras AES e uma geração de chave que consumiu $1,611$ ms.

7.4.6.2 Modelos de ataques

Para todos os modelos de ataque será considerada uma rede VANET com comunicação V2V em perímetro urbano, com n nós a uma velocidade média de 40 km/h, em um dia útil típico. Os nós nesta rede realizam trocas de mensagens entre seus vizinhos para notificação das condições do tráfego, alertas de emergência ou comunicações diretas entre eles.

A identificação inicial ocorre através de *beacons*, para informar a presença do nó na rede e também para perceber os demais nós que estão entrando ou que já estão presentes na rede, assim, os nós enviam *beacons* de tempo em tempo. A comunicação entre os nós ocorre sob um canal de transmissão de rede sem fio, a uma determinada frequência, através de um protocolo de comunicação previamente combinado.

Cada nó está apto a se comunicar com os seus vizinhos dentro de um intervalo de comunicação, e estes vizinhos com seus demais vizinhos, disseminando as mensagens *broadcasts*. Este modelo assume que qualquer nó da rede pode tornar-se malicioso em momentos oportunos. Foram considerados três diferentes cenários de ataque, eles são:

1º cenário: o atacante é um agente externo ao grupo, que analisa o movimento da rede, seja seguindo trajetos aleatórios próximo aos membros do grupo ou de maneira estacionária. Em um determinado momento, ele inicia a coleta passiva das informações trafegadas, pois conhece a forma de comunicação da rede sem fio e seu protocolo, com objetivo de obter as informações de identidade dos demais nós e suas mensagens. Essas informações podem ser usadas para fins maliciosos de diferentes naturezas [80, 152].

Este ataque é suprimido pelo uso da cifragem entre os membros do grupo. O atacante poderá coletar um grande rol de dados na rede, porém todas as informações estarão cifradas sob força da chave de criptografia de grupo utilizada pelo *Framework*. Vale ressaltar que a escolha do sistema de criptografia e o tamanho da chave é fundamental para possibilitar esta proteção. Destaca-se que o escopo deste trabalho é a proteção da informação dos membros do grupo contra atacantes externos. Além disso, que o nó atacante é considerado malicioso do início ao fim, e detectado rapidamente quando é admitido no grupo como nível 1. Assim, detectar a alteração de comportamento de um nó após ele ser considerado confiável e realizar reclassificação dos nós foge à finalidade deste trabalho.

2º cenário: este cenário utiliza o ataque de *collusion* para estabelecer acordos secretos de nós, a princípio confiáveis, com um nó atacante. Nesta primeira fase, o ataque forma um grupo de nós em coluio. O número deve ser tão grande quanto possível. Em um movimento de dia normal, em uma via com tráfego intenso, este grupo pode realizar o ataque enviando informações sobre uma falsa manutenção na pista, indicando um desvio desnecessário aos demais nós. Desta forma, o grupo atacante obtém acesso livre na avenida antes congestionada. Esta abordagem mescla o ataque de *collusion* com o *bogus attack*.

Este ataque pode ser mitigado por meio da estrutura do *Framework*. O GSeF4V fornece uma estrutura de nível e serviços para cada fase do grupo. Embora muitos serviços sejam indicados como base para o seu funcionamento, novos serviços podem, também, ser implementados com base na estrutura fornecida. Neste contexto, várias maneiras [153, 78, 154, 155] de detectar o coluio podem ser implementadas na forma de serviço. Este serviço pode ser atribuído a membros dos níveis 2 ou 3 para detectar o atacante em coluio e o remover do grupo. As informações comprometidas se limitarão às do período compreendido entre o seu início no grupo e o momento da detecção do seu ataque. Este nó será marcado como malicioso e não será mais aceito como membro de grupo por este *Framework*.

Em relação ao *bogus attack*, os demais nós do grupo perceberão que a informação trafegada por um determinado nó não corresponde à realidade e o classificarão como malicioso. Mesmo que a mensagem seja de um nó confiável, porém adulterada pelo atacante *bogus*, este ataque será percebido através da verificação da integridade, por meio do uso das funções MAC. Quando o nó é classificado como malicioso, é excluído do grupo e uma nova chave é gerada após a sua exclusão.

3º cenário: o último cenário de ataque combina o ataque de *eavesdropping*, *alteration* e *replay*. Neste cenário, o atacante está em movimento e inserido na rede. Ele coleta, secretamente, informações divulgadas na rede em *broadcast*, com dados reais e de alta relevância para todos os nós, e as guarda para um momento oportuno. Ao chegar o momento do ataque, o nó atacante altera ligeiramente a mensagem, inserindo um dado sobre um acidente em determinado local, e a repassa em *broadcast* como uma mensagem válida; com isto o atacante pode obter acesso

privilegiado à avenida ou mesmo causar pânico entre os participantes da rede, prejudicando o uso positivo da rede veicular.

Este terceiro cenário é totalmente inviabilizado pelas funções de verificação de mensagem MAC, pois quando a mensagem é alterada, a *tag* resultante da função MAC não será a mesma que a recebida. Ainda que o atacante não demonstre nenhum comportamento malicioso ao longo do tempo e consiga coletar informações de tráfego, entre outras, o seu ataque será inviabilizado pela garantia da integridade deste *Framework*.

7.5 CONSIDERAÇÕES SOBRE O CAPÍTULO

Este capítulo apresentou as avaliações do GSeF4V com foco na formação de grupos multiníveis e da segurança. O simulador *The* ONE foi utilizado para gerar a carga de trabalho, com traços de entrada para os testes que representam um modelo de movimento próximo à realidade. Este capítulo, também introduziu o GSeF4V-*Simulator*, um simulador construído para testar este *Framework*. Ele foi construído para consumir a carga de trabalho, realizar as chamadas das funções que implementam todas as regras do *Framework* para formar grupos e entrar nos níveis, além de reportar os resultados finais. O criptossistema implementado foi adaptado da biblioteca MIRACL, a fim de obter os custos para garantir a integridade e confidencialidade. Os principais resultados desta avaliação são o impacto dos nós maliciosos, do parâmetro $pointsL1$, do tempo de permanência, da confiança e da segurança.

Na avaliação de nós maliciosos, foi alterada a quantidade desses nós em 10%, 20%, 30% e 40%. Nesta avaliação, foram medidos o impacto da variação de nós maliciosos nos grupos multiníveis, o impacto da variação de nós maliciosos no total de nós e o impacto da variação de nós maliciosos na formação de grupo. Os resultados apresentados nas Figuras 7.3, 7.4 e 7.5 demonstram o mesmo comportamento, independentemente da densidade de nós. Assim, é possível concluir que, com o aumento de número de nós maliciosos, haverá menos nós nos níveis 2 e 3, e no nível 1 não haverá alteração, pois neste nível os nós precisam apenas de contato e permanência.

Na avaliação do parâmetro $pointsL1$, foi alterado o valor de $pointsL1$ em 2, 4, 6, 8 e 10. Nesta avaliação, foram medidos o impacto da variação do $pointsL1$ nos grupos multiníveis, o impacto da variação do $pointsL1$ no total de nós, o impacto da variação do $pointsL1$ na formação de grupos, o impacto da variação do $pointsL1$ na permanência dos nós no grupo e o impacto da variação do $pointsL1$ na quantidade de nós no grupo. Os resultados apresentados nas Figuras 7.6, 7.7, 7.8, 7.9, 7.10 mostram que quanto maior o $pointsL1$, menor o número de grupos, menor o número de nós em cada grupo e, consequentemente, menor a quantidade de nós nos níveis, porém, maior é o tempo de permanência dos nós nos grupos. Como hipótese para trabalhos futuros, pode ser mensurado o número de nós em cada nível por grupo, variando-se o parâmetro $pointsL1$. Na presente avaliação, esse número foi mensurado considerando-se todos os nós, não apenas por grupo; no entanto, espera-se que o comportamento seja similar.

Na avaliação do tempo de permanência, foi alterado o valor de permanência em 10, 20, 30, 40, 50 e 60. Nesta avaliação, foram medidos o impacto da variação dos segundos nos grupos multiníveis, o impacto da variação dos segundos no total de nós, o impacto da variação dos segundos na permanência em um grupo, o impacto da variação dos segundos na formação de grupos e o impacto da variação dos segundos na quantidade de nós no grupo. Os resultados apresentados nas Figuras 7.11, 7.12, 7.13, 7.14 e 7.15 mostram que, conforme aumenta o tempo de permanência, diminui a quantidade de número de nós nos grupos multiníveis e a quantidade de nós por grupo, porém aumenta a permanência destes nós nos grupos.

Na avaliação da confiança, foi alterado o valor de *trust* em 1000, 5000, 10000, 15000 e 20000. Nesta avaliação, foram medidos o impacto da variação do *trust* nos grupos multiníveis e o impacto da variação do *trust* no total de nós. Através dos resultados apresentados nas Figuras 7.16 e 7.17, é possível verificar que esta variação influencia apenas nos níveis 2 e 3, pois estes requerem confiança para o ingresso do nó; além disso, pode-se observar que houve um maior impacto nas avaliações que consideram 150 e 300 nós; na avaliação que considera 450 nós não houve impacto, pois os nós estão mais conectados e, por isso, incrementa mais pontos referente à confiança, assim os valores estipulados não sofreram impactos.

Através destes resultados, podemos concluir que a escolha dos pontos de confiança e do tempo de confiança causa impacto direto no número de grupos formados, no tamanho médio dos grupos e no tempo que os nós permanecem em um grupo. Esses parâmetros podem ser ajustados pelo *designer*, a depender de seus objetivos e limitações:

- Variando o $pointsL1$:
  - A escolha de 8 a 10 pontos de confiança melhora a segurança, formando poucos grupos confiáveis. Os grupos são pequenos e possuem maior tempo de permanência, embora possam demorar muito para se formar e alguns nós possam nunca fazer parte de um grupo. Esta escolha também impacta a pontuação exigida para atribuir a confiança a um nó; dessa forma, os poucos nós que farão partes dos grupos serão mais confiáveis do que em outras escolhas e parâmetros.
  - A escolha de 6 pontos representa uma abordagem intermediária entre a quantidade de nós em um grupo e o tempo de permanência de cada nó no grupo. Vale observar que, embora este valor aponte um equilíbrio, ele não oferece grupos tão seguros, tampouco nós densos. Este equilíbrio também se refere à confiança. Com menos pontos, a confiança é atribuída mais facilmente entre os nós.
  - A escolha de 2 ou 4 pontos de confiança gera poucos grupos grandes. Os grupos são formados muito rapidamente, e os nós podem mudar de grupo com muita frequência. Esta situação faz com que as chaves compartilhadas dos grupos sejam alteradas frequentemente. A escolha de valores baixos não incorre em pontuações adicionais para obter a confiança, ou seja, as demais métricas de confiança ($pointsL2$ e $pointsL3$) serão os únicos critérios para atingir essa confiança. Vale observar que estas métricas são as responsáveis pela confiança, porém o $pointsL1$ age como complemento para o alcance desta confiança, além de atuar como parâmetro para número de contato e tempo de permanência.
- Variando a permanência:
  - A escolha de 50 e 60 segundos aumenta a permanência dos nós no grupo, porém, diminui a quantidade de nós nos grupos e a quantidade de grupos na rede. Esta escolha é adequada quando o objetivo do implementador é oferecer serviços no grupo que demandam estabilidade de conectividade entre os nós, ou seja, embora sejam poucos nós, eles estarão por mais tempo conectados entre si de forma segura.
  - A escolha de 30 e 40 segundos é intermediária, possuindo uma média permanência dos nós no grupo. Estes valores representam a necessidade de implementar serviços em grupo em que a permanência ou densidade não sejam fatores restritivos.
  - A escolha de 10 e 20 segundos possui muitos grupos, com muitos nós em cada, e com baixa permanência dos nós. Esta escolha aplica-se ao implementador, que

visa a oferecer serviços em grupos que demandam alta densidade, ou seja, serviços em que o número de nó seja o principal foco, seja para disseminar mensagens mais rapidamente ou realizar serviços que necessitam de maior diâmetro de rede para cobrir uma área superior à área com as outras escolhas deste parâmetro.

- Variando a confiança - este valor impacta diretamente na densidade de nós no ambiente, desta forma, o valor do *trust* deve ser alterado conforme o interesse do implementador:

  - Para o ambiente com 150 nós, os valores do *trust* de 1000, 5000, e 10000 influenciaram na quantidade de nós nos grupos e no total de nós na rede. Conforme aumenta o valor de *trust*, diminui a quantidade de nós. Os demais valores, de 15000 e 20000, tiveram o mesmo comportamento de 10000; ou seja, observa-se que após alcançar pontos próximos ao valor de 10000, a variação deste parâmetro se torna irrelevante, pois nenhum nó alcançou pontos superior a 10000.
  - Para o ambiente que considera 300 nós, os valores de *trust* de 1000 e 5000 têm o mesmo comportamento. O que mostra que é um valor facilmente alcançado pelos nós. Conforme o valor de *trust* aumenta, a quantidade de nós diminui, ou seja, menos nós alcançam esta pontuação. Este parâmetro começa a agir como elemento restritivo e pode ser considerado como um filtro de confiança.
  - Para o ambiente que considera 450 nós, os valores de *trust* de 1000, 5000, 10000, 15000 e 20000 não influenciaram no comportamento dos nós, o que mostra que são valores facilmente alcançados por eles. Pelo fato deste ambiente possuir vários nós que realizam muitos contatos entre si, e permitir que a análise de comportamento seja feita constantemente, eles alcançam os pontos mais rápido que as densidades anteriores, que demoraram mais tempo para fazer estas análises; assim, devido à alta densidade, a maioria dos veículos alcançaram os pontos até 20000, e estes pontos não servirão de elementos restritivos. É possível concluir que pontos maiores devem ser considerados para densidades maiores.

Observe que toda configuração tem suas vantagens e desvantagens, e que a escolha dos parâmetros é uma decisão de implementação do projetista. Essas escolhas estão diretamente ligadas ao objetivo do implementador e ao seu foco, seja ele para formar muitos grupos com poucos nós mais confiáveis ou poucos grupos com muitos nós menos confiáveis, entre outros.

Na avaliação da segurança, foram medidos os custos de processamento e os números de mensagem necessários para estabelecer a chave de segurança, bem como o tempo de processamento para cifrar, decifrar e realizar as funções MAC. Os resultados mostraram que adotar o sistema ECIES para integrar o *Framework* resultou em tempos viáveis para um cenário de redes veiculares urbanas com comunicação V2V. Vale ressaltar que a cifragem e a decifragem com o AES-128 representam um dos melhores custos, devido ao nível de segurança conhecido na literatura, como apresentado no capítulo 4.1.1. Embora o custo das mensagens não seja calculado, acredita-se que ele não inviabilizará o uso do *Framework*, pois o implementador pode escolher protocolos de roteamento que se adaptem melhor às suas necessidades e às mensagens necessárias para o GSeF4V.

8 CONSIDERAÇÕES FINAIS

A VANET é uma rede *ad-hoc* que permite a comunicação entre os veículos, representados por nós em movimento, em diferentes cenários. Estas redes possuem características únicas, como topologia altamente dinâmica, desconexões frequentes, mobilidade restrita, quantidade de nós, entre outras. A conectividade da VANET permite a comunicação entre os nós em locais sem infraestrutura, através da comunicação V2V, na qual eles atuam de forma distribuída e descentralizada. No entanto, a segurança desta rede é um desafio que este trabalho buscou mitigar através da abordagem da comunicação em grupos multiníveis.

A principal contribuição desta tese é oferecer uma solução com serviço de confidencialidade e integridade para VANETs, usando a comunicação V2V, sistemas distribuídos e independentes de protocolos e de infraestruturas. Esta solução foi possível por meio do uso de grupos multiníveis, para oferecer um ambiente adequado a uma segura aplicação da comunicação em grupo sem o ônus de comunicações diretas com criptografia de chave pública, beneficiando-se da formação de grupo e do sistema de criptografia híbrida ECIES para comunicações íntegras e confidenciais dentro de um único grupo, com uso de chaves simétricas compartilhadas entre os níveis dos grupos na VANET.

Foram realizados vários testes para analisar a viabilidade e o desempenho deste *Framework*; porém, como não há um trabalho diretamente correlato a este que faça a mesma abordagem em segurança distribuída com grupos multiníveis, as avaliações foram restritas ao próprio *Framework* e ao sistema de criptografia utilizado. A avaliação foi projetada para variar os principais parâmetros, de modo a obter a melhor forma de auxiliar o implementador com as melhores decisões para cada ambiente a ser planejado.

Nos testes realizados, foram analisados vários cenários com um rol de variadas possibilidades, para que cada implementador pudesse escolher a melhor configuração para atingir os seus objetivos; além disso, o *Framework* resulta em um cenário com um grupo já formado com nós confiáveis, prontos para implementar os serviços. Neste cenário, a utilização de um criptossistema híbrido mostrou-se totalmente viável para garantir a confidencialidade e a integridade, mesclando os passos desse criptossistema com o processo de formação de grupo e níveis de cada grupo, ou seja, por meio da utilização de uma chave compartilhada para comunicação segura. Os resultados mostraram que o desempenho é viável para a rede VANET com comunicação V2V e que o GSeF4V é viável para utilização; além disso, foram mostrados cenários de ataques para ilustrar a proteção que este *Framework* pode oferecer.

Alguns pontos ainda podem ser explorados para extensão deste *Framework*. Os principais pontos são a implementação de um suporte multigrupo para os nós na VANET, mitigar o número de mensagens necessárias para realizar os consensos e mensurar o custo das transmissões de mensagens e como isso pode impactar neste *Framework*. Tais pontos são indicados como trabalhos futuros e extensão da avaliação do GSeF4V.

O GSeF4V permite que um nó pertença a apenas um grupo por vez; ou seja, se um determinado nó já fizer parte de um grupo e encontrar novo grupo, solicitará que eles sejam mesclados em um único grupo. No entanto, uma abordagem multigrupo pode ser implementada, fazendo com que cada nó possa ser membro de múltiplos grupos e cada grupo possuir múltiplos níveis independentes entre si. Uma das vantagens de assumir esta abordagem é permitir a criação de grupos orientados a objetivos, ou seja, as exigências e serviços oferecidos em cada grupo serão atribuídos conforme seus objetivos, evitando-se, dessa forma, a mesclagem de grupos com objetivos distintos.

Um dos gargalos do GSeF4V é o número de mensagens que devem ser trocadas durante os processos de consenso. Este consenso é realizado na adição, remoção e alteração de nível entre os membros do grupo; assim, o custo de sua realização impacta diretamente no desempenho de todo o sistema. Uma possível forma de abrandar este impacto é modificar o consenso de mensagens diretas para uso de *blockchain*. O uso de *blockchain* no contexto de consenso em VANETs pode ser encontrado em [156, 157, 158]. Esta sugestão aproveita-se da abordagem descentralizada deste *Framework*; porém, com as cadeias de blocos *hashs*, a confirmação e a verificação entre os nós podem ser realizadas de forma menos onerosa. A abordagem com *blockchain* deve ser estudada em detalhes, com foco na sua viabilidade para incorporar o *Framework*, além dos custos comparados ao *Framework* atual.

Além disso, novos testes com este *Framework* podem ser realizados com foco em medir o custo da transmissão de mensagens, dado que o estabelecimento de chaves necessita de mensagens *broadcast* ou envio direto de mensagens para $n-1$ ou $n-2$ nós no grupo. Esses testes podem direcionar o implementador a escolher protocolos de roteamento que façam um *trade off* entre os objetivos da implementação e o desempenho no GSeF4V.

REFERÊNCIAS

[1] Stephan Eichler. Performance evaluation of the IEEE 802.11p WAVE communication standard. In *Vehicular Technology Conference*, pages 2199–2203, 2007.

[2] Roberto A. Uzcategui, Antonio Jose De Sucre, and Guillermo Acosta-Marum. Wave: A tutorial. *IEEE Communications Magazine*, 47(5):126–133, 2009.

[3] William Stallings. *Criptografia e segurança de redes.* Pearson Prentice Hall, 2008.

[4] Hannes Hartenstein and Kenneth Laberteaux. *VANET: vehicular applications and internetworking technologies*, volume 1. John Wiley & Sons, 2009.

[5] Elias C Eze, Si-Jing Zhang, En-Jie Liu, and Joy C Eze. Advances in vehicular ad-hoc networks (VANETs): Challenges and road-map for future development. *International Journal of Automation and Computing*, 13(1):1–18, 2016.

[6] Stefan Mihai, Nedzhmi Dokuz, Meer Saqib Ali, Purav Shah, and Ramona Trestian. Security aspects of communications in VANETs. In *International Conference on Communications*, pages 277–282. IEEE, 2020.

[7] Ademar Takeo Akabane, Edmundo Roberto Mauro Madeira, and Leandro Aparecido Villas. Collaborative and infrastructure-less vehicular traffic rerouting for intelligent transportation systems. In *Anais Estendidos do XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 193–200. SBC, 2020.

[8] Muhammad Sameer Sheikh, Jun Liang, and Wensong Wang. A survey of security services, attacks, and applications for vehicular ad hoc networks (VANETs). *Sensors*, 19(16):3589, 2019.

[9] Fang Li, Wei Chen, Yishui Shui, Ji Wang, Kun Yang, Lida Xu, Junyi Yu, and Changzhen Li. Connectivity probability analysis of VANETs at different traffic densities using measured data at 5.9 GHz. *Physical Communication*, 35:100709, 2019.

[10] Sneha Kanchan and Narendra S. Chaudhari. Signcrypting the group signature with non-transitive proxy re-encryption in VANET. In Pankaj Kumar Sa, Sambit Bakshi, Ioannis K. Hatzilygeroudis, and Manmath Narayan Sahoo, editors, *Recent Findings in Intelligent Computing Techniques*, pages 15–23, Singapore, 2019. Springer Singapore.

[11] M. A. Shahid, A. Jaekel, C. Ezeife, Q. Al-Ajmi, and I. Saini. Review of potential security attacks in VANET. In *2018 Majan International Conference (MIC)*, pages 1–4, March 2018.

[12] Irshad Ahmed Sumra, Halabi Bin Hasbullah, and Jamalul-lail Bin AbManan. Attacks on security goals (confidentiality, integrity, availability) in VANET: A survey. In Anis Laouiti, Amir Qayyum, and Mohamad Naufal Mohamad Saad, editors, *Vehicular Ad-hoc Networks for Smart Cities*, pages 51–61, Singapore, 2015. Springer Singapore.

[13] Richard Gilles Engoulou, Martine Bellaïche, Samuel Pierre, and Alejandro Quintero. VANET security surveys. *Computer Communications*, 44:1–13, 2014.

[14] Kui Liu and Changle Li. An efficient adaptive frame aggregation scheme in vehicular ad hoc networks. In *Wireless Communications and Signal Processing*, pages 1–6. IEEE, 2017.

[15] D. F. Macedo T. R. Oliveira, C. M. Silva and J. M. S. Nogueira. SNVC: Social Nerworks for vehicular certification. *Computer Networks*, 111:129–140, 2016.

[16] W. Li and H. Song. ART: An attack-resistant trust management scheme for securing vehicular ad hoc networks. *Transactions on Intelligent Transportation Systems*, 17(4):960–969, 2016.

[17] N. Panwar S. Dolev, Ł. Krzywiecki and M. Segal. Vehicle authentication via monolithically certified public key and attributes. *Wireless Networks, Springer*, 22(3):879–896, 2016.

[18] Renan Greca and Luiz Carlos Pessoa Albini. TruMan: Trust management for vehicular networks. *Symposium on Computers and Communications*, 2018.

[19] Hamssa Hasrouny, Carole Bassil, Abed Ellatif Samhat, and Anis Laouiti. Group-based authentication in V2V communications. In *International Conference on Digital Information and Communication Technology and its Applications*, pages 173–177. IEEE, 2015.

[20] Lei Zhang, Qianhong Wu, Bo Qin, Josep Domingo-Ferrer, and Bao Liu. Practical secure and privacy-preserving scheme for value-added applications in VANETs. *Computer Communications*, 71:50–60, 2015.

[21] K. K. Chauhan, S. Kumar, and S. Kumar. The design of a secure key management system in vehicular ad hoc networks. In *Conference on Information and Communication Technology*, pages 1–6, Nov 2017.

[22] G Kumaresan and T Adiline Macriga. Group key authentication scheme for VANET intrusion detection (GKAVIN). *Wireless Networks*, 23(3):935–945, 2017.

[23] Hamssa Hasrouny, Carole Bassil, Abed Ellatif Samhat, and Anis Laouiti. Security risk analysis of a trust model for secure group leader-based communication in VANET. In *Vehicular Ad-Hoc Networks for Smart Cities*, pages 71–83. Springer, 2017.

[24] Shrikant Tangade and Sunilkumar S Manvi. CBTM: Cryptography based trust management scheme for secure vehicular communications. In *International Conference on Control, Automation, Robotics and Vision*, pages 325–330. IEEE, 2018.

[25] Elaine Alves Rocha Pires, Ivan Luiz Pedroso Pires, and Luiz Carlos Pessoa Albini. Supporting Confidentiality and Integrity on V2V Communications. In *International Conference on Connected Vehicle and Expo (ICCVE)*, pages 1–6, 2022.

[26] Yu Wang and Fan Li. *Vehicular Ad Hoc Networks*, pages 503–525. Springer London, London, 2009.

[27] F. Li and Y. Wang. Routing in vehicular ad hoc networks: A survey. *Vehicular Technology Magazine*, 2(2):12–22, June 2007.

[28] Mohamed Nidhal Mejri, Jalel Ben-Othman, and Mohamed Hamdi. Survey on VANET security challenges and possible cryptographic solutions. *Vehicular Communications*, 1(2):53–66, 2014.

[29] Haowen Tan, Dongmin Choi, Pankoo Kim, Sungbum Pan, and Ilyong Chung. Secure certificateless authentication and road message dissemination protocol in VANETs. *Wireless Communications and Mobile Computing*, 2018:1–13, 2018.

[30] S. S. Husain, A. Kunz, A. Prasad, E. Pateromichelakis, and K. Samdanis. Ultra-high reliable 5G V2X communications. *IEEE Communications Standards Magazine*, 3(2):46–52, 2019.

[31] V. Sharma, I. You, and N. Guizani. Security of 5G-V2X: Technologies, standardization and research directions. *IEEE Network*, pages 1–9, 2020.

[32] D. Jiang and L. Delgrossi. IEEE 802.11p: Towards an international standard for wireless access in vehicular environments. In *VTC Spring 2008 - IEEE Vehicular Technology Conference*, pages 2036–2040, May 2008.

[33] H. Zimmermann. OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communications*, 28(4):425–432, 1980.

[34] Aneel Rahim, Muhammad Sher, Adeel Javed, Imran Ahmad, and Rehman Hameed. Performance analysis of TCP in VANETs by using 802.11e. In *International Conference on Networks*, pages 1–3, 2008.

[35] Jun-Li Kuo, Chen-Hua Shih, and Yaw-Chung Chen. Performance analysis of real-time streaming under TCP and UDP in VANET via OMNET. In *International Conference on ITS Telecommunications*, pages 116–121, 2013.

[36] B. Khorashadi, A. Chen, D. Ghosal, C.-N. Chuah, and M. Zhang. Impact of Transmission Power on the Performance of UDP in Vehicular Ad Hoc Networks. In *International Conference on Communications*, pages 3698–3703, 2007.

[37] Michael Lee and Travis Atkison. VANET applications: Past, present, and future. *Vehicular Communications*, 28:100310, 2021.

[38] Shalini. S and Annapurna P Patil. Survey of hybrid VANET design for provisioning infotainment application. In *International Conference on Advances in Information Technology*, pages 140–145, 2019.

[39] Jyoti Grover, Ashish Jain, Sunita Singhal, and Anju Yadav. Real-time VANET applications using fog computing. In Arun K. Somani, Sumit Srivastava, Ankit Mundra, and Sanyog Rawat, editors, *International Conference on Smart System, Innovations and Computing*, page 683–691, Singapore, 2018. Springer Singapore.

[40] Othman S. Al-Heety, Zahriladha Zakaria, Mahamod Ismail, Mohammed Mudhafar Shakir, Sameer Alani, and Hussein Alsariera. A Comprehensive Survey: Benefits, Services, Recent Works, Challenges, Security, and Use Cases for SDN-VANET. *IEEE Access*, 8:91028–91047, 2020.

[41] Li Zhu, Fei Richard Yu, Yige Wang, Bin Ning, and Tao Tang. Big data analytics in intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 20(1):383–398, 2019.

[42] Kakan C. Dey, Ashok Mishra, and Mashrur Chowdhury. Potential of intelligent transportation systems in mitigating adverse weather impacts on road mobility: A review. *Intelligent Transportation Systems*, 16(3):1107–1119, 2015.

[43] Rasheed Hussain, Jooyoung Lee, and Sherali Zeadally. Trust in VANET: A survey of current solutions and future research opportunities. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–19, 2020.

[44] 16091. IEEE Draft Trial-Use Standard for Wireless Access in Vehicular Environments (WAVE) - Resource Manager. *IEEE Std P1609.1/D17, Jul 2006*, pages 1–66, 2019.

[45] 16092. IEEE Draft Standard for Wireless Access in Vehicular Environments (WAVE) - Certificate Management Interfaces for End Entities. *IEEE P1609.2.1/D6, January 2022*, pages 1–257, 2022.

[46] 16093. IEEE Approved Draft Standard for Wireless Access in Vehicular Environments (WAVE) - Networking Services. *IEEE P1609.3v3/D6, November 2015*, pages 1–162, 2016.

[47] 16094. IEEE Approved Draft Standard for Wireless Access in Vehicular Environments (WAVE) – Multi-Channel Operation - Corrigendum 1: Miscellaneous corrections. *IEEE P1609.4-2016/Cor1/D3, April 2019*, pages 1–12, 2019.

[48] Hamaciré El Hadj Kalil, Ahmed Dooguy Kora, and Selma Boumerdassi. Security in VANETs: Lightweight Protocol for Group-of-Vehicles Masters (LPGVM). In *International Conference on Advanced Communication Technology (ICACT)*, pages 6–11, 2020.

[49] Georgios Karagiannis, Onur Altintas, Eylem Ekici, Geert Heijenk, Boangoat Jarupan, Kenneth Lin, and Timothy Weil. Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions. *IEEE Communications Surveys Tutorials*, 13(4):584–616, 2011.

[50] Michael Lee and Travis Atkison. VANET applications: Past, present, and future. *Vehicular Communications*, page 100310, 2020.

[51] Koosha Paridel, Yves Vanrompay, Davy Preuveneers, Yolande Berbers, et al. Teamwork on the road: Efficient collaboration in VANETs with context-based grouping. *Procedia Computer Science*, 5:48–57, 2011.

[52] O. Ermiş, Ş. Bahtiyar, E. Anarim, and M. U. Çağlayan. Open problems for group-key agreement protocols on vehicular ad-hoc networks. In *2013 International Conference on Connected Vehicles and Expo (ICCVE)*, pages 828–831, Dec 2013.

[53] X. Zhu, S. Jiang, L. Wang, H. Li, W. Zhang, and Z. Li. Privacy-preserving authentication based on group signature for VANETs. In *Global Communications Conference*, pages 4609–4614, Dec 2013.

[54] Y. Park and S. Seo. Fast and secure group key dissemination scheme for out-of-range V2I communication. *IEEE Transactions on Vehicular Technology*, 64(12):5642–5652, Dec 2015.

[55] C. Xu, M. Ma, X. Huang, and H. Bao. A cross-domain group authentication scheme for LTE-A based vehicular network. In *International Conference on Communication Software and Networks*, pages 595–599, May 2017.

[56] R. Waghmode, R. Gonsalves, and D. Ambawade. Security enhancement in group based authentication for VANET. In *International Conference on Recent Trends in Electronics, Information Communication Technology*, pages 1436–1441, May 2016.

[57] Hamssa Hasrouny, Abed Ellatif Samhat, Carole Bassil, and Anis Laouiti. A security solution for V2V communication within VANETs. In *2018 Wireless Days (WD)*, pages 181–183. IEEE, April 2018.

[58] A. Kumar and R. P. Nayak. An efficient group-based safety message transmission protocol for VANET. In *International Conference on Communication and Signal Processing*, pages 270–274, April 2013.

[59] C. Caballero-Gil, P. Caballero-Gil, and J. Molina-Gil. Tool to simulate groups in vehicular networks using NS-2 and TraceGraph. In *European Conference on Circuits and Systems for Communications*, pages 272–273, Nov 2010.

[60] Mayank Verma and Dijiang Huang. SeGCom: Secure Group Communication in VANETs. In *Consumer Communications and Networking Conference*, pages 1–5, 2009.

[61] C. Caballero-Gil, P. Caballero-Gil, and J. Molina-Gil. Group formation through cooperating node in VANETs. In Yuhua Luo, editor, *Cooperative Design, Visualization, and Engineering*, pages 105–108, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[62] Ikram Ullah, Munam Ali Shah, Abid Khan, Carsten Maple, and Abdul Waheed. Virtual pseudonym-changing and dynamic grouping policy for privacy preservation in VANETs. *Sensors*, 21(9), 2021.

[63] Lianhai Liu, Yujue Wang, Jingwei Zhang, and Qing Yang. A secure and efficient group key agreement scheme for VANET. *Sensors*, 19(3):482, 2019.

[64] Thi Ngoc Diep Pham and Chai Kiat Yeo. Adaptive trust and privacy management framework for vehicular networks. *Vehicular Communications*, 13:1–12, 2018.

[65] Hamssa Hasrouny, Abed Ellatif Samhat, Carole Bassil, and Anis Laouiti. Trust model for group leader selection in VANET. *International Journal of Digital Information and Wireless Communications*, 8(2):139–143, 2018.

[66] Fabian Schneider, Anja Feldmann, Balachander Krishnamurthy, and Walter Willinger. Understanding online social network usage from a network perspective. In *Conference on Internet Measurement*, page 35–48. Association for Computing Machinery, 2009.

[67] Wanita Sherchan, Surya Nepal, and Cecile Paris. A survey of trust in social networks. *ACM Comput. Surv.*, 45(4), aug 2013.

[68] Rasheed Hussain, Jooyoung Lee, and Sherali Zeadally. Trust in VANET: A Survey of Current Solutions and Future Research Opportunities. *Intelligent Transportation Systems*, 22(5):2553–2571, 2021.

[69] Qing Yang and Honggang Wang. Toward trustworthy vehicular social networks. *Communications Magazine, IEEE*, 53:42–47, 08 2015.

[70] Tong Cheng, Guangchi Liu, Qing Yang, and Jianguo Sun. Trust assessment in vehicular social network based on three-valued subjective logic. *IEEE Transactions on Multimedia*, 21(3):652–663, 2019.

[71] Razi Iqbal, Talal Ashraf Butt, Muhammad Afzaal, and Khaled Salah. Trust management in social internet of vehicles: Factors, challenges, blockchain, and fog solutions. *International Journal of Distributed Sensor Networks*, 15(1):1550147719825820, 2019.

[72] Chaker Abdelaziz Kerrache, Nasreddine Lagraa, Abderrahim Benslimane, Carlos T Calafate, and Juan-Carlos Cano. On the human factor consideration for VANETs security based on social networks. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2018.

[73] Jingjing Guo, Xinghua Li, Zhiquan Liu, Jianfeng Ma, Chao Yang, Junwei Zhang, and Dapeng Wu. TROVE: A Context-Awareness Trust Model for VANETs Using Reinforcement Learning. *Internet of Things Journal*, 7(7):6647–6662, 2020.

[74] Federico Concone, Fabrizio De Vita, Ajay Pratap, Dario Bruneo, Giuseppe Lo Re, and Sajal K. Das. A novel recruitment policy to defend against sybils in vehicular crowdsourcing. In *International Conference on Smart Computing*, pages 105–112, 2021.

[75] Raph Levien and Alexander Aiken. Attack-resistant trust metrics for public key certification. In *Conference on USENIX Security Symposium*, volume 7 of *SSYM'98*, page 18, USA, 1998. USENIX Association.

[76] Avleen Kaur Malhi, Shalini Batra, and Husanbir Singh Pannu. Security of vehicular ad-hoc networks: A comprehensive survey. *Computers & Security*, 89:101664, 2020.

[77] D. P. Choudhari and S. S. Dorle. Maximization of packet delivery ratio for DADCQ protocol after removal of eavesdropping and DDoS attacks in VANET. In *International Conference on Computing, Communication and Networking Technologies*, pages 1–8, 2019.

[78] M. Z. A. Bhuiyan and J. Wu. Collusion attack detection in networked systems. *Dependable, Autonomic and Secure Computing*, pages 286–293, 2016.

[79] S. Ali, P. Nand, and S. Tiwari. Secure message broadcasting in VANET over wormhole attack by using cryptographic technique. In *International Conference on Computing, Communication and Automation*, pages 520–523, 2017.

[80] S. Zhang, Q. Cheng, and X. Wang. Impersonation attack on two identity-based authenticated key exchange protocols. In *International Conference on Information Engineering*, volume 2, pages 113–116, Aug 2010.

[81] Atinderpal Singh and Tejinderdeep Singh. Review on detection and prevention of sink hole attack in network. *Global Journal of Computers & Technology*, 5(2):289–292, 2016.

[82] K. Lim, K. M. Tuladhar, and H. Kim. Detecting location spoofing using ADAS sensors in VANETs. In *Consumer Communications Networking Conference*, pages 1–4, 2019.

[83] N. Lo and H. Tsai. Illusion attack on VANET applications - a message plausibility problem. In *Globecom Workshops*, pages 1–8, 2007.

[84] M. N. Mejri and J. Ben-Othman. Detecting greedy behavior by linear regression and watchdog in vehicular ad hoc networks. In *Global Communications Conference*, pages 5032–5037, 2014.

[85] Hamssa Hasrouny, Abed Ellatif Samhat, Carole Bassil, and Anis Laouiti. VANET security challenges and solutions: A survey. *Vehicular Communications*, 7:7–20, 2017.

[86] R. Mishra, A. Singh, and R. Kumar. VANET security: Issues, challenges and solutions. In *International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pages 1050–1055, March 2016.

[87] Jesús Téllez Isaac, Sherali Zeadally, and José Sierra Camara. Security attacks and solutions for vehicular ad hoc networks. *IET communications*, 4(7):894–903, 2010.

[88] Yasser Safinejhad1 and Mehran Abdali. Various types of attacks in VANETs. *International Journal of Computer & Information Technologies*, November 2015.

[89] Sheng Zhong, Hong Zhong, Xinyi Huang, Panlong Yang, Jin Shi, Lei Xie, and Kun Wang. *Connecting Things to Things in Physical-World: Security and Privacy Issues in Vehicular Ad-hoc Networks*, pages 101–134. Springer International Publishing, Cham, 2019.

[90] Farhan Ahmad, Asma Adnane, Virginia N. L. Franqueira, Fatih Kurugollu, and Lu Liu. Man-in-the-middle attacks in vehicular ad-hoc networks: Evaluating the impact of attackers' strategies. *Sensors*, 18(11), 2018.

[91] A. A. Celes and N. E. Elizabeth. Verification based authentication scheme for bogus attacks in VANETs for secure communication. In *International Conference on Communication and Signal Processing*, pages 0388–0392, April 2018.

[92] Tim Leinmuller, Robert K Schmidt, Elmar Schoch, Albert Held, and Gunter Schafer. Modeling roadside attacker behavior in VANETs. In *Globecom Workshops*, pages 1–10. IEEE, 2008.

[93] John R Douceur. The sybil attack. In *International Workshop on Peer-to-Peer Systems*, pages 251–260. Springer, 2002.

[94] Al-Sakib Khan Pathan. *Security of Self-Organizing Networks: MANET, WSN, WMN, VANET*. Auerbach Publications, Boston, MA, USA, 1 edition, 2010.

[95] Fatih Sakiz and Sevil Sen. A survey of attacks and detection mechanisms on intelligent transportation systems: VANETs and IoV. *Ad Hoc Networks*, 61:33–50, 2017.

[96] Pedro Cirne, André Zúquete, and Susana Sargento. TROPHY: Trustworthy VANET routing with group authentication keys. *Ad Hoc Networks*, 71:45–67, 2018.

[97] William Stallings. *Criptografia e segurança de redes. Princípios e práticas*. Pearson Prentice Hall, 2006.

[98] Joan Daemen and Vincent Rijmen. *The design of Rijndael: AES-the advanced encryption standard*. Springer Science & Business Media, 2013.

[99] Whitfield Diffie and Martin Hellman. New directions in cryptography. *Information Theory*, 22(6):644–654, 1976.

[100] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications*, 21(2):120–126, 1978.

[101] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[102] Michael J Wiener. Cryptanalysis of short RSA secret exponents. *IEEE Transactions on Information theory*, 36(3):553–558, 1990.

[103] Victor S Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology*, pages 417–426. Springer, 1986.

[104] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[105] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 119–132. Springer, 2004.

[106] Nathaniel Couture and Kenneth B Kent. The effectiveness of brute force attacks on RC4. In *Communication Networks and Services Research*, pages 333–336. IEEE, 2004.

[107] Peter L Montgomery. A survey of modern integer factorization algorithms. *Centrum voor Wiskunde en Informatica, quarterly*, 7(4):337–366, 1994.

[108] Kevin S McCurley. The discrete logarithm problem. In *Proc. of Symp. in Applied Math*, volume 42, pages 49–74. USA, 1990.

[109] F Vercauteren. Elliptic curve discrete logarithm problem. *Kotholieke Universite it Leuven*, 2005.

[110] Darrel Hankerson, Alfred J Menezes, and Scott Vanstone. *Guide to elliptic curve cryptography*. Springer Science & Business Media, 2006.

[111] Laurie Law, Alfred Menezes, Minghua Qu, Jerry Solinas, and Scott Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28(2):119–134, Mar 2003.

[112] Hamssa Hasrouny, Abed Ellatif Samhat, Carole Bassil, and Anis Laouiti. Trust model for secure group leader-based communications in VANET. *Wireless Networks*, 25(8):4639–4661, 2019.

[113] Shrikant Tangade, Sunilkumar S Manvi, and Pascal Lorenz. Trust management scheme based on hybrid cryptography for secure communications in VANETs. *IEEE Transactions on Vehicular Technology*, 69(5):5232–5243, 2020.

[114] Rasheed Hussain, Jooyoung Lee, and Sherali Zeadally. Trust in VANET: A survey of current solutions and future research opportunities. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[115] Zhen Huang, Sushmita Ruj, Marcos A Cavenaghi, Milos Stojmenovic, and Amiya Nayak. A social network approach to trust management in VANETs. *Peer-to-peer networking and applications*, 7(3):229–242, 2014.

[116] R Venitta Raj and Kannan Balasubramanian. Trust aware similarity-based source routing to ensure effective communication using game-theoretic approach in VANETs. *Journal of Ambient Intelligence and Humanized Computing*, 12(6):6781–6791, 2021.

[117] Umar Farooq Minhas, Jie Zhang, Thomas Tran, and Robin Cohen. A multifaceted approach to modeling agent trust for effective communication in the application of mobile ad hoc vehicular networks. *Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 41(3):407–420, 2010.

[118] Md Zakirul Alam Bhuiyan and Jie Wu. Collusion attack detection in networked systems. In *Conference on Dependable, Autonomic and Secure Computing, Conference on Pervasive Intelligence and Computing, Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress*, pages 286–293. IEEE, 2016.

[119] William Stallings. *Criptografia e segurança de redes: princípios e práticas.* Pearson Prentice Hall, 6 edition, 2015.

[120] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Message authentication using hash functions: The HMAC construction. *RSA Laboratories' CryptoBytes*, 2(1):12–15, 1996.

[121] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *Annual International Cryptology Conference*, pages 1–15. Springer, 1996.

[122] Wai Chen and Shengwei Cai. Ad hoc peer-to-peer network architecture for vehicle safety communications. *IEEE Communications Magazine*, 43(4):100–107, April 2005.

[123] H. O. Al Falasi and N. Mohamed. Preference-based trust rules for group formation in VANETs. In *International Conference on Computational Intelligence, Modelling and Simulation*, pages 333–338, Sep. 2013.

[124] Yasuharu Ohta, Tomoyuki Ohta, and Yoshiaki Kakuda. An autonomous clustering-based data transfer scheme using positions and moving direction of vehicles for VANETs. In *IEEE Wireless Communications and Networking Conference*, pages 2900–2904. IEEE, 2012.

[125] Q. Wang and B. Ayalew. A probabilistic framework for tracking the formation and evolution of multi-vehicle groups in public traffic in the presence of observation uncertainties. *Intelligent Transportation Systems*, 19(2):560–571, Feb 2018.

[126] S. Kanchan, G. Singh, and N. S. Chaudhari. Re-encrypting secure and efficient routing in VANET groups using sharable clouds. In *International Conference on Recent Advances in Information Technology*, pages 1–6, March 2018.

[127] S. Kanchan and N. S. Chaudhari. SRCPR: SignReCrypting Proxy Re-Signature in Secure VANET Groups. *IEEE Access*, 6:59282–59295, 2018.

[128] Q. Li, H. Wu, L. Liu, B. Pan, and L. Dong. A group based dynamic mix zone scheme for location privacy preservation in VANETs. In *International Conference on Security of Smart Cities, Industrial Control System and Communications*, pages 1–5, Oct 2018.

[129] Xiaohan Yue, Bing Chen, Xibo Wang, Yong Duan, Mingchao Gao, and Yuan He. An efficient and secure anonymous authentication scheme for VANETs based on the framework of group signatures. *IEEE Access*, 6:62584–62600, 2018.

[130] D. Tiwari, M. Bhushan, A. Yadav, and S. Jain. A novel secure authentication scheme for VANETs. In *International Conference on Computational Intelligence Communication Technology*, pages 287–297, Feb 2016.

[131] K. Lim, K. M. Tuladhar, X. Wang, and W. Liu. A scalable and secure key distribution scheme for group signature based authentication in VANET. In *Ubiquitous Computing, Electronics and Mobile Communication Conference*, pages 478–483, Oct 2017.

[132] P. Vijayakumar, S. Bose, and A. Kannan. Chinese remainder theorem based centralised group key management for secure multicast communication. *IET Information Security*, 8(3):179–187, May 2014.

[133] N Alangudi Balaji, R Sukumar, and M Parvathy. Enhanced dual authentication and key management scheme for data authentication in vehicular ad hoc network. *Computers & Electrical Engineering*, 76:94–110, 2019.

[134] Xinxin Liu, Yanyan Yang, Erfeng Xu, and Zhijuan Jia. An authentication scheme in VANETs based on group signature. In *International Conference on Intelligent Computing*, pages 346–355. Springer, 2019.

[135] Hamssa Hasrouny, Abed Ellatif Samhat, Carole Bassil, and Anis Laouiti. Trust model for secure group leader-based communications in VANET. *Wireless Networks*, pages 1–23, 2018.

[136] Adnan Mahmood, Bernard Butler, Wei Emma Zhang, Quan Z Sheng, and Sarah Ali Siddiqui. A hybrid trust management heuristic for VANETs. In *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 748–752. IEEE, March 2019.

[137] The working group for WLAN standards. `https://www.ieee802.org/11/`, 2021. acessado em 20 de junho de 2020.

[138] Jyoti Grover, Manoj Singh Gaur, and Vijay Laxmi. Trust establishment techniques in VANET. In *Wireless Networks and Security*, pages 273–301. Springer, 2013.

[139] D. Huang, Z. Zhou, X. Hong, and M. Gerla. Establishing email-based social network trust for vehicular networks. In *Consumer Communications and Networking Conference*, pages 1–5, Jan 2010.

[140] Daemen Joan and Rijmen Vincent. The design of rijndael: AES-the advanced encryption standard. *Information Security and Cryptography*, 2002.

[141] Gurpreet Singh. A study of encryption algorithms (RSA, DES, 3DES and AES) for information security. *International Journal of Computer Applications*, 67(19), 2013.

[142] Joppe W Bos, J Alex Halderman, Nadia Heninger, Jonathan Moore, Michael Naehrig, and Eric Wustrow. Elliptic curve cryptography in practice. In *International Conference on Financial Cryptography and Data Security*, pages 157–175. Springer, 2014.

[143] Ryan Glabb, Laurent Imbert, Graham Jullien, Arnaud Tisserand, and Nicolas Veyrat-Charvillon. Multi-mode operator for SHA-2 hash functions. *Journal of Systems Architecture*, 53(2-3):127–138, 2007.

[144] Henri Gilbert and Helena Handschuh. Security analysis of SHA-256 and sisters. In *International workshop on selected areas in cryptography*, pages 175–193. Springer, 2003.

[145] Rini Indrayani, Hanung Adi Nugroho, Risanuri Hidayat, and Irfan Pratama. Increasing the security of MP3 steganography using AES encryption and MD5 hash function. In *International Conference on Science and Technology-Computer*, pages 129–132. IEEE, 2016.

[146] S Nivetha, N Edna Elizabeth, T Prasanya Padmasha, and I Gohulalakshmi. Secure authentication process in smart cards. In *International Conference on Intelligent Systems and Control*, pages 1–5. IEEE, 2016.

[147] Florian Mendel, Norbert Pramstaller, Christian Rechberger, and Vincent Rijmen. Analysis of step-reduced sha-256. In *International workshop on fast software encryption*, pages 126–143. Springer, 2006.

[148] Akashi Satoh and Tadanobu Inoue. ASIC-hardware-focused comparison for hash functions MD5, RIPEMD-160, and SHS. *Integration*, 40(1):3–10, 2007.

[149] Ari Keranen, Jorg Ott, and Teemu Karkkainen. The ONE Simulator for DTN Protocol Evaluation. In *International Conference on Simulation Tools and Techniques*, New York, NY, USA, 2009. ICST.

[150] Package java.nio. `https://docs.oracle.com/javase/8/docs/api/java/nio/package-summary.html`, 2021.

[151] CertiVox. Multiprecision integer and rational arithmetic cryptographic library (MIRACL). `http://github.com/miracl`, 2014.

[152] M. H. Yılmaz and H. Arslan. Impersonation attack identification for secure communication. In *Globecom Workshops*, pages 1275–1279, Dec 2013.

[153] Xuejiao Liu, Oubo Ma, Wei Chen, Yingjie Xia, and Yuxuan Zhou. HDRS: A Hybrid Reputation System With Dynamic Update Interval for Detecting Malicious Vehicles in VANETs. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–12, 2021.

[154] Yong Hao, Jin Tang, and Yu Cheng. Cooperative sybil attack detection for position based applications in privacy preserved VANETs. In *Global Telecommunications Conference*, pages 1–5, 2011.

[155] Yu-Chih Wei and Yi-Ming Chen. Evaluation of collusion resistance on trust management systems for VANETs. In *International Conference on ITS Telecommunications*, pages 356–360, 2012.

[156] Sowmya Kudva, Shahriar Badsha, Shamik Sengupta, Ibrahim Khalil, and Albert Zomaya. Towards secure and practical consensus for blockchain based VANET. *Information Sciences*, 545:170–187, 2021.

[157] Shirshak Raja Maskey, Shahriar Badsha, Shamik Sengupta, and Ibrahim Khalil. ALICIA: Applied Intelligence in blockchain based VANET: Accident Validation as a Case Study. *Information Processing & Management*, 58(3):102508, 2021.

[158] Rakesh Shrestha, Rojeena Bajracharya, and Seung Yeob Nam. Blockchain-based message dissemination in VANET. In *International Conference on Computing, Communication and Security*, pages 161–166, 2018.

APÊNDICE A – CÓDIGO FONTE DA IMPLEMENTAÇÃO

Este apêndice apresenta o código-fonte da implementação do GSeF4V-*Simulator*. Os arquivos estão organizados em pacotes e neste apêndice eles são apresentados nas seções.

A.1 MAIN PACKAGE

A.1.1 Main.java

```
1  package main;
2  import java.util.ArrayList;
3  import java.util.Iterator;
4  import filehandler.FileHandler;
5  public class Main {
6     public static void main(String[] args) {
7        ArrayList<Parameter> parameters = new ArrayList<Parameter>();
8        // Variar o parâmetro para a simulação desejada
9        parameters.add(new Parameter(10,20,10,10000,5,5,10,450));//05
10       int step = 0;
11       long startTotal = System.currentTimeMillis();
12       for (Iterator<Parameter> iterator = parameters.iterator();
            iterator.hasNext();) {
13          long start = System.currentTimeMillis();
14          Parameter p = iterator.next();
15          new FileHandler(p.filename).run(p);
16          step++;
17          System.out.println("Simulação "+step+" finalizada em "+(
               System.currentTimeMillis()-start));
18       }
19       System.out.println("Fim das simulações em "+(System.
            currentTimeMillis()-startTotal));
20    }
21 }
```

A.1.2 Parameter.java

```
22 package main;
23 public class Parameter {
24    public int malicious_rate;
25    public int social_trust_rate;
26    public int trust_time;
27    public int beta_point;
28    public int n; // gamma
29    public int x; // delta
30    public int w; // epsilon
31    public int number_nodes;
32    public String filename;
```

```
33      public Parameter(int malicious_rate, int social_trust_rate, int
            trust_time, int beta_point, int n, int x, int w,
34              int number_nodes) {
35          this.malicious_rate = malicious_rate;
36          this.social_trust_rate = social_trust_rate;
37          this.trust_time = trust_time;
38          this.beta_point = beta_point;
39          this.n = n;
40          this.x = x;
41          this.w = w;
42          this.number_nodes = number_nodes;
43          this.filename = "src/logs/default_scenario_" + this.
                number_nodes + "_ConnectivityONEReport.txt_otimizado";
44      }
45  }
```

A.2 FILEHANDLER PACKAGE

A.2.1 FileHandler.java

```
46  package filehandler;
47  import java.io.IOException;
48  import java.nio.charset.StandardCharsets;
49  import java.nio.file.Files;
50  import java.nio.file.Paths;
51  import java.util.LinkedHashSet;
52  import java.util.StringTokenizer;
53  import framework.Framework;
54  import main.Parameter;
55  public class FileHandler {
56      private LinkedHashSet<StringData> dataset;
57      public FileHandler(String fileName) {
58          this.dataset = new LinkedHashSet<StringData>();
59          loadData(fileName);
60      }
61      private void loadData(String fileName) {
62          try {
63              String str = new String(Files.readAllBytes(Paths.get(
                    fileName)), StandardCharsets.UTF_8);
64              StringTokenizer st = new StringTokenizer(str, "\n");
65              while (st.hasMoreTokens()) {
66                  String[] data = st.nextToken().split(" ");
67                  double timestamp = Double.parseDouble(data[0]);
68                  int va = Integer.parseInt(data[1]);
69                  int vb = Integer.parseInt(data[2]);
70                  boolean connected = (data[3].compareTo("up") == 0) ? true
                        : false;
71                  this.dataset.add(new StringData(timestamp, va, vb,
                        connected));
72              }
```

```
73            st = null;
74            str = null;
75         } catch (IOException e) {
76            e.printStackTrace();
77         }
78         assert (this.dataset.size() == 0) : "Não há dados";
79      }
80      public void run(Parameter p) {
81         new Framework(p, dataset).run();
82         this.dataset = null;
83         p = null;
84      }
85   }
```

A.2.2 StringData.java

```
86   package filehandler;
87   public class StringData {
88      public double timeStamp;
89      public int va;
90      public int vb;
91      public boolean connected;
92      public StringData(double timeStamp, int va, int vb, boolean
           connected) {
93         this.timeStamp = timeStamp;
94         this.va = va;
95         this.vb = vb;
96         this.connected = connected;
97      }
98   }
```

A.3 FRAMEWORK PACKAGE

A.3.1 Framework.java

```
99   package framework;
100  import java.io.FileWriter;
101  import java.io.IOException;
102  import java.util.ArrayList;
103  import java.util.HashMap;
104  import java.util.HashSet;
105  import java.util.Iterator;
106  import filehandler.StringData;
107  import main.Parameter;
108  public class Framework {
109     public int countVehicle;
110     private int countMalicious;
111     private HashSet<StringData> dataset;
112     private HashMap<Integer, Vehicle> vehicles;
```

```
113     public Parameter parameter;
114     // Resultados
115     public HashSet<Double> groups;
116     public int maxVG;
117     public double maxPG;
118     private int maxLG1;
119     private int maxLG2;
120     private int maxLG3;
121     private ArrayList<Integer> vehiclesInLevel1;
122     private ArrayList<Integer> vehiclesInLevel2;
123     private ArrayList<Integer> vehiclesInLevel3;
124     public Framework(Parameter parameter, HashSet<StringData> data) {
125        this.parameter = parameter;
126        this.dataset = data;
127        this.vehicles = new HashMap<Integer, Vehicle>();
128        this.countVehicle = 0;
129        this.countMalicious = (parameter.malicious_rate == 0) ? 0 : (
              parameter.number_nodes * parameter.malicious_rate) / 100;
130        // Resultados
131        this.groups = new HashSet<Double>();
132        this.vehiclesInLevel1 = new ArrayList<Integer>();
133        this.vehiclesInLevel2 = new ArrayList<Integer>();
134        this.vehiclesInLevel3 = new ArrayList<Integer>();
135     }
136     public void run() {
137        for (Iterator<StringData> iterator = dataset.iterator();
              iterator.hasNext();) {
138           StringData data = iterator.next();
139           double ts = data.timeStamp;
140           boolean connected = data.connected;
141           Vehicle va = getVehicle(data.va);
142           Vehicle vb = getVehicle(data.vb);
143           // Vizinhança
144           va.neighborwood.put(vb, connected);
145           vb.neighborwood.put(va, connected);
146           // Contato
147           va.setContact(vb, ts);
148           vb.setContact(va, ts);
149           // Tempo de permanência
150           va.setStay(ts);
151           vb.setStay(ts);
152           // Confiança (analisa o comportamento e rede social)
153           va.setTrust(ts);
154           vb.setTrust(ts);
155           // Gerenciamento de grupo
156           va.setGroup(vb, ts);
157           vb.setGroup(va, ts);
158           // Atualiza os níveis no grupo
159           va.setLevel(ts);
160           vb.setLevel(ts);
161        }
```

```
162         setLog();
163     }
164     private Vehicle getVehicle(int id) {
165         if (vehicles.containsKey(id))
166             return this.vehicles.get(id);
167         else {
168             boolean malicious = false;
169             if (parameter.malicious_rate > 0 && this.countMalicious > 0
                    && ++this.countVehicle % 2 == 0) {
170                 malicious = true;
171                 this.countMalicious--;
172             }
173             Vehicle v = new Vehicle(id, malicious, this.countVehicle,
                    this);
174             this.vehicles.put(id, v);
175             return v;
176         }
177     }
178     public void maxPG(double time) {
179         this.maxPG = (this.maxPG > time) ? this.maxPG : time;
180     }
181     public void maxVG(int size) {
182         this.maxVG = (this.maxVG > size) ? this.maxVG : size;
183     }
184     public void maxLevelGroup(int level, int max) {
185         if (level == 1)
186             this.maxLG1 = (this.maxLG1 > max) ? this.maxLG1 : max;
187         else if (level == 2)
188             this.maxLG2 = (this.maxLG2 > max) ? this.maxLG2 : max;
189         else if (level == 3)
190             this.maxLG3 = (this.maxLG3 > max) ? this.maxLG3 : max;
191     }
192     public void vehicleInLevel(int id, int level) {
193         if (level == 1 && !vehiclesInLevel1.contains(id))
194                 vehiclesInLevel1.add(id);
195         else if (level == 2 && !vehiclesInLevel2.contains(id)) {
196                 vehiclesInLevel2.add(id);
197         }
198         else if (level == 3 && !vehiclesInLevel3.contains(id))
199                 vehiclesInLevel3.add(id);
200     }
201     private void setLog() {
202         try {
203             /* **** TOTAIS **** */
204             String output = "";
205             output += "N1/grupo: " + this.maxLG1 + "\n";
206             output += "N2/grupo: " + this.maxLG2 + "\n";
207             output += "N3/grupo: " + this.maxLG3 + "\n";
208             output += "N1/veiculos: " + this.vehiclesInLevel1.size() + "
                    \n";
```

```
209             output += "N2/veiculos: " + this.vehiclesInLevel2.size() + "
                    \n";
210             output += "N3/veiculos: " + this.vehiclesInLevel3.size() + "
                    \n";
211             output += "Max V/grupo: " + this.maxVG + "\n";
212             output += "Max P/grupo: " + this.maxPG + "\n";
213             output += "Nr Grupos: " + this.groups.size() + "\n";
214             String filename = "autolog_"+parameter.number_nodes+"_" +
                    parameter.malicious_rate + "-" + parameter.
                    social_trust_rate + "-" + parameter.trust_time
215                     + "-" + parameter.beta_point + "-" + parameter.n + "-"
                            + parameter.x + "-" + parameter.w + ".txt";
216             FileWriter log = new FileWriter(filename);
217             log.write(output);
218             log.close();
219         } catch (IOException e) {
220             e.printStackTrace();
221         }
222     }
223 }
```

A.3.2 Group.java

```
224 package framework;
225 import java.util.Iterator;
226 import java.util.TreeSet;
227 public class Group {
228     public double id;
229     public double timeStamp;
230     private TreeSet<Vehicle> members;
231     public Group(double id, double ts) {
232         this.id = id;
233         this.timeStamp = ts;
234         this.members = new TreeSet<Vehicle>();
235     }
236     public void add(Vehicle v) {
237         if(!this.members.contains(v))
238         this.members.add(v);
239     }
240     public void addAll(TreeSet<Vehicle> members) {
241         for (Iterator<Vehicle> iterator = members.iterator(); iterator.
                hasNext();)
242             iterator.next().g = this;
243         this.members.addAll(members);
244     }
245     public void del(Vehicle v) {
246         if (this.members.contains(v))
247             this.members.remove(v);
248     }
249     public int size() {
```

```
250         return this.members.size();
251     }
252     public boolean contains(Vehicle vehicle) {
253         return this.members.contains(vehicle);
254     }
255     public Iterator<Vehicle> iterator() {
256         return this.members.iterator();
257     }
258     public TreeSet<Vehicle> getMembers() {
259         return this.members;
260     }
261     public void clear() {
262         this.members.clear();
263     }
264     @Override
265     public String toString() {
266         String out = "G"+id+" ["+members.size()+"] [";
267         for (Vehicle v : members)
268             out += v.id + " ";
269         out = out.substring(0, out.length() - 1);
270         out += "]";
271         return out;
272     }
273 }
```

A.3.3 LocalTrust.java

```
274 package framework;
275 import main.Parameter;
276 public class LocalTrust implements Comparable<LocalTrust> {
277     private Vehicle vehicle;
278     private double timeStamp;
279     private int level;
280     private int alpha; // contact + stay;
281     private boolean beta; // trustable
282     private double behavior;
283     private Parameter parameter;
284     public LocalTrust(Vehicle vehicle, double timeStamp, Parameter
          parameter) {
285         this.vehicle = vehicle;
286         this.timeStamp = timeStamp;
287         this.level = 0;
288         this.alpha = 0;
289         this.beta = false;
290         this.behavior = 0.0;
291         this.parameter = parameter;
292     }
293     public Vehicle getVehicle() {
294         return vehicle;
295     }
```

```
296     public double getTimeStamp() {
297        return timeStamp;
298     }
299     public void setTimeStamp(double timeStamp) {
300        if (this.timeStamp < timeStamp)
301           this.timeStamp = timeStamp;
302     }
303     public void addContact() {
304        this.alpha++;
305     }
306     public void delContact() {
307        this.alpha--;
308        this.alpha = (this.alpha > 0) ? this.alpha : 0;
309
310     }
311     public int getPoints() {
312        return this.alpha;
313     }
314     public void addStay(int points) {
315        this.alpha += points;
316     }
317     public void delStay(int points) {
318        this.alpha -= points;
319        this.alpha = (this.alpha > 0) ? this.alpha : 0;
320     }
321     public boolean isTrustable() {
322        return this.beta;
323     }
324     public void setTrustable(boolean trustable) {
325        this.beta = trustable;
326     }
327     public int getLevel() {
328        return level;
329     }
330     public void setLevel(Framework fw) {
331
332        if (this.alpha >= (parameter.n + parameter.w) && this.beta && !
              this.vehicle.malicious) {
333           this.level = 3;
334           fw.vehicleInLevel(this.vehicle.id, 3);
335        } else if (this.alpha >= (parameter.n + parameter.x) && this.
              beta && !this.vehicle.malicious) {
336           this.level = 2;
337           fw.vehicleInLevel(this.vehicle.id, 2);
338        } else if (this.alpha >= parameter.n) {
339           this.level = 1;
340           fw.vehicleInLevel(this.vehicle.id, 1);
341
342        }
343     }
344     public void setLevel(int level, Framework fw) {
```

```
345         if (level != this.level) {
346            this.level = level;
347            if (level == 1) {
348               fw.vehicleInLevel(this.vehicle.id, 1);
349
350            } else if (level == 2) {
351               fw.vehicleInLevel(this.vehicle.id, 2);
352            } else if (level == 3) {
353               fw.vehicleInLevel(this.vehicle.id, 3);
354            }
355         }
356      }
357      public double getBehavior() {
358         return behavior;
359      }
360      public void setBehavior(double behaviorPoints) {
361         this.behavior = this.behavior + (behaviorPoints / 1000);
362         this.behavior = (this.behavior > 0.0) ? this.behavior : 0;
363         this.setBeta();
364      }
365      private void setBeta() {
366         if ((this.behavior) >= (parameter.beta_point))
367            this.beta = true;
368         else
369            this.beta = false;
370
371      }
372      @Override
373      public String toString() {
374         return vehicle.id + ": alpha=" + this.alpha + " Comportameto =
               " + this.behavior + " beta=" + this.beta + " Lvl="
375               + this.level;
376         // return vehicle.id+"";
377      }
378      @Override
379      public int compareTo(LocalTrust o) {
380         if (vehicle.id > o.vehicle.id)
381            return 1;
382         else if (vehicle.id < o.vehicle.id)
383            return -1;
384         return 0;
385      }
386   }
```

A.3.4 Vehicle.java

```
387   package framework;
388   import java.util.HashMap;
389   import java.util.Iterator;
390   import java.util.Random;
```

```
391 public class Vehicle implements Comparable<Vehicle> {
392     public int id;
393     private int idai; // id auto increment
394     public Group g;
395     public boolean malicious;
396     public HashMap<Vehicle, Boolean> neighborwood;
397     public HashMap<Vehicle, LocalTrust> localTrust;
398     private Framework fw;
399     private static boolean ACCEPT = true;
400     public Vehicle(int id, boolean malicious, int idai, Framework fw)
            {
401         this.id = id;
402         this.malicious = malicious;
403         this.idai = idai;
404         this.g = null;
405         this.neighborwood = new HashMap<Vehicle, Boolean>();
406         // this.localTrust = new LinkedList<LocalTrust>();
407         this.localTrust = new HashMap<Vehicle, LocalTrust>();
408         this.localTrust.put(this, new LocalTrust(this, 0, fw.parameter)
                ;
409         this.fw = fw;
410     }
411     /* **** ALPHA **** */
412     public void setContact(Vehicle v, double ts) {
413         LocalTrust t = getTrust(v, ts);
414         t.setTimeStamp(ts);
415         if (this.neighborwood.get(v))
416             t.addContact(); // Se o contato está conectado
417         else
418             t.delContact(); // Se o contato está desconectado
419     }
420     public void setStay(double ts) {
421         for (LocalTrust t : localTrust.values()) {
422             Vehicle v = t.getVehicle();
423             double interval = ts - t.getTimeStamp();
424             if(interval > 0 && v != this && this.neighborwood.
                    containsKey(v)) {
425                 if (interval > fw.parameter.trust_time) {
426                     int points = (int) Math.ceil(interval / fw.parameter.
                            trust_time);
427                     if (this.neighborwood.get(t.getVehicle()))
428                         t.addStay(points);
429                     else
430                         t.delStay(points);
431                 }
432             }
433         }
434     }
435     /* **** TRUST **** */
436     public LocalTrust getTrust(Vehicle v, double ts) {
437         LocalTrust t = localTrust.get(v);
```

```
438         if (t != null)
439             return t;
440         LocalTrust nt = new LocalTrust(v, ts, fw.parameter);
441         // Social Trust Rate
442         if (++this.idai % fw.parameter.social_trust_rate == 0)
443             nt.setBehavior(fw.parameter.beta_point);
444 
445         localTrust.put(v, nt);
446         return nt;
447     }
448     private LocalTrust getTrust(Vehicle v) {
449         return localTrust.get(v);
450     }
451     public void setTrust(double ts) {
452         if (this.g != null)
453             for (Iterator<Vehicle> iterator = this.g.getMembers().
                    iterator(); iterator.hasNext();) {
454                 Vehicle v = iterator.next();
455                 LocalTrust t = getTrust(v, ts);
456                 t.setTimeStamp(ts);
457                 if (v.malicious && !v.getBehavior())
458                     t.setBehavior(-1);
459                 else
460                     t.setBehavior(1);
461             }
462     }
463     /* **** BEHAVIOR **** */
464     public boolean getBehavior() {
465         return (new Random(System.currentTimeMillis()).nextInt(2) == 0)
                 ? true : false;
466     }
467     /* **** GROUPS **** */
468     public void setGroup(Vehicle v, double ts) {
469         if (this.localTrust.get(v).getPoints() >= fw.parameter.n) {
470             if (this.g != null && v.g != null && this.g != v.g) {
471                 // mescla grupos
472                 // Cria um novo grupo
473                 if (inviteToGroup(v) == ACCEPT) {
474                     fw.maxPG(ts - this.g.timeStamp);
475                     Group g = new Group(ts, ts);
476                     mergeUpdate(this, v, g, ts);
477                     fw.maxVG(g.getMembers().size());
478                     if (!fw.groups.contains(g.id)) {
479                         fw.groups.add(g.id);
480                     }
481                 }
482 
483             } else if (this.g == null && v.g == null && inviteToGroup(v)
                     == ACCEPT) {
484                 // cria um novo gupo
485                 Group g = new Group(ts, ts);
```

```
486             g.add(this);
487             this.g = g;
488             g.add(v);
489             v.g = g;
490             if (!fw.groups.contains(g.id)) {
491                fw.groups.add(g.id);
492             }
493             fw.maxVG(2);
494             this.localTrust.get(v).setLevel(1, fw);
495             v.localTrust.get(this).setLevel(1, fw);
496
497          } else if (this.g == null && v.g != null && inviteToGroup(v)
                == ACCEPT
498                && consensus(v, this, ts) == ACCEPT) {
499             // a vai para o gb
500             v.g.add(this);
501             fw.maxVG(v.g.getMembers().size());
502             fw.maxPG(ts - v.g.timeStamp);
503             v.g.timeStamp = ts;
504             this.g = v.g;
505
506          } else if (this.g != null && v.g == null && inviteToGroup(v)
                == ACCEPT
507                && consensus(this, v, ts) == ACCEPT) {
508             // b vai para o ga
509             this.g.add(v);
510             fw.maxVG(this.g.getMembers().size());
511             fw.maxPG(ts - this.g.timeStamp);
512             this.g.timeStamp = ts;
513             v.g = this.g;
514          }
515       } else {
516          if (this.g != null && this.g.getMembers().contains(v) &&
              consensus(v, ts)) {
517             g.del(v);
518             v.g = null;
519          }
520       }
521    }
522    private void mergeUpdate(Vehicle va, Vehicle vb, Group g, double
       ts) {
523       HashMap<Vehicle, LocalTrust> tmp = new HashMap<Vehicle,
          LocalTrust>();
524       // Trust A
525       HashMap<Vehicle, LocalTrust> trustA = new HashMap<Vehicle,
          LocalTrust>();
526       for (Iterator<Vehicle> iterator = va.g.getMembers().iterator();
           iterator.hasNext();) {
527          Vehicle v = iterator.next();
528          trustA.putAll(v.localTrust);
529          g.add(v);
```

```
530             v.g = g;
531         }
532         // Trust B
533         HashMap<Vehicle, LocalTrust> trustB = new HashMap<Vehicle,
                LocalTrust>();
534         for (Iterator<Vehicle> iterator = vb.g.getMembers().iterator();
                iterator.hasNext();) {
535             Vehicle v = iterator.next();
536             trustB.putAll(v.localTrust);
537             g.add(v);
538             v.g = g;
539         }
540         // updateGA
541         for (Iterator<Vehicle> iterator = va.g.getMembers().iterator();
                iterator.hasNext();) {
542             Vehicle v = iterator.next();
543             tmp.putAll(v.localTrust);
544             v.localTrust.putAll(trustB);
545             v.localTrust.putAll(tmp);
546             tmp.clear();
547         }
548         // updateGB
549         for (Iterator<Vehicle> iterator = vb.g.getMembers().iterator();
                iterator.hasNext();) {
550             Vehicle v = iterator.next();
551             tmp.putAll(v.localTrust);
552             v.localTrust.putAll(trustA);
553             v.localTrust.putAll(tmp);
554             tmp.clear();
555         }
556         trustA = null;
557         trustB = null;
558         tmp = null;
559     }
560     private boolean inviteToGroup(Vehicle v) {
561         return (!this.malicious) ? true : false;
562     }
563     // Consenso para adicionar no grupo
564     private boolean consensus(Vehicle va, Vehicle vb, double ts) {
565         if (!vb.malicious) {
566             // Atualiza a tabela de confiança dos demais membros do
                    grupo sobre vb
567             for (Iterator<Vehicle> iterator = va.g.getMembers().iterator
                    (); iterator.hasNext();)
568                 iterator.next().getTrust(vb, ts);
569             return true;
570         }
571         return false;
572     }
573     // Consenso para atribuir o nível
574     private boolean consensus(Vehicle v, int level, double ts) {
```

```
575         if (v.malicious || this.malicious)
576             return false;
577         boolean consensus = false;
578         if (level > 1 && existInLevel(level)) {
579             int yes = 0, no = 0;
580             for (Iterator<Vehicle> it = g.getMembers().iterator(); it.
                    hasNext();) {
581                 Vehicle member = it.next();
582                 LocalTrust t = this.getTrust(member);
583                 if (member != v && t != null && t.getLevel() >= level) {
584                     LocalTrust tm = member.getTrust(v);
585                     if (tm != null && tm.getLevel() == level)
586                         yes++;
587                     else
588                         no++;
589                 }
590             }
591             consensus = (yes >= no) ? true : false;
592         } else
593             consensus = true;
594         if (consensus) {
595             // Atualiza todos
596             for (Iterator<Vehicle> it = g.getMembers().iterator(); it.
                    hasNext();) {
597                 Vehicle member = it.next();
598                 LocalTrust mt = member.getTrust(v);
599                 if (mt != null && mt.getLevel() != level)
600                     mt.setLevel(level, fw);
601             }
602         }
603         return consensus;
604     }
605     // Consenso para remover do grupo
606     private boolean consensus(Vehicle v, double ts) {
607         if (this.malicious)
608             return false;
609         for (Iterator<Vehicle> iterator = g.getMembers().iterator();
                iterator.hasNext();) {
610             LocalTrust t = iterator.next().localTrust.get(v);
611             if (t != null && t.getPoints() >= fw.parameter.n)
612                 return false;
613         }
614         return true;
615     }
616     /* **** LEVEL **** */
617     public void setLevel(double ts) {
618         if (g != null) {
619             int l1 = 0, l2 = 0, l3 = 0;
620             for (Iterator<Vehicle> it = g.getMembers().iterator(); it.
                    hasNext();) {
621                 Vehicle v = it.next();
```

```
622                 LocalTrust t = this.localTrust.get(v);
623                 if (v != this && t != null) {
624                     int oldLevel = t.getLevel();
625                     t.setLevel(fw);
626                     if (t.getLevel() != oldLevel) {
627                         if (!consensus(v, t.getLevel(), ts))
628                             t.setLevel(oldLevel, fw);
629                     }
630 
631                     if (t.getLevel() == 1)
632                         l1++;
633                     else if (t.getLevel() == 2)
634                         l2++;
635                     else if (t.getLevel() == 3)
636                         l3++;
637                 }
638             }
639             fw.maxLevelGroup(1, l1);
640             fw.maxLevelGroup(2, l2);
641             fw.maxLevelGroup(3, l3);
642         }
643     }
644     private boolean existInLevel(int level) {
645         int count = 0;
646         for (Iterator<Vehicle> iterator = this.g.getMembers().iterator
            (); iterator.hasNext();) {
647             LocalTrust t = this.localTrust.get(iterator.next());
648             if (t != null && t.getLevel() >= level)
649                 if (++count >= 2)
650                     return true;
651         }
652         return false;
653     }
654     @Override
655     public int compareTo(Vehicle o) {
656         if (this.id > o.id)
657             return 1;
658         else if (this.id < o.id)
659             return -1;
660         return 0;
661     }
662     @Override
663     public String toString() {
664         if (this.malicious)
665             return this.id + "M ";
666 
667         return this.id + "";
668     }
669 }
```