Detecção incremental de comportamento malicioso em VANETs

Fernando Dutra[1,2], **Kenniston Bonfim**[1], **Carlos Travagini**[1]
Rodolfo I. Meneguette[4], **Aldri Santos**[3], **Lourenço Alves Pereira**[1]

[1]Divisão de Ciência da Computação
Instituto Tecnológico de Aeronáutica (ITA) – São José dos Campos, SP – Brasil

[2]Exército Brasileiro – Brasil

[3]Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG) – Belo Horizonte, MG – Brasil

[4]Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo (USP) – São Carlos, SP – Brasil

{dutrafsd,kenniston,carlos.travagini,ljr}@ita.br,aldri@dcc.ufmg.br

Abstract. *Position forgery in VANETs may cause disastrous effects on traffic. This work presents the DISMISS-BSM, an algorithm that is able to detect this kind of misbehavior in C-ITS enviroment, resorting to two new features and a new way of grouping the messages in sliding analysis windows of incremental size, combining agility and efficiency in the misbehavior detection. To evaluate the performance, five machine learning models were used, showing remarkable results in detecting random offset type attacks, surpassing other works in the area, and the cost-benefit of the Decision Tree model, with an average training and inference time of,* 47.617 *and* 0.085 *sec, respectively.*

Resumo. *Falsificação de posição em VANETs pode causar efeitos desastrosos no trânsito, sendo um problema em aberto na literatura. Este trabalho apresenta o DISMISS-BSM, um algoritmo capaz de identificar os ataques de falsificação de posição, utilizando dois novos atributos preditores e uma nova forma de agrupamento das mensagens em janelas deslizantes de análise de tamanho incremental, aliando agilidade e eficácia na detecção. Para avaliação do desempenho foram utilizados cinco modelos de aprendizado de máquina. Destacam-se o desempenho na detecção dos ataques random offset, superando os demais trabalhos da área, e o custo-benefício do modelo Decision Tree, com média de tempo de treinamento e inferência de,* $47,617$ *e* $0,085$ *seg, respectivamente.*

1. Introdução

Os sistemas de transporte inteligentes cooperativos (C-ITS) interconectam todas as entidades participantes do trânsito, como os próprios veículos, a infraestrutura de trânsito e até mesmo os pedestres. Esses sistemas podem ser implementados em qualquer ambiente, sendo especialmente adaptáveis para o funcionamento nas chamadas cidades inteligentes, trazendo diversos benefícios para a melhoria do fluxo de tráfego, colaborando para que os veículos permaneçam menos tempo nas vias. Portanto, melhoram a qualidade

de vida ao proporcionar menores engarrafamentos, consumo de combustíveis e emissões de poluentes, além de tornar o trânsito mais seguro e eficiente, diminuindo o número de acidentes [Gohar and Nencioni 2021].

C-ITSs representam infraestruturas críticas relacionadas ao sistema de transporte, sendo essenciais em ambiente de cidades inteligentes. Falhas nesses sistemas podem causar danos a vidas e comprometer cadeias de suprimentos. Observa-se que C-ITSs apresentam vulnerabilidades de segurança exploradas por agentes mal intencionados, apesar dos esforços de segurança proativos [van der Heijden et al. 2019], como o uso do padrão IEEE 1609.2 [IEEE 2016] que provê mecanismos de autenticação, permitindo um controle sobre os participantes e reduzindo os vetores de ataques nessas redes. Periodicamente, os veículos informam dados relevantes para o tráfego, visando proporcionar um ambiente de consciência situacional que possa indicar o estado de cada veículo na via. Tais mensagens são chamadas de *Basic Safety Messages* (BSMs), definidas no padrão SAE J2735, e permitem a propagação periódica de informações de localização, velocidade, sentido da direção entre outras, tipicamente transmitidas 10 vezes por segundo [Committee 2020]. Entretanto, faltam mecanismos para atestar a corretude dos dados dessas mensagens. Como exemplo de aplicação, destaca-se a formação de comboio (*platoon*) [Kamoi et al. 2021], em que um conjunto de veículos é agrupado para proporcionar maior eficiência no trânsito, deslocando-se com menores distâncias entre os veículos e velocidade sincronizada. Logo, comportamentos maliciosos que alterem o conteúdo das mensagens de sinalização (e.g., BSMs) podem causar efeitos desastrosos.

As abordagens existentes na detecção de falsificação de posição em BSM ou ocorrem para cada mensagem [van der Heijden et al. 2018, Ercan et al. 2021, Gyawali and Qian 2019], ou por conjunto de mensagens [So et al. 2018]. Por outro lado, em relação às formas de detecção, usa-se verificações de plausibilidade [van der Heijden et al. 2018, So et al. 2018], ou aprendizado de máquina [So et al. 2018, Ercan et al. 2021, Gyawali and Qian 2019]. No entanto, ao observar as soluções mais citadas [van der Heijden et al. 2018, So et al. 2018, Ercan et al. 2021, Gyawali and Qian 2019], fica evidente a necessidade de aperfeiçoamento ou criação de novos preditores para atingir maior efetividade e usabilidade, destacando-se como pontos de investigação em aberto a obtenção de eficácia aceitável, uso de *features* reproduzíveis em cenários realísticos, e métricas que envolvam tempos de treinamento e inferência.

Este artigo apresenta o DISMISS-BSM (***D**etect**I**on **S**ystem for **MIS**behavior in **BSM***), um algoritmo para detecção de conteúdo falsificado em *Basic Safety Messages* (BSM). Para tanto, foram desenvolvidos dois novos preditores de falsificação, um de intensidade do sinal e outro de conformidade de deslocamento, que fazem uso apenas de informações disponíveis ao hospedeiro ($host$) em cenários reais. Além disso, foi concebido um sistema de múltiplas janelas deslizantes de análise com diferentes tamanhos, que podem ser usadas simultaneamente, consequentemente conferindo agilidade e eficácia na detecção dos ataques. Foram realizados 630 experimentos para a avaliação do desempenho dos preditores e das janelas deslizantes de análise, para os quais utilizou-se cinco modelos de aprendizado de máquina (RF (*random forest*), DT (*decision tree*), K-NN (*k-nearest neighbors*), MLP (*multilayer perceptron*) e LSTM (*long short-term memory*)) e três métricas (f1-score, tempo de treinamento e tempo de teste). Como contribuição, destacam-se a possibilidade de realização de inferência com apenas 2 BSMs, o excelente

desempenho do preditor de Conformidade de deslocamento na detecção dos ataques do tipo *random offset*, obtendo *f1-score* de $1,0$, superando os demais trabalhos da área e o custo-benefício do modelo DT, que obteve pontuação média *f1-score* de $0,878$, com média de tempo de treinamento normalizada de $47,617$ segundos e média de tempo de inferência normalizada de $0,085$ segundos.

O restante do artigo está organizado da seguinte maneira: a Seção 2 apresenta os trabalhos relacionados. A Seção 3 descreve a proposta. A Seção 4 detalha a avaliação e os resultados obtidos. A Seção 5 apresenta a conclusão e os trabalhos futuros.

2. Trabalhos Relacionados

A correta identificação e solução para o problema de falsificação do conteúdo das *basic safety messages* (BSM) ainda permanece aberto na literatura. Muito embora, haja um infraestrutura de chaves possibilitando que agentes externos sejam incapazes de injetar, interceptar ou alterar mensagens no sistema, veículos pertencentes a rede podem ser comprometidos (e.g., por malwares) e, com isso, informar valores inadequados capazes de comprometer o funcionamento adequado do sistema [van der Heijden et al. 2018]. Assim, o trabalho de [van der Heijden et al. 2018] trata da análise de plausibilidade do conteúdo recebido e tenta filtrar as mensagens recebidas com base em seu estado local. No entanto, percebe-se a falta de generalização do método e um estudo aprofundado que permita medir melhor a robustez face a diferentes cenários. Já em outra abordagem, [So et al. 2018], além das verificações de plausibilidade, passou a utilizar-se de modelos de aprendizado de máquina para realizar a detecção das mensagens falsificadas. Porém, a forma de pré-processamento exigia que, a cada verificação, todo o conjunto de mensagens fosse considerado, fazendo com que o veículo seja obrigado a armazenar todas as mensagens recebidas de todos os veículos para que execute a detecção continuamente.

Dentre as abordagens que fazem uso de aprendizado de máquina, os artigos [Ercan et al. 2021, Ercan et al. 2022] apresentam novas *features* e combinação de *features* para ajudar na detecção das mensagens falsificadas, extraindo novas informações dos dados já disponíveis nas BSM, aperfeiçoando a detecção de certos tipos de ataque, embora faltem dados sobre o treinamento e propagação dos modelos. Outra abordagem, [Sharma and Jaekel 2022], faz uso da infraestrutura para obter o conjunto de mensagens e treiná-las. Assim, é possível derivar um modelo de modo *offline* e posteriormente propagá-lo aos veículos. No entanto, o processo de treinamento é centralizado e os veículos recebem o modelo para aferição, logo o tempo de propagação pode ser insuficiente em ambientes que compartilham modelos e identificam novos ataques. Deve se observar a criticalidade da aplicação, porque os atrasos podem ocasionar riscos aos usuários do sistema. Além disso, o método é muito dependente de uma alta densidade de RSU (*road side unit*).

Como destacado em [De Carvalho Bertoli et al. 2021], além de produzir modelos de aprendizado, o passo no sentido de sua operacionalização (MLOps) é importante. Logo, a observância de modelos plausíveis de implementação nos dispositivos-alvo deve ser considerado. Fatores como tempo de inferência, capacidade de armazenamento requerida, tamanho do modelo para compartilhamento e possibilidade evolucionária dos treinamento devem ser levados em consideração. Como observado, os trabalhos relacionados focam apenas em produzir um modelo com bom desempenho na fase de treinamento. No entanto, faltam com uma discussão relacionada à aplicabilidade da solução

em ambiente real. [Hawlader et al. 2021] geram uma série de modelos em função do dataset VeReMi e testam em uma simulação em que os atacantes alternam entre atividade legítima e maliciosa. Por um lado, essa abordagem de validação traz realismo à solução; porém, requer 23 mensagens para inferir um comportamento anômalo. Assim, pode-se derivar uma estratégia furtiva para contornar esse mecanismo de proteção.

Os níveis do RSSI presentes no dataset VeReMi mostraram-se realistas em comparação com dados empíricos do projeto iTETRIS [So et al. 2019]. Nesse estudo os autores tomam uma abordagem *stateless* para avaliação e confrontam o RSSI percebido pelo receptor e a posição do veículo emissor. Assim, com base no modelo de atenuação do sinal é possível verificar a coincidência da posição reportada e sua atenuação. No entanto, essa abordagem está suscetível ao erro estocástico inerente do sistema, mas, por outro lado, ela serve como medida auxiliar a ser incorporada ao dataset.

A maioria dos trabalhos realizam o treinamento de modelos de forma centralizada. No entanto, questões de privacidade e o treinamento local podem ser abordados como diferencial no contexto de VANETS. Principalmente quando levados em consideração implantações de soluções em ambiente real, em que pode se observar evoluções nos padrões de ataque e compartilhamento de novos mecanismos de proteção. Nesse sentido, [Uprety et al. 2021] foi o único artigo a apresentar uma proposta para essa solução. Os autores focam em questões de privacidade e evidenciam resultados promissores. Porém, o tempo de treinamento não foi abordado e atributos que relacionam-se com a sequência de BSMs também ficam de fora do estudo.

Como será observado, o DISMISS-BSM diferencia-se das outras soluções por apresentar um algoritmo capaz de prover um treinamento descentralizado (local) e realizar a detecção de falsificação de posição com janelas deslizantes de tamanho variado, de 2 a 23 BSMs. Assim, destaca-se pelo baixo tempo de treinamento e inferência, em contraste com os demais trabalhos, porém, ainda é capaz de atingir alto desempenho de detecção conforme o tamanho da janela de análise aumenta.

3. Algoritmo DISMISS-BSM

Esta seção apresenta os pormenores do algoritmo proposto DISMISS-BSM e o ecossistema visualizado ao seu funcionamento, que incluem as ferramentas necessárias para a detecção de anomalias no contexto de C-ITS. O DISMISS-BSM armazena e separa as mensagens em janelas deslizantes de análise, conforme o número de mensagens aumenta, acionando então o pré-processamento dos preditores. Por fim, realiza a detecção propriamente dita, por meio de um modelo de aprendizado de máquina. O DISMISS-BSM apresenta duas novidades principais, sendo a primeira a forma de processamento da detecção em múltiplas janelas deslizantes de análise, 2, 3, 8, 13, 18 e 23 BSMs, aliando, portanto, agilidade e eficácia na detecção de falsificação de BSMs. Outra diferença é a concepção de dois novos preditores de falsificação, Intensidade do sinal e Conformidade de deslocamento, além do emprego da combinação de ambos preditores. Embora outros trabalhos já tenham utilizado o RSSI e a distância entre os veículos como atributos de detecção, DISMISS-BSM diferencia-se na forma de empregá-los, agrupando em janelas de análise.

3.1. Arquitetura multinível

Assim como elencado em [Grover et al. 2021], as soluções modernas de computação raramente são resolvidas em apenas um nível, pois suas demandas possuem um alto grau de

exigência tanto de potência computacional, quanto de latência e confiabilidade. Assim, a fim de prover suporte para o algoritmo DISMISS-BSM, visualizou-se a utilização de uma arquitetura multinível, fracionando a responsabilidade pelas tarefas ao longo de diferentes níveis computacionais, frequentemente combinando computação em nuvem, computação de borda, computação local e dispositivos IoT para sustentar os serviços centrados no usuário. Dessa forma, a Figura 1 ilustra a arquitetura para o DISMISS-BSM, onde as responsabilidades são divididas entre os níveis, permitindo uma grande flexibilidade no processo de identificação de ataques e compartilhamento de soluções.

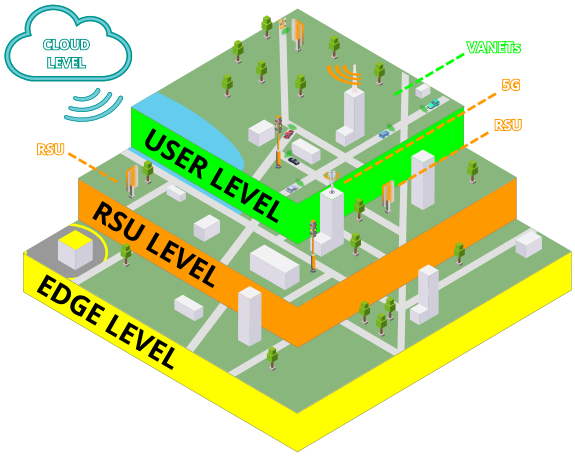

Figura 1. Arquitetura multinível permite alocar tarefas dinamicamente. A nuvem possui maior potência computacional e maior latência, a borda tem grande potência computacional e baixa latência e a infraestrutura gerencia a rede.

A computação de nuvem é o primeiro nível, com grande potência computacional e capacidade de armazenamento, geralmente localizada a grandes distâncias do usuário final, tornando a comunicação mais suscetível a falhas, aumento de latência e diminuição de confiabilidade. Portanto, sua vocação é mais voltada para a solução de problemas computacionais complexos e armazenamento de grandes quantidades de dados, o que permite que seja mais eficiente em tarefas de análise de dados e tarefas sem um requisito estrito de latência e confiabilidade, liberando recursos computacionais mais próximos aos usuários.

O segundo nível é a computação de borda, a qual recebe tarefas que requeiram algum grau de orquestração ou uma maior potência computacional, mas ainda possuam requisitos exigentes de latência e confiabilidade. O próximo nível da hierarquia computacional é a da infraestrutura, a qual comporta as RSU e as estruturas de comunicações móveis. Este nível é responsável pelas tarefas de gerenciamento e controle da rede, e pode ser aperfeiçoado trabalhar em conjunto com os demais níveis para um melhor desempenho, como as redes 5G, que permitem a execução de determinadas funções de rede em computação de borda. O último nível é o do usuário final, incluindo os veículos equipados para participar dos sistemas de transportes inteligentes. Nesse nível estão as VANETs, onde ocorrem as transmissões entre os veículos e com a infraestrutura, os equipamentos coletam e transmitem dados, e podem processar, com alguma restrição em capacidade de processamento, a informação mais iminente. Com uma arquitetura multinível flexível, torna-se possível atender as demandas crescentes dos usuários e sistemas computacionais autônomos, delegando as tarefas entre os níveis conforme a disponibilidade e exigência, portanto, permitindo atender melhor os casos de uso existentes.

3.2. Detecção ágil

Um importante requisito para o algoritmo DISMISS-BSM é a capacidade de identificar ataques o mais rapidamente possível, pois tratam-se de ataques de manipulação de conteúdo de BSMs, no qual os atacantes podem manipular suas informações, como sua posição e velocidade, podendo acarretar em reações de mecanismos de segurança dos demais veículos. BSMs auxiliam nas tomadas de decisões e interferem no sistema de tráfego, portanto, BSMs com conteúdo falsificado podem causar prejuízos desde o primeiro ataque. No entanto, enquanto os ataques persistirem, aumentará a probabilidade de mais veículos serem afetados, consequentemente, é fundamental que esses ataques cessem o quanto antes, a fim de diminuir seu potencial danoso. Um ponto crucial para facilitar a identificação desses ataques, é a construção de modelos capazes de serem utilizados o mais rapidamente possível. Assim, modelos que entreguem menores tempos de execução e treinamento são preferidos. Logo, criou-se um modelo de janela deslizante de análise, que pode ser utilizada a partir do recebimento de apenas 2 BSMs, agilizando a detecção e permitindo, desta maneira, que ações corretivas possam ser aplicadas mais rapidamente, diminuindo a exposição do comportamento malicioso.

3.3. Preparação do dataset VeReMi

O *dataset* VeReMi [van der Heijden et al. 2018] replica diversos cenários de tráfego da cidade de Luxemburgo, utiliza o simulador VEINS, englobando 225 simulações, cada uma contendo um arquivo de rotulação (*ground truth*) e um conjunto de registro das mensagens recebidas por cada veículo. As simulações tem variações de três níveis de densidade veicular, três níveis de densidade de atacantes e cinco diferentes tipos de ataques de falsificação de posição (Tabela 1), e cada combinação de parâmetros foi replicada cinco vezes, totalizando 225 rodadas de simulação. Ainda, os registros das mensagens podem ser de dois tipos, mensagens do tipo 2, que são de dados de telemetria do veículo, e.g. GPS, e mensagens do tipo 3, contendo as BSMs recebidas dos outros veículos por DSRC.

Visando atender as caraterísticas utilizadas nos modelos de detecção, foi necessária a reunião dos rótulos das mensagens, todos os registros recebidos do tipo 3 e os dados da última posição conhecida do destinatário, contida nas mensagens de tipo 2, de forma a concentrar em um registro único, todos os dados disponíveis do dataset, por consequência, viabilizando o pré-processamento conforme a engenharia de atributos. Uma das deficiências conhecidas do dataset VeReMi diz respeito ao ataque *eventual stop*, em que o atacante inicia a simulação transmitindo mensagens legítimas e a chance de iniciar o ataque aumenta a cada mensagem transmitida, eventualmente iniciando um ataque de posição fixa, transmitindo a mesma posição repetidamente, enquanto continua a movimentar-se normalmente. O problema é que as mensagens legítimas, antes do ataque iniciar, estão rotuladas erroneamente como classe 16, confundindo o modelo de detecção, uma vez que essas mensagens deveriam possuir o rótulo 0, de mensagem legítima. Portanto, como solução, apenas as mensagens rotuladas erroneamente foram descartadas, tanto para o treinamento, quanto para o teste dos modelos.

3.4. Engenharia de atributos preditores (*feature engineering*)

A fim de realizar a detecção das mensagens de ataque, foram desenvolvidos dois modelos de preditores a serem usados pelas técnicas de aprendizado de máquina, pois os dados brutos das mensagens não carregam informações relevantes por si só, mas obtém ganho

Tabela 1. Descrição dos 5 tipos ataques de falsificação de posição do VeReMi.

ID: Ataque	Detalhes	Parâmetros
1: *Constant*	Atacante transmite uma localização fixa	$x = 5560, y = 5820$
2: *Constant offset*	Atacante transmite uma localização fixa adicionada à posição real	$\Delta_x = 250, \Delta_y = -150$
4: *Random*	Atacante informa uma posição aleatória dentro da área de simulação	Posição aleatória dentro do ambiente de simulação
8: *Random offset*	Atacante informa uma posição aleatória em um retângulo ao redor do veículo	Δ_x, Δ_y uniformemente aleatórios entre $[-300, 300]$
16: *Eventual stop*	Atacante inicia a simulação agindo legitimamente eventualmente passando a transmitir a mesma posição repetidamente até o fim da simulação	Probabilidade de parada aumenta em 0,025 para cada atualização de posição

de informação quando trabalhados de um modo particular para obter significado dentro do contexto do trabalho, permitindo, assim, a detecção dos ataques de falsificação de posição.

Preditor de Intensidade do sinal No desenvolvimento deste preditor, utilizou-se as propriedades físicas que determinam que, em geral, a distância de transmissão é inversamente proporcional à intensidade do sinal recebido (RSSI), como demonstrado na Figura 2. Foram utilizadas as informações de posições dos veículos remetente e destinatário, de modo a calcular a distância euclideana entre os veículos, que é dada pela distância entre dois pontos $\mathbf{p} = (p_x, p_y, p_z)$ e $\mathbf{q} = (q_x, q_y, q_z)$, e é definida pela equação 1.

$$dist(\mathbf{p}, \mathbf{q}) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2 + (p_z - q_z)^2} \tag{1}$$

Preditor de Conformidade de deslocamento O racional lógico para concepção deste preditor parte da premissa que um veículo que transmite suas BSM legitimamente, tende a manter um deslocamento mais uniforme em contraste ao deslocamento relatado pelos veículos atacantes, que informam posições inconsistentes com seu movimento real. Este preditor, representado na Figura 3, calcula o deslocamento que o veículo remetente percorreria entre duas mensagens e determina a posição que supostamente deveria ocupar na segunda mensagem, para isso, utiliza informações de velocidade e tempo de transmissão contidas na primeira mensagem, conforme a Equação 2, calculando então a diferença entre a posição estimada e a posição informada na segunda mensagem, utilizando a distância euclideana, conforme a equação 3, sendo esta diferença o valor final do preditor.

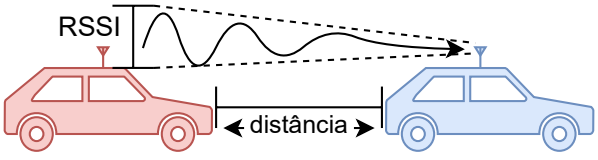

Figura 2. Intensidade do sinal: Conforme a distância aumenta, o RSSI diminui e essa relação pode ser capturada pelos modelos de detecção.

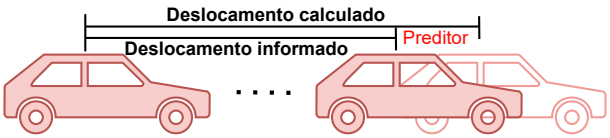

Figura 3. Conformidade de deslocamento: A falsificação de posição causa uma discrepância entre o deslocamento estimado e o informado pelo atacante.

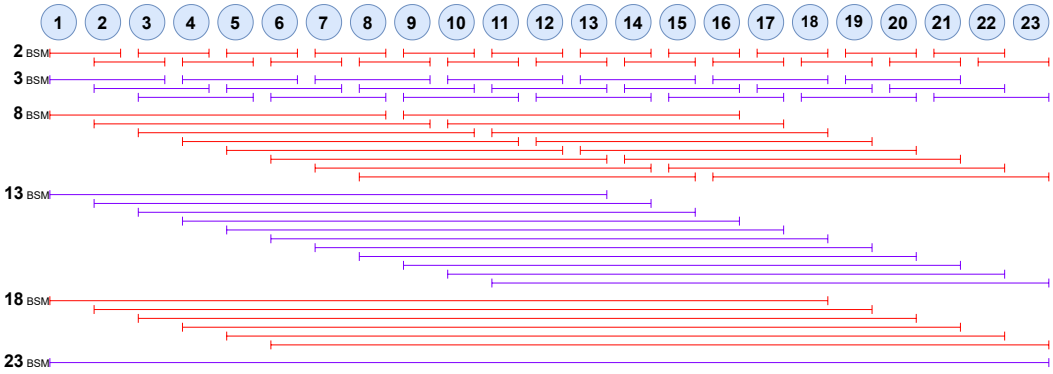

Figura 4. Exemplo da aplicação das janelas deslizantes de análise nas 23 primeiras BSMs recebidas pelo host.

$$posEstimada_{t+1} = pos_t + (vel_t \times \Delta_t) \tag{2}$$

$$dist(posReportada_{t+1}, posEstimada_{t+1}) \tag{3}$$

A $posReportada$ é obtida diretamente da posição informada na BSM, e a $posEstimada$ é proporcional a velocidade do veículo. Onde t representa o tempo atual e serve de índice para a última BSM recebida; pos_t corresponde a última posição informada; vel_t é a última velocidade informada, e Δ_t é a diferença de tempo entra as duas últimas mensagens recebidas.

Preditor de combinação Utiliza a combinação dos preditores de intensidade do sinal e de conformidade de deslocamento, simultaneamente.

3.5. Janela de análise

Quando analisada isoladamente, uma BSM geralmente é incapaz de oferecer informações suficientes para detectar padrões que revelem se esta mensagem é legítima ou anômala. A fim de sanar essa dificuldade, utilizou-se uma abordagem de agrupamento de mensagens para análise, aumentando a possibilidade de um padrão emergir e ser capturado pelos modelos de aprendizado de máquina, conforme a Figura 4.

Para a definição do número de mensagens que compõem a janela de análise, foi feito um estudo do conjunto de dados, onde foram separadas todas as comunicações ocorridas ao longo de todas as simulações. No total, no dataset VeReMi, existem 2.900.722 comunicações entre dois veículos, e o número de mensagens recebidas em cada interação

é variável, tendo um mínimo de 1 e o máximo de 200 mensagens, com uma média de 13, 5 mensagens por interação. Como uma das bases para o desenvolvimento do DISMISS-BSM é a agilidade na detecção, a fim de diminuir o intervalo de tempo de detecção, o tamanho da menor janela de detecção foi definida em 2 BSMs. Para a definição da maior janela de detecção a ser estudada neste trabalho, utilizou-se como referência o estudo feito em [Hawlader et al. 2021], e o maior intervalo de detecção foi estabelecido em 23 mensagens. Para estudo e comparação de custo-benefício, experimentou-se os seguintes tamanhos de janelas de análise: 2, 3, 8, 13, 18 e 23. Permitindo, dessa forma, medir a detecção mais ágil e que apresente maior eficácia no mesmo algoritmo.

3.6. Modelos para predição de falsificação de posição em BSMs

Para avaliar a eficácia dos novos preditores concebidos pela engenharia de atributos preditores, foram selecionados 5 modelos de aprendizado de máquina para a detecção das mensagens de ataque, utilizando as técnicas *k-nearest neighbors* (KNN), *Decision Tree* (DT), *Random Forest* (RF), *Multilayer perceptron* (MLP) and *Long short-term memory* (LSTM), sendo as duas últimas, técnicas de aprendizado profundo. Todos modelos foram treinados e testados no ambiente do *Google's COLAB*, usando a biblioteca Keras para modelar e treinar as redes neurais e a biblioteca scikit-learn para as demais técnicas.

A configuração da arquitetura de nossa MLP, na camada de entrada, depende do tamanho da janela de análise e do preditor utilizado, uma vez que o preditor de intensidade do sinal possui 2 componentes, o preditor de conformidade de deslocamento possui 1 componente e o preditor de combinação possui 3 componentes, a quantidade de neurônios de entrada é o tamanho da janela de análise multiplicado pela quantidade de componentes do preditor utilizado, e.g. se a janela de análise possuir tamanho de 8 mensagens e utilizar o preditor de intensidade do sinal, que contém 2 componentes, o número de neurônios de entrada será 16. A configuração da MLP ainda possui 2 camadas escondidas, contendo 7 neurônios cada e a camada de saída, contendo 2 (binário) ou 6 (multiclasse) neurônios. As camadas escondidas foram configuradas com a função de ativação ReLU (*Rectified Linear Unit*), enquanto a camada de saída foi configurada com a função de ativação Softmax, com o objetivo de aferir a probabilidade de classificação de cada registro. Os modelos foram treinados com o algorítimo de otimização Adam, e a função de perda escolhida foi a *Categorical cross-entropy* para atuar em conjunto com a função de ativação *Softmax*. Foi construído também um modelo LSTM, arquitetura especialmente adequada para séries temporais, utilizando duas camadas LSTM empilhadas, aprofundando a aprendizagem do modelo, cada uma contendo 32 neurônios. O algorítimo de otimização e a camada de saída, bem como sua função de perda e função de ativação, possuem a mesma configuração da arquitetura MLP.

3.7. Planejamento dos experimentos

Para a avaliação do desempenho dos atributos preditores e dos modelos de aprendizado de máquina, desenvolveu-se todas as combinações possíveis de experimentos contidos na Tabela 2, englobando o total de 630 experimentos, de forma a obter uma compreensão mais completa e abrangente dos resultados, permitindo, portanto, aferir mais assertivamente a respeito de seu rendimento. Um dos fatores elencados trata da Abordagem de rotulação adotada, dividida entre os rótulos individuais de cada ataque, classificação binária e classificação multiclasse. Para a abordagem de detecção de cada ataque isoladamente, os rótulos de classe 1**,** 2**,** 4**,** 8 **e** 16 (Tabela 1) foram pré-processados de forma

Tabela 2. Planejamento dos experimentos. Cada um dos fatores são combinados entre si, resultando em 630 experimentos.

Fatores	Níveis
Janelas de análise	2, 3, 8, 13, 18, 23 BSMs
Modelos	RF, DT, KNN, MLP, LSTM
Preditores	Intensidade do sinal, Conformidade de deslocamento, Combinação
Abordagens de rotulação	Cada ataque separadamente (*constant, constant offset, random, random offset, eventual stop*), Binária, Multiclasse
Métricas	F1-score, Tempo de treinamento, Tempo de teste

a manter o rótulo da classe em evidência como a classe positiva, e a classe 0 (mensagens legítimas), como a classe negativa. Por exemplo, na rotulação do Ataque *random offset*, classe 8, todos os registros rotulados como classe 8 e classe 0 permaneceram inalterados, já os registros de todas as demais classes foram ignorados. Isso permitiu medir a proficiência do modelo em aprender a identificar especificamente os registros de classe 8.

Para a abordagem de **classificação binária**, o conjunto de dados possui os registros de seis classes, sendo cinco tipos de ataques e um de mensagens legítimas. Nesta abordagem, aquelas legítimas mantiveram seus rótulos de classe 0 inalterados, enquanto todos os ataques foram reclassificados para o rótulo de classe 1 (maligno). Dessa forma, avaliamos a capacidade dos modelos em diferenciar entre mensagens de ataque e legítimas, independente do tipo de ataque realizado. Para a abordagem de **classificação multiclasse**, os dados de classe presentes no conjunto de dados foram preservados, mantendo a distinção entre os seis ataques (classes), possuindo um dos rótulos, 0, 1, 2, 4, 8 ou 16, conforme o tipo de ataque descrito na Tabela 1. Nesta abordagem, os modelos de aprendizado de máquina foram adaptados para possibilitar a predição da classe mais provável, habilitando, portanto, a avaliação do desempenho dos modelos em identificar corretamente as classes dos registros.

Em relação aos modelos para classificação, cada uma das 5 técnicas de aprendizagem de máquina foi combinada com cada um dos 3 preditores, resultando em 15 modelos de classificação, os quais foram nomeados da seguinte forma: `KNNfeat1`, `KNNfeat2`, `KNNfeat3`, `DTfeat1`, `DTfeat2`, `DTfeat3`, e assim por diante. Onde `feat1` refere-se ao **Preditor de intensidade do sinal** e `feat2` ao **Preditor de conformidade de deslocamento** e `feat3` ao **Preditor de combinação**. As métricas adotadas foram o *f1-score*, tempo de treinamento e tempo de teste. O **f1-score** é definido como a média harmônica entre *Precision* e *Recall*, dado pela equação (4), de modo a trazer um indicador único que represente a qualidade geral do um modelo, especialmente útil quando utilizado em um conjunto de dados com desbalanceamento entre as classes, como é o caso do *dataset* VeReMi. O *f1-score* binário foi utilizado para mensurar todas as abordagens de rotulação, exceto a abordagem multiclasse, para a qual foi utilizado o *f1-score* macro.

$$
\begin{aligned}
Precision &= \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN} \\
F1 - score &= \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}
\end{aligned}
\tag{4}
$$

Finalmente, as demais métricas são o **tempo de treinamento** e **tempo de teste**,

permitindo a avaliação de aplicabilidade dos modelos em cenários reais, os quais possivelmente exigiriam uma constante atualização dos modelos em uso, tornando necessária a repetição das tarefas de treinamento dos modelos. Para as tarefas de treinamento foram usados 80% dos dados, enquanto 20% dos dados foram reservados para as tarefas de teste.

4. Avaliação dos resultados

A avaliação dos resultados considera a combinação das abordagens de rotulação para cada um dos 15 modelos-preditores e 6 janelas de análise, resultando em 630 experimentos. Além disso, as métricas aplicadas estão descritas na Tabela 2.

A Tabela 3 apresenta os resultados que indicam que o algoritmo DISMISS-BSM atingiu um excelente desempenho na detecção de falsificação de posição em BSMs, obtendo as melhores pontuações na detecção dos ataques dos tipos *random*, *random offset* e *eventual stop*, superando os melhores e mais recentes trabalhos da área, tendo ainda atingindo uma pontuação *f1-score* de 0,984 na classificação binária, cujo objetivo é identificar se uma mensagem qualquer é legítima ou anômala, desempenhando junto ao topo do estado da arte atual.

Tabela 3. comparação de *f1-score* entre o DISMISS-BSM e a reprodução de 4 artigos de referência.

Ataques	**DISMISS-BSM**	[van der Heijden et al. 2018]	[So et al. 2018]	[Gyawali and Qian 2019]	[Ercan et al. 2021]
constant	0,994 (23 BSM)	0,621	0,816	0,994	**1,0**
constant offset	0,967 (23 BSM)	0,261	0,364	0,885	**0,998**
random	**1,0** (2 BSM)	0,984	0,994	**1,0**	**1,0**
random offset	**1,0** (8 BSM)	0,305	0,944	0,948	0,973
eventual stop	**0,972** (23 BSM)	0,724	0,810	0,942	0,958
binária	0,984 (23 BSM)	0,695	0,834	0,950	**0,986**
multiclasse	0,970 (23 BSM)	NA	0,676	0,933	**0,985**

Ao apreciar os resultados, foi possível atestar que o incremento da janela de análise resultou em uma melhoria logarítmica da performance da detecção, conforme valores na Tabela 4, com a janela de análise de 23 BSMs tendo obtido os melhores resultados, apesar da diferença para os resultados da janela de 8 BSMs ter sido relativamente pequena, de apenas $0,031$, abrindo oportunidade para uma consideração dos benefícios sobre os custos em termos de tempo extra dispendido em uma detecção tardia, e.g., esperar pela transmissão de 23 mensagens. No entanto, nada impede o uso simultâneo de múltiplas janelas de análise ao longo da acumulação de mensagens, melhorando a detecção conforme o número de mensagens recebidas aumentam, dessa forma, aperfeiçoando a capacidade de detecção durante o próprio processamento.

Tabela 4. médias dos resultados em função dos tamanhos de janela de análise. O *f1-score* melhora logaritmicamente com o aumento da janela de análise.

Métricas	02 BSM	03 BSM	08 BSM	13 BSM	18 BSM	23 BSM
f1-score	0,683	0,77	0,827	0,841	0,842	**0,858**
treinamento (seg)	**265,62**	529,02	835,94	930,96	563,6	501,07
inferência (seg)	**10,75**	14,81	754,87	386,89	223,93	133,22

Em termos de tempo de treinamento e teste dispendidos em cada tamanho de janela de análise, seria lógico que o tempo total aumentasse conforme a quantidade de

mensagens na janela de análise, entretanto observou-se que os tamanhos com os maiores tempos totais foram as janelas de 8 e 13 mensagens, isso se deve ao fato de que o tempo dispendido é diretamente proporcional ao número de *features* e ao número de amostras, e muito embora a quantidade de *features* aumente com o número de mensagens consideradas, a quantidade de BSMs recebidas pelo host é limitada ao tempo de interação entre os dois veículos, sendo que a média da quantidade de BSMs recebidas por cada host no dataset VeReMi é de $13,5$ mensagens, implicando em menos amostras para treinamento e teste nas janelas de análise de 18 e 23 BSMs, resultando, portanto, em uma diminuição dos seus tempos de treinamento e de teste.

Tabela 5. comparação das médias dos resultados entre modelos de machine learning. RF e DT possuem as melhores performances em f1-score. DT possui o menor custo de tempo total, apresentando o melhor custo-benefício.

Métricas	RF	DT	K-NN	MLP	LSTM
f1-score	**0,897**	0,878	0,81	0,663	0,768
treinamento (seg)	683,77	47,62	**1,41**	67,55	2221,5
inferência (seg)	5,43	**0,09**	1240,58	6,2	19,0

Em relação ao desempenho dos modelos de aprendizagem de máquina, mostrado na Tabela 5, os resultados demonstraram que os algoritmos RF e DT obtiveram os melhores desempenhos na detecção, no entanto, os tempos de treinamento e de teste do algoritmo DT foram, respectivamente, 14 e 60 vezes menores do que os do algoritmo RF. Portanto, pode-se dizer que o algoritmo DT se apresentou como o melhor modelo custo-benefício, uma vez que a diferença de desempenho *f1-score* médio dos modelos que utilizaram DT foram apenas $0,019$ menores do que os modelos de RF.

Tabela 6. comparação das médias dos resultados entre preditores por tipo de ataque (média f1-score). A combinação dos preditores aproveitou as melhores pontuações de cada preditor para cada tipo de ataque.

Preditor	*constant*	*constant offset*	*random*	*random offset*	*eventual stop*	binária	multiclasse
intensidade do sinal	0,719	0,6	**1,0**	0,723	0,546	0,791	0,688
conformidade de deslocamento	0,837	0,396	0,969	**0,975**	**0,834**	0,853	0,747
combinação	**0,857**	**0,625**	**1,0**	0,97	0,813	**0,886**	**0,819**

Sobre a performance dos atributos preditores, foi possível atestar que cada preditor possui diferentes propensões na identificação de tipos diferentes de ataques, conforme apresentados na Figura 5 e na Tabela 6. O preditor de intensidade do sinal demonstrou maior qualidade na detecção dos ataques *constant offset* e *random*, enquanto o preditor de Conformidade de deslocamento desempenhou melhor nos ataques *constant*, *random offset* e *eventual stop*, e ainda, nas classificações binária e multiclasse. Entretanto, o preditor de combinação, aproximou-se ou superou os preditores de intensidade do sinal e conformidade de deslocamento em seus melhores resultados individuais, levando-nos a inferir que, ao combinar os preditores, os modelos de aprendizado de máquina aproveitaram as melhores qualidades de cada preditor para cada tipo de ataque, franqueando, que sejam realizados novos testes de combinações com outros preditores de forma a verificar a generalização desse comportamento.

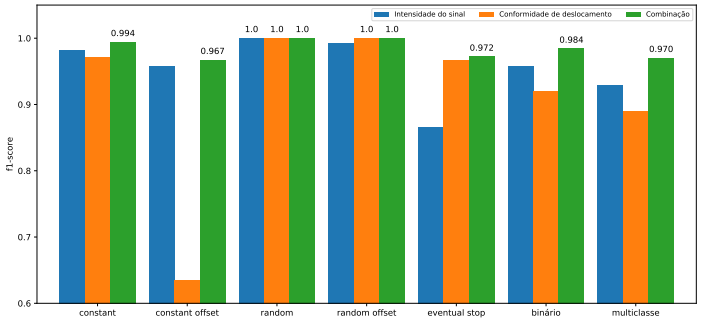

Figura 5. O Preditor de combinação proporciona ganho de desempenho ao combinar os melhores resultados individuais dos demais preditores.

5. Conclusão

Este trabalho apresentou o DISMISS-BSM, um algoritmo para detecção de falsificação de conteúdo de *Basic Safety Messages* em C-ITS, o qual permite que a detecção ocorra a partir das 2 primeiras mensagens recebidas, pois foram estabelecidas seis janelas deslizantes de análise (2, 3, 8, 13, 18 e 23 BSMs) para detecção das mensagens. Ainda, foram concebidos 2 novos atributos preditores, preditor de intensidade do sinal e preditor de conformidade de deslocamento, além da utilização de um terceiro preditor, que é a combinação dos dois preditores anteriores. Foram então testados com cinco diferentes modelos de aprendizado de máquina, RF, DT, K-NN, MLP, e LSTM, resultando em um total de 630 experimentos, os quais tiveram seus resultados avaliados por meio das seguintes métricas: f1-score, tempo de treinamento e tempo de inferência. Com base nos resultados atingidos, as janelas deslizantes de análise se provaram eficazes na detecção, permitindo conciliar agilidade e eficácia na detecção de ataques. O algoritmo DT apresentou o melhor custo-benefício em tempo/desempenho, obtendo pontuação média *f1-score* de $0,878$, com média de tempo de treinamento normalizada de $47,617$ segundos e média de tempo de inferência normalizada de $0,085$ segundos. Já o preditor de Conformidade de deslocamento destacou-se por sua grande eficácia na detecção dos ataques do tipo *random offset*, obtendo *f1-score* de $1,0$, superando os resultados dos demais trabalhos.

Como trabalhos futuros, planeja-se conceber novos atributos preditores e combinações de preditores para a detecção de ataques, validar a capacidade da combinação de preditores de aproveitar as características individuais de cada preditor e estudar a possibilidade de combinação de resultados simultâneos de múltiplas janelas de análise como forma de validar os resultados. Todos os códigos utilizados neste artigo estão disponíveis no repositório github.

Referências

Committee, V. C. T. (2020). *V2X Communications Message Set Dictionary*.

De Carvalho Bertoli, G., Pereira Júnior, L. A., Saotome, O., Dos Santos, A. L., Verri, F. A. N., Marcondes, C. A. C., Barbieri, S., Rodrigues, M. S., and Parente De Oliveira, J. M. (2021). An end-to-end framework for machine learning-based network intrusion detection system. *IEEE Access*, 9:106790–106805.

Ercan, S., Ayaida, M., and Messai, N. (2021). New features for position falsification detection in vanets using machine learning. In *ICC 2021 - IEEE International Conference on Communications*, pages 1–6.

Ercan, S., Ayaida, M., and Messai, N. (2022). Misbehavior detection for position falsification attacks in vanets using machine learning. *IEEE Access*, 10:1893–1904.

Gohar, A. and Nencioni, G. (2021). The role of 5g technologies in a smart city: The case for intelligent transportation system. *Sustainability*, 13(9).

Grover, H., Alladi, T., Chamola, V., Singh, D., and Choo, K.-K. R. (2021). Edge computing and deep learning enabled secure multitier network for internet of vehicles. *IEEE Internet of Things Journal*, 8(19):14787–14796.

Gyawali, S. and Qian, Y. (2019). Misbehavior detection using machine learning in vehicular communication networks. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pages 1–6.

Hawlader, F., Boualouache, A., Faye, S., and Engel, T. (2021). Intelligent misbehavior detection system for detecting false position attacks in vehicular networks. In *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*.

IEEE (2016). Ieee standard for wireless access in vehicular environments–security services for applications and management messages. *IEEE Std 1609.2-2016 (Revision of IEEE Std 1609.2-2013)*, pages 1–240.

Kamoi, R. N., Júnior, L. A. P., Verri, F. A. N., Marcondes, C. A. C., Ferreira, C. H. G., Meneguette, R. I., and Cunha, A. M. D. (2021). Platoon grouping network offloading mechanism for vanets. *IEEE Access*, 9:53936–53951.

Sharma, A. and Jaekel, A. (2022). Machine learning based misbehaviour detection in vanet using consecutive bsm approach. *IEEE Open Journal of Vehicular Technology*.

So, S., Petit, J., and Starobinski, D. (2019). Physical layer plausibility checks for misbehavior detection in v2x networks. In *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '19, page 84–93, New York, NY, USA. Association for Computing Machinery.

So, S., Sharma, P., and Petit, J. (2018). Integrating plausibility checks and machine learning for misbehavior detection in vanet. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 564–571.

Uprety, A., Rawat, D. B., and Li, J. (2021). Privacy preserving misbehavior detection in iov using federated machine learning. In *2021 IEEE 18th Annual Consumer Communications Networking Conference (CCNC)*, pages 1–6.

van der Heijden, R. W., Dietzel, S., Leinmüller, T., and Kargl, F. (2019). Survey on misbehavior detection in cooperative intelligent transportation systems. *IEEE Communications Surveys Tutorials*, 21(1):779–811.

van der Heijden, R. W., Lukaseder, T., and Kargl, F. (2018). Veremi: A dataset for comparable evaluation of misbehavior detection in vanets. In Beyah, R., Chang, B., Li, Y., and Zhu, S., editors, *Security and Privacy in Communication Networks*, pages 318–337, Cham. Springer International Publishing.