



UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO

Leonardo Passig Horstmann

**Cyber-Physical Systems Intrusion Detection based on Artificial Immune
Systems with Temporal Behavior and Physical Plausibility**

Florianópolis
2026

Leonardo Passig Horstmann

**Cyber-Physical Systems Intrusion Detection based on Artificial Immune
Systems with Temporal Behavior and Physical Plausibility**

PhD thesis submitted to Programa de Pós-Graduação
em Ciências da Computação from Universidade Fed-
eral de Santa Catarina.
Supervisor: Prof. Antônio A. M. Fröhlich, PhD.

Florianópolis
2026

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Passig Horstmann, Leonardo
Cyber-Physical Systems Intrusion Detection based on
Artificial Immune Systems with Temporal Behavior and
Physical Plausibility / Leonardo Passig Horstmann ;
orientador, Antônio Augusto Medeiros Fröhlich, 2026.
129 p.

Tese (doutorado) - Universidade Federal de Santa
Catarina, Centro Tecnológico, Programa de Pós-Graduação em
Ciência da Computação, Florianópolis, 2026.

Inclui referências.

1. Ciência da Computação. 2. Cyber-Physical Systems. 3.
Intrusion Detection Systems. 4. Artificial Immune Systems.
I. Medeiros Fröhlich, Antônio Augusto. II. Universidade
Federal de Santa Catarina. Programa de Pós-Graduação em
Ciência da Computação. III. Título.

Leonardo Passig Horstmann

**Cyber-Physical Systems Intrusion Detection based on Artificial Immune
Systems with Temporal Behavior and Physical Plausibility**

This work, at the Doctorate level, was evaluated and approved by an examining board composed of the following members.

Prof. Mauro Roisenberg, PhD
Universidade Federal de Santa Catarina

Prof. Evandro Leonardo Silva Teixeira, PhD
Faculdade do Gama, Universidade de Brasília

Prof. Roberto Milton Scheffel, PhD
Universidade Tecnológica Federal do Paraná, Campus Toledo

We certify that this is the **original and final version** of the final project that was deemed suitable for obtaining the degree of PhD in Computer Science.



Documento assinado digitalmente

Carina Friedrich Dorneles

Data: 04/02/2026 16:54:28-0300

CPF: ***.834.250-**

Verifique as assinaturas em <https://v.ufsc.br>

Coordenação do Programa de
Pós-Graduação



Documento assinado digitalmente

Antonio Augusto Medeiros Frohlich

Data: 02/02/2026 15:28:41-0300

CPF: ***.696.400-**

Verifique as assinaturas em <https://v.ufsc.br>

Prof. Antônio A. M. Fröhlich, PhD.
Supervisor:

Florianópolis, 2026.

This work is dedicated to my dear parents, Andreia and
Pedro, and my dear brother Lucas.

ACKNOWLEDGEMENTS

First, I would like to thank my mother, Andreia Passig Horstmann, my father, Pedro Ipolito Horstmann, and my brother, Lucas Passig Horstmann, for all the emotional support and encouragement provided.

I would like to thank my advisor, Prof. Antônio Augusto Medeiros Fröhlich, for the scientific guidance, support, and helpful discussions that led to this work conclusion. I would also like to thank my dear friend, José Luis Conradi Hoffmann, for the scientific discussions that helped me during the development of this work and for the encouraging words. Many thanks also to all colleagues at LISHA.

I am also grateful to Prof. Marcus Völp, Prof. Wolfgang Schröder-Preikschat, and Prof. Rüdiger Kapitza, for hosting me during my years of research abroad.

Last but not least, many thanks to CAPES-PrInt, FAPESC, and University of Luxembourg for the financial support during my years of research abroad.

ABSTRACT

Modern Cyber-Physical Systems (CPS), such as autonomous vehicles, increasingly integrate Machine Learning (ML) and connectivity features to enhance autonomy and user experience. While these additions improve functionality, they also expand the surfaces for attack on the system they compose, exposing them to a wide range of cyber threats, including remote exploitation and physical-layer attacks. Conventional security solutions designed for general-purpose computing are often unsuitable for CPS due to their stringent real-time, safety, and resource constraints. This work proposes an intrusion detection system (IDS) tailored to CPS, leveraging Artificial Immune Systems (AIS) to model and monitor the temporal properties of messages exchanged between software components, as well as the physical plausibility of their data. The AIS-based IDS is trained in a controlled environment prior to deployment, assuming the availability of mechanisms to guarantee no interference between components on the underlying computing platform, and the physical plausibility is assessed using ML-based techniques to identify the intricate properties and functions that govern the CPS operation. The proposed approach is evaluated in terms of overhead, latency, and detection accuracy, using a case study based on a relevant dataset on IDS for CPS. The results demonstrate the potential of the proposed AIS-based IDS with temporal behavior and physical plausibility monitoring to achieve low false-positive rates while preserving CPS performance and safety requirements.

Keywords: Cyber-Physical Systems (CPS), Intrusion Detection Systems (IDS), Artificial Immune Systems (AIS), Temporal Behavior, Physical Plausibility

RESUMO

Sistemas Ciber-Físicos (CPS) modernos, como veículos autônomos, integram cada vez mais recursos de aprendizado de máquina (ML) e conectividade para aumentar a autonomia e a experiência do usuário. Embora essas adições melhorem a funcionalidade, elas também expandem as superfícies de ataque no sistema que compõem, expondo-os a uma ampla gama de ameaças cibernéticas, incluindo exploração remota e ataques na camada física. As soluções de segurança convencionais projetadas para computação de uso geral são frequentemente inadequadas para CPS devido às suas restrições rigorosas de tempo real, segurança e recursos. Este trabalho propõe um sistema de detecção de intrusão (IDS) adaptado a CPS, aproveitando Sistemas Imunológicos Artificiais (AIS) para modelar e monitorar as propriedades temporais das mensagens trocadas entre componentes de software, bem como sua plausibilidade física. O IDS baseado em AIS é treinado em um ambiente controlado antes da implantação, assumindo a disponibilidade de mecanismos para garantir nenhuma interferência entre os componentes na plataforma de computação subjacente, e a plausibilidade física é avaliada usando técnicas baseadas em ML para identificar as propriedades e funções intrincadas que regem a operação do CPS. A abordagem proposta é avaliada em termos de sobrecarga, latência e precisão de detecção, usando um estudo de caso baseado em um conjunto de dados relevante sobre IDS para CPS. Os resultados demonstram o potencial do IDS baseado em AIS proposto com monitoramento temporal e de plausibilidade para alcançar baixas taxas de falsos positivos enquanto preserva os requisitos de desempenho e segurança do CPS.

Palavras-chave: Sistemas Ciberfísicos (CPS); Sistemas de Detecção de Intrusão (IDS); Sistemas Imunológicos Artificiais (AIS), Comportamento Temporal, Plausibilidade Física

RESUMO EXPANDIDO

Introdução

Sistemas Ciber-Físicos (CPS) integram tecnologias digitais e processos físicos. Muitos CPS são desenvolvidos para suportar aplicações de segurança crítica com requisitos funcionais e não-funcionais rigorosos, incluindo operação à prova de falhas e em tempo real. Não obstante, a crescente utilização e os recentes progressos em Aprendizado de Máquina (ML) e tecnologias de conectividade levaram CPS modernos, como veículos autônomos, a adotar tais características para promover níveis mais altos de autonomia e conveniência para o usuário.

A adição de tais funcionalidades frequentemente expõe CPS antes isolados a uma variedade de ameaças de cibersegurança (BENMALEK, 2024; XING; SHEN, 2024). Embora muitas dessas ameaças sejam herdadas de sistemas de computação de propósito geral, as soluções geralmente aplicadas a esses sistemas não são diretamente aplicáveis a cenários de CPS, principalmente devido a vulnerabilidades específicas e às restrições dos dispositivos na Internet das Coisas (IoT) (DIRO et al., 2017). O leque de ataques na IoT é, de fato, muito mais amplo quando comparado aos ataques vistos atualmente na Internet e inclui inutilização de dispositivos (*bricking*), interferência em canais de comunicação (*jamming*), negação de serviço, infiltração e exfiltração de dados e até mesmo ataques físicos diretos. Isso foi demonstrado em diversos cenários, incluindo veículos autônomos, casas inteligentes, redes inteligentes (*smart grids*), redes de saúde e provedores de infraestrutura (CORERO NETWORK SECURITY, n.d.; KOLIAS et al., 2017; LYU et al., 2017; BENMALEK, 2024; XING; SHEN, 2024).

Exemplos de ciberataques contra CPS incluem o ataque de 2015 ao sistema de infoentretenimento do Jeep Cherokee, no qual os pesquisadores Charlie Miller e Chris Valasek violaram de maneira remota um Jeep Cherokee através da conexão celular do veículo e obtiveram controle sobre componentes críticos como direção e frenagem, levando a Fiat Chrysler a realizar um *recall* de 1,4 milhão de veículos no primeiro *recall* do tipo relacionado à cibersegurança (GREENBERG, 2015). Em 2016, pesquisadores do Keen Security Lab realizaram com sucesso um ataque remoto a um Tesla Model S, comprometendo o sistema de infoentretenimento via Wi-Fi e ganhando controle de frenagem e outras funções vitais (GREENBERG, 2016). Em 2024, pesquisadores descobriram que as APIs *web* de controle de veículos da Kia podiam ser acessadas sem a devida autenticação. Isso expôs milhões de veículos a ataques remotos capazes de destrancar portas, ligar motores e divulgar localizações (COX, 2024). No início de 2025, a plataforma Starlink da Subaru demonstrou expor serviços de *back-end*, potencialmente permitindo que invasores rastreassem a localização dos veículos

e emitissem comandos remotos (COX, 2025). Finalmente, *exploits* na camada física também persistem, como demonstrado em 2025, quando os sistemas de entrada passiva sem chave da Hyundai foram encontrados vulneráveis a ataques de retransmissão (*relay attacks*). Usando equipamento de rádio de baixo custo, invasores podiam roubar veículos em menos de 20 segundos sem acesso à chave original (HERN, 2025).

Os *designs* atuais de CPS também consideram a integração de componentes com diferentes níveis de criticidade em uma única plataforma de computação (BURNS; DAVIS, 2017; OLIVEIRA et al., 2025; LEE, J.; KOH, 2023; HOORNAERT et al., 2024). Além disso, a crescente demanda por poder computacional em tais sistemas também os encaminha para a adoção de plataformas multiprocessadas. De acordo com Burns e Davis (BURNS; DAVIS, 2017), a maioria dos CPS complexos nos setores automotivo e de aviação está evoluindo para Sistemas com Criticalidade Mista (MCS) para atender a requisitos não-funcionais rigorosos de custo, espaço, peso, dissipação de calor e consumo de energia. A integração de múltiplos componentes de software em uma única plataforma computacional, no entanto, expõe componentes críticos a falhas, intrusões e ataques que podem ser adaptados para explorar fraquezas de componentes menos críticos, muitas vezes não certificados e projetados com medidas de segurança menos rigorosas (MAJUMDER et al., 2020; OLIVEIRA et al., 2025). De fato, padrões e regulamentos como a ISO 26262 afirmam que componentes de software relevantes para a segurança devem apresentar liberdade de interferência de componentes não relevantes para a segurança. Além disso, o comportamento temporal e a assinatura geral dos componentes de software podem ser afetados por restrições geradas pela utilização de recursos compartilhados, como a hierarquia de memória e periféricos (OLIVEIRA et al., 2025). Tal fenômeno foi observado por Horstmann e Fröhlich no contexto de um Sistema de Detecção de Intrusão (IDS) baseado no monitoramento de desempenho de componentes de software (HORSTMANN, Leonardo Passig et al., 2024). Não obstante, numerosos estudos têm investigado soluções de isolamento de recursos em sistemas críticos (WEST et al., 2016; KADAR, 2022; HOORNAERT et al., 2024).

Dado o cenário acima descrito, a detecção de componentes de software em falha e de intrusão em CPS é, de fato, desafiadora. Vários trabalhos abordaram problemas relacionados à otimização e à observação de requisitos funcionais através do uso de mecanismos de monitoramento em tempo de execução (*runtime monitoring*) juntamente com modelos do comportamento do sistema (CONRADI HOFFMANN; FROHLICH, Antonio Augusto, 2021; DONYANAVARD et al., 2016). Nessas soluções, os mecanismos de monitoramento em tempo de execução são responsáveis por rastrear a execução de um sistema (i.e., um conjunto de componentes de software) de

acordo com as assinaturas arquitetônicas de sua execução. Além disso, os modelos de comportamento do sistema são construídos através de mecanismos estatísticos e de ML que operam sobre os dados coletados através do monitoramento em tempo de execução.

Aprender traços de segurança e detectar desvios pode ser visto como uma aplicação de técnicas de detecção de anomalias (CHANDOLA et al., 2009), que consistem em buscar padrões nos dados que não estejam em conformidade com uma noção bem definida de normalidade. Nesse sentido, algumas técnicas de ML foram adotadas para automatizar o processo de detecção de anomalias em sistemas embarcados por muitos trabalhos, por exemplo, (EZEME et al., 2018; VIEGAS et al., 2017; PARK et al., 2020) e (ZHOU et al., 2015). Algoritmos de classificação de ML estão entre as alternativas mais adotadas para detecção de anomalias (GHOSH; SCHWARTZBARD, 1999; OTT; MAHAPATRA, 2018; EZEME et al., 2018). Essas técnicas operam como detectores de correspondência de padrões (*pattern-matching*) sob a suposição de que o comportamento normal pode ser aprendido a partir do conjunto de dados (CHANDOLA et al., 2009). Alternativamente, Forrest et al. (FORREST et al., 1994) introduziram um algoritmo de detecção de mudança baseado no Sistema Imunológico Humano para discriminação de comportamento próprio/não-próprio (*self-nonself*) em um computador. Sistemas Imunológicos Artificiais (AIS) têm sido aplicados em vários trabalhos no contexto de detecção de anomalias, detecção de intrusão e outros aspectos de segurança de computadores, como indicado em Fernandes et al. (FERNANDES et al., 2017).

Não obstante, como os CPS fundamentalmente ligam operações de software a processos físicos, soluções eficazes de IDS para este domínio devem transcender as técnicas tradicionais centradas no ciberespaço para considerar os processos físicos associados (FARAMONDI et al., 2021). De fato, trabalhos como os de Mitchell e Chen (MITCHELL, R.; CHEN, I.-R., 2015) e Cho e Shin (CHO; SHIN, 2016) propuseram mecanismos de IDS nos quais os modelos consideram características que representam o processo físico. Enquanto o primeiro integra dados coletados de sensores para validar o comportamento de aplicações médicas, o segundo considera o desvio de tempo (*time skew*) para monitorar mensagens CAN, uma característica herdada do canal onde as mensagens são transmitidas. A premissa é considerar modelos ou invariantes informados pela física para detectar inconsistências que podem ser refletidas em propriedades físicas específicas, comportamento temporal, leituras de sensores, etc., visando correlacionar anomalias tanto no domínio digital quanto no físico. Neste contexto, Raissi et al. (RAISSI et al., 2019) propuseram Redes Neurais informadas pela Física (PINNs), que são redes neurais treinadas para resolver

tarefas de aprendizado supervisionado enquanto respeitam quaisquer leis da física dadas, descritas por equações diferenciais parciais não lineares gerais. A aplicação de PINNs no contexto de IDS tem sido investigada por trabalhos recentes como (MISHRA; CHAURASIYA, 2025), no qual uma PINN é usada para prever se ataques poderiam manipular dados de IoT.

Em paralelo, os CPS modernos também estão enfrentando uma transição nos métodos de projeto, afastando-se do modelo tradicional, centrado em tarefas e local, para uma visão mais ampla, centrada em dados e distribuída (ZHANG, J. et al., 2011; JIANG et al., 2018; FRÖHLICH, Antônio Augusto, 2018; SARKER, 2021). Veículos autônomos, por exemplo, utilizam dados para calcular seu estado atual, planejar ações e atuar, tudo em direção a um objetivo específico de alcançar um destino sem colisões. Os CPS são inerentemente sistemas de tempo real, exigindo atuação periódica para garantir a execução correta (KOPETZ; BAUER, 2003). Como os dados impulsionam esses sistemas, requisitos temporais são impostos à geração, comunicação, transformação e consumo de dados (FRÖHLICH, Antônio Augusto, 2018; HOFFMANN; FRÖHLICH, Antônio Augusto, 2022).

Neste contexto, o conceito de *SmartData* (FROHLICH, Antonio A.; RESNER, 2018) surge como uma abstração de alto nível para as primitivas de processamento, escalonamento, sincronização e comunicação utilizadas pelos CPS. Ele aproveita a miríade de recursos disponíveis nas plataformas embarcadas para fornecer uma interface centrada em dados para sensoriamento, atuação e controle. SmartData encapsula dados com metadados que os tornam autocontidos em termos de semântica, localização espacial, temporalidade e confiabilidade. Pretende-se que seja a única interface visível à aplicação na plataforma e, portanto, que medie implicitamente todos os serviços de nível de sistema, expressando a interação entre componentes através de relações de Interesse, com atributos como *freshness* (frescor), periodicidade e segurança sendo herdados por tais relações.

Nesta tese, aproveitamos os conceitos de SmartData e sua capacidade de modelar sistemas complexos para construir uma solução de IDS para CPS que leva em consideração os domínios cibernético e físico. A solução de IDS considera o comportamento temporal e a plausibilidade física das mensagens SmartData trocadas entre os componentes do CPS. Por fim, a solução se baseia nas propriedades do SmartData e explora seus metadados, que encapsulam propriedades de tempo e a semântica dos dados usados para verificar a plausibilidade física.

A adequação do mecanismo de IDS proposto estende-se a diversos domínios de CPS,

onde modelar a dependência inerente entre ativos cibernéticos e físicos é primordial. No campo dos veículos autônomos, a solução pode ser explorada considerando a consistência temporal dos comandos de controle e as leis físicas que regem a dinâmica do veículo, prevenindo desfechos catastróficos de intervenções remotas não autorizadas. Da mesma forma, em redes de telemedicina e saúde, o sistema garante a integridade do monitoramento crítico à vida e da prestação de tratamentos automatizados, verificando se os dados dos dispositivos médicos alinham-se à plausibilidade biológica e aos rigorosos requisitos de tempo real na interação entre componentes. Além destes, outros CPSS, como redes elétricas inteligentes (smart grids) e infraestruturas industriais, também podem se beneficiar da capacidade de modelar e detectar anomalias temporais ou mudanças de estado fisicamente inconsistentes para identificar intrusões. Por fim, ao fundamentar a detecção baseada em AIS nas características temporais e propriedades dos processos físicos relacionados ao CPS, a solução fornece uma camada de defesa robusta para qualquer infraestrutura de missão crítica onde a operação correta depende fundamentalmente da coerência entre os domínios cibernético e físico.

Proposta e Objetivo

À luz do cenário acima, funcionalidades como ML e conectividade são cruciais para que os CPS modernos aumentem a autonomia e melhorem a experiência do usuário. No entanto, tais funcionalidades frequentemente aumentam a exposição desses CPS a uma variedade de ciberataques. Dados os mecanismos discutidos para o *design* de componentes, isolamento de recursos, monitoramento e detecção de anomalias, este trabalho visa a detecção de intrusão em sistemas ciber-físicos, com baixa taxa de falso-positivos, detectando a violação das propriedades temporais dos componentes cibernéticos de tais sistemas, bem como a plausibilidade física dos dados que eles usam para modelar os processos físicos do CPS. O comportamento temporal é observado através do monitoramento das mensagens trocadas entre os componentes e é expresso por meio de modelos baseados em AIS, treinados em um ambiente controlado, antes da implantação do sistema, e assumindo que as plataformas de computação nas quais tais componentes executam fornecem mecanismos de particionamento de recursos. Enquanto isso, a plausibilidade física é obtida usando técnicas baseadas em ML para aproximar e modelar as propriedades e funções que governam a operação do CPS.

A solução proposta considera que os componentes sejam modelados seguindo uma abordagem orientada a dados (*data-driven*), o que permite extrair propriedades e modelos para a plausibilidade física dos dados que eles usam para modelar o estado

de operação do CPS. Além disso, aproveitamos o tempo em que as mensagens são trocadas para extrair as características temporais que modelam o comportamento temporal dos componentes que as trocam, as quais serão usadas juntamente com os metadados das mensagens para construir os modelos baseados em AIS.

O trabalho proposto é avaliado em termos de acurácia e *overhead* computacional. No primeiro caso, um conjunto de dados relevante sobre IDS para CPS é considerado. No segundo, uma plataforma com isolamento de recursos foi selecionada, e métricas como utilização de memória e *overhead* computacional serão observadas. Finalmente, a complexidade computacional para o modelo AIS final, que incorpora comportamento temporal e plausibilidade física, também é calculada.

Objetivos Específicos

O mecanismo proposto para detecção de intrusão será projetado de acordo com o seguinte conjunto de objetivos específicos:

- Adaptar o projeto do IDS baseado em AIS proposto por (HORSTMANN, Leonardo Passig, 2020; HORSTMANN, Leonardo Passig; FROHLICH, Antonio Augusto, 2022) para modelar propriedades temporais observadas nas mensagens trocadas por componentes de software de CPS.
- Adaptar a solução de IDS baseada em AIS para abranger a plausibilidade física dos dados trocados pelos componentes de software de CPS, com base em propriedades extraídas de seu respectivo modelo SmartData.
- Definir cenários de avaliação e construir os modelos de comportamento seguro (isto é, normal, esperado). A definição do cenário de avaliação incluirá a especificação de tecnologias aplicadas para o isolamento de recursos de componentes de software e a seleção de conjuntos de dados relevantes para aplicações de IDS em CPS.
- Avaliar o desempenho da solução em termos de sobrecarga (*overhead*), latência e acurácia na detecção de mau comportamento.

Contribuições Esperadas

As contribuições deste trabalho incluem o projeto, a implementação e a avaliação de um mecanismo para detecção de intrusão em CPS de acordo com propriedades temporais e plausibilidade física. As principais contribuições esperadas são resumidas a seguir:

- O projeto e a implementação de um mecanismo baseado em AIS para monitorar o

comportamento de um CPS, considerando características temporais, através das mensagens trocadas por seus componentes.

- A extensão do projeto do IDS baseado em AIS para considerar a plausibilidade física dos dados trocados pelos componentes do CPS com base em propriedades extraídas do modelo orientado a dados desses componentes.
- A avaliação do desempenho do IDS proposto em termos de (*overhead*), latência e da acurácia dos modelos de comportamento.

Resultados e Considerações Finais

Este trabalho apresenta um mecanismo de detecção de intrusão baseado em AIS adaptado para modelar, monitorar e distinguir propriedades temporais de componentes de software de um CPS. A solução proposta monitora as propriedades temporais através das mensagens trocadas por cada componente e aumenta a confiabilidade ao considerar a plausibilidade física dos dados trocados nas referidas mensagens. A validação experimental, considerando uma do mundo real com conjuntos de dados de CPS de alta relevância no contexto de IDS, demonstra a eficácia da abordagem proposta e sua adequação para aplicações de CPS. Desta forma, as contribuições detalhadas nesta tese pavimentam o caminho para desenvolvimentos de soluções para CPS seguros e protegidos.

Etapas Fundamentais Alcançadas Rumo à Detecção de Intrusão para CPS Baseada em Comportamento Temporal e Plausibilidade Física

Este trabalho oferece uma solução para a detecção de intrusão em CPS com baixas taxas de falso-positivo, baseada no conceito de AIS introduzido pela primeira vez pelo autor em Horstmann et al (HORSTMANN, Leonardo Passig, 2020; HORSTMANN, Leonardo Passig; FROHLICH, Antonio Augusto, 2022) e estendido nesta tese para incorporar aspectos do comportamento temporal e da plausibilidade física dos CPS monitorados. Primeiro, revisamos e atualizamos a versão original do **AIS** (HOFFMANN; FROHLICH, Antonio Augusto, 2022), removendo suas ligações com o *framework* de monitoramento EPOS (HORSTMANN, Leonardo Passing et al., 2019) e promovendo-o como um componente de software individual. Em seguida, a estratégia de amostragem do AIS foi modificada para acomodar mensagens e amostras de conjuntos de dados de rede típicos, o que exigiu diferentes estratégias para o pré-processamento de dados, nomeadamente, codificação de dados e tratamento de variáveis com amostragem não sincronizada. Em seguida, propusemos e incorporamos características temporais seguindo o modelo orientado a dados (*data-driven*) dos componentes do CPS e suas dependências de dados.

Uma vez que o AIS passou a suportar o tratamento de mensagens, a amostragem não sincronizada e as características temporais, revisamos alternativas para verificar a plausibilidade do processo físico relacionado aos componentes cibernéticos do CPS. Apresentamos abordagens relacionadas à física e intensivas em dados, aproveitando o modelo orientado a dados de subsistemas de CPS tanto para a identificação e elaboração de funções de plausibilidade quanto para a definição da aplicabilidade de uma medida de plausibilidade ao modelo AIS.

Finalmente, a validação experimental demonstra tanto a viabilidade de incorporar a solução proposta para operação em tempo de execução (*runtime operation*) em CPS quanto o potencial de melhoria ao considerar os mundos cibernético e físico ao abordar a detecção de intrusões. Por um lado, o sistema foi avaliado em termos de desempenho, utilização de memória e sobrecarga (*overhead*) em uma plataforma complexa, operando em uma máquina virtual (VM) configurada com isolamento de recursos e compartilhando a plataforma com uma carga de trabalho secundária típica de um CPS e um Sistema Operacional (OS) de propósito geral. Por outro lado, foi avaliado quanto à sua capacidade de detectar intrusões através de um conjunto de dados coletados em uma aplicação relevante de CPS focada no desenvolvimento de IDS.

Passamos agora a revisar os principais marcos alcançados nesta tese, em linha com os objetivos específicos definidos para este trabalho.

1. **Verificação de características temporais em IDS baseado em AIS:** O projeto original do AIS foi aprimorado para suportar a inclusão de características temporais observadas nas mensagens trocadas pelos componentes do CPS. A extração de características temporais foi introduzida no processo de treinamento do AIS para tornar a classificação ciente do comportamento temporal do sistema. Como o comportamento temporal é um fator crítico na operação correta de um CPS, isso representa uma melhoria em relação ao modelo original do IDS baseado em AIS, que considerava apenas valores absolutos para as variáveis monitoradas como características disponíveis. As características temporais introduzidas no processo de treinamento representam os intervalos entre amostras e entre eventos, a latência de comunicação e o *jitter* entre componentes do CPS, a frequência de eventos e o número de ocorrências em um dado intervalo. As características mais significativas são então selecionadas de acordo com sua relevância para a detecção de intrusões na aplicação alvo por meio de um processo de seleção de características multifatorial, resultando em um modelo consistente da operação do CPS.

Considerando isso, abordamos o primeiro objetivo específico deste trabalho: **Adaptar o projeto de IDS baseado em AIS proposto por (HORSTMANN, Leonardo**

Passig, 2020; HORSTMANN, Leonardo Passig; FROHLICH, Antonio Augusto, 2022) para modelar propriedades temporais observadas em mensagens trocadas por componentes de software de CPS.

2. **Medida de Plausibilidade Física:** A solução IDS baseada em AIS foi enriquecida ainda mais pela adição de uma medida de plausibilidade dos dados que compõem a representação do estado do sistema. O presente trabalho propõe a utilização de mecanismos baseados em ML para identificar e modelar por aproximação as funções que governam a operação física dos componentes do CPS. Os modelos operacionais são gerados para as variáveis que compõem a representação do estado do sistema e fornecem um meio para calcular uma medida de plausibilidade dos dados usados pelos processos cibernéticos para modelar os processos físicos relacionados ao CPS. A medida de plausibilidade é integrada à classificação do IDS como um fator de tolerância para o desvio entre o comportamento observado e o modelado, controlando ativamente o limiar para a detecção de anomalias. Além disso, definimos o processo para selecionar as medidas de plausibilidade a serem aplicadas para a detecção de intrusão, considerando as relações orientadas a dados (*data-driven*) entre os componentes do CPS.

Desta forma, alcançamos o segundo objetivo específico deste trabalho: **Adaptar a solução IDS baseada em AIS para abranger a plausibilidade física dos dados trocados pelos componentes de software de CPS com base em propriedades extraídas por seu respectivo modelo SmartData.**

3. **Implementação de Prova de Conceito:** A abordagem proposta foi implementada para dois cenários de validação. O primeiro cenário de validação considera sua capacidade de modelar o comportamento do CPS através dos pacotes trocados pelos componentes do CPS, modelando aspectos de seu comportamento temporal, bem como modelando a plausibilidade dos dados usados para modelar a operação do CPS. Primeiro, observamos os dados de tráfego de rede, amostras foram coletadas, características temporais foram extraídas, as características que melhor representam o tráfego normal foram selecionadas através da análise de sua correlação com a presença de ataques, e os modelos AIS foram gerados. Os dados físicos também foram analisados, e uma aproximação baseada em ML das funções e propriedades que governam a operação do CPS foi gerada para cada característica que compõe o modelo do estado de operação do CPS. Primeiro, para cada uma das características, selecionamos um subconjunto com as características mais correlacionadas para treinar preditores baseados em Redes Neurais (NN) que foram usados como funções de plausibilidade e utilizados para extrair medidas de plausibilidade para cada amostra. Ambos os mecanismos foram então integrados para a avaliação final do tráfego de rede. O segundo cenário de validação considerou o desempenho computacional e a sobrecarga (*overhead*) introduzida pela operação do IDS baseado

em AIS em uma plataforma equipada com um mecanismo de isolamento de recursos baseado em técnicas de virtualização e na presença de duas outras VMs, uma com uma carga de trabalho típica de aplicações de CPS no domínio de veículos autônomos operando em um Sistema Operacional de Tempo-Real (RTOS), e outra considerando um OS de propósito geral.

Assim, cumprimos nosso segundo objetivo específico para este trabalho: **Definir cenários de avaliação e construir os modelos de comportamento seguro (i.e., normal, esperado). A definição do cenário de avaliação incluirá a especificação de tecnologias aplicadas para o isolamento de recursos de componentes de software e a seleção de um conjunto de dados relevante para aplicação de IDS em CPS.**

4. **Desempenho de Classificação, Importância da Plausibilidade Física e Sobrecarga:** Os resultados obtidos através do primeiro cenário de avaliação demonstraram a capacidade da presente implementação de nosso mecanismo de detecção de intrusão baseado em AIS de modelar o comportamento normal dos componentes do CPS com base nas mensagens que eles trocam. De fato, com um total de 6 características, sendo duas temporais, e 129 células T, a presente solução baseada em AIS foi capaz de capturar 99% do comportamento do sistema. Apesar de reduzir a latência de detecção, a política de detectar comportamento anormal baseada em anomalias pontuais (analisando pontos de dados individuais), no entanto, demonstrou ineficiência na detecção de uma janela de ataques MITM. De fato, tais ataques foram notados apenas em momentos individuais ao longo da janela de ataque. No entanto, a incorporação de uma medida de plausibilidade física aprimorou o mecanismo IDS proposto para detectar os efeitos de tal ataque, aumentando nossas taxas de *recall* para quase 90%. Além disso, os resultados obtidos através do segundo cenário de avaliação demonstraram a eficiência computacional do mecanismo de classificação IDS proposto, já que o tempo de inferência não excedeu 0.28 ms para 1000 células T. Além disso, nenhum impacto perceptível foi observado na execução da aplicação secundária de CPS rodando na mesma plataforma, o que demonstra que a sobrecarga do AIS não excedeu os limites do isolamento fornecido pelo mecanismo de virtualização que foi adotado.

Assim, cumprimos o objetivo específico final deste trabalho: **Avaliar o desempenho da solução em termos de sobrecarga, latência e acurácia na detecção de mau comportamento.**

Em conclusão, a solução proposta oferece uma técnica IDS computacionalmente eficiente e robusta que se baseia em um mecanismo baseado em AIS para modelar o comportamento normal de um CPS e em mecanismos baseados em ML para aprender as regras e funções que governam a operação do CPS. O mecanismo baseado

em AIS modela o comportamento do sistema através das mensagens trocadas pelos componentes do CPS, extraíndo e observando características temporais juntamente com outras variáveis de comunicação. Enquanto isso, mecanismos baseados em ML são usados para aproximar e modelar as funções e propriedades que governam a operação do CPS para extrair uma medida da plausibilidade física dos dados usados para modelar o estado de operação de tais sistemas dinâmicos. Finalmente, ambos os mecanismos são integrados para uma decisão final sobre o comportamento normal ou anormal do sistema. A combinação de tais técnicas permite a construção de mecanismos IDS sem a dependência de conhecimento especializado de domínio para determinar aspectos da operação correta do CPS. Trabalhos futuros podem considerar a investigação da aplicabilidade e escalabilidade da presente solução em diferentes cenários de CPS, como veículos autônomos. Outra oportunidade significativa reside na investigação de alternativas para navegar no trade-off entre a qualidade do modelo e otimizações de desempenho, devido ao mecanismo de treinamento iterativo descrito nas seções anteriores. Um exemplo típico seria a fusão de células T sobrepostas com base em sua proximidade espacial e raio de afinidade. Ao consolidar detectores que representam a mesma variedade comportamental (manifold), o modelo poderia alcançar uma granularidade maior e uma representação mais compacta do espaço "self", melhorando, em última análise, a precisão do IDS em ambientes complexos de CPS.

LIST OF FIGURES

Figure 1 – The SmartData Interface (FRÖHLICH, Antônio Augusto, 2018). . . .	9
Figure 2 – Example of Censoring. Figure based on the algorithm proposed by Forrest et al. (FORREST et al., 1994)	17
Figure 3 – AIS-based IDS considering Temporal Features and Physical Plausibility	49
Figure 4 – Water Distribution Testbed Schematic adaptation from (FARAMONDI et al., 2021). In the Figure, the right blue dotted rectangle represents the simulated subsystem. Blue arrows represent virtual water flowing between the two subsystems.	54
Figure 5 – Correlation Analysis According to Pearson Correlation Coefficient from each Network Feature to the Traffic Class	59
Figure 6 – Correlation Analysis According to Pearson Correlation Coefficient from each Network Feature to Normal vs Abnormal Behavior	60
Figure 7 – Correlation Analysis According to Spearman's Rank Correlation Co- efficient from each Network Feature to the Traffic Class	61
Figure 8 – Correlation Analysis According to Spearman's Rank Correlation Co- efficient from each Network Feature to Normal vs Abnormal Behavior	62
Figure 9 – Physical Plausibility Measure during MITM Attack from Scenario 1.1	70
Figure 10 – Physical Plausibility Measure during MITM Attack from Scenario 1.3	71
Figure 11 – Physical Plausibility Measure during MITM Attack from Scenario 1.5	71
Figure 12 – Physical Plausibility Measure during MITM Attack from Scenario 1.7	72
Figure 13 – Physical Plausibility Measure during MITM Attack from Scenario 1.8	72
Figure 14 – Physical Plausibility Measure during MITM Attack from Scenario 2.12	73
Figure 15 – Physical Plausibility Measure during MITM Attack from Scenario 3.6	74
Figure 16 – Physical Plausibility Measure during MITM Attack from Scenario 3.7	75
Figure 17 – AIS Inference Time vs Number of T cells	84

LIST OF TABLES

Table 1 – Comparison of Intrusion Detection Systems (IDS) Works in Cyber-Physical Systems	30
Table 2 – Tank Direct Connections	53
Table 3 – First acquisition	56
Table 4 – Second acquisition	57
Table 5 – Third acquisition	58
Table 6 – Spearman’s Rank Correlation Coefficient for Temporal Features . . .	60
Table 7 – AIS Grid-Search Parameters	63
Table 8 – Correlation Analysis for the Variables Composing the State of the Physical Components	65
Table 9 – AIS Configurations	67
Table 10 – AIS-Only Classification Results	67
Table 11 – Average Plausibility Measure Scores For Normal Behavior	68
Table 12 – MITM Attack Scenarios according to (FARAMONDI et al., 2021) . . .	69
Table 13 – Experiment Scenarios	83
Table 14 – Performance measurements with varying numbers of T cells.	84
Table 15 – AIS Memory Consumption	85

LIST OF ABBREVIATIONS AND ACRONYMS

AE	Autoencoders
AI	Artificial Intelligence
AIS	Artificial Immune System
ANN	Artificial Neural Network
CAN	Controller Area Network
CFI	Control-Flow Integrity
CNN	Convolutional Neural Network
CPS	Cyber-Physical System
CPS	Cyber-Physical Systems
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DoS	Denial-of-Service
ECU	Electronic Control Unit
HMI	Human-Machine Interface
HPC	Hardware Performance Counter
IDS	Intrusion Detection Systems
IoT	Internet of Things
K-Means	K-Means
KNN	K-Nearest-Neighbors
LSTM	Long Short Term Memory
MCS	Mixed-Criticality Systems
MITM	Man-In-The-Middle
ML	Machine Learning
NB	Naïve Bayes
NN	Neural Networks
OS	Operating System
PCA	Principal Component Analysis
PINN	Physics-Informed Neural Network
PINNs	Physics-Informed Neural Networks
RNN	Recurrent Neural Networks
RTOS	Real-Time Operating System
SCADA	Supervisory Control and Data Acquisition
SNR	Signal-to-Noise Ratio
SVM	Support Vector Machine
TTA	Time-Triggered Architecture
VM	Virtual Machine
WDT	Water Distribution Testbed

CONTENTS

1	INTRODUCTION	1
1.1	PROPOSAL AND GOAL	4
1.2	SPECIFIC GOALS	5
1.3	ENVISIONED CONTRIBUTIONS	5
1.4	DOCUMENT ORGANIZATION	6
2	BACKGROUND	8
2.1	DATA-DRIVEN CYBER-PHYSICAL SYSTEMS DESIGN	8
2.1.1	SmartData	9
2.1.1.1	Modeling Data Relationships	9
2.2	MACHINE LEARNING FOR ANOMALY AND INTRUSION DETECTION	10
2.2.1	Intrusion Detection Methods	12
2.2.1.1	Anomaly Types	12
2.2.1.2	Anomaly Detection Operational Models	13
2.2.2	Feature Selection	13
2.2.3	Association Rules	14
2.2.4	Neural Networks (NN)	15
2.3	ARTIFICIAL IMMUNE SYSTEMS - AIS	16
3	LITERATURE REVIEW	21
3.1	INTRUSION DETECTION SYSTEMS	21
3.1.1	IDS for Cyber-Physical Systems	22
3.2	CPS IDS DATASETS	27
3.3	SUMMARY OF LITERATURE REVIEW	29
4	ARTIFICIAL IMMUNE SYSTEMS-BASED IDS WITH TEMPORAL FEATURES AND PHYSICAL PLAUSIBILITY	32
4.1	SELF AND NONSELF DEFINITION	33
4.2	T CELLS	34
4.3	BINDING FUNCTION	34
4.4	CENSORING	35
4.5	DISCRIMINATION SELF AND NONSELF	37
4.6	TIME COMPLEXITY ANALYSIS	37
4.7	TRAINING AND OPERATION	39
4.8	SUPPORTING TEMPORAL BEHAVIOR FEATURES	39
4.8.1	Deriving Temporal Features	40
4.8.1.1	Inter-Sample and Inter-Event Intervals	40
4.8.1.2	Latency and Jitter	41
4.8.1.3	Event Frequency and Count	42
4.8.1.4	Component Interactions	42

4.8.2	Incorporating Temporal Features into Samples	42
4.9	INTEGRATING PHYSICAL PLAUSIBILITY BASED ON CPS SMART-DATA MODEL	43
4.9.1	Machine Learning Mechanisms for Modeling CPS Properties . .	45
4.9.2	Plausibility Model Generation	46
4.9.3	Physical Plausibility and AIS-based IDS Integration	47
5	PROOF-OF-CONCEPT IMPLEMENTATION AND VALIDATION . . .	52
5.1	DATASET ANALYSIS	52
5.2	DATASET PREPROCESSING AND FEATURE SELECTION	56
5.2.1	Preprocessing and Feature Selection Results	58
5.3	PERFORMANCE EVALUATION	62
5.3.1	Building AIS Monitors	62
5.3.1.1	Model Construction	63
5.3.2	Physical Plausibility Strategy	63
5.3.3	Evaluation Metrics	64
5.3.4	AIS Performance Results	66
5.3.5	Discussion	74
5.4	OVERHEAD EVALUATION	77
5.4.1	Evaluation Scenarios	82
5.4.2	Computational Performance, Memory Utilization, and Overhead Results	83
5.4.3	Discussion	85
6	FINAL REMARKS	88
6.1	FUNDAMENTAL STEPS ACHIEVED TOWARDS INTRUSION DETECTION FOR CPS WITH TEMPORAL BEHAVIOR AND PHYSICAL PLAUSIBILITY	88
	REFERENCES	93

1 INTRODUCTION

Cyber-Physical Systems (CPS) integrate digital technologies and physical processes. Many CPS are developed to support safety-critical applications with strict functional and non-functional requirements, including fail-safe, real-time operation. Notwithstanding, the increasing utilization and recent progress in Machine Learning (ML) and connectivity technologies have led modern CPS like autonomous vehicles to adopt such features to promote higher levels of autonomy and user convenience.

The addition of such features often exposes once-isolated CPS to a variety of cybersecurity threats (BENMALEK, 2024; XING; SHEN, 2024). While many of these threats are inherited from general-purpose computing systems, the solutions usually applied to these systems are not directly applicable to CPS scenarios, mainly because of specific vulnerabilities and IoT devices constraints (DIRO et al., 2017). In fact, the range of IoT attacks is much broader than those seen today on the Internet and includes bricking devices, jamming communication channels, denial of service, data infiltration and exfiltration, and even direct physical attacks. This has been demonstrated in several scenarios, including autonomous vehicles, smart homes, smart grids, healthcare networks, and infrastructure providers (CORERO NETWORK SECURITY, n.d.; KOLIAS et al., 2017; LYU et al., 2017; BENMALEK, 2024; XING; SHEN, 2024).

Examples of cyberattacks over CPS include the 2015 attack on the Jeep Cherokee infotainment system, where researchers Charlie Miller and Chris Valasek remotely compromised a Jeep Cherokee via the vehicle's cellular connection, and gained control over critical components such as steering and braking, prompting Fiat Chrysler to recall 1.4 million vehicles in the first cybersecurity-related recall of its kind (GREENBERG, 2015). In 2016, researchers at Keen Security Lab successfully performed a remote attack on a Tesla Model S by compromising the infotainment system via Wi-Fi, gaining control of braking and other vital functions (GREENBERG, 2016). In 2024, researchers found that Kia's vehicle control web APIs could be accessed without proper authentication. This exposed millions of vehicles to remote attacks capable of unlocking doors, starting engines, and disclosing locations (COX, 2024). In early 2025, Subaru's Starlink platform was shown to expose backend services, potentially allowing attackers to track vehicle locations and issue remote commands (COX, 2025). Finally, physical-layer exploits also persist, as demonstrated in 2025 when Hyundai's passive keyless entry systems were found vulnerable to relay attacks. Using inexpensive radio equipment, attackers could steal vehicles in under 20 seconds without access to the original key fob (HERN, 2025).

Current designs of Cyber-Physical Systems also consider the integration of components with different criticality levels onto a single computing platform (BURNS; DAVIS, 2017; OLIVEIRA et al., 2025; LEE, J.; KOH, 2023; HOORNAERT et al., 2024). More-

over, the growing demand for computational power in such systems also points to the adoption of multiprocessed platforms. According to Burns and Davis (BURNS; DAVIS, 2017), most of the complex CPS in the automotive and avionics sectors are evolving into MCS to meet stringent non-functional requirements of cost, space, weight, heat dissipation, and power consumption. The integration of multiple software components into a single computational platform, however, exposes critical components to failures, intrusion, and attacks that could be tailored to exploit weaknesses of less-critical components, often non-certified and designed with fewer stringent security measures (MAJUMDER et al., 2020; OLIVEIRA et al., 2025). In fact, standards and regulations such as ISO 26262 state that safety-relevant software components must present freedom from interference from non-safety-relevant components. Other than that, the temporal behavior and overall fingerprint of software components can be affected by constraints generated by the utilization of shared resources, like the memory hierarchy and peripherals (OLIVEIRA et al., 2025). Such a phenomenon was observed by Horstmann and Fröhlich in the context of a IDS solution based on performance monitoring of software components (HORSTMANN, Leonardo Passig et al., 2024). Nevertheless, numerous studies have investigated resource isolation solutions in critical systems (WEST et al., 2016; KADAR, 2022; HOORNAERT et al., 2024).

Considering all the above, the detection of faulty components and intrusion on CPS is, in fact, challenging. Several works addressed problems related to optimization and the observation of functional requirements through the usage of runtime monitoring mechanisms alongside models of the system behavior (CONRADI HOFFMANN; FROHLICH, Antonio Augusto, 2021; DONYANAVARD et al., 2016). In these solutions, runtime monitoring mechanisms are responsible for tracing the execution of a system (i.e., a set of software components) according to the architectural footprints of its execution. Moreover, the models of the system behavior are built through statistical and ML mechanisms that operate over the data collected through runtime monitoring.

Learning safety traces and detecting deviations can be seen as an application of anomaly detection (CHANDOLA et al., 2009) techniques, which consist of searching for data patterns that do not conform to a well-defined notion of normal. In this sense, some ML techniques have been adopted to automate the process of detecting anomalies in embedded systems by many works, e.g., (EZEME et al., 2018; VIEGAS et al., 2017; PARK et al., 2020), and (ZHOU et al., 2015). ML classification algorithms are among the most adopted alternatives for anomaly detection (GHOSH; SCHWARTZBARD, 1999; OTT; MAHAPATRA, 2018; EZEME et al., 2018). These techniques operate as pattern-matching detectors under the assumption that normal behavior can be learned from the dataset (CHANDOLA et al., 2009). Alternatively, Forrest et al. (FORREST et al., 1994) introduced a change detection algorithm based on the Human Immune System for self-nonsel self discrimination in a computer. Artificial Immune

Systems (AISs) have been applied in several works in the context of anomaly detection, intrusion detection, and other aspects of computer security, as indicated in Fernandes et al. (FERNANDES et al., 2017).

Notwithstanding, as CPS fundamentally links software operations to physical processes, effective IDS solutions for this domain must transcend the traditional cyber-centered techniques to consider the physical processes associated (FARAMONDI et al., 2021). In fact, works like Mitchell and Chen (MITCHELL, R.; CHEN, I.-R., 2015) and Cho and Shin (CHO; SHIN, 2016) proposed IDS mechanisms in which models consider features representing the physical process. While the former integrates data collected from sensors to validate the behavior of medical applications, the latter considers time skew to monitor CAN messages, a characteristic inherited from the channel where messages are transmitted. The main idea is to consider physics-informed models or invariants to detect inconsistencies that can be reflected in specific physical properties, temporal behavior, sensor readings, etc, aiming to correlate anomalies across both the digital and physical domains. In this context, Raissi et al. (RAISSI et al., 2019) proposed PINNs, which are neural networks trained to solve supervised learning tasks while respecting any given laws of physics described by general nonlinear partial differential equations. The application of PINNs in the context of IDS has been investigated by recent works like (MISHRA; CHAURASIYA, 2025) in which a PINN is used to predict whether attacks could manipulate IoT data.

In the meantime, Modern CPS are also facing a transition in design methods, departing from the traditional task-centric, local model to a broader data-centric, distributed view (ZHANG, J. et al., 2011; JIANG et al., 2018; FRÖHLICH, Antônio Augusto, 2018; SARKER, 2021). Autonomous vehicles, for instance, utilize data to compute their current state, plan actions, and actuate, all towards a specific goal of reaching a destination with no collisions. CPS are inherently real-time systems, requiring periodic actuation to ensure correct execution (KOPETZ; BAUER, 2003). As data drive these systems, temporal requirements are imposed on data generation, communication, transformation, and consumption (FRÖHLICH, Antônio Augusto, 2018; HOFFMANN; FRÖHLICH, Antônio Augusto, 2022).

In this context, *SmartData* (FROHLICH, Antonio A.; RESNER, 2018) rises as a high-level abstraction for the processing, scheduling, synchronization, and communication primitives used by CPSs. It leverages the myriad of features available on the embedded platforms to deliver a data-centric interface for sensing, actuating, and controlling. SmartData encapsulates data with metadata that makes it self-contained in terms of semantics, spatial location, timing, and trustworthiness. It is meant to be the only application-visible construct in the platform and therefore implicitly mediates all system-level services, expressing interaction between components through Interest relations, with attributes like freshness, periodicity, and security being inherited by such

relations.

In this thesis, we leverage the SmartData construct and its ability to model complex systems to build a IDS solution for CPS that takes into consideration the cyber and physical domains. The IDS solution considers temporal behavior and physical plausibility of the SmartData messages exchanged amongst components of the CPS. Finally, the solution builds upon SmartData properties and explores its metadata, which encapsulates timing properties and the data semantics used to verify physical plausibility.

The suitability of the proposed IDS mechanism extends across several CPS domains where modeling the inherent dependency between cyber and physical assets is paramount. In the realm of autonomous vehicles, the solution can be explored considering the temporal consistency of control commands and the governing physical laws of the vehicle's dynamics, preventing the catastrophic outcomes of unauthorized remote interventions. Similarly, in e-medicine and healthcare networks, the system ensures the integrity of life-critical monitoring and automated treatment delivery by verifying that data from medical devices align with biological plausibility and strict real-time requirements on component interaction. Beyond these, other CPSs such as smart grids and industrial infrastructures can also benefit from the ability to model and detect timing anomalies or physically inconsistent state changes to identify intrusions. By grounding the AIS-based detection in the timing characteristics and properties of the physical processes related to the CPS, the solution provides a robust defense layer for any mission-critical infrastructure where correct operation is fundamentally dependent on the coherence between cyber and physical domains.

1.1 PROPOSAL AND GOAL

In light of the scenario above, features like ML and connectivity are key for modern CPS to increase autonomy and user convenience. However, such features often increase the exposure of these CPS to a variety of cyberattacks. Given the discussed mechanisms for component design, resource isolation, monitoring, and anomaly detection, this work aims at detecting intrusion in cyber-physical systems, with a low false-positive rate, by detecting the violation of the temporal properties of the cybernetic components of such systems, as well as the physical plausibility of the data they use to model CPS physical processes. The temporal behavior is observed by monitoring the messages exchanged between components and expressed through AISs-based models trained in a controlled environment, before the deployment of the system, and assuming that the computing platforms on which such components execute provide resource partitioning mechanisms. Meanwhile, the physical plausibility is obtained by using ML-based techniques for approximating and modeling the properties and functions that govern the operation of the CPS.

The proposed solution considers components to be modeled following a data-

driven approach, which allows for extracting properties and models for the physical plausibility of the data they use to model the state of the operation CPS. Moreover, we take advantage of the time messages are exchanged to extract the temporal features that model the temporal behavior of the components exchanging them, which will be used alongside the messages metadata to build the AIS-based models.

The proposed work is evaluated in terms of accuracy and computational overhead. In the former one, a relevant dataset on IDS for CPS is considered. For the latter, a platform with resource isolation was selected, and metrics like memory utilization and computational overhead will be observed. Finally, the computational complexity for the final AIS model, which incorporates temporal behavior and physical plausibility, is also calculated.

1.2 SPECIFIC GOALS

The proposed mechanism for detecting intrusion will be designed according to the following set of specific goals:

- Adapt the AIS-based IDS design proposed by (HORSTMANN, Leonardo Passig, 2020; HORSTMANN, Leonardo Passig; FROHLICH, Antonio Augusto, 2022) to model temporal properties observed on messages exchanged by CPS software components.
- Adapt the AIS-based IDS solution to encompass the physical plausibility of the data exchanged by the CPS software components based on properties extracted by their respective SmartData model.
- Define evaluation scenarios and build the models of safe (i.e., normal, expected) behavior. The evaluation scenario definition will include the specification of technologies applied for the resource isolation of software components and the selection of relevant datasets for IDS applications on CPS.
- Evaluate the performance of the solution in terms of overhead, latency, jitter, and misbehavior detection accuracy.

1.3 ENVISIONED CONTRIBUTIONS

The contributions of this work include the design, implementation, and evaluation of a mechanism for detecting intrusion in CPS according to temporal properties and physical plausibility. The main envisioned contributions are summarized as follows:

- The design and implementation of a mechanism based on AIS to monitor the behavior of a CPS, accounting for temporal features, through the messages exchanged by its components.

- The extension of the design AIS-based IDS to consider the physical plausibility of the data exchanged by CPS components based on properties extracted from the data-driven model of these components.
- The assessment of the performance of the proposed IDS in terms of overhead, latency, and the accuracy of the behavior models.

1.4 DOCUMENT ORGANIZATION

The remainder of this text is organized as follows: Chapter 2 presents the Background for this work, including the fundamental concepts that support its development, e.g., data-driven Cyber-Physical Systems (CPS), SmartData, resource isolation in modern computing systems, and machine learning elements for intrusion detection. Chapter 3 presents state-of-the-art solutions regarding intrusion detection with special focus on CPS-dedicated solutions. Chapter 4 presents the design of the Artificial Immune System (AIS)-based IDS solution, the proposed mechanisms for modeling temporal behavior and physical plausibility, including a time complexity analysis of the proposed solution. Chapter 5 presents our proof-of-concept implementation and results. We describe the chosen real-world CPS application that encompasses network and physical process data, the preprocessing steps, the training of the AIS models, the mechanism for assessing physical plausibility, and the platform and configurations chosen for performance measurements. Finally, we present the results for the IDS training and classification, demonstrating the effectiveness of the proposed approach. Chapter 6 concludes this work with a recapitulation of the main contributions and final remarks.

2 BACKGROUND

This chapter is dedicated to presenting the basic concepts supporting the development of this work that are not directly connected to its contributions: Data-driven CPS design, SmartData concepts, Machine Learning, and Artificial Immune Systems basic concepts. Related work is presented later in this work, in Chapter 3.

2.1 DATA-DRIVEN CYBER-PHYSICAL SYSTEMS DESIGN

CPS are inherently real-time systems, requiring periodic actuation to ensure correct execution. In this sense, TTA (KOPETZ; BAUER, 2003) is a widespread paradigm for designing critical systems. TTA promotes a task-centered view of the system in which real-time applications are decomposed into nearly autonomous clusters and nodes. Design considers architecture and node levels. In the architecture design, the application is decomposed into clusters and nodes, followed by the specification of timing and values of the communication interfaces to define the time-division schedules of the communication network. At the node design stage, the software must be validated in terms of the temporal constraints established during the architecture design.

Notwithstanding, a CPS can be seen as a composition of software and hardware components, where the outputs produced by some are used as inputs by others. Zhang (ZHANG, L., 2014) states that CPS integrate two distinct components: a set of sensors that produce data and a set of communication and computation systems that aggregate data. In fact, works like Wuyts et al. (WUYTS et al., 2005) proposed a data-centric component-based system design, where a component consists of an entity that reads and writes data available at its ports, and may produce effects in the physical world. In their scheme, ports represent data that is read or written by a component, and may be shared with other components. In multi-component systems, a connector is used to specify the data-sharing relationships between components, describing their interaction.

Another approach for data-driven design is Dataflow-based designs, such as Synchronous Dataflow Graphs, introduced by Lee et al. (LEE, E. A.; MESSERSCHMITT, 1987). The approach proposed by Lee et al. represents tasks as nodes in a directed graph, and nodes are connected by edges that specify the data dependencies. The dataflow paradigm introduced by (LEE, E. A.; MESSERSCHMITT, 1987) motivated data-driven design methodologies such as AUTOSAR (AUTOSAR, 2016). In his work on specification and verification of safety properties in data-driven CPS, Hoffmann (HOFFMANN, 2025) explored and extended concepts of SmartData (FRÖHLICH, Antônio Augusto, 2018) to model data-driven CPS. The remainder of this section introduces SmartData and its main concepts relevant to the present work.

2.1.1 SmartData

SmartData is a self-contained data construct proposed by Fröhlich (FROHLICH, Antonio A.; RESNER, 2018) as an alternative to model CPS components through the data they produce and consume. The SmartData Interface, presented in Figure 1, comprises information regarding location, temporal requirements (period and expiry), and specific operation modes for modeling CPS sensors, controllers, and actuators. SmartData is meant to be the only application-visible construct to mediate all system-level services, including communication, synchronization, and interaction with transducers and actuators.

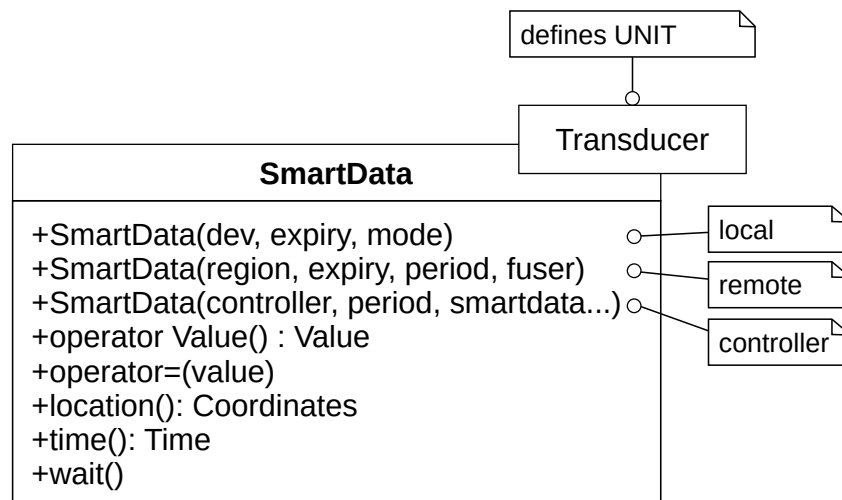


Figure 1 – The SmartData Interface (FRÖHLICH, Antônio Augusto, 2018).

A CPS modeled with SmartData is composed of components that interact with each other through messages implicitly exchanged over a communication channel to guarantee the consistency of the application-visible data. Communication is transparent and can be done over Network, CAN, or any other connection supported by the underlying operating system.

Instances of SmartData can be defined as Local, Remote, or Controller. A local SmartData abstracts local transducers (sensors or actuators), directly connected to the device where the SmartData is running. Remote SmartData acts as a local proxy for transducers on a remote SmartData. A controller SmartData encapsulates control algorithms that tie sensors and actuators.

2.1.1.1 Modeling Data Relationships

Of special interest for this work is how SmartData models the data interactions between the different CPS components. In the SmartData design, CPS components and subsystems are viewed as nodes that interact following a publish-subscribe policy. Subscribers send *Interest* messages $i(type, interval, period | event, expiry)$ to express their interest in a given *type* of data produced in the local network during a given time

interval. Nodes that match the given criteria produce *Response* messages respecting the timing expressed by *period* or on the occurrence of a given *event*. The data carried in a response message is assumed to be valid for the Interested applications until it expires at the time instant *expiry*.

A response message $r(\text{type}, \text{timestamp}, \text{expiry}, \text{data})$ carries the requested data along with information about its *type*, a *timestamp*, and an *expiry*. Interest messages can be dynamically revoked or injected into the network by nodes reacting to special circumstances or changes in CPS operating mode.

The relationships established among SmartData via Interest messages intrinsically define a formal representation of the data dependencies between CPS components. In this way, whenever a SmartData A depends on data produced by a SmartData B , it issues an Interest request expressing the sampling mode along with the required periodicity, location, and interval of interest. Upon receiving the Interest declaration, SmartData B will adapt its behavior (i.e., the periodicity in which it produces and communicates data) to match the requirements expressed by SmartData A .

Given this formal interaction, a CPS C can be expressed as $C = \{S, I\}$, where S represents the set of SmartData components that compose a CPS and I defines the set of interests that define their interactions and data dependencies. This formal relationship between CPS components will later be explored in this work for the classification process as a means to determine the plausibility properties associated with each communicating CPS component.

2.2 MACHINE LEARNING FOR ANOMALY AND INTRUSION DETECTION

Machine Learning methods are AI techniques that automatically improve themselves with experience (MITCHELL, T. M., 1997), by learning meaningful patterns from previously collected information and adapting their structure to better fit their objective. ML methods have inherent limitations. However, effective algorithms have already been developed for certain types of learning tasks (MITCHELL, T. M., 1997). One can separate learning tasks into two major groups: supervised and unsupervised. For supervised learning, algorithms learn to predict the class or value of unobserved data points based on a concept modeled from a training dataset (BADILLO et al., 2020). The learning problem can be seen as a function approximation problem, more specifically, the approximation of an unknown function $F(X)$ that maps the inputs X into a value Y , thus, approximating $F(X) = Y$ or $F : X \mapsto Y$ based on the inputs X with corresponding label Y . In fact, function approximation problems can be separated into two classes: classification problems and regression problems (NISSEN, 2007).

Classification problems define tasks where the output is **discrete** (NISSEN, 2007). Algorithms in this class aim to categorize (i.e., classify) input into one or more groups. They learn patterns from a labeled dataset (inputs mapped to classes) to

differentiate between classes. In this way, a classification problem approximates a function that maps an N-dimensional input to a set of discrete class labels:

$$f : \mathbb{R}^n \rightarrow \{C_1, C_2, \dots, C_k\}$$

Regression problems define tasks where the output is a **real value** (NISSEN, 2007). In other words, regression ML models produce numerical outputs instead of a class. Similar to classification, these algorithms learn to map an N-dimensional input to a corresponding output. Therefore, one can model a regression problem as learning a function f whose output is a continuous number:

$$f : \mathbb{R}^n \rightarrow \mathbb{R}^1$$

A classification problem is often implemented by the learning algorithm as a set of binary outputs, one per class, determining input membership. This approximation can be performed aggressively, using only a few input features (NISSEN, 2007). Conversely, regression problems typically require a smooth approximation, often achieved by fitting a mathematical function for real values.

Meanwhile, when considering unsupervised learning tasks, algorithms are built to identify unknown patterns in data without pre-existing knowledge, like groups or classes (BADILLO et al., 2020).

Clustering mechanisms, such as KNN, DBSCAN, and K-Means, that define a function g that maps an input X to an internal cluster assignment Z :

$$Z = g(X)$$

and frequent patterns detection, such as association rules (discussed later in this chapter), are common mechanisms for this scenario.

The main difference from supervised learning scenarios is that, when dealing with unsupervised learning methods such as clustering and pattern detection, one is performing tasks related to discovering patterns and similarities in data, not approximating a function. Semi-supervised learning scenarios allow for combining both applications. For instance, using clustering to discover underlying data relationships and behavioral patterns, or using regression techniques to learn the functions that govern the correct behavior of a CPS.

For the specific case of considering intrusion detection as an instance of discriminating self from nonself, this work addresses a classification problem with two classes (self and nonself). Nevertheless, as the proposed IDS solution considers only the concept of normal system behavior, we are dealing with a semi-supervised learning task.

2.2.1 Intrusion Detection Methods

Intrusion detection focuses on identifying nonconforming behavior, or system misbehavior. Techniques for detecting misbehavior are categorized into:

- **Misuse Detection or Signature-based Detection:** it models known system attacks as specific patterns (i.e., signatures) and systematically scans the system for their occurrence. These techniques identify predefined misbehaviors but **cannot deal with unseen patterns**.
- **Anomaly Detection:** it defines misbehavior as a **deviation from established normal behavior**. Detection requires first profiling the system's normal state. Anomaly detection handles unseen abnormal patterns, though it **cannot discriminate between different misbehaviors** (GHOSH; SCHWARTZBARD, 1999).
- **Hybrid Detection:** It considers the application of both mechanisms, either by using signature-based mechanisms to classify an abnormal behavior or using anomaly detection to verify unmatched observed patterns.

As the present work aims to detect any intrusion, we will focus on **anomaly detection** concepts.

2.2.1.1 Anomaly Types

Anomaly detection techniques can be classified according to the type of anomaly sought (CHANDOLA et al., 2009):

- **Point Anomalies:** An individual data instance is identified as anomalous when compared to the rest of the data. Example: A data point that does not belong to any cluster representing normal behavior.
- **Contextual Anomalies:** The data exhibits anomalous behavior given its context, but would be considered normal in a different context. Contextual attributes can include time or geographical location. Example: A high-temperature reading is anomalous in winter but normal in summer.
- **Collective Anomalies:** A collection of related data instances is anomalous considering the entire dataset, even if each individual instance is normal. The occurrence as a collection (e.g., a sequence) indicates the anomaly. Example: An invalid sequence of data instances.

The present work performs detection by considering each data sample individually (mechanism presented in Chapter 4), thus searching for point anomalies.

2.2.1.2 Anomaly Detection Operational Models

Anomaly detection techniques operate under three models, classified by the extent of training data label availability (CHANDOLA et al., 2009):

- **Supervised Anomaly Detection:** Training data contains labeled instances for both normal and anomalous classes. A typical approach is building a predictive model for normal vs. anomaly classes to classify data instances. A challenge is the frequent scarcity of labeled anomalous data compared to normal data, which can induce model overfitting.
- **Semi-supervised Anomaly Detection:** Training data consists **only of labeled instances of the normal class**. The typical approach is building a model of normal behavior and using it to identify deviations. An alternative is treating the only labeled data instances as anomalous, which aligns with misuse detection techniques that model specific attacks (GHOSH; SCHWARTZBARD, 1999).
- **Unsupervised Anomaly Detection:** These techniques **do not require labeled data** for training. They rely on the implicit assumption that normal instances are far more frequent than abnormal ones; otherwise, the techniques suffer from a high false alarm rate.

Given the nature of the intrusion detection problem, this work proposes a technique operating in the **semi-supervised anomaly detection** mode, assuming normal behavior can be collected to build a model of self. When a program is misused (attacked or violated), its behavior deviates from normal usage (GHOSH; SCHWARTZBARD, 1999). Consequently, if normal program behavior is adequately captured in a compact representation, its behavioral features can be utilized for intrusion detection. Most intrusion detection tools are based on anomaly detection through the profiling of normal program behavior (GHOSH; SCHWARTZBARD, 1999).

2.2.2 Feature Selection

A key for developing reliable and accurate Machine Learning (ML) models is the construction of a high-quality dataset, characterized by features that meaningfully represent the target application. This process inherently involves filtering, selecting, and extracting relevant features from the available datasets. Visalakshi et al. (VISALAKSHI; RADHA, 2014) assert that the primary objective of feature selection is the systematic elimination of inappropriate and redundant data, a practice which directly contributes to enhanced performance of subsequent learning algorithms.

The strategic application of feature selection offers several critical advantages. First, it reduces the dimensionality of the feature space, which not only mitigates the computational complexity associated with training and prediction but also addresses

another problem often associated with the dataset dimensionality, where data sparsity degrades model performance. Second, by focusing the model on the most informative variables, feature selection often yields more accurate models by reducing noise and improving generalization ability, avoiding problems related to overfitting. Third, the resulting models are typically more interpretable because they rely on a smaller, more relevant subset of variables, offering clearer insights into the underlying data generation process. Finally, utilizing fewer features leads to faster training and execution times and lower memory requirements.

Feature selection techniques are conventionally categorized into three main paradigms (BOLÓN-CANEDO et al., 2012): Filters, Wrappers, and Embedded methods. Filter methods operate as a preliminary preprocessing step and evaluate features based solely on the intrinsic characteristics of the data, independent of the learning algorithm (e.g., Pearson's Correlation Coefficient, Information Gain, and Relief). Wrapper methods incorporate a specific predictive model, treating the feature selection process as an optimization problem where the performance of the predictor is maximized (e.g., Wrapper C4.5, and Recursive Feature Elimination). Embedded methods integrate feature selection directly into the model training procedure, making them inherently specific to the chosen learning method (e.g., Lasso, and FS-Perceptron).

It must be noted, as highlighted by Bolón-Canedo et al. (BOLÓN-CANEDO et al., 2012), that no single algorithm constitutes the "best method" for feature selection. A large diversity of feature selection algorithms exists, and the quality of their results is heavily contingent upon the specific characteristics and particularities of the dataset under scrutiny. Consequently, a more robust, data-independent, and less algorithm-specific solution can often be attained by combining the outcomes from multiple feature selection methods (MOLINA et al., n.d.). In the context of this work's experimental design, a multiple-algorithm feature selection approach will be employed. The goal is to verify the disparities in the results produced by various methods, thereby supporting the reasoned selection of the algorithm or combination of algorithms that yield the most representative feature set for the intrusion detection problem within the selected case study.

2.2.3 Association Rules

Association rules are a type of Contrast Patterns (CP) (AGRAWAL et al., 1993; LOYOLA-GONZÁLEZ et al., 2020) in which patterns are associated with attributes or classes. In CP, a pattern p is an expression that describes a collection of objects according to a given property or set of characteristics that are usually represented by a set of relational statements rs over the monitored variables defined as

$$rs = [v_j < operator > < value >] \quad (1)$$

where v_j is the value of the j^{th} monitored variable in the sample S_i and the *operator* is a relational operator (e.g., $>$, $<$, $\in$, $\notin$, $=$, $\neq$).

In this way, one can define a pattern p as a set of $m \geq 1$ relational statements combined using Boolean operators (i.e., $\wedge$ and $\vee$), for instance

$$p := rs \mid p_1 \wedge p_2 \mid p_1 \vee p_2 \quad (2)$$

Finally, an association rule is defined by associating the occurrence of a pattern p_i with the occurrence of another pattern p_k in the form $p_i \implies p_k$.

Association rules are extracted by searching historical data for repeating if-then patterns and selecting relevant relationships by considering their support and confidence, with support meaning how frequently a pattern p_i is observed and confidence indicating how often the association $p_i \implies p_k$ is demonstrated to be valid.

2.2.4 Neural Networks (NN)

NNs (MCCULLOCH; PITTS, 1943) constitute one of the most widely adopted families of ML algorithms, particularly effective in regression problems where the desired output is a numerical value rather than a class or label. Key applications include autoencoders, data prediction, and filtering. The most common ANN structure is the multilayer feedforward ANN (NISSEN, 2007).

ANNs are structured as layers of interconnected artificial neurons, with connections typically flowing only in a forward direction between layers. A fundamental ANN architecture is composed of three primary layer types: the input layer, which receives the input data; the hidden layer (often extended to a multilayer structure to increase complexity); and the output layer, which yields the final result of the ANN.

Each neuron is a computational unit that aggregates a set of incoming connections C , a bias b , a weight vector W (composed of weights for each incoming connection), and a specific activation function σ . The activation function $\sigma(x)$ is responsible for smoothing the neuron's output based on the weighted sum of its inputs. Consequently, the knowledge acquired by an ANN during training is encapsulated within the weights of its connections. The objective of ANN training is thus to minimize the deviation between the output layer's result and the desired output, specifically by reducing the regression or classification error. The backpropagation algorithm (WERBOS, 1974) is the most prevalent training method; it operates by propagating the error signal from the output layer backward to the input layer, thereby adjusting the connection weights iteratively.

Beyond fully connected layers, ANNs can integrate RNN or CNN layers to address different data structures. The core concept of RNNs is the introduction of short-term memory from previous computational steps. LSTM networks (HOCHREITER; SCHMIDHUBER, 1997) are a prominent extension of the recurrent layer structure, incorporating both short-term (hidden state) and long-term memory (cell state). This

design enables the network to learn temporal correlations within time series data. By learning to modulate its connections (often through gates weighted in the range $[0,1]$), the LSTM layer can selectively retain or discard information, facilitating the learning of correlations across present, short-term, and long-term contexts. However, recurrent layers are generally more computationally intensive than standard fully connected ANNs due to the sequential nature of the time series data they process.

Conversely, the CNN class of ANNs finds its primary application in image processing (GOODFELLOW et al., 2016), such as semantic segmentation, but is also applicable to time-series analysis (CHEN, X. et al., 2021). Each hidden layer in a CNN typically comprises a set of convolutional layers, which extract features using shared-weight filters, followed by a pooling layer, which summarizes the extracted features by deriving a summary statistic of nearby outputs. For seismic data, a potential application involves feature extraction after the data is transformed into an image-like representation, such as through a wavelet transform. Ultimately, these various layer types can be combined within a single architecture to leverage the strengths of each technique. For example, a hybrid model could use CNN layers for feature extraction, LSTM layers for learning temporal correlations, and a final fully connected layer to compose the sequence data and produce the desired output.

2.3 ARTIFICIAL IMMUNE SYSTEMS - AIS

The Artificial Immune System (AIS), as initially proposed by Forrest et al. (FORREST et al., 1994), draws its analogy from the T-cell negative selection mechanism of the human immune system. This biological process generates T-cells (lymphocytes) with surface receptors designed to detect foreign bodies (antigens) upon protein binding.

In the computational model, the AIS generates a set of random detectors, where each detector is a string that must not match any instance of protected data (considered "self"). Both the detector strings and the protected data strings are partitioned into equal-length segments, also referred to as "self" strings. This segmentation, according to the original authors, simplifies the mathematical analysis and enhances the capability to detect specific types of computer viruses.

To ensure that no detector incorrectly identifies "self" data as an anomaly, the randomly generated set of detectors undergoes a censoring process. This process is governed by a parameter r , which defines the matching criterion. Specifically, two strings, s_i and s_{ij} , are considered a match if and only if they share a similarity (or match) at least r contiguous locations (or segments). Detectors matching any "self" string during censoring are eliminated, ensuring the remaining set is tolerant to the notion of *self*.

Figure 2 illustrates the censoring process for an initial set of randomly generated

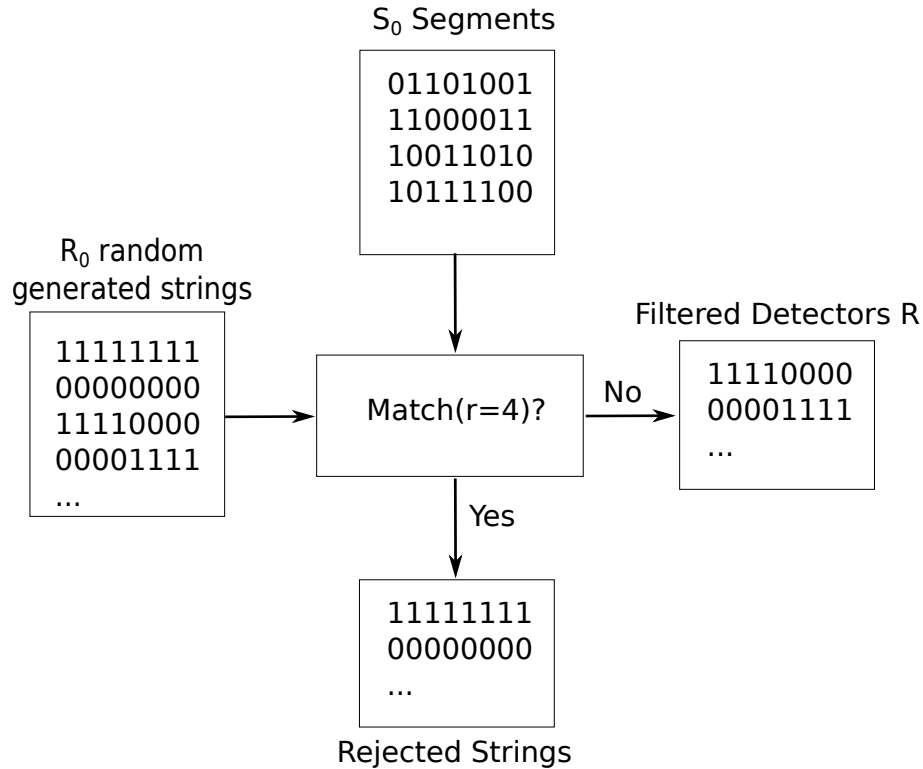


Figure 2 – Example of Censoring. Figure based on the algorithm proposed by Forrest et al. (FORREST et al., 1994)

detectors, R_0 . Consider a protected string $S_0 = 01101001110000111001101010111100$, partitioned into segments of size $L = 8$. The matching criterion is defined by $r = 4$, which corresponds to a binding threshold of 50% similarity (i.e., $r/L = 4/8 = 0.5$) between two strings. In this context, anomaly detection occurs whenever a protected segment s_i matches a detector s_{ij} . Furthermore, the matching rule is defined as s_i and s_{ij} must have at least r equal characters in sequence.

The methodology proposed by Forrest et al. directly mirrors the biological process of T-cell generation and negative selection within the human immune system. The biological process involves T-cells, which are lymphocytes equipped with surface detectors for foreign proteins. These cells are randomly generated and subsequently migrate to the Thymus, an organ responsible for their maturation. Here, they undergo negative selection (or censoring) to eliminate any T-cell that recognizes "self-peptides," thereby preventing autoimmune responses. The surviving T-cells form the immune system's defense repertoire.

The computational analogy models T-cells as randomly generated strings called detectors (or sometimes antibodies in other AIS contexts). These detectors are subjected to a computational censoring process that implements the negative selection principle. The resultant set of non-matching detectors constitutes the system's defense mechanism against intrusions. The efficacy of this technique is quantified by Forrest et al. through a mathematical analysis of the probability of failure to detect an intrusion.

This probability is calculated based on the N modeled detectors and their collective inability to register the presence of an anomalous pattern.

We inspire our solution in the concept of using AIS for self-nonsel discrimination. However, instead of adopting the original approach proposed by Forrest et al. (FORREST et al., 1994), in which the AIS detectors are built to perform nonself detection, we model our set of detectors to recognize self. Fernandes et al. (FERNANDES et al., 2017) stated AISs are modeled with negative or positive samples to distinguish self, which they classify as the set that characterizes the normal behavior of the monitored system, from nonself, which they classify as the samples representing abnormalities.

Boracchi and Roveri (BORACCHI; ROVERI, 2014) introduced a change detection technique that models *self* behavior based on the self-similarity of time-series data segments. Their approach detects a structural change in a data segment by analyzing a change indicator value, which is derived from comparing the current data path to the most similar path previously observed. A structural change is detected precisely when this self-similarity is broken. However, this technique is fundamentally designed for a univariate time-series, which consists of only a single variable and its corresponding timestamps.

In contrast, systems often involve multivariate time-series, where multiple variables are monitored simultaneously. This context offers an advantage: the increased dimensionality allows for a more precise definition of the system's normal behavior. Dasgupta and Gonzalez (DASGUPTA; GONZALEZ, 2002) argue that analyzing the individual values of separate variables in isolation can lead to inaccurate classification of abnormality. They demonstrate that the combination of values across different monitored variables is necessary to indicate abnormalities with significantly higher precision.

Dasgupta and Gonzalez (DASGUPTA; GONZALEZ, 2002) compared two techniques for intrusion detection based on Artificial Immune Systems (AIS): positive characterization and negative characterization. For positive characterization, the self space was defined directly using the self (normal) samples. Detection of non-self (anomalies) was achieved by comparing the minimum distance between a new sample and any member of the self space against a predefined threshold.

In the negative characterization approach, detectors were modeled as a set of rules based on the feature vector. These rules were evolved using genetic algorithms, with the objective of optimizing the balance between the number of detectors and the coverage of the abnormality subspace.

The results showed that positive characterization, which relies on defining the self space explicitly with samples, was superior in detection performance. It achieved up to 8.9% better true positive rates while maintaining a maximum false alarm rate of 1%, and was capable of detecting more attack types (5 versus 4 for negative characterization). However, positive characterization demanded significantly higher memory and

time resources. The increased memory utilization stems from the strategy of retaining all self samples to establish the system's complete notion of self space. This same factor also contributed to the increased execution time.

3 LITERATURE REVIEW

In this work, we assume intrusion and misbehavior detection as an application of anomaly detection (CHANDOLA et al., 2009). For instance, abnormal patterns on the messages exchanged by a CPS can be caused by attacks such as DDoS and Physical Attacks (SINGH et al., 2011; PRATIBHA; VASHIST, 2014). Therefore, this section will cover relevant state-of-the-art approaches in the context of ML-based anomaly and intrusion detection for CPS.

3.1 INTRUSION DETECTION SYSTEMS

Intrusion Detection Systems (IDS) are a vital component in the scope of cybersecurity, acting like a mechanism that recognizes the various forms of intrusion (ABDULGANIYU et al., 2023). Many works investigate IDS applications, mainly in the scope of Network IDS (DINA; MANIVANNAN, 2021; ZHONG et al., 2024). Kunal and Dua (KUNAL; DUA, 2019) reviewed typical ML approaches for intrusion detection, from the preprocessing and dataset handling to the ML methods and detection strategies. Abdulganiyu et al. (ABDULGANIYU et al., 2023) reviewed signature-based (misuse detection), anomaly-based, and hybrid intrusion detection techniques. The authors pointed out that IDS solutions are predominantly built considering anomaly-based techniques, with the most used techniques being NN-based mechanisms like CNN, ANN, LSTM, and AE, and the most accurate models using a combination of CNN and AE, which achieved nearly 100% performance. Nevertheless, authors also point the high utilization of computational resources may hinder the adoption of such mechanisms.

A similar conclusion is taken by Lansky et al. (LANSKY et al., 2021), which reviewed deep learning-based IDS techniques, considering different methods, implementations, datasets, and evaluation metrics, but also pointed the high computational needs, training latency, and computation complexity. Neupane et al. (NEUPANE et al., 2022) reviewed explainable AI solutions for IDS. They discuss current challenges related to the design of IDS and review black box and white box approaches and tradeoffs between their performance and ability to produce explanations. They close the review by proposing a solution that considers human-in-the-loop to be used as a guideline when designing an explainable IDS.

Otoum and Nayak (OTOUM; NAYAK, 2021) proposed a hybrid approach combining anomaly- and signature-based IDS for IoT applications. In their approach, the signature-based IDS subsystem applies a Lightweight Neural Network using Human Mental Search (HMS), a population-based metaheuristic algorithm, for traffic clustering in the hidden layer and Boyer-Moore, an efficient string-search algorithm, to search particular signatures in the output layer, classifying them as intruder, normal, or unknown. The anomaly-based IDS module then utilizes deep learning techniques to identify un-

known attacks, employing SNR and bandwidth analysis to classify data. Overall, their approach demonstrates the ability to reduce false positives in network traffic analysis when compared to methods focusing solely on anomaly- or signature-based detection.

Duque and Omar (DUQUE; OMAR, 2015) investigated K-means clustering in the context of IDS. Their approach proved to yield better results when configuring the expected number of clusters to the number of different types of data represented in the target dataset. The dependency on a previously known number of clusters, however, limits the applicability of such a mechanism to real-world dynamic systems. To address this limitation, a combination of different techniques, including signature-based approaches, is recommended by the authors.

Kim et al. (KIM et al., 2020) proposed a method that integrates Deep Learning techniques for real-time web intrusion detection. Their technique uses a CNN-LSTM model for spatial feature learning to extract the main characteristics of the real-time web traffic. They also combine their solution with a labeling tool, which allows for better distinguishing sophisticated attacks from normal behavior. Besides the improvements in results, a drawback for this approach is the dependence on security analysts to label data and the utilization of complex machine learning models.

When considering using Deep Learning ML models for IDS, Shurman et al. (SHURMAN et al., 2020) proposed a hybrid and an LSTM Deep Learning-based IDS focusing on DoS attacks. Aktar and Yasin Nur proposed a Deep Learning approach using a contractive AE for anomaly-based DoS and Distributed DoS detection. They trained models to learn normal traffic patterns from the compacted representation of input data and apply a stochastic threshold to detect attacks. Lastly, Satpathy et al. (SATPATHY et al., 2022) also investigated Deep Learning solutions in this context, using CNN models for DoS and DDoS attack detection.

Gharaee and Hosseinvand. (GHARAEI; HOSSEINVAND, 2016) proposed an IDS mechanism based on SVM and Genetic algorithms. The authors propose a novel feature selection method based on Genetic that is able to reduce dimensionality, which leads to an increased true positive ratio. The genetic algorithm is used as a wrapper feature selection method tied to a SVM classifier, and the increase in true positives comes from a change in the fitness function for the genetic algorithm, in order to decrease the false alarm rate and increase the true positive rate simultaneously, while also aiming at minimizing the number of features for computational efficiency.

3.1.1 IDS for Cyber-Physical Systems

Performing intrusion detection on CPS poses a series of different challenges when compared to traditional IT applications. As previously mentioned, many of these different challenges come from the inherent constraints of many CPS (DIRO et al., 2017). Islam et al. (ISLAM et al., 2024) state that traditional IDS that operate on IT need

to be adapted for CPS applications due to the inherent differences in terms of communication, standards, and unique operation. In this sense, several papers investigated IDS in the context of CPS. Jun and Chi (JUN; CHI, 2014) proposed a mechanism for IDS based on Complex Event Processing (CEP), which is a technique for filtering and processing events in real-time, to deal with data streams in IoT applications, leading to reduced memory utilization and processing time. Shahriar et al. (SHAHRIAR et al., 2020) adopted Generative Adversarial Networks (GAN) to assist the training of IDS solutions. The authors aimed to address imbalanced data and missing samples, a typical issue in CPS applications that directly impacts the quality of IDS models.

In the context of focusing on the specifics of the CPS domain, Humayed et al. (HUMAYED et al., 2017), provided a survey on CPS security and reviewed security vulnerabilities and IDS requirements for different CPS applications. Meanwhile, Ali (ALI, 2021) surveyed detection techniques and application layer for different IDS solutions for CPS in order to extract meaningful trends, identify techniques to improve performance, and layers commonly vulnerable to cyberattacks.

Jeffrey et al. (JEFFREY et al., 2023) surveyed anomaly detection strategies for CPS. The authors argue that due to the extreme diversity of CPS applications, there is no unique solution for anomaly detection that can be generalized for all CPS. According to them, as detection methods increase in generality to aid rapid test case development, they necessarily reduce fidelity for real-world applications, which makes them less representative. Finally, the authors state that the best solution may be a low-level generic framework with a modular architecture that allows plugins to be developed for the unique characteristics of a particular CPS.

Mitchell and Chen (MITCHELL, R.; CHEN, I.-R., 2014) surveyed IDS solutions for CPS. They reviewed design principles and techniques for intrusion detection on CPS and provide a taxonomy for existing CPS IDS techniques and discuss their merits and drawbacks. Their main target consisted of works evaluating the physical processes that govern the behavior of physical devices. They classify IDS solutions for CPS regarding detection technique and auditing material. For the auditing material, the authors considered two possible solutions: host-based techniques, which consider logs and other audit data maintained by each node to verify whether it is behaving correctly, and network-based techniques, which consider the network activity of each node. In this taxonomy, the proposed solution would fit both techniques as we analyze the data produced and exchanged between devices alongside the messages and temporal behavior from CPS software components.

When considering the detection techniques, the authors proposed two categories of works, namely knowledge-based and behavior-based. Knowledge-based IDS mechanisms look for features that match a specific behavior or pattern, i.e., signature- or misuse-based detection, while behavior-based solutions focus on finding features that

deviate from the normal. In their research, Mitchell and Chen highlighted the advantages of behavior- and behavior-specification-based intrusion detection mechanisms when compared to other classes of solutions. These mechanisms aim at building models for normality and detecting deviations and are, in general, less prone to false positives and better-suited for dealing with unknown attack behavior. A similar technique is explored in this work, considering the problem through the optics of modeling the normal behavior and looking for deviations from it. In fact, Mitchell and Chen (MITCHELL, R.; CHEN, I.-R., 2015) proposed a behavior-rule specification-based IDS for Medical CPS. In their discussion, authors argue that the key difference in designing IDS for safety-critical CPS when compared to other types of systems is that the intrusion detection is closely tied with the physical components of the CPS.

Cho and Shin (CHO; SHIN, 2016) proposed CIDS (Clock-based IDS), an anomaly-based IDS mechanism that aims at addressing a common issue for some of the existing IDS solutions, the absence of the transmitters' information in the messages exchanged inside a vehicle. CIDS exploits the periodicity of in-vehicle messages to estimate the clock and tag (fingerprinting) the transmitters for uniquely identifying ECUs. The solution proposed by Cho and Shin builds a normality model for the temporal behavior of a message and detects deviation from this normality model. Other than that, their solution also presents the possibility of correlating the temporal behavior of two different periodic messages from the very same transmitter in order to detect deviations and reduce false-positive rates.

Works like Mitchell and Chen (MITCHELL, R.; CHEN, I.-R., 2015) and Cho and Shin (CHO; SHIN, 2016) proposed IDS mechanisms in which models consider some features of the physical process. While the former integrates data collected from sensors to validate the behavior of medical applications, the latter considers time skew to monitor CAN messages, a characteristic inherited from the channel where messages are transmitted. The present work also fits the idea of modeling normal behavior and focusing on detecting deviations as a means to perform intrusion detection, taking into consideration the physical plausibility of data to weigh into the decision.

In the context of considering physics-informed models or invariants to detect inconsistencies that can be reflected in specific physical properties, Raissi et al. (RAISSI et al., 2019) proposed PINNs, which are neural networks trained to solve supervised learning tasks while respecting any given laws of physics described by general nonlinear partial differential equations. The application of PINNs in the context of IDS has been investigated by recent works like (MISHRA; CHAURASIYA, 2025) in which a PINN is used to predict whether attacks could manipulate IoT data. Alternatively, Scheffel et al. (SCHEFFEL, R. M.; FRÖHLICH, A. A., 2018; SCHEFFEL, Roberto M.; FRÖHLICH, Antônio A., 2019) propose an unsupervised mechanism to assign a confidence value indicating whether a piece of data is believed to be correct in relation to correlated

sensors and variables. Such a mechanism has been adopted in the context of fault detection in CPS applications like (SCHEFFEL, Roberto Milton et al., 2020).

Some works proposed IDS mechanisms that focused on the architectural properties of CPS applications, using HPC to profile system behavior. Ott and Mahapatra (OTT; MAHAPATRA, 2018) employ a subset of performance counters, specifically branches executed, conditional branches executed, loads executed, and stores executed, arguing this selection minimizes interference from other running programs. They evaluated Hidden Markov Models (HMM) and Long Short-Term Memory (LSTM) networks for anomaly detection on these traces, achieving high accuracy (95% for HMM, 98% for LSTM). SiPTA (ZADEH et al., 2014) leverages the intrinsic periodicity of embedded applications to perform an offline, trace-based anomaly detection using signal processing. This technique assigns time series to traces, extracts periodic features using the Discrete Fourier Transform (DFT) to define a trace through metrics like maximum magnitude and a DC-Significance Metric, and achieves a 100% detection rate across its case studies. Nevertheless, SiPTA cannot be applied online and is only suitable for monitoring temporal behavior, not assessing the underlying effects of intrusions on the components of the system.

Recognizing the need for less intrusive runtime analysis, Leng et al. (LENG et al., 2017) developed a lightweight technique for online intrusion detection. They extract application fingerprints using counters such as committed instructions, number of function calls, number of integer instructions, and number of load instructions. The detection mechanism operates by comparing observed HPC values, collected at a 100Hz rate, against expected value intervals established from fault-free execution profiles. This approach confirms the feasibility of identifying normal system behavior and detecting misbehavior by analyzing runtime collected samples in a controlled environment.

Kadar (KADAR, 2022) investigated the integration of IDS on embedded MCS. The IDS mechanisms adopted in her work used code instrumentation techniques to monitor system calls and Hardware Performance Counters (HPC) and Hardware-assisted CFI monitoring with ARM CoreSight. In her work, Kadar also relies on a separation kernel to isolate software running on the user level as a means to guarantee freedom from interference and independence between user-level components. We also consider resource isolation to be a relevant aspect of modern IDS solutions. Nevertheless, guaranteeing temporal and spatial isolation is out of the scope for this work, as we consider it as provided by some external tool. Furthermore, by limiting the IDS mechanism to analyzing architectural fingerprints, we limit the scope of our IDS to CPS applications running in the very same hardware platform. To avoid such a constraint, we monitor messages exchanged by the different CPS components, relying on the periodic behavior of messages exchanged in cyber-physical systems and their physical plausibility to perform intrusion detection.

Almuqren et al. (ALMUQREN et al., 2023) proposed a IDS mechanism using an Improved Elman Neural Network. Their solution also considers the necessity to improve explainability and understanding of the neural network solution. Ericson et al. (ERICSON et al., 2025) also investigated lightweight ML-based mechanisms for IDS, focusing on Edge devices. Their solution also adopted explainable AI solutions for increasing transparency. Rehman et al. (REHMAN et al., 2025) propose a hybrid framework for intrusion detection, combining different techniques in order to cover detection on each layer of a CPS application. In their work, different types of IDS mechanisms are analyzed on their capability to perform one of 12 proposed activities for the desired IDS solution, ranging from the detection of known and unknown threats, to fast decision making, accurate detection, and detecting suspicious behavior.

Zhu et al. (ZHU et al., 2024) explored Federated Learning to build a collaborative intrusion detection mechanism for CPS. They propose a trustworthy model update strategy to detect threats on industrial CPS that enables multiple industrial CPS to collaboratively build an IDS model. Catillo et al. (CATILLO et al., 2023) proposed CPS-GUARD, an IDS approach based on an autoencoder. They used an outlier-aware threshold for distinguishing normal operations from attacks. The outlier-aware threshold definition aimed at mitigating imperfections of the training data. More recently, Catillo et al. (CATILLO et al., 2025) proposed MultiCIDS, an IDS mechanism that focuses on detecting collective anomalies (or intrusions), which are characterized by a sequence of consecutive anomalous data. Their mechanism focuses on time series analysis and, similar to this work, models data as N-dimensional data points where N is defined as the number of monitored variables on the target CPS application.

Santos et al. (SANTOS et al., 2023) provided a comparative study of five offline and three online ML-based IDS solutions using seven different CPS-specific datasets. Their work demonstrates that time and memory constraints, typical of CPS applications, have a negative impact on overall ML-based IDS performance. This was clearer when considering unbalanced datasets (i.e., datasets with clear discrepancies in the occurrences of each learned behavior). The solution presented in this work is independent of dataset balancing. In fact, the number of samples representing a valid behavior in the CPS can be as small as a single occurrence, depending on training parameters. Another finding pointed out by Santos et al. is that online ML-based IDS methods tend to be more adaptable, as they learn new behaviors faster, and are more suitable for real-time applications, due to increased responsiveness (i.e., smaller response time).

Islam et al. (ISLAM et al., 2024) proposed an IDS mechanism based on Generative AI and cognitive computing for Industrial CPS. The authors also adopt a Long Short-Term Memory-based Sparse Variational Autoencoder (LSTM-SVAE) for feature selection and a Bidirectional Recursive Neural Network with Hierarchical Attention (BiRNN-HAID) for detecting complex intrusion patterns with high precision. Finally, a

Cognitive Enhancement for Contextual Intrusion Awareness (CECIA) is introduced for reducing false negatives and false positives. The approach does combine several techniques in order to achieve good results. Nevertheless, the complex process hinders its adoption for real-time operation on real-world CPS.

Faramondi et al. (FARAMONDI et al., 2021) investigated different approaches for IDS on CPS, including KNN, Random Forest, SVM, and NB on a Water Distribution Testbed dataset, where network and physical attacks are introduced. The investigated algorithms presented good results when detecting attacks on the physical data. Nevertheless, the overall accuracy has a significant drop when analyzing network attacks. Motivated by these results, the authors concluded that to achieve better results, the chosen IDS mechanism should consider both network and physical data.

3.2 CPS IDS DATASETS

There are a myriad of IDS datasets available. Mondragon et al. (MONDRAGON et al., 2025) present a benchmark with 14 recent and preprocessed datasets and analyze 7 different open-source algorithms for Network Intrusion Detection based on Network Flows, namely, Leader and Class Confidence Decision Ensemble, Energy Flow Based Classifier, NN, CNN, Deep Belief Network Model, E-GraphSage Based Classifier, and Contrast Network Flow Classifier. The resulting dataset is publicly available.

Some of the datasets adopted by related works are CIC DDoS 2019, from Sharafaldin et al. (SHARAFALDIN et al., 2019), NSL-KDD (TAVALLAEE et al., 2009), KDD 99 (UCI KDD ARCHIVE, 1999), CIC IDS 2017 (SHARAFALDIN et al., 2018), CSE CIC IDS 2018 (SHARAFALDIN et al., 2018; CANADIAN INSTITUTE FOR CYBERSECURITY (CIC), 2018), ToN-IoT (MOUSTAFA, 2019; BOOIJ et al., 2022), and others. All these datasets, however, are strictly focused on Network IDS.

Santos et al. (SANTOS et al., 2023) proposed an investigation over CPS-specific datasets, namely, SWaT, BATADAL, Morris, and Ereno. Developed by the iTrust Lab for Cyber-Physical Systems (CPS) studies, the Secure Water Treatment dataset (GOH et al., 2016) (SWaT) is collected from a scaled-down version of a real-world industrial water treatment plant. This dataset contains 946,722 samples, each with 53 attributes (measurements from sensors and actuators), collected over 11 days. Its primary focus for intrusion detection research is simulating attacks through the injection of various forms of fake data (SANTOS et al., 2023).

The BATADAL (Battle of Attack Detection Algorithms) dataset (TAORMINA et al., 2018) was created for a competition to test attack detection solutions on a Water Distribution System. The dataset is composed of simulated SCADA observations and holds a total of nearly 14,000 samples, with 43 attributes and over 14 attacks. The Ereno dataset (QUINCOZES et al., 2022) is composed of samples from seven different Use

Cases (scenarios) collected from a synthetic traffic generation framework, generating seven different sequences of attacks and normal operations.

The Morris datasets are related to Industrial Control Systems (ICS). The datasets were analyzed by Hink et al. (BORGES HINK et al., 2014), Morris and Gao (MORRIS, T.; GAO, 2014), and Morris et al. (MORRIS, T. H. et al., 2015). The former presents 76,035 samples, each with 128 attributes, from the operation of a system with four smart-relay-controlled circuit breakers connected to a supervisory control and data acquisition system. In this scenario, attacks consist of injecting commands from the substation switch network. The second dataset holds data from MODBUS traffic from two distinct laboratory-scale SCADA systems—a pipeline network for gas and a water storage tank—for intrusion detection analysis. The dataset includes normal operational logs and seven different attack scenarios for each system. The latter dataset expands the pipeline network simulation, including 35 labeled random cyberattacks.

Although these datasets are focused on Cyber-Physical Systems (CPS), the presented attacks do not fully permeate both the cyber and physical domains of the system. In this sense, Faramondi et al. (FARAMONDI et al., 2021) presented a dataset to support the validation of IDS solutions based on AI and ML for the detection and categorization of threats in CPS encompassing both physical and network data to highlight the consequences of attacks in the physical process as well as in network traffic.

The CPS model proposed by Faramondi et al. consists of hardware-in-the-loop WDT that emulates water flowing between eight tanks via solenoid-valves, pumps, pressure, and flow sensors. Their testbed is composed of a real subsystem virtually connected to a simulated one. The real subsystem consists of five tanks, twenty solenoid valves, four pumps, and five pressure sensors on each tank. Other than that, eight manual valves are considered to simulate water leaks from tanks or pipes. To increase the complexity of the scenario, three tanks, two pumps, four flow sensors, two solenoid valves, and three pressure sensors for each tank are added to the CPS testbed. All samples are timestamped. Cyberattacks considered were MITM, 3 different variants of DoS, and 4 different variants of Scanning attack. Physical attacks considered were water leaks and breakdowns of sensors and pumps.

The yielded dataset encompasses data from both the physical system (9,206 samples and 41 features) and the network (nearly 20 million samples and 15 features). Thus, they represent a perfect fit for the experiments conducted in this paper, highlighting attacks in the physical process as well as in network traffic. Network data consists of packet-based features extracted from all network segments. Network traffic was captured with Wireshark. The variables encompass a timestamp, the source and destination IP addresses, source and destination MAC addresses, source port and destination port, protocol, TCP flags, payload size, ModBus function code, ModBus

response value, number of packets from the same source in the last two seconds, and number of packets with the same destination in the last two seconds.

3.3 SUMMARY OF LITERATURE REVIEW

This chapter was dedicated to presenting the Literature Review on IDS, CPS-focused IDS solutions, and discussing relevant CPS datasets for IDS. The literature review established that Intrusion Detection Systems (IDS) applied to Cyber-Physical Systems (CPS) predominantly rely on Machine Learning (ML) for anomaly detection. This implies the identification of normal operational patterns, frequently using deep learning architectures such as Neural Networks (NN), Long Short-Term Memory (LSTM), and Autoencoders (AE) for feature extraction and classification, which are often too complex for CPS application. The specialized nature of CPS, with unique communication constraints and safety-critical functions, requires tailored IDS mechanisms that often incorporate knowledge of the physical process to enhance detection fidelity and reduce false positives. This is highlighted by works like (MITCHELL, R.; CHEN, I.-R., 2014, 2015; CHO; SHIN, 2016).

However, a significant gap remains: current methodologies tend to treat the cyber and physical domains as isolated silos. Network-centric solutions often overlook the physical context, while physics-informed approaches rarely integrate the underlying cyber-temporal behavior into a unified model. The solution proposed in this thesis addresses this gap by combining these perspectives into a resource-efficient, AIS-based framework. By exploring the inherent synergy between cyber-temporal footprints and physical process plausibility, this work provides a holistic approach to IDS tailored for the operational constraints of modern CPSs.

A qualitative comparison between this work and the related works on IDS solutions for CPS is summarized in Table 1.

Table 1 – Comparison of Intrusion Detection Systems (IDS) Works in Cyber-Physical Systems

Work	Strategy	Physical Variables	Cyber Variables	Type
(MITCHELL, R.; CHEN, I.-R., 2015)	Behavior-Rule Specification	✓	✓	Anomaly
(CHO; SHIN, 2016)	Clock-based / Statistical	✓*	✓	Anomaly
(SCHEFFEL, R. M.; FRÖHLICH, A. A., 2018)	ANN	✓	-	Anomaly
(MISHRA; CHAURASIYA, 2025)	PINN	✓	-	Signature
(OTT; MAHAPATRA, 2018)	HMM and LSTM	-	✓	Anomaly
(ZADEH et al., 2014)	Signal Processing	-	✓	Anomaly
(LENG et al., 2017)	Manual Analysis	-	✓	Anomaly
(KADAR, 2022)	CFI	-	✓	Anomaly
(ALMUQREN et al., 2023)	Improved Elman NN	-	✓	Anomaly
(ERICSON et al., 2025)	ML	-	✓	Anomaly
(REHMAN et al., 2025)	Multiple IDS	-	✓	Hybrid
(ZHU et al., 2024)	Federated Learning	-	✓	Signature
(CATILLO et al., 2023)	AE	-	✓	Anomaly
(CATILLO et al., 2025)	MultiCIDS	-	✓	Anomaly
(SANTOS et al., 2023)	ML	-	✓	Anomaly
(ISLAM et al., 2024)	Generative AI and LSTM	-	✓	Anomaly
(FARAMONDI et al., 2021)	KNN, SVM, Random Forest, and NB	✓*	✓*	Anomaly
This work	AIS	✓	✓	Anomaly

4 ARTIFICIAL IMMUNE SYSTEMS-BASED IDS WITH TEMPORAL FEATURES AND PHYSICAL PLAUSIBILITY

The intrusion detection mechanism proposed in this work relies on monitoring messages exchanged by the CPS software components to profile the system's correct behavior in order to detect intrusions by discriminating samples representing the correct behavior of the system (hereby named self samples) and samples representing behavior that deviates from the models for correctness (or simply nonself samples).

The present work adopts a strategy based on anomaly detection, as described in Section 2.2. In this way, the IDS mechanism described in this section operates by modeling the notion of self-behavior, i.e., the IDS model defines the behavior produced by the target CPS as the behavior to be recognized. This notion of correct behavior is modeled based on an AIS-based mechanism, with the construction of the set of detectors based on the generation of T cells by the Human Immune System.

The Human Immune System, or Natural Immune System, is the part of the human body responsible for recognizing foreign cells or molecules that may represent a threat to health. To be able to perform this task, the immune system must distinguish between cells of the own body (self) and external cells (nonself). This process is handled by T cells, which are lymphocytes with receptors on their surface (T-cell receptor, or TCR) that can bind to peptides on antigens, triggering a reaction of the immune system. The T cells are created in the hematopoietic stem cells on the bone marrow. After creation, they are sent to the Thymus gland to mature and pass through a process that censors self-reactive T cells. Finally, the T cells that are not censored compose the Human Immune System.

Despite being inspired in the Human Immune System, the T cells modeled in the proposed solution are designed to detect exactly the opposite as they are programmed to detect in the Human Immune System, i.e., the T cells of the AIS built in this work are a species of "anti-antibodies" once they perform the inverse matching as they are originally designed to. In this way, the notion of self is defined by the T cells (detectors), and an intrusion is detected whenever an incoming cell (i.e., a sample extracted from the exchanged messages) does not bind to any of the T cells.

The following of this chapter describes the original AIS design earlier proposed by the Author (HORSTMANN, Leonardo Passig, 2020; HORSTMANN, Leonardo Passig; FROHLICH, Antonio Augusto, 2022) and extends it by proposing a new construction methodology which relies on modeling the temporal behavior of CPS' software components.

4.1 SELF AND NONSELF DEFINITION

In the present AIS-based IDS solution, self and nonself concepts are modeled around the definition of T cells. A T cell, in turn, is modeled according to the definition of a sample on the target CPS. Following the original definition proposed by the author in Horstmann et al. (HORSTMANN, Leonardo Passing et al., 2019) and (HORSTMANN, Leonardo Passig; FROHLICH, Antonio Augusto, 2022), a sample S_i is a snapshot of the state of the system, represented by the set of all variables V (3) monitored within this system, and collected at a specific time t during its execution.

$$V = \{v_1, v_2, \dots, v_n\} \mid V.v_i \text{ is a monitored variable, and } n = N \quad (3)$$

In this way, a sample S_i is represented by an N -dimensional data-point composed of the values for the N variables that are monitored on the system from which S_i has been collected (4). For the sake of keeping track of the time and the moment of the execution when a sample S_i has been captured, every sample S_i is tagged with a timestamp $S_i.t$ indicating the exact moment S_i acquisition is triggered.

$$S_i = \{t, v_1, v_2, \dots, v_n\}, \text{ where } n = N \quad (4)$$

It is important to notice that, for variables whose values are not synchronously updated, their respective sampling moment will not coincide with $S_i.t$. Therefore, assuming $v_j.t$ as the time a variable $v_j \in V$ was sampled, we have that a sample S_i and its variables respect the property expressed by (5).

$$S_i = \{t, v_1, v_2, \dots, v_n\}, \text{ where } n = N \wedge v_j.t \leq S_i.t \forall v_j \in V \quad (5)$$

With these definitions in mind, one can model the concepts of self and nonself. To better understand this, let us consider that whenever a sample S_i is collected from a CPS software component that has not been compromised (i.e., with correct behavior), the behavior described by S_i represents a valid self behavior. In this way, the nonself concept stands for the behavior described by the samples S_i collected from components executing under any circumstance considered not valid. Notwithstanding, when considering the timestamp for samples and for each monitored variable, one can extend this definition to:

$$S_i = \{t, v_1, v_2, \dots, v_n\} \text{ is self} \iff S_i.t \wedge v_j.t \in [t_0, t_f] \forall v_j \in V, \quad (6)$$

where $[t_0, t_f]$ denotes an interval during which the execution of the monitored CPS was not compromised.

4.2 T CELLS

Once the concepts of sample, self, and nonself are defined, one can model the concepts of T cells. As previously stated, the T cells in this work are meant to represent snapshots of the system captured while it is behaving as expected. In this way, the T cells modeled in this work are built from the set of self samples, S_{self} , collected during the system's execution and filtered by the Thymus in the censoring process. Therefore, S_{self} is a subset of the set S of all the samples, where all $S_i \in S_{self}$ are expected to respect (6). This can be expressed by:

$$S_{self} \in S \mid \forall S_i \in S_{self} \wedge \forall v_j \in S_i, S_i.t \wedge v_j.t \in [t_0, t_f] \forall v_j \in V \quad (7)$$

Each T cell T_i must represent a part of the self notion. In other words, a T cell T_i is an N-dimensional data-point that represents a valid self behavior. Therefore, a T cell T_i is a sample S_i that must respect the rules described by equation (6) and (7). This condition, described by equation (8) is complemented by the fact that the T cells set T are meant to be the subset of the set S_{self} resulting of the censoring conducted by the Thymus.

$$T \in S_{self} \mid \forall T_i \in T, T_i = S_i = \{v_1, v_2, \dots, v_n\} \wedge S_i \in S_{self} \quad (8)$$

4.3 BINDING FUNCTION

As previously stated, the definition of the set T of T cells depends on the censoring process. Nevertheless, the censoring itself is defined around the concept of a T-cell Receptor (TCR), which binds to other cells, triggering a reaction that, in the AIS solution proposed in this work, indicates that the incoming cell (sample) is a part of the already-known model of self behavior.

The binding relationship between a given T-Cell T_i and a given sample S_j is a function of the difference between them, calculated as the Euclidean distance of the two N-dimensional data points, as described in (9).

$$dist(T_i, S_j) = \sqrt{\sum_{k=1}^N (T_i.v_k - S_j.v_k)^2} \quad (9)$$

Once this distance between two samples is calculated, the decision of whether a T cell T_i binds with S_j consists of verifying whether said distance does not exceed a predefined *threshold* (10).

$$binds(T_i, S_j, threshold) = dist(T_i, S_j) \leq threshold \quad (10)$$

As one can notice, this definition for the binding function makes the decision of whether a given sample S_j is recognized by a T cell T_i to be an instance of the problem of deciding the cluster membership between a data point and a cluster centroid, where the radius for the cluster is defined by the binding threshold. The strategy for building the model that will represent the concept of self behavior, however, differs from typical clustering solutions.

4.4 CENSORING

Once we have established the rules that define the binding relationship between samples (cells), we can now move on to the specification of the censoring process that will filter the incoming T cells and generate the final version of our set T that models the notion of self in the system.

Unlike the Human Immune System, our T cells are not generated in a pseudo-random process, and they are indeed self-reactive, which means they do bind with self-cells. As in the Human Immune System, our censoring is responsible for filtering the cells that will compose the set of T cells T . The censoring here is modeled as a function of the Thymus, which in our case, is the component responsible for holding the necessary information and methods to build the set of T cells T and decide whether a sample S_j represents self behavior or not.

As the Thymus is responsible for conducting the censoring of the incoming T cells, the first step to build the set T with the T cells that will define the notion of self and model the behavior of the system is to initialize the Thymus structure with the binding threshold and the minimum number of bindings *min_members* to keep a T cell T (Algorithm 1). The first parameter determines the maximum distance in which an incoming cell should be recognized by existing T cells, i.e., the incoming cell (sample) binds to known members of the T set and thus must be censored. The latter parameter expresses a filtering condition to be applied at the end of the maturation process for the T cells. T cells that have not recognized at least *min_members* incoming cells during censoring are considered to model an abnormal behavior and thus must be censored to avoid compromising the AIS ability to detect intruders.

Algorithm 1 Thymus Initialization

```

1: procedure Initialize (threshold, min_members)
2:    $T \leftarrow \emptyset$ 
3:   min_members  $\leftarrow$  min_members
4:   threshold  $\leftarrow$  threshold
5:   training_threshold  $\leftarrow$  0

```

Once the Thymus structure is initialized, the censoring can be initialized. During

the censoring phase, the S_i samples collected on sane execution moments are individually submitted to the Thymus' censoring method (Algorithm 2). Besides our T cells model self notion, the censoring process still eliminates T cells that bind to self during maturation. In our case, however, we only eliminate T cells that bind to the already known notion of self.

In this way, the `censoring` consists of identifying if S_i binds with any of the T_i existing T cells. If S_i binds with a T cell T_i , the T_i with the smallest distance to S_i increments its number of members $T[i].members$. If S_i does not bind with any of the existing T cells, it is inserted into the set of T cells, and its number of members is set to 0, configuring an iterative clustering process. Finally, as monitored variables can present different proportions, the censoring is also expected to submit the sample S_i to some scaling or standardization technique. Typical examples of these methods are Min-Max Normalization and Z-score or Standard Scaler (AMORIM et al., 2023).

Algorithm 2 Thymus Censoring

```

1: procedure Censoring ( $S_i$ )
2:    $S_i \leftarrow scale(S_i)$ 
3:    $min\_dist \leftarrow \infty$ 
4:    $index \leftarrow -1$ 
5:   for each  $T_i \in T$  do
6:     if  $dist(T_i, S_i) < min\_dist$  then
7:        $min\_dist \leftarrow dist(T_i, S_i)$ 
8:        $index \leftarrow T.index(T_i)$ 
9:   if not ( $binds(T_{index}, S_i, threshold)$ ) then
10:     $T.append(S_i)$ 
11:     $T[T.index(S_i)].members \leftarrow 0$ 
12:    if  $min\_dist > training\_threshold$  then
13:       $training\_threshold \leftarrow min\_dist$ 
14:   else
15:      $T[index].members \leftarrow T[index].members + 1$ 

```

Particular attention is also to be given to monitored variables whose value is cumulative (e.g., the number of occurrences of a given event, the time in a certain operation mode, the overall energy consumption of a component, and others). When dealing with these cumulative variables, the R_i sample to be submitted to the censoring or the process to discriminate self and nonself is composed of the relative values of the monitored variables, i.e., the difference from the actual sampled value to the previous sampling value, as given by equation (11).

$$R_i = \{S_i.v_j - S_{i-1}.v_j \mid j \in [1, n]\}, \text{ where } n = N \quad (11)$$

After all training samples representing a self behavior of the system have been

submitted to the censoring, the training can be finished through the method *finalize* of the Thymus (Algorithm 3), which will remove the T cells T_i whose member count does not exceed the minimum value established on the initialization from the T set. Moreover, the *threshold* is adjusted to the maximum value observed during censoring for any of the samples in a T cell T_i in the T set.

Algorithm 3 Thymus Finalize

```

1: procedure Finalize
2:    $T\_cp \leftarrow \text{copy}(T)$ 
3:   for each  $T_i \in T$  do
4:     if  $T[T.index(T_i)].members < min\_members$  then
5:        $T\_cp.remove(T_i)$ 
6:    $T \leftarrow \text{copy}(T\_cp)$ 
7:    $threshold \leftarrow training\_threshold$ 

```

4.5 DISCRIMINATION SELF AND NONSELF

Once censoring has been finished by the method *finalize*, any new incoming sample S_i can be classified between self and nonself. For classification purposes, a self will be pointed whenever the incoming sample S_i binds with any of the configured T cells in the T set (12). In a complementary fashion, a nonself is detected whenever the self value of S_i , $self(S_i)$, is false.

$$self(S_i) = \exists T_i \in T \mid binds(T_i, S_i, threshold) \quad (12)$$

4.6 TIME COMPLEXITY ANALYSIS

This section aims to analyze the time complexity of the proposed methods using the big-O notation. First, as Thymus initialization, presented in Algorithm 1, is composed of only constant operations, we assume its time complexity to be $\mathcal{O}(1)$.

The Thymus *finalize* method that is responsible for ending the training, presented in Algorithm 3, consists mainly of running the set of members of each T cell and remove the ones that have less than the minimum required members. Assuming the comparison (line 4 of the code) and the remove operation (line 5 of the code) are $\mathcal{O}(1)$, and the size of the T set of T cells to be equal to $\|T'\|$, with T' as the set of T cells before the method to be executed, we have the time complexity of the main part of the *finalize* method as: $\mathcal{O}(\|T'\|) * \mathcal{O}(1) = \mathcal{O}(\|T'\|)$. Assuming the worst case, which in this case will be to not remove any of the T cells, the resulting set T of T cells will be equal to T' , therefore, both copy functions of the beginning (line 2 of the code) and the end of

the function (lines 8 and 9 of the code) will have time complexity of $\mathcal{O}(\|T'\|)$. Therefore, the final time complexity of the `finalize` method is:

$$\mathcal{O}(\|T'\|) + 2 * (\mathcal{O}(\|T'\|)) = 3 * \mathcal{O}(\|T'\|) \quad (13)$$

Or, simply:

$$\mathcal{O}(\|T'\|) \quad (14)$$

The `Thymus censoring` method is responsible to generate the T set of T cells and is presented in Algorithm 2, consists mainly of verifying the minimum distance between the incoming sample S_i and the existent T cells in the T set and verifying if this distance is lesser than or equal to the binding *threshold*. Considering that the sample scaling (line 2 of the code) will include the transformation from a cumulative value to a relative value for all N monitored variables, the time complexity of this method is $2 * \mathcal{O}(\|N\|)$.

The distance between a T cell and the sample S_i (lines 6 and 7, and implicitly, line 11 of the code) have time complexity $\mathcal{O}(\|N\|)$, according to the equation ((9)). As we assumed the size of the T cells set T before the training finalization is expressed by $\|T'\|$, this operation will be repeated $\|T'\|$ times. Assuming the other operations before the main loop (lines 3 and 4) and the complexity of the lines 12 to 15, and the line 18 of the code are $\mathcal{O}(1)$, the time complexity of this method is defined as:

$$2 * \mathcal{O}(\|T'\|) + \|T'\| * \mathcal{O}(\|N\|) + \mathcal{O}(\|N\|) \quad (15)$$

Which could be simplified to:

$$\mathcal{O}(\|T'\| * \|N\|) \quad (16)$$

Note that, for the censoring algorithm, the execution flow would be reduced by breaking the loop that calculates the distance between the incoming sample S_i and the T cells (lines 5 to 10 of the code) as soon as the sample S_i bonded to the T cell it is being compared. This modification would not cause negative impacts on the detection of nonself samples. However, it would impact negatively on the false positive rate.

For the classification of an incoming sample S_i , described at equation (12), if we assume the worst case, in which it will be necessary to compare S_i to the entire set T of T cells, and the training finalization (method `finalize` of the `Thymus`) will not remove any of the T cells generated at the `censoring`, we would have the same time complexity as described at equation (16), since we will have $\|T'\|$ distance verifications, depicted by equation (9), plus a comparison with the binding *threshold*.

This means all the operations of our AIS have a time complexity that is, at the worst-case, linear to the number of T cells used for detection. It is important to notice that for offline execution, where the whole data-set is processed instead of a single sample, the time complexity will be, in the worst case, equal to the size of the data-set

$\|D\|$ times the original complexity since this will be the number of activations of the algorithms, i.e., $\mathcal{O}(\|T'\| * \|N\| * \|D\|)$. This also represents the overall time complexity of the algorithm when considering the actuation during all the system's execution time, instead of verifying it for every activation.

Finally, the operations of `censoring` and classification can be executed in isolation from the other software components composing the CPS. Furthermore, these methods are executed at different moments of the system's lifetime (i.e., training and normal execution). In other words, the `censoring` and the classification will not be executed for the same sample during the system execution. Sampling rate is defined according to the temporal properties of the monitored CPS. Nevertheless, the execution of these algorithms will be proportional to this sampling rate.

4.7 TRAINING AND OPERATION

The present work aims at monitoring and modeling the behavior of a CPS based on the messages exchanged between CPS components. Therefore, when considering training and operation, the user must guarantee that the AIS module gets access to every message exchanged by the CPS components. Once all messages are visible by the AIS module, for the training, we assume all samples S_i submitted to the censoring process were collected during sane execution moments (i.e., $S_i.moment = sane \forall S_i \in S$). This means the training of the model is expected to run in a controlled environment.

In this way, in case the training was not finalized by calling the `Thymus'` method `finalize`, the AIS actuates whenever a message is observed on the CPS in order to submit collected samples S_i to the censoring process via the censoring method of the `Thymus`.

On the other hand, in case the training was finalized by calling the method `finalize` of the `Thymus`, the AIS actuates when new messages are observed by submitting every collected sample S_i to be discriminated between self and nonself via the method expressed by the equation (12).

4.8 SUPPORTING TEMPORAL BEHAVIOR FEATURES

The AIS mechanism described early in this Chapter models the T cells (cluster centroids) that define behavior accepted as self around the variables monitored in a CPS and detects nonself behavior via the deviation of the values of these monitored variables observed in new samples. Nevertheless, this mechanism was initially conceived assuming data to be acquired via periodic sampling on a single source. Therefore, samples are viewed as being a periodic and relatively uniform representation of the behavior of the monitored system. Such an assumption resonates with the AIS utilization in previous works (HORSTMANN, Leonardo Passig, 2020; HORSTMANN, Leonardo

Passig; FROHLICH, Antonio Augusto, 2022; HORSTMANN, Leonardo Passig et al., 2024), where the set of variables V used to describe the system behavior consisted of HPC collected periodically by an operating system running the monitored application.

This work proposes an extension of the original approach by taking advantage of the SmartData model of the system to build the concepts of self and nonself around the temporal behavior of the CPS software components. This shift necessitates incorporating temporal features that account for the inherent asynchronous, event-driven, or multi-rate periodic nature of message exchanges, which the original design did not consider.

SmartData (Section 2.1.1) construct, and the Interest relations provide essential information for this process. Every message to be exchanged is specified by an Interest and has a SmartData source and a SmartData destination. Moreover, Interest relationships are also responsible for specifying the operation mode (event-driven or periodic). All this information is a key aspect for building new types of variables that quantify the system's temporal behavior. These new temporal variables are crucial for building a notion of self that captures not only the values of a set V of monitored variables, but characteristics of the CPS temporal signature.

4.8.1 Deriving Temporal Features

We introduce a set of temporal features, V_T , derived from the exchanged messages and the associated SmartData metadata. These features are calculated based on the timestamps t of consecutive messages and can be focused on specific entities, such as a particular CPS component (source C_s or destination C_d) or a particular event/message type M .

First, let us say that for every message m observed, a new Sample S_i is registered. Then, let S_i be the current sample and S_{i-1} be the sample representing the immediately preceding message of the same specific type, source, or destination being analyzed. Finally, let $S_i.t$ denote the time the sample S_i was captured.

4.8.1.1 Inter-Sample and Inter-Event Intervals

One of the most basic temporal features is the interval between two successive occurrences of a specific event or message type. From this, one can extract variables reflecting the interval between messages and variations that can potentially indicate a deviant behavior or system instability, possibly caused by an attack or intrusion.

To estimate the interval between messages (inter-arrival time), one must observe the time elapsed since the previous sample S_{i-1} of the same type M was collected, i.e., the time between messages with the same type. This helps to model the expected period or frequency of message generation as

$$\tau(M) = S_j.t - S_{j-1}.t \quad (17)$$

where $\tau(M)$ is the inter-arrival time for the type M .

Alternatively, one may want to capture variations of $\tau(M)$. One way to do so is to calculate the average interval for an event or message of type M to be observed within a given time window. In this way, the average interval $\bar{\tau}$ over a moving time window of size W is calculated, focusing on a specific message type M :

$$\bar{\tau}_{M,W}(t) = \frac{1}{N_{M,W}} \sum_{k=1}^{N_{M,W}} (t_k - t_{k-1}) \quad (18)$$

where $N_{M,W}$ is the count of messages of type M observed in the window W ending at time t , and t_k are their timestamps.

It is important to notice that the size W of the time window for calculating such variation is dependent on the CPS application being monitored. Thus, the definition of an optimal value for W is considered to be out of the scope for this work. For instance, one can consider a hyper-period of the system, i.e., the necessary time for all components to complete a cycle of operation. Alternatively, one can tie the value of W to the temporal characteristics of the messages with type M .

4.8.1.2 Latency and Jitter

Latency and Jitter features can be extracted by considering the interval between messages. More specifically, latency represents the time interval between a message and its respective response. If considering the message types M and M' as a pair of types representing a message and its respective response, one can calculate the latency $\tau(M, M')$ between M and M' as

$$\tau(M, M') = S_j.t - S_i.t \quad (19)$$

where $S_j.type = M'$ and $S_i.type = M$.

To calculate jitter, one must calculate the variation among consecutive latency values. Considering $\tau(M, M')[i]$ as the latency for the pair (S_i, S_j) of message and response, jitter can be calculated as

$$\Delta\tau(M, M') = \tau(M, M')[i] - \tau(M, M')[i-1] \quad (20)$$

where $\tau(M, M')[i-1]$ is the last pair of messages with same types M and M' before S_i and S_j .

4.8.1.3 Event Frequency and Count

Another example of features that may be useful for tracing the temporal behavior of CPS components is to observe the intensity of activity for a specific component or the number of occurrences of a given event over a period. One of the ways to do so is to calculate the frequency of an event or message with the same type M . The frequency f for an event or message with type M on the time window with size W is given by the inverse of equation (18).

$$f_{M,W}(t) = \frac{1}{\bar{\tau}_{M,W}(t)} \quad (21)$$

On another perspective, one may consider the number of messages of type M or events observed in a time window of length W . This represents a direct measure of the count of a specific message type M as in equation (22):

$$\eta_{M,W}(t) = \text{Count of messages of type } M \text{ in } [t - W, t] \quad (22)$$

or messages from a specific component C_s as in equation (23):

$$\eta_{C_s,W}(t) = \text{Count of messages from source } C_s \text{ in } [t - W, t] \quad (23)$$

or messages to a specific component C_d as in equation (24):

$$\eta_{C_d,W}(t) = \text{Count of messages from source } C_d \text{ in } [t - W, t] \quad (24)$$

within a time window W . As one can notice, in all cases, this measure is represented by a counter variable that is reset or calculated over the window W .

4.8.1.4 Component Interactions

Previous sections discussed ways to model the behavior of individual components over time. In another perspective, one can also be interested in measuring the interactions of components over time. One way to model the relationship between components in a given period is to measure the rate of messages exchanged between specific components. For instance, the frequency of messages sent from a source C_s to a destination C_d within a time window W is given by equation (25).

$$f_{C_s \rightarrow C_d,W}(t) = \frac{\eta_{C_s \rightarrow C_d,W}(t)}{W} \quad (25)$$

where $\eta_{C_s \rightarrow C_d,W}(t)$ is the count of messages sent from C_s to C_d in window W .

4.8.2 Incorporating Temporal Features into Samples

Once incorporating temporal features, the final sample S_i to be used for Thymus censoring (Algorithm 2) and self/nonself discrimination, as in equation (12), is now

composed of the values of the original monitored features F and the newly introduced temporal features V_T .

$$S_i = \{F \cup V_T\},$$

$$\text{or simply } S_i = \{V\}, \text{ where } V = F \cup V_T$$
(26)

where V_T is the set of selected temporal features.

By including V_T , the T cells can capture and discriminate not only amongst the features extracted from messages, but also considering the time, frequency, and period in which such messages are exchanged, providing a more robust model that accounts also for the system temporal dynamics. In other words, anomalies are detected not only by deviations in data values, but also by deviations in the timing and frequency of communications.

4.9 INTEGRATING PHYSICAL PLAUSIBILITY BASED ON CPS SMARTDATA MODEL

The correct operation of a CPS is not solely dependent on its temporal behavior or the cybersecurity of its components, but also related to the integrity of the cybernetic processes that model the physical processes related to that CPS. This can often be measured through values used for actuation or collected from the sensors used to monitor and control components.

Attacks and intrusions can affect those cybernetic processes or the data they use in varying degrees. In this sense, the present work proposes integrating an evaluation of the plausibility of the data used to model the physical processes related to a CPS as part of the IDS mechanism. In this section, we discuss approaches for assessing physical plausibility by exploiting data-driven CPS design paradigm, with special focus on the SmartData model of CPS components, and propose mechanisms to integrate the physical plausibility measure into the AIS-based IDS classification process.

CPS are dynamical systems, which means their state, represented by the set of variables that encapsulate the necessary information to describe the system at any given moment, evolves over time. Such a state evolution is often described through mathematical models extracted from theoretical knowledge of the system behavior or through methods to identify relationships between system components.

Dynamical systems can present different characteristics. Some dynamical systems can have state variables that change continuously over time (continuous systems) or at specific intervals (discrete systems). The parameters of a dynamical system may change over time (time-variant) or be constant (time-invariant), and their overall state evolution can be described by linear or non-linear equations. Of special interest here are the dynamical system's invariant properties.

Invariant properties are the characteristics of a dynamical system that remain constant over time. Such properties are often related to physical, mechanical, and engineering characteristics of the dynamical system and can be expressed mathematically as functions defined over the state of the system that remain constant throughout its operation.

The functions defining invariant properties vary significantly in complexity. Simple functions may consist of physical bounds like minimum or maximum values for a variable:

$$v_{\min} \leq v(t) \leq v_{\max}$$

For instance, a pressure sensor reading cannot be negative, and an absolute temperature T must satisfy $T \geq 0$ Kelvin.

Other functions can consider engineering, mechanical, and physical properties of the CPS to define the ranges in which a variable can change during a given interval of operation. For instance, the maximum physical acceleration or the maximum response capability of the CPS, derived from the system's intrinsic limitations, engineering and mechanical properties:

$$\left| \frac{dv(t)}{dt} \right| \leq \left(\frac{dv}{dt} \right)_{\max}$$

Such a verification ensures that observed variables do not exhibit large drifts or deviations, a typical signal of abnormal behavior.

Alternatively, those properties can consist of intricate functions that define relationships between components and are defined over multiple variables composing the system state. A typical and straightforward example of multi-variable verifications according to the physics would be a water tank CPS with a sensor for water level h , an inflow rate control F_{in} , and an outflow rate control F_{out} . One can easily provide the physical model $P(h, F_{in}, F_{out})$ for the Δh variation of h according to F_{in} , F_{out} , and the specific properties of such a water tank.

A more complex plausibility function may consider equations that determine the dynamics of a vehicle, how it moves, and responds to forces. Such equations may consider properties of the vehicle body, tires, friction, suspension, and individual models for different actuators, like the steering, throttling, and braking. The verification of such properties can consider whether actuation is perceived through sensing values, whether the relationships between variables respect physical laws, or whether there is compliance with the system model.

The ability to identify such properties and model the functions that express them is a key factor for exploring the plausibility of the cybernetic processes that control the operation of the CPS. In this work, we rely on ML approaches for identifying the intricate

invariant properties of the CPS and model the functions that govern the interactions between its components.

4.9.1 Machine Learning Mechanisms for Modeling CPS Properties

The invariant properties that model the behavior of a CPS can be defined in multiple ways. Alternatives range from theoretical physical knowledge to engineering and mechanical constraints of the CPS to data-intensive approaches based on the observation of such properties on the data produced during CPS operation. In this work, we rely on the data-intensive approaches for identifying and modeling the functions that approximate such properties for the verification of the plausibility of the system state representation.

In this sense, ML techniques leverage the growing computational resources and increasing data availability for identifying the underlying properties and relationships between CPS components. Such data-driven approaches represent a powerful mechanism for extracting insights and constructing models from complex, high-dimensional datasets, without relying on explicit physical knowledge or simplified assumptions.

One of the alternatives to express such properties is to assess compliance with previously identified relationships between dependent variables. One way to do so is to analyze the behavior of all variables involved in the operation of a CPS subsystem through the optics of a regression algorithm, e.g., linear regression. A similar solution was adopted by Cho and Shin (CHO; SHIN, 2016) when calculating the clock skew feature for their IDS mechanism. One could also verify the compliance of sensed values with correlations observed with techniques like the Pearson Correlation Coefficient.

Another typical solution is to use data-mining mechanisms, such as the extraction of association rules that represent patterns of behavior that have associated occurrences. A similar mechanism was adopted by Mitchell and Chen (MITCHELL, R.; CHEN, I.-R., 2015), which used association rules to specify behavior rules in the form of properties or conditions that imply the observation of secondary properties or conditions for the operation of the CPS.

In another approach, NN-based models have also been used to extract such relationships. NN-based techniques such as ANN, CNN, and RNN, AE, and LSTM have shown great promise in modeling nonlinear dynamical systems, capturing intricate input-output relationships. In fact, Hornik (HORNIK, 1991) has stated that simple feed-forward ANNs work as universal function approximators.

AE (GOODFELLOW et al., 2016) are known as a subclass of NNs that solve a regression problem. It consists of an unsupervised learning technique with two main components, an encoder function E and a decoder function D . More specifically, two NNs that produce an output represented as $X' = D(E(x))$. An autoencoder is usually evaluated based on the amount of information loss it implies, which is usually expressed

as a function of its reconstruction error $R_e = |X' - X|$. Alternatively, techniques like CNN, RNN, and LSTMs are able to identify temporal dependencies from data, making them well-suited for a wide range of regression tasks.

In this way, the present work operates under the assumption that ML-based techniques are able to identify the invariant properties and relationships between CPS components and approximate the intricate mathematical functions that govern their operation.

4.9.2 Plausibility Model Generation

For building the respective models that will identify the underlying properties and approximate the functions that govern the operation of the CPS in order to provide a plausibility measure of the operation of the CPS based on the data used to model the state of the system, a dataset will be collected consisting of the data observed in the messages exchanged by CPS components during operation.

Plausibility model generation will consider all variables representing the system state. The selected ML mechanism can consist of NN-based function approximations in the form of predictors for single or multiple features (e.g., AE), regression mechanisms to observe mathematical relationships between variables, association rules extracted from the dataset, or any other mechanism able to detect and model properties and relationships between CPS components.

For ML-based predictors, the choice of features to compose each model will rely on feature selection mechanisms to identify underlying relationships between the data produced and consumed by each CPS component. ML models should be generated only for variables presenting multiple highly correlated features. Similarly, when considering the extraction of association rules, only rules with high support and confidence should be considered. These constraints are applied to reduce false positives introduced in the resulting IDS solution by unreliable ML models. An alternative for such constraints is the utilization of methods like AE and PCA that are capable of reducing data dimensionality and extracting relevant features from data.

For ML mechanisms that depend on training, the training process will consider variations of hyperparameters, such as the number of neurons per hidden layer, activation functions, and learning rate for NN-based techniques.

The best ML models will be used for plausibility evaluation. Their output must be converted into one of two plausibility measures. The first type of plausibility measure consists of a Boolean indicating whether the system operation is considered to be normal or not. The second type of plausibility measure is indicated in the form of a plausibility degree, indicating how much the data evaluated is considered to be plausible.

Finally, plausibility models are associated with the SmartData component that

produces the data they use to generate a plausibility measure. In this way, a plausibility model can be associated with multiple SmartData components. The association of plausibility models with SmartData and with the AIS-based IDS classification process is discussed later in this chapter.

4.9.3 Physical Plausibility and AIS-based IDS Integration

As previously mentioned, the integrity of the cybernetic processes that model the physical part of a CPS, critical for their correct operation, may be assessed through the analysis of the data it uses. Considering this aspect, a data-driven design for CPS components allows for directly mapping physical processes, software components, sensors, and actuators associated with each CPS subsystem. Such a data dependency hierarchy is expressed by SmartData's Interest relationships. In this section, we outline the integration of plausibility functions into the AIS-based IDS classification process.

Let us consider the CPS C to be defined as $C = \{S, I\}$, where S represents the set of SmartData components and subsystems that compose the CPS and I is the set of interests that define their interactions and data dependencies. Then, let us consider $s'.P$ to be the set of ML-based plausibility functions modeled over a SmartData component s' defined as

$$s'.P = \{p_1, p_2, \dots, p_n\} \mid s'.data \in p_i.parameters \forall p_i \in s'.P \quad (27)$$

As aforementioned, our IDS samples are tied to the messages exchanged by the CPS components. Therefore, let us consider the components involved in the sample S_i to be determined according to the SmartData components involved in the communication. Then, for every sample S_i , there is a SmartData $S_i.s'$ acting as the source of the message and a SmartData $S_i.s''$ acting as the message destination. From the definition of the Interest relationships that define the data dependencies between SmartData components, if $S_i.s'$ sends a piece of data to $S_i.s''$, then $S_i.s''$ must have declared an Interest $i \in I$ to $S_i.s'$. Therefore, one must consider that the plausibility of a receiver $S_i.s''$ is also dependent on the plausibility of the information it uses to operate. In this way, the $P(S_i)$ set of properties to be used for the validation of S_i is given by the properties of both SmartData s' and s'' , which are given by:

$$P(S_i) = s'.P \cup s''.P \quad (28)$$

Once we have defined the $P(S_i)$ set of properties to be verified for a sample S_i , the application of the plausibility measure produced by the functions in $P(S_i)$ to the classification process depends on whether the plausibility measure represents a plausibility degree or a Boolean value (true or false for whether S_i is plausible).

For binary plausibility measures, the evaluation of whether a sample S_i represents a self behavior, originally depicted by equation (12) in Section 4.5, would be given by

$$self(S_i) = AIS(S_i) \wedge P(S_i) \quad (29)$$

which is equivalent to

$$self(S_i) = \exists T_i \in T \mid binds(T_i, S_i, threshold) \wedge P(S_i) \quad (30)$$

On the other hand, whenever $P(S_i)$ provides a plausibility degree, one could use the output from $P(S_i)$ to weigh in the classification process by directly applying $P(S_i)$ to the binding function as a factor representing the tolerance to which we accept deviations from the modeled behavior of the CPS, which would imply a classification rule as:

$$self(S_i) = \exists T_i \in T \mid binds(T_i, S_i, threshold * P(S_i)) \quad (31)$$

where $P(S_i) \in [0,1]$.

Finally, the classification process for the AIS-based IDS with temporal features and physical plausibility is summarized by algorithm 4.

Algorithm 4 Classification Function

```

1: procedure Classify ( $S_i$ )
2:    $P \leftarrow P(S_i)$ 
3:    $S_i \leftarrow S_i \cup temporal\_features(S_i)$ 
4:    $Membership\_Score = \min(dist(S_i, T_j) \mid T_j \in T)$ 
5:    $Plausibility\_Score \leftarrow \prod p_i \mid p_i \in P$ 
6:   if  $Plausibility\_Score \in \mathbb{R}$  then
7:      $Self(S_i) = Membership\_Score \leq threshold * Plausibility\_Score$ 
8:   else
9:      $Self(S_i) = (Membership\_Score \leq threshold) \wedge Plausibility\_Score$ 

```

It is important to notice that, while the integration of the plausibility function as a tolerance factor for the AIS-based IDS classification process restricts $P(S_i)$ to return a plausibility degree, it offers a direct mapping from the function $P(S_i)$ into the classification process. Therefore, for the solution expressed in equation (31), each small variation on the output of $P(S_i)$ has the precise same impact on the AIS classification. Meanwhile, for solutions that fall into equation (30), a change in the classification results would only be noticed in case the output of $P(S_i)$ changes truth value, which can be defined as another classification problem or by comparing the membership degree to a given threshold. Notwithstanding, the reduced sensitivity from equation (30) also makes it less susceptible to the impacts of misreadings or outliers. Notwithstanding, the choice

of methods is dependent on available plausibility functions, how accurate they are, and their sensitivity to the incoming data.

SUMMARY OF AIS-BASED IDS WITH TEMPORAL FEATURES AND PHYSICAL PLAUSIBILITY

This chapter presented a detailed exposition of an Artificial Immune System (AIS)-based Intrusion Detection System (IDS) specifically designed for Cyber-Physical Systems (CPS), significantly extending the original approach by incorporating temporal features and physical plausibility.

The core of the proposed IDS is modeled after the Human Immune System, using T cells as detectors to define and recognize self-behavior (correct system operation). The T cells are generated from self samples (S_{self}) through a rigorous censoring process modeled on the Thymus, which iteratively builds the final set of T cells, T , using the Euclidean distance and a binding threshold for cluster membership (Algorithm 2). After training, a sample is classified as self if it binds to any T cell in T , and nonself (an intrusion) otherwise.

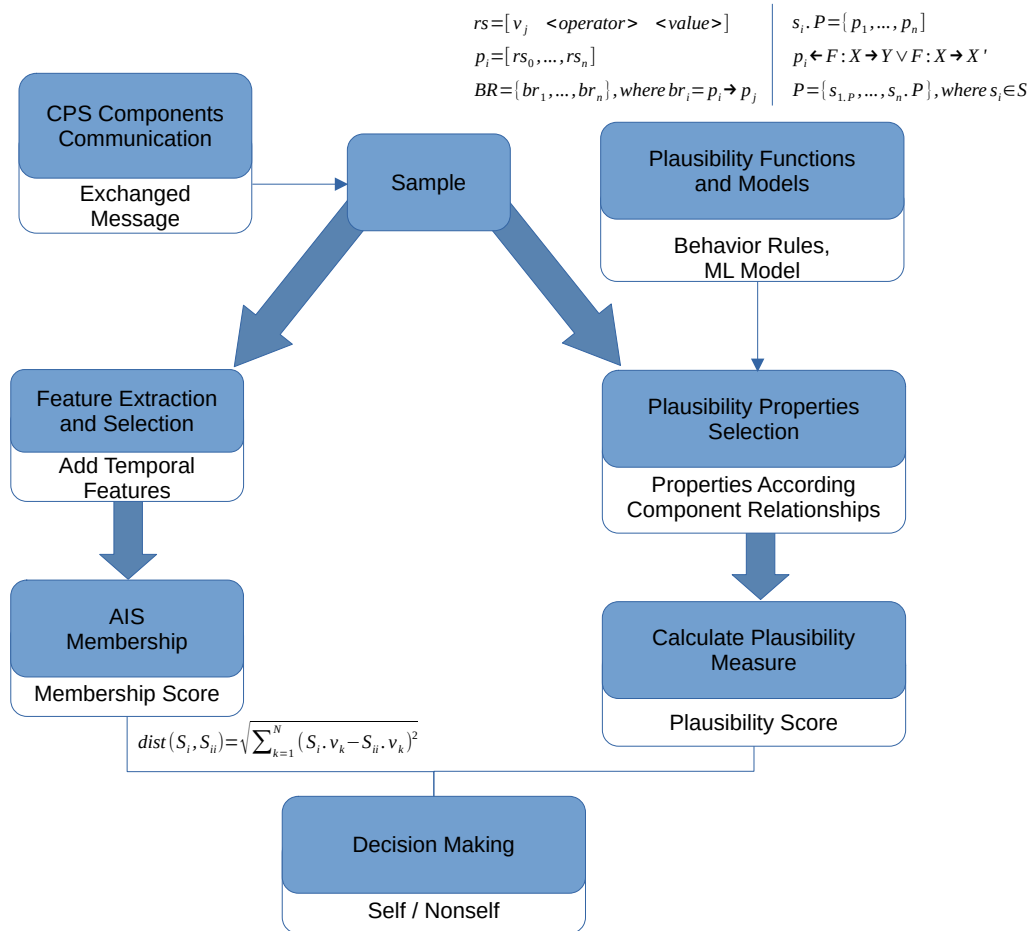


Figure 3 – AIS-based IDS considering Temporal Features and Physical Plausibility

This work's main contribution lies in two key new mechanisms. First, we model the behavior of the target CPS after messages exchanged by its components. In the training process, we introduce temporal features (V_T) extracted from these messages, such as inter-sample intervals, latency, jitter, and event frequency. This enhancement allows the AIS model to capture the temporal signature and dynamic behavior of CPS components, enhancing the robustness of the model of self. Then, we also integrate the verification of physical plausibility by leveraging Machine Learning (ML) techniques to model the CPS's invariant properties and relationships between components. This plausibility measure can be integrated into the classification process either as a Boolean condition ($\text{self}(S_i) = \text{AIS}(S_i) \wedge P(S_i)$) or as a tolerance factor weighing the binding threshold ($\text{self}(S_i) = \exists T_i \in T \mid \text{binds}(T_i, S_i, \text{threshold} \cdot P(S_i))$). The decision-making process is expressed by Figure 3. By merging the temporal and physical domains, the proposed AIS-based IDS provides a more comprehensive and resilient mechanism for detecting sophisticated intrusions in modern CPS environments.

5 PROOF-OF-CONCEPT IMPLEMENTATION AND VALIDATION

This section details the methodology adopted to evaluate the proposed IDS framework in terms of overhead and performance in terms of detection. Section 5.1 presents a detailed description of our dataset. Section 5.2 outlines preprocessing and feature selection steps for the generation of our AIS models.

Section 5.3 focuses on the evaluation of the performance in terms of the ability to detect intrusions, describing the process of building AIS models, the selected plausibility function, and the metrics selected for evaluating our IDS. Finally, Section 5.4 outlines the configuration of the framework, including the selection and configuration of the resource isolation mechanism to ensure correct system behavior, and defines the experimental evaluation scenarios considered for measuring computational performance, memory utilization, and overhead.

5.1 DATASET ANALYSIS

This section outlines the dataset in-depth analysis. Chapter 3 presented the dataset selected for this case study, the Water Distribution Testbed WDT proposed by Faramondi et al. (FARAMONDI et al., 2021). The hardware-in-the-loop simulation produced two labeled datasets representing the evolution of the state of the physical process as well as network communication. In this chapter, we will go over each monitored variable in both the physical and network datasets, as well as revisit all network attack scenarios, how they are modeled, and what their impacts are on cyber and physical data.

Their WDT is composed of a real and a simulated subsystem. The real subsystem consists of 5 tanks ($T_1 \dots T_5$) of different dimensions. Tanks T_1 and T_2 have a 45 cm height and 90 cm circumference, T_3 and T_4 have a 36 cm height and 70 cm circumference, and T_5 has a 40 cm height and 100 cm circumference. Each tank is monitored by a pressure sensor ($S_1 \dots S_5$) installed under the tank. The real-world subsystem also holds 8 manual valves for simulating water leaks from tanks or pipes. 20 solenoid valves ($V_1 \dots V_{20}$), and 4 pumps ($P_1 \dots P_4$).

Each tank has multiple outlet valves for modulating the output flow. T_1 is equipped with outlet valves V_1 , V_2 , V_3 and V_4 ; T_2 with V_5 , V_6 , V_7 and V_8 ; T_3 with V_9 , V_{10} , V_{11} and V_{12} ; T_4 with V_{13} , V_{14} and V_{15} and T_5 with V_{19} and V_{20} . Tanks are connected by pipes, and pumps are used to transport water upwards. Pumps P_1 and P_3 transport water from the reservoir to tank T_2 , pump P_2 from the reservoir to tank T_1 , and pump P_4 from tank T_1 to tank T_5 . Valves are also used to control the water distribution between tanks. V_9 controls the water flow from T_1 to T_2 , V_{16} controls the flow from T_4 to T_3 , V_{17} controls the flow from T_2 to T_3 , and V_{18} controls the flow from T_1 to T_4 .

The simulated subsystem consists of 3 tanks ($T_6 \dots T_8$) with the same dimen-

sions as T_5 . Each tank is monitored by a pressure sensor ($S_6 \dots S_8$). The subsystem also contains 2 pumps (P_5 and P_6), 4 flow sensors ($F_1 \dots F_4$), and 2 solenoid valves (V_{21} and V_{22}). The connection between components is given by the following: T_6 is connected to the outlet valve V_{20} . P_5 takes the water from T_6 to T_7 . The connection between the two is monitored by F_1 . P_6 takes the water from T_7 to T_8 . This connection is monitored by F_2 . V_{21} works as outlet valve for T_7 . The flow is monitored by F_3 . Finally, V_{22} works as an outlet valve for T_8 , and this flow is monitored by F_4 . Table 2 summarizes the direct relationships for each tank composing the WDT.

Table 2 – Tank Direct Connections

Tank	Components with Direct Connection
T_1	$P_2, P_4, V_1, V_2, V_3, V_4, V_9, V_{18}$
T_2	$P_1, P_3, T_1, V_5, V_6, V_7, V_8, V_9, V_{17}$
T_3	$T_2, T_4, V_{10}, V_{11}, V_{12}, V_{16}, V_{17}$
T_4	$T_1, V_{13}, V_{14}, V_{15}, V_{16}, V_{18}$
T_5	T_1, P_4, V_{19}, V_{20}
T_6	T_5, V_{20}, P_5, F_1
T_7	$T_6, P_5, F_1, V_{21}, F_3, P_6, F_2$
T_8	$T_7, P_6, F_2, V_{22}, F_4$

The operation of the WDT is modeled in 4 stages ($S1 \dots S4$). In $S1$, water is pumped from the reservoir towards two different paths. In the first path, the water is pumped by P_1 towards T_2 , which starts to fill up T_3 through V_1 . When the water level reaches a specific threshold, V_{10} , V_{11} , and V_{12} are activated in order to get water back to the reservoir. In the second path, the water is pumped by P_2 towards T_1 . Then, P_4 is activated to fill up T_5 . When the water level reaches a specific threshold, P_4 is deactivated, and the remaining water in T_1 is drained towards T_4 by opening V_{18} and then through valves V_{13} , V_{14} , and V_{15} towards the reservoir.

$S2$ starts when the water level in T_5 reaches a predefined threshold. When started, V_{20} is opened and starts to fill up T_6 , for which the water level increases as much as the level in T_5 decreases (i.e., V_{20} is virtually deviated towards T_6). This starts $S3$, which enables P_5 , filling up T_7 while F_1 records water flow downstream of the pump. Finally, $S4$ controls water flowing from T_7 towards T_8 through P_6 . The flow is measured by F_2 . When the water level in T_8 reaches a specific threshold, V_{22} is opened to virtually drain the water towards the reservoir. Figure 4 presents the WDT as described in (FARAMONDI et al., 2021), with all components and .

The physical dataset is composed of the data monitored by the flow sensors $F_1 \dots F_4$, the pressure sensors $S_1 \dots S_8$, the control value of the pumps $P_1 \dots P_6$, and the control values for the valves $V_1 \dots V_{22}$. Samples are collected at the frequency of 1 Hz, and the collected data is timestamped.

To understand the network dataset, we first need to recap the network model

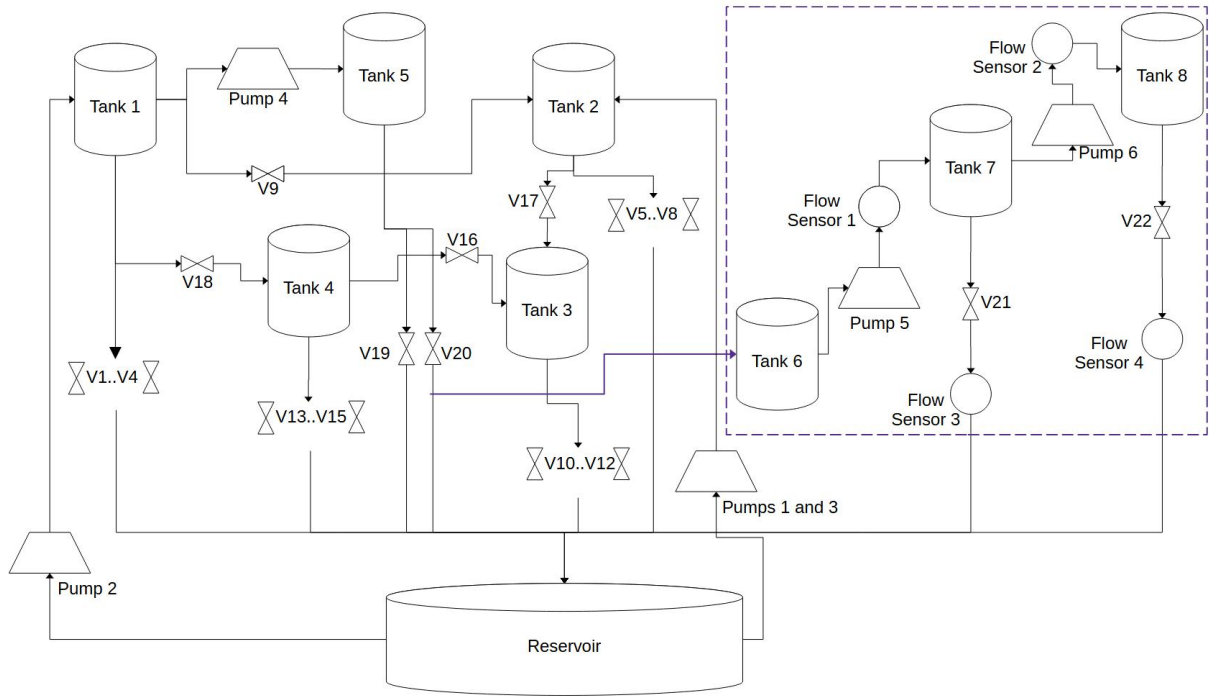


Figure 4 – Water Distribution Testbed Schematic adaptation from (FARAMONDI et al., 2021). In the Figure, the right blue dotted rectangle represents the simulated subsystem. Blue arrows represent virtual water flowing between the two subsystems.

adopted by Faramondi et al. (FARAMONDI et al., 2021). Their network architecture consists of a three-layer SCADA architecture using the MODBUS TCP/IP communication protocol. The first layer consists of sensors and actuators connected to the control subsystems for each stage via wired links. All of them are connected to the I/O analog or digital module of the respective control subsystem, with the exception of flow sensors F_1 and F_2 , which are modeled as MODBUS TCP/IP sensors with their own IP addresses. The second layer consists of the control subsystems. The third layer consists of the supervisory system, which collects and stores data from the communication and control subsystems and is equipped with an HMI.

To compose the network dataset, the traffic was monitored during operation (normal and under attacks) using Wireshark. The following features were observed:

- Time: The date and time of acquisition. When the packet is observed;
- Src IP Address (ip_s): The IP address of the source of the packet;
- Dst IP Address (ip_d): The IP address of the destination of the packet;
- Src Port: The source port;
- Dst Port: The destination port;

- Src MAC Address (mac_s): The source MAC address;
- Dst MAC Address (mac_d): The destination MAC address;
- Proto: The protocol used for communication;
- Flags: TCP flags - CWR | ECN | URG | ACK | PSH | RST | SYN | FIN;
- Size: Size of the packet payload;
- MODBUS Function: MODBUS function code;
- MODBUS Response: MODBUS response value;
- N^o Pkt Src: Number of packets from the same source in the last 2 seconds;
- N^o Pkt Dst: Number of packets with the same destination in the last 2 seconds.

An assumption regarding the dataset is that IP and MAC addresses are considered to be statically assigned to the network entities. Thus, ARP (Address Resolution Protocol) messages, used to identify the MAC address for a given IP, are not expected. Other than that, the features collected assume source and destination ports are static and well defined for the applications composing a CPS, and protocols are limited and well-defined. It is important to notice, the network dataset already incorporates temporal features (the numbers of packets with the same source and packets with the same destination in the last two seconds).

Once knowing the characteristics of the sampling for the network and physical datasets, we can now dive into the attack scenarios modeled by Faramondi et al. (FARAMONDI et al., 2021). Three main classes of cyber attacks were modeled in their work, MITM, DoS, and the Scan attack.

The MITM attack, modeled as ARP poisoning, consists of an attack against the MODBUS communication protocol intended to modify packets' data payload by fixing the returned value on the last value observed before the fault. The DoS attack is performed with three different mechanisms, namely, TCP flood, ICMP flood, and Land attack. The DoS targets single hosts to saturate their resources and disconnect them from the network. Finally, the Scanning attack, which is executed as four variants, namely, SYN scan, FIN scan, Null scan, and XMAS scan, targets hosts with the intention of gathering network information about those devices.

The attacks are distributed alongside three rounds of data acquisition, each one of composing one dataset. Table 3 introduces the cyber attacks on the first data acquisition cycle. In this dataset, only MITM attacks were performed, and all of them affected the observable behavior of the physical components for one of the sensors.

Table 4 presents the cyber attacks on the second data acquisition experiment, and Table 5 presents the cyber attack scenarios executed on the third data acquisition experiment.

Table 3 – First acquisition

# scenario	Time Frame		Type of attack	Effects on physical process
	Start	End		
1.1	09/04/2021 18:25:48	09/04/2021 18:28:14	MITM against S2 and S3 on S ₆	✓
1.3	09/04/2021 18:34:11	09/04/2021 18:35:38	MITM against S1 and S2 on S ₅	✓
1.5	09/04/2021 18:43:52	09/04/2021 18:45:54	MITM against S3 and S4 on S ₇	✓
1.7	09/04/2021 18:58:05	09/04/2021 18:59:32	MITM against F ₁ and S3 on F ₁	✓
1.8	09/04/2021 19:00:40	19/04/2021 19:02:07	MITM against F ₂ and S3 on F ₂	✓

It is important to notice that, while some of the cyber attacks present a direct effect on the CPS physical operation, this is not a true assumption for all executed attacks. In fact, this represents a common factor for IDS solutions that map both the network and the physical scenarios. As previously introduced in Chapter 3, the solutions that consider data from the physical domain consider them as a unique factor or separated from network traffic data. Nevertheless, the main reason for the selection of this dataset is the fact that it presents a realistic scenario with cyber attacks that have an impact on the cybernetic processes that model and control the physical components of the CPS, with proper identification, and the availability of data from both scenarios, allowing for the construction of a proof-of-concept for the IDS solution proposed here.

5.2 DATASET PREPROCESSING AND FEATURE SELECTION

The present work adopts two steps for data preprocessing. First, we extracted temporal features from the dataset following the definitions of Section 4.8. As one can notice, temporal features representing the frequency of events, on the network dataset, namely, the number of packets with the same source and the number of packets with the same destination in the last 2 seconds. Therefore, we calculated features for the interval between samples with the same type, considering 5 parameters, namely, protocol, source port, destination port, source MAC address, and destination MAC address. Moreover, the latency between a MODBUS request and response was also calculated and integrated into the dataset.

After introducing new samples, categorical values were transformed into discrete numbers by assigning a sequential number to each different value to each different value. That is the case for variables like flags, MODBUS function, protocol, and IP and

Table 4 – Second acquisition

# scenario	Time Frame		Type of attack	Effects on physical process
	Start	End		
2.1	19/04/2021 15:38:52	19/04/2021 15:38:52	SYN scan against S1	-
2.2	19/04/2021 15:40:09	19/04/2021 15:40:09	FIN scan against S2	-
2.3	19/04/2021 15:41:10	19/04/2021 15:41:10	XMAS scan against S3	-
2.4	19/04/2021 15:42:03	19/04/2021 15:42:03	Null scan against S4	-
2.5	19/04/2021 15:43:46	19/04/2021 15:44:22	ICMP flood against S1	✓
2.7	19/04/2021 15:51:16	19/04/2021 15:51:16	Null scan against S3	-
2.9	19/04/2021 15:58:40	19/04/2021 15:58:40	SYN scan against S1	-
2.10	19/04/2021 15:59:57	19/04/2021 16:00:10	ICMP flood against S1	-
2.11	19/04/2021 16:04:39	19/04/2021 16:04:39	FIN scan against S2	-
2.12	19/04/2021 16:08:26	19/04/2021 16:10:01	MITM against S3 and S4 on S ₇	✓
2.13	19/04/2021 16:11:46	19/04/2021 16:12:14	TCP flood against S1	✓

MAC addresses. It is important to notice that, as the proposed solution is based on clustering and calculates cluster memberships based on the distance to a centroid, all variables composing the training and inference process must present numerical values. After all numerical features are available, we standardized the data using the z-score normalization as in equation (32) in order to make the data normally distributed.

$$z = \frac{x - \mu}{\sigma} \quad (32)$$

$$\text{where } \mu = \frac{\sum_{i=1}^n x_i}{n}, \sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}}$$

The resulting dataset was submitted to the feature selection stage, which aims to reduce all the variables sampled during the data acquisition to a set with the N

Table 5 – Third acquisition

# scenario	Time Frame		Type of attack	Effects on physical process
	Start	End		
3.2	09/04/2021 19:45:52	09/04/2021 19:46:21	ICMP flood against HMI	-
3.5	09/04/2021 19:55:29	09/04/2021 19:56:15	ICMP flood against S1 with huge payload	✓
3.6	09/04/2021 19:58:02	09/04/2021 19:59:09	MITM against S3 and S4 on S ₇	✓
3.7	09/04/2021 20:01:18	09/04/2021 20:02:03	MITM against S3 and S4 on S ₇	-

most relevant, increasing the overall quality of the produced models and reducing the possibility of overfitting. To do so, we considered two algorithms, Pearson's Correlation Coefficient and Spearman's Rank Correlation Coefficient.

Pearson correlation is a statistical measure that quantifies the intensity and direction of a linear relationship between a pair of continuous variables. It provides a value ranging from -1 to 1, where the more distant the value from 0, the more it is correlated, with 0 meaning no correlation at all. Meanwhile, Spearman's rank measures the intensity of monotonic relationships between two ranked variables. As it represents a non-parametric test, it does not meet the assumptions of linear relationships like Pearson's. The results for the preprocessing and feature selection are presented in the following section.

5.2.1 Preprocessing and Feature Selection Results

This section presents and discusses the results from the preprocessing and feature selection processes. Figure 5 presents the results for the Pearson Correlation Coefficient, considering the problem of classifying the type of intrusion and Figure 6 presents the results for the Pearson Correlation Coefficient, considering the problem of detecting anomalies (self and nonself distinguishing).

Figure 7 presents the results for Spearman's Rank Correlation Coefficient, considering the problem of classifying the type of intrusion. Meanwhile, Figure 8 presents the results for Spearman's Rank Correlation Coefficient, considering the problem of detecting anomalies (self and nonself distinguishing).

In both scenarios, labeled data was used to perform feature ranking. Specifically, label_n consisted of a numerical indicator of the presence of an anomaly (i.e., 0 for normal data and 1 for abnormal behavior), and label_encoded consisted of a numerical value indicating what class of behavior each sample represented (i.e., numerical values

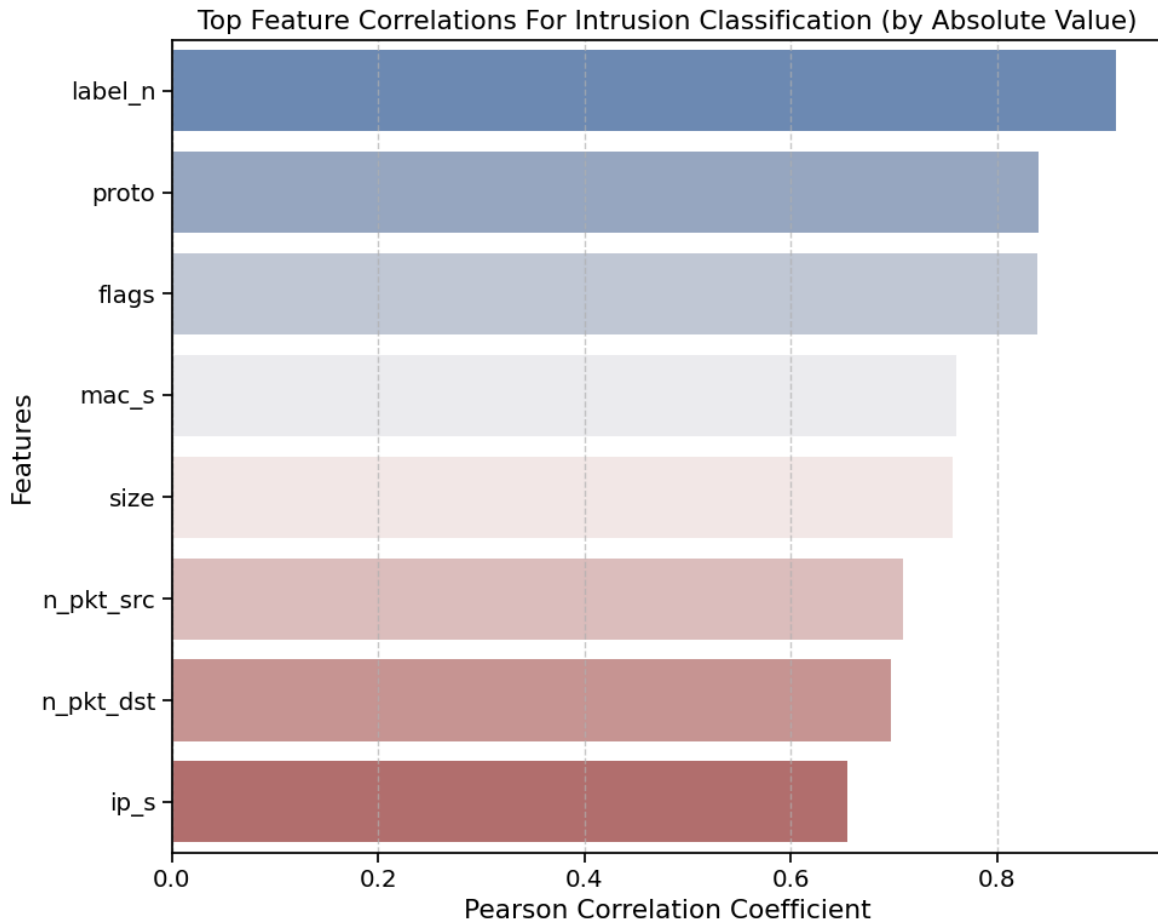


Figure 5 – Correlation Analysis According to Pearson Correlation Coefficient from each Network Feature to the Traffic Class

set for each kind of behavior). Such variables, besides presenting high correlation to the detection problem, were not further considered as the IDS mechanism is supposed to detect anomalies in unlabeled data.

Considering the results for both correlation methods, a total of 6 variables were selected for the AIS model training. Variable selection respected the criteria of selecting variables evaluated with a correlation value of at least 0.5 by both techniques. The selected variables were: flags, MAC address from the source, MODBUS function, Protocol, and two temporal features: number of packets with the same destination in the last two seconds, and number of packets from the same source in the last two seconds.

An interesting factor is that when considering the additional temporal features that were inserted into the dataset, their overall correlation was lower than that observed in the other variables and the two temporal features already existing in the dataset. The best results were obtained using Spearman's Rank Correlation Coefficient. Table 6 presents the obtained correlations. Despite presenting a correlation above 40% in some of the cases, the correlation values for those variables are far below the ones selected to compose the AIS model.

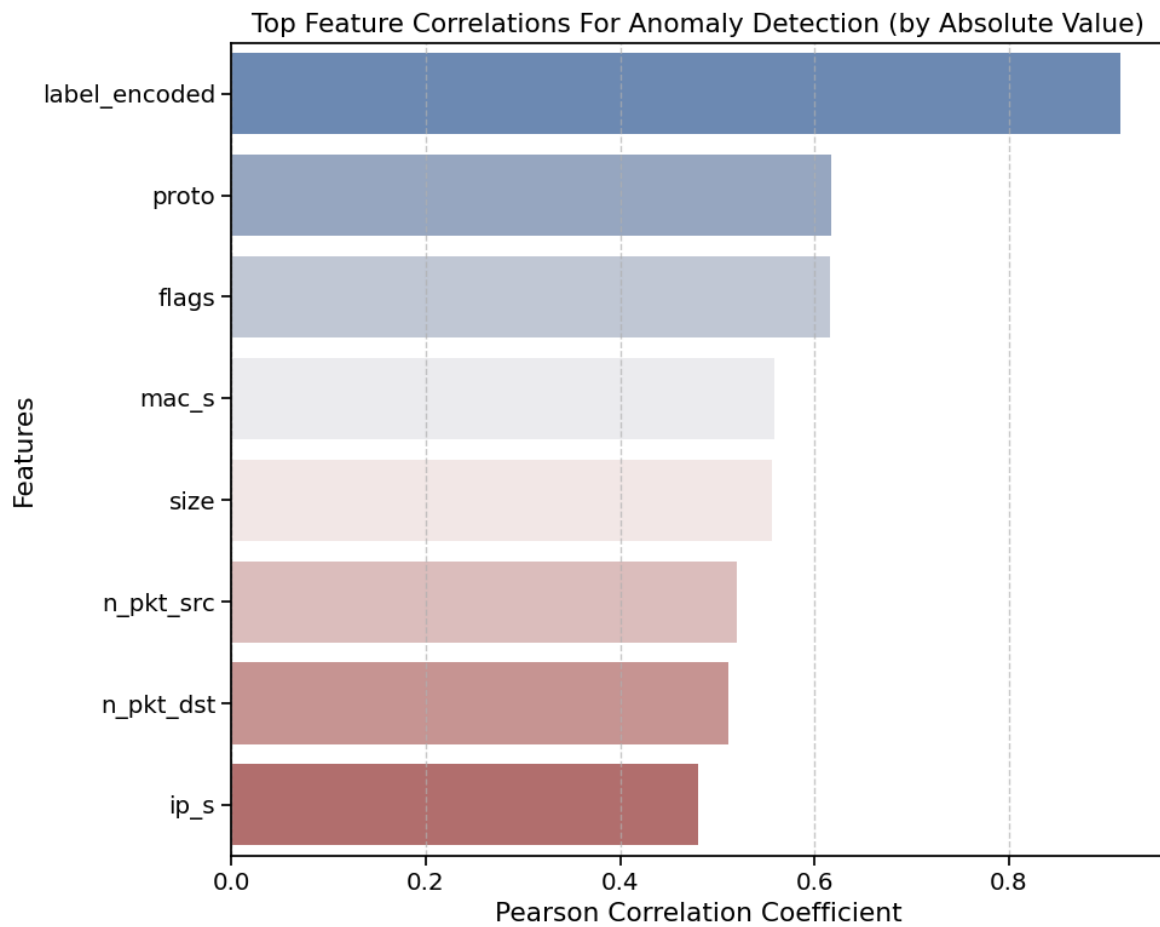


Figure 6 – Correlation Analysis According to Pearson Correlation Coefficient from each Network Feature to Normal vs Abnormal Behavior

Table 6 – Spearman's Rank Correlation Coefficient for Temporal Features

Feature	Correlation
MODBUS latency	30.8%
Interval between d_port	44.28%
Interval between s_port	45.66%
Interval between mac_d	45.74%
Interval between mac_s	44.13%
Interval between proto	32.27%
Number of packets by destination in 2s	51.48%
Number of packets by source in 2s	52.67%

The correlation results presented in Table 6 point to the conclusion that the attacks, in general mode, did not fully disrupt the temporal behavior of the monitored CPS application. In fact, more often than not, they do increase the frequency of message exchanges, as demonstrated by the higher correlation values for the features representing

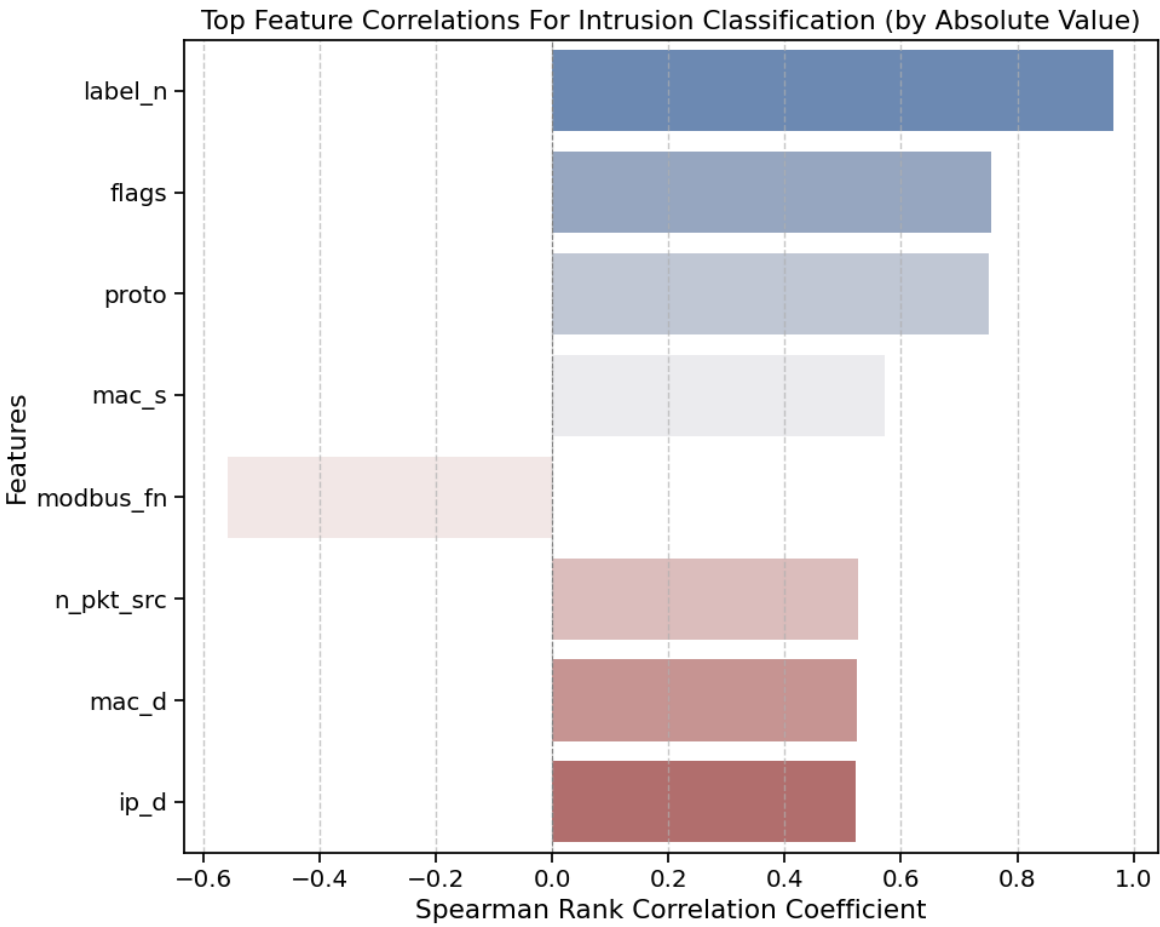


Figure 7 – Correlation Analysis According to Spearman’s Rank Correlation Coefficient from each Network Feature to the Traffic Class

such a behavior.

Another topic of discussion is the possibility of adding all variables for the clustering process. Nevertheless, adding features with lower correlation to a ML model can lead to a drop in their performance, especially when considering the impacts of possible overfitting. A typical motivator for conducting a feature selection process prior to generating models.

A similar discussion arises from the idea of including physical features amongst the features considered for clustering. As depicted by Tables 3, 4, and 5, the impacts of cyber attacks do not always extend to the cybernetic systems that model and control the physical processes. The relationship between variables and a class (i.e., an attack type or an anomaly) in terms of correlation is expressed as a covariance calculation, representing a measure of how much a pair of variables change together. Therefore, the correlations between physical features and network features when considering the problem of detecting intrusion are not expected to be strong enough so that the utilization of both metrics as part of the same model would enhance the overall detection capabilities of the ML models. Moreover, as previously mentioned, the inclusion of more

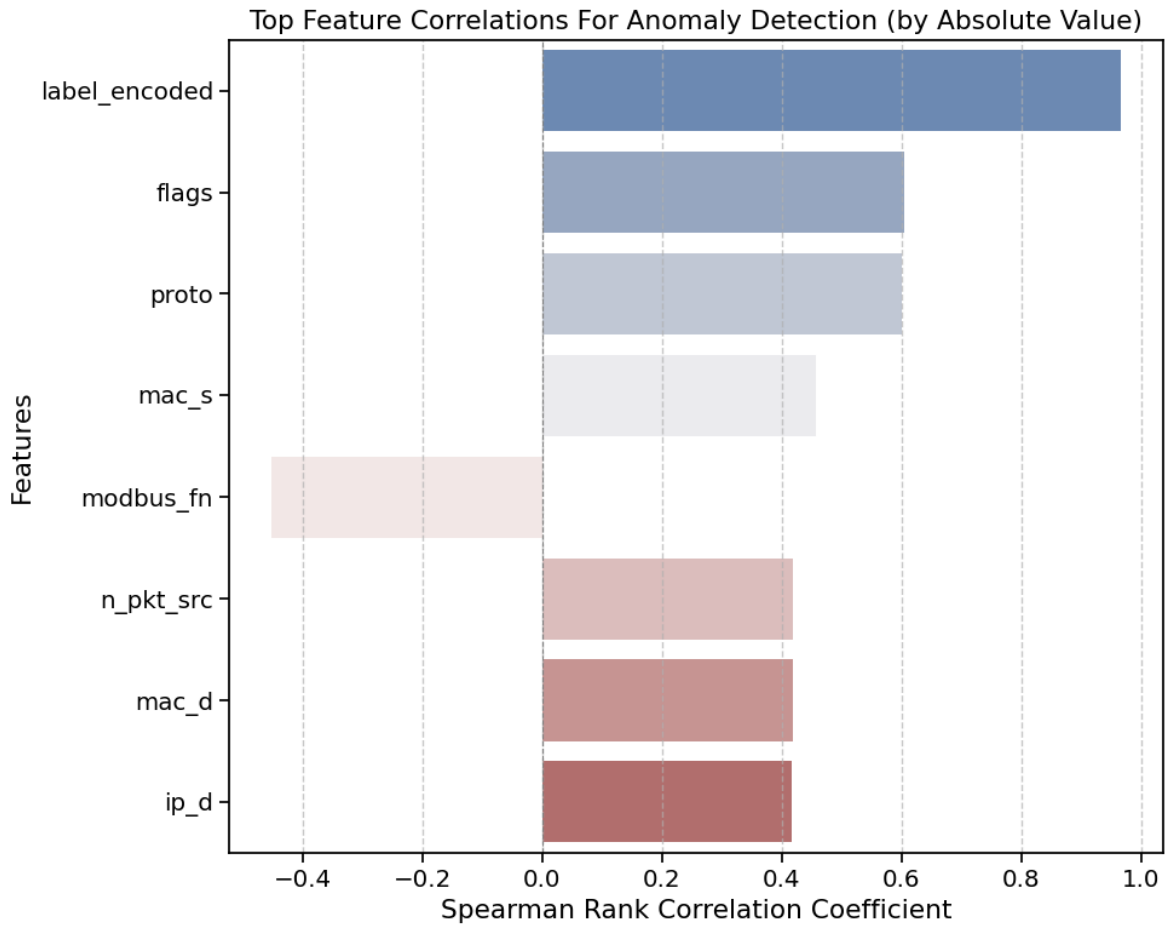


Figure 8 – Correlation Analysis According to Spearman’s Rank Correlation Coefficient from each Network Feature to Normal vs Abnormal Behavior

features, especially with lower correlation, can lead to the deterioration of the IDS performance. This reasoning partially motivated the decision to model the proposed solution as a two-factor detection mechanism instead of simply integrating the features from the different domains.

5.3 PERFORMANCE EVALUATION

This section outlines the process for evaluation of the performance in terms of the ability of the AIS to detect intrusions, describing the process of generating models, the plausibility function, and the metrics selected for evaluating our IDS.

5.3.1 Building AIS Monitors

As in Section 4.8, we build our IDS mechanism considering relevant variables and the temporal behavior of the CPS measured from the messages exchanged by its components. This section describes the process to prepare the dataset to generate our

AIS models for the system. The model generation steps are divided into three steps: preprocessing, feature selection, and training.

5.3.1.1 Model Construction

Once the set of features to represent the system behavior has been selected, the training can be carried out. This phase aims at generating the AIS model by building the set of T cells, i.e., configuring the set of cluster centroids that will be used as the system's detectors. In this process, AIS will be generated with different configurations of the initial threshold and minimum number of members to accept a T cell.

First, as AIS training only considers the CPS self behavior, we separated the dataset into samples representing moments under attack and not. Finally, we split training and testing data using a 70-30% split with a pseudo-random static seed.

The configurations considered for the training process are depicted in Table 7. Smaller values for the minimum number of members were not considered due to the number of samples in the chosen dataset. Finally, the ideal configuration is the one that combines a small false-positive rate, a small false-negative rate, a large number of minimum members, a small number of clusters, and a small value for the final threshold.

Table 7 – AIS Grid-Search Parameters

AIS Parameter	Configurations
Min. Members	100, 250, 500, 1000
Initial Threshold	{0.05, 0.075, 0.1, 0.15, 0.2, 0.25, 0.3}

5.3.2 Physical Plausibility Strategy

For the sake of the validation of the present work, we adopt NN-based models for identifying and approximating the function that defines the value of a target variable based on the underlying properties and data relationships that govern the operation of the CPS to build predictors for each variable composing the state of the system based on a set of correlated features defined according to a correlation analysis.

First, the physical dataset was submitted to the z-score normalization, as in equation (32). As we wanted the predictors to learn the correct system behavior, the dataset was then filtered to keep only samples representing normal execution of the CPS.

Finally, the Pearson Correlation Coefficient was used to yield a set of correlated features. To avoid compromising or overfitting the model with data with low correlation, only five features were selected to train the models. The results of the correlation analysis are depicted in Table 8. Features for which the Pearson Correlation Coefficient could not find variables with a covariance relationship are omitted from the table. We

extended this analysis for the equivalent feature. Examples are the valves V_{10} , V_{11} , and V_{12} that control the output for Tank 3 (S_3) and are all opened at the same time. The same is observed for valves V_{13} , V_{14} , and V_{15} for Tank 4 (S_4). Predictors for variables that did not present at least 3 non-equivalent features with more than 50% correlation (0.5) were not generated to avoid generating false-positives. The existence of variables for which no correlated features were found, or the correlated features have a small correlation score, is partially explained by the level to which those features are involved to the main operation flow determined by the stages S_1 to S_4 .

A small grid search was conducted to tune hyperparameters like the number of neurons in the hidden layers, activation functions, and optimizer. The resulting predictors are then explored to derive a value that measures the data plausibility. The plausibility values are derived through the concept of confidence attribution proposed by Scheffel et al. (SCHEFFEL, R. M.; FRÖHLICH, A. A., 2018; SCHEFFEL, Roberto M.; FRÖHLICH, Antônio A., 2019), which evaluates data points with a confidence level ranging from 0 to 100%, indicating how much an observed value is believed to be correct.

To calculate this measure, the actual observed value v and the value produced by the predictor $\hat{v}$ are used in the function $f(v, \hat{v})$ to determine the current reading's confidence level, as shown in equation (33).

$$f(v, \hat{v}) = \begin{cases} 1, & \text{if } |v - \hat{v}| \leq \beta \times MAE \\ 1 - \frac{|v - \hat{v}| - \beta \times MAE}{\alpha \times MAE}, & \text{otherwise} \end{cases} \quad (33)$$

$$MAE = \frac{1}{n} \sum_{i=1}^n |v_i - \hat{v}_i| \quad (34)$$

The α and β in the equation are values that can be used to fine-tune the sensitivity of the formula. β determines the *tolerance* on the deviation from the predicted value, in which values are still considered correct. *MAE* stands for *Mean Absolute Error* (equation (34)), and should be retrieved from the predictor training process.

By using the *MAE*, equation (33) forces the confidence of a given value to be dependent on the accuracy of the predictor used to generate $\hat{v}$. This solution tends to reduce false positives in the final classification of AIS introduced by less-accurate predictors, once the tolerance for how much a variable can deviate from the prediction before a significant drop in confidence level is observed is directly dependent on how good the predictor demonstrated to be during training.

5.3.3 Evaluation Metrics

Regarding the AIS performance, the present work will be evaluated in terms of accuracy, recall, false positive rate, and false negative rate. The metrics will be

Table 8 – Correlation Analysis for the Variables Composing the State of the Physical Components

Target Variable	Correlated Features by Absolute Correlation
S_1	V_{20} : 0.73, S_6 : 0.70, P_4 : 0.62, F_2 : 0.54, P_6 : 0.53
S_2	V_{17} : 0.65, S_4 : 0.49, V_{10} : 0.42, V_{11} : 0.42, V_{12} : 0.42
S_3	V_{12} : 0.72, V_{11} : 0.72, V_{10} : 0.72, P_1 : 0.69, S_4 : 0.50
S_4	V_{15} : 0.80, V_{13} : 0.80, V_{14} : 0.80, S_3 : 0.50, V_{10} : 0.49
S_5	P_2 : 0.72, V_{18} : 0.60, F_1 : 0.59, P_5 : 0.58, S_7 : 0.49
S_6	S_1 : 0.70, V_{20} : 0.63, P_4 : 0.60, P_5 : 0.51, F_1 : 0.49
S_7	P_6 : 0.73, F_2 : 0.73, F_1 : 0.65, P_5 : 0.64, S_5 : 0.49
S_8	F_2 : 0.74, P_6 : 0.74, F_4 : 0.63, V_{22} : 0.54, S_6 : 0.49
P_1	V_{10} : 0.91, V_{12} : 0.91, V_{11} : 0.91, S_3 : 0.69, S_4 : 0.47
P_2	S_5 : 0.72, V_{20} : 0.44, S_7 : 0.42, F_1 : 0.37, P_5 : 0.37
P_4	S_1 : 0.62, S_6 : 0.60, V_{20} : 0.59, S_8 : 0.37, V_{15} : 0.35
P_5	F_1 : 0.99, S_7 : 0.64, S_5 : 0.58, S_6 : 0.51, F_4 : 0.43
P_6	F_2 : 0.99, S_8 : 0.74, S_7 : 0.73, S_1 : 0.53, V_{20} : 0.45
F_1	P_5 : 0.99, S_7 : 0.65, S_5 : 0.59, S_6 : 0.49, V_{22} : 0.43
F_2	P_6 : 0.99, S_8 : 0.74, S_7 : 0.73, S_1 : 0.54, V_{20} : 0.45
F_4	V_{22} : 0.95, S_8 : 0.63, S_5 : 0.43, F_1 : 0.43, P_5 : 0.43
V_{10}	V_{11} : 1.00, V_{12} : 1.00, P_1 : 0.91, S_3 : 0.72, S_4 : 0.50
V_{11}	V_{12} : 1.00, V_{10} : 1.00, P_1 : 0.91, S_3 : 0.72, S_4 : 0.50
V_{12}	V_{11} : 1.00, V_{10} : 1.00, P_1 : 0.91, S_3 : 0.72, S_4 : 0.50
V_{13}	V_{15} : 1.00, V_{14} : 1.00, S_4 : 0.80, S_1 : 0.52, V_{20} : 0.47
V_{14}	V_{15} : 1.00, V_{13} : 1.00, S_4 : 0.80, S_1 : 0.52, V_{20} : 0.47
V_{15}	V_{13} : 1.00, V_{14} : 1.00, S_4 : 0.80, S_1 : 0.52, V_{20} : 0.47
V_{17}	S_2 : 0.65, S_5 : 0.38, V_{18} : 0.24, P_2 : 0.24, S_3 : 0.23
V_{18}	S_5 : 0.60, S_4 : 0.41, V_{20} : 0.37, P_5 : 0.35, F_1 : 0.35
V_{20}	S_1 : 0.73, S_6 : 0.63, P_4 : 0.59, V_{15} : 0.47, V_{13} : 0.47
V_{22}	F_4 : 0.95, S_8 : 0.54, S_5 : 0.46, F_1 : 0.43, P_5 : 0.43

calculated both at the individual scopes of physical attacks and network attacks, and the overall performance.

Considering:

- TP : True Positive (Nonself correctly classified)
- TN : True Negative (Self correctly classified)
- FP : False Positive (Self incorrectly classified as nonself)
- FN : False Negative (Nonself incorrectly classified as Self)

The metrics are calculated as follows:

Accuracy: the proportion of the total number of predictions that were correct.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Recall: the proportion of actual positive cases (nonself) that are correctly identified.

$$Recall = \frac{TP}{TP + FN}$$

False Positive Rate (FPR): the proportion of actual negative cases (self) that are incorrectly identified as positive.

$$FPR = \frac{FP}{FP + TN}$$

False Negative Rate (FNR): the proportion of actual positive cases (nonself) that are incorrectly identified as negative.

$$FNR = \frac{FN}{FN + TP}$$

Finally, a comparison between the current method and the methods originally experimented with by Faramondi et al. (FARAMONDI et al., 2021) when introducing the adopted dataset will also be presented.

5.3.4 AIS Performance Results

This section presents the evaluation of the AIS-based IDS solution proposed in this work in terms of its ability to detect anomalies, attacks, and intrusions in CPS datasets. The results are based on models constructed according to the process described in Section 5.3.1 of the experimental setup.

During the training, no limitation is actually applied regarding the minimum number of members a cluster should have by the end of the censoring phase for it to be kept as part of the self model. Therefore, models were trained with the limitation set to 100, i.e., the smallest value for the configuration of the minimum number of samples. The tests presented later will simply adjust this parameter when loading AIS models by only loading a T cell when the number of samples that are tied to it exceeds the minimum. The yielded training results are presented in Table 9.

When considering the total number of self samples of 9.851.133, with 70% of this value (i.e., 6.895.793) being used for training, the AIS generated, in the worst case, 0.01% of the training samples as T cells. For the evaluation, we selected configuration 4 to establish a balanced tradeoff between false positive and false negative rates. This tradeoff is reduced by the selection of the maximum proposed configuration for the minimum number of members (i.e., 1000) per T cell according to Table 7. Such a configuration yields a model with 129 T cells. At this point, we analyze the AIS model

Table 9 – AIS Configurations

Configuration	Initial Threshhold	Final Threshhold	#T cells
1	0.05	0.045	687
2	0.075	0.072	460
3	0.1	0.097	393
4	0.15	0.144	240
5	0.2	0.197	186
6	0.25	0.246	134
7	0.3	0.298	109

Table 10 – AIS-Only Classification Results

Network Traffic Class	Accuracy
Normal	99.15%
DoS	98.75%
MITM	1.32%
Scan	100%

quality prior to the integration of the plausibility functions. The obtained results are depicted in Table 10.

As noticed by the presented results, the generated AIS models demonstrated good results on the detection of the DoS and Scan attacks. Nevertheless, using the chosen set of features, the AIS alone demonstrated a poor ability to detect MITM attacks. Despite that, the overall accuracy for the IDS was 88.28%, with a 68.65% recall, 31.35% false negative rate, and only 0.85% false positives. Such results are already better when compared to the ones presented by Faramondi et al., in which the only configuration that yielded a higher recall had a considerably lower accuracy (FARAMONDI et al., 2021).

The inability of the AIS to properly discriminate MITM attacks from normal traffic is explained by the characteristics of this attack, how it impacted the network, and how the AIS operates. As described early this section, the MITM attack consisted of fixing the value of the attacked sensor to the last value before the attack impairs its readings. This presents direct results on the physical application whenever the value of the sensor changes in the real world, as such a change would not be recognized by other CPS components, such as the control mechanism.

In the network dataset, the MITM attack is noticeable mainly as it affects the MODBUS value and the source MAC address features. Nevertheless, this attack makes punctual changes to the network dataset and at a frequency smaller than the frequency messages are exchanged. Therefore, as AIS is a point-based anomaly detection, it

would be able to detect the sample in which these values were changed by the attack, but the remainder samples would not be characterized as abnormal. Finally, as the feature selection techniques applied to this work consider the problem through the optics of variation and covariation of features, and the label for the dataset is static throughout the entire duration of the attack, the correlation value of the necessary features is considerably low, especially on the MODBUS value, which is only directly affected by this attack.

The next step was to train the predictors and calculate the plausibility measure to be integrated into the AIS classification process. Once predictors were generated, the confidence attribution was calculated, and the plausibility measure results were verified against the normal behavior of each variable in a dataset without the presence of attacks that could corrupt the system's state representation, which was also provided by (FARAMONDI et al., 2021). This process aims at assessing the overall quality of the approximations of the functions that govern the operation of each component. The average plausibility measure values in a range of 0 to 1 for the features are presented in Table 11. Equivalent variables such as V_{10} , V_{11} , and V_{12} and V_{13} , V_{14} , and V_{15} use the predictor with highest score amongst them.

Table 11 – Average Plausibility Measure Scores For Normal Behavior

Feature	Average Plausibility
S_1	0.86
S_3	0.84
S_5	0.83
S_6	0.94
S_7	0.91
S_8	0.92
P_4	0.93
P_5	0.93
P_6	0.99
F_1	0.99
F_2	0.99
$V_{10...12}$	0.92
$V_{13...15}$	1.0
V_{20}	0.93

Finally, we measured the impacts of the application of the plausibility measures to the AIS classification. In particular, considering the ability to detect the MITM attacks, which were not previously detected.

First, let us review the variables involved in describing the state of the CPS subsystem responsible for controlling the components involved in each attack. For the sake of facilitating the visualization, Table 12 summarizes the configurations of the performed MITM attacks.

Table 12 – MITM Attack Scenarios according to (FARAMONDI et al., 2021)

Scenario	Impact on Physical Process	Subsystem	Timespan
1.1	Affected Pressure Sensor of Tank 6 (S_6)	S2, S3	146 s
1.3	Affected Pressure Sensor of Tank 5 (S_5)	S1, S2	87 s
1.5	Affected Pressure Sensor of Tank 7 (S_7)	S3, S4	122 s
1.7	Affected Flow Sensor 1 (F_1)	S3	87 s
1.8	Affected Flow Sensor 2 (F_2)	S3	87 s
2.12	Affected Pressure Sensor of Tank 7 (S_7)	S3, S4	95 s
3.6	Affected Pressure Sensor of Tank 7 (S_7)	S3, S4	67 s
3.7	No observed effect	-	45 s

Following such a definition, we start the evaluation of the physical plausibility integration by analyzing the ML-based plausibility model. The plausibility measures to be adopted were primarily selected as the models for the pressure sensors of Tanks 5, 6, and 7, alongside the models for the flow sensors 1 and 2. Given that attacks in the proposed dataset are labeled across their entire duration—rather than only the precise messages that constitute the attack—we configured our IDS to consider these expanded event windows for all classifications. In this way, we not only selected the plausibility measures related to each message exchanged during this interval, but also considered the plausibility measures for the attacked subsystems. Alternatively, one could measure whether the attack was detected at some point during its occurrence. The choice in here is made solely to demonstrate how the impacts of the attack manifest themselves in the system's state representation over time and to facilitate comparisons and the estimation of performance metrics. Figures 9, 10, 11, 12, 13, 14, 15, and 16 depict the plausibility measures for the target variables during the MITM attack intervals, i.e., plots start and finish at the first and last samples labeled as during the execution of the attack.

As the ML-based plausibility measure consists of a value indicating a plausibility degree, the results in the AIS classification consist of 3 steps. First, the plausibility measure values of all the variables composing the system state are calculated. Then, the resulting dataset was aligned in time with the network samples. As the sampling rate of the two datasets is considerably different, for each sample in the network dataset, the assigned plausibility measure value consisted of the value assigned to the target variable in the closest sample of the physical dataset prior to the network sample.

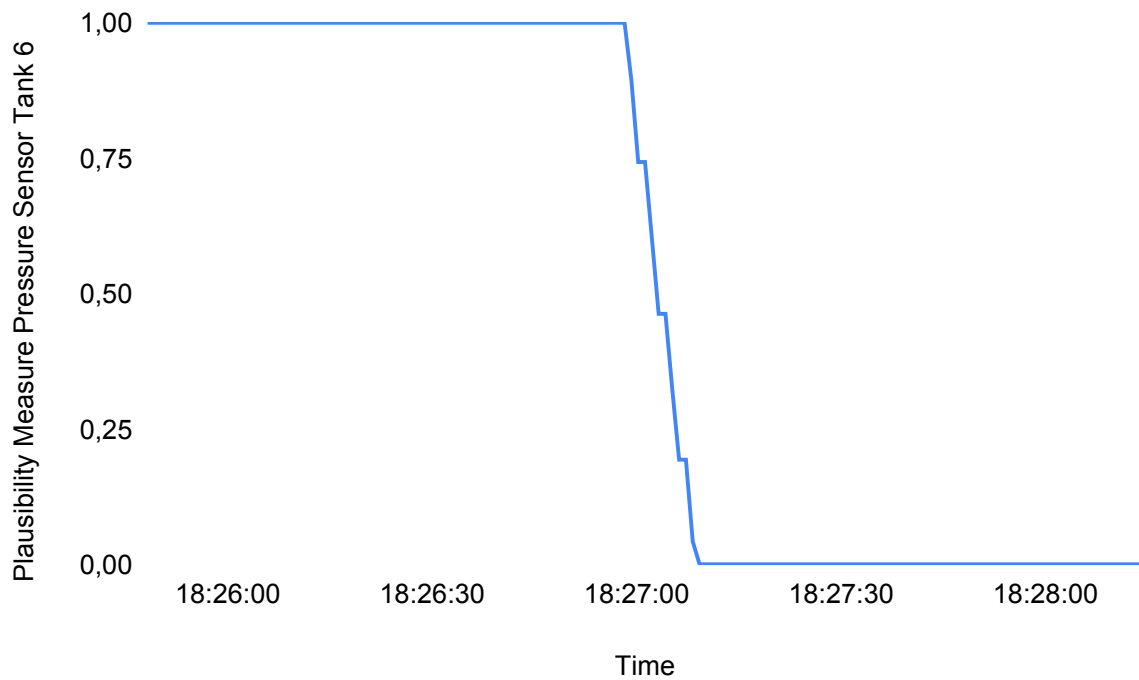


Figure 9 – Physical Plausibility Measure during MITM Attack from Scenario 1.1

Particular attention is given to scenario 1.8, presented by Figure 13. In this case, despite the MITM attack holding the value of the attacked sensor (F_2) to the last value prior to the attack, no drop is observed in the plausibility measure for the attacked sensor. Such a fact may represent an inaccurate prediction that leads the plausibility measure to evaluate the observed value with a high probability of being correct. Nevertheless, the imposed distortion of F_2 value caused a disruption to the system state representation, which is visible in other components. This comes from the fact that F_2 and F_1 measure output and input water flows from Tank 7, which is directly connected to Tank 6 (S_6), and Pump 5 (P_5). Therefore, by applying the principle of selecting the lowest plausibility measure score, the value used for this scenario is the one assigned to F_1 .

Finally, with the datasets fully aligned, the plausibility measure for the data used by the cybernetic processes to model the physical domain of the CPS is attached to each of the samples. The next step was to run the AIS classification process, considering the plausibility measure as a limiter for the tolerance of the deviation from the modeled behavior.

The resulting analysis yields a 69.5% detection rate for MITM attack. This would increase our recall to 89.7%, reducing the false negative rate to 10.3%. Such an increase in recall represents a considerable improvement to the original AIS ability to detect intrusion. The impact of the adoption of a plausibility function in this case, however, is reduced by the fact that scenario 3.7 did not present noticeable effects on the

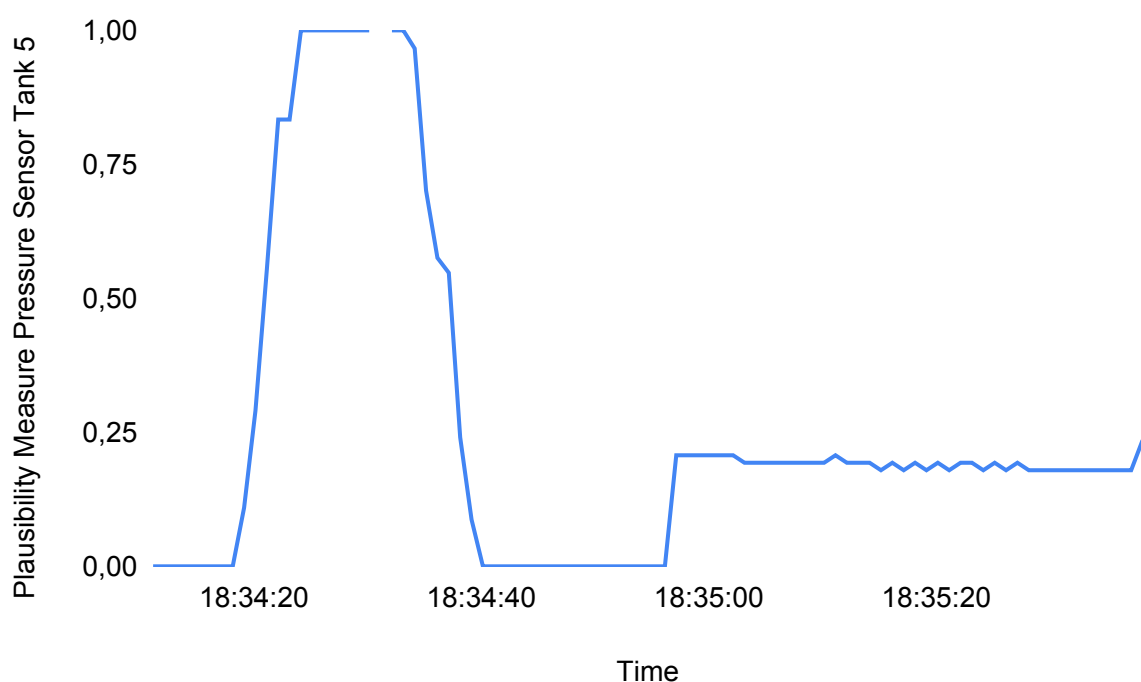


Figure 10 – Physical Plausibility Measure during MITM Attack from Scenario 1.3

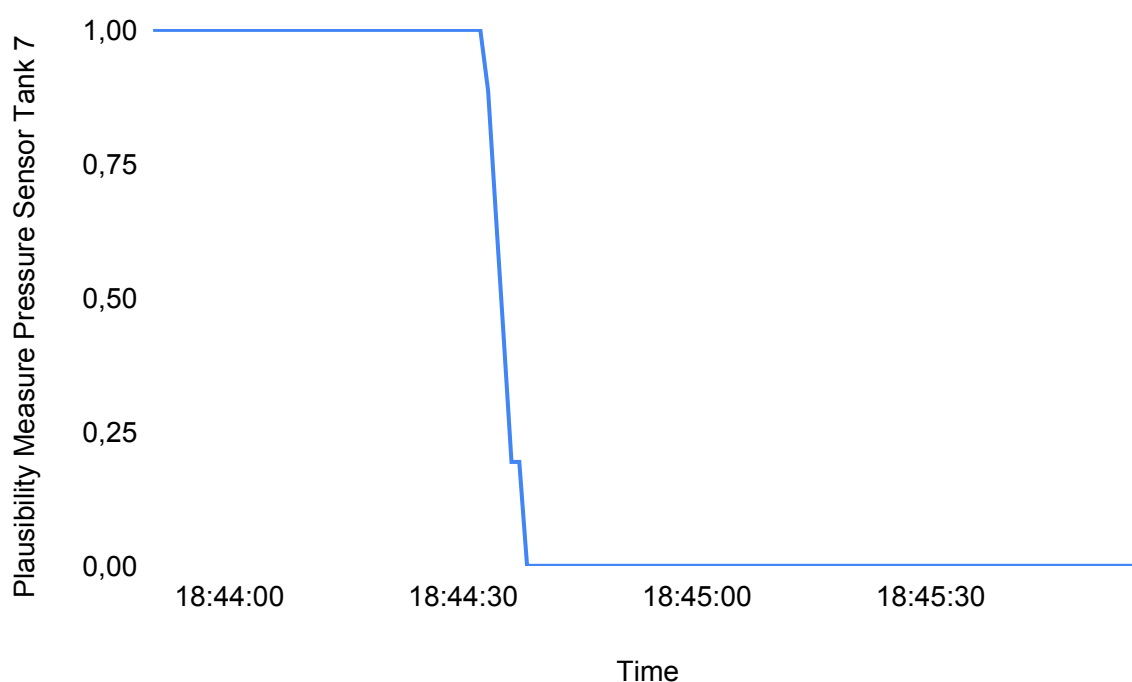


Figure 11 – Physical Plausibility Measure during MITM Attack from Scenario 1.5

data representing the physical process.

In some of the attacks, the physical plausibility evaluation presents a detection delay, as evidenced in Scenarios 1.1, 1.5, 1.7, and 2.12. In these cases, the plausibility measure remains high for a brief interval after the MITM attack starts. This delay is

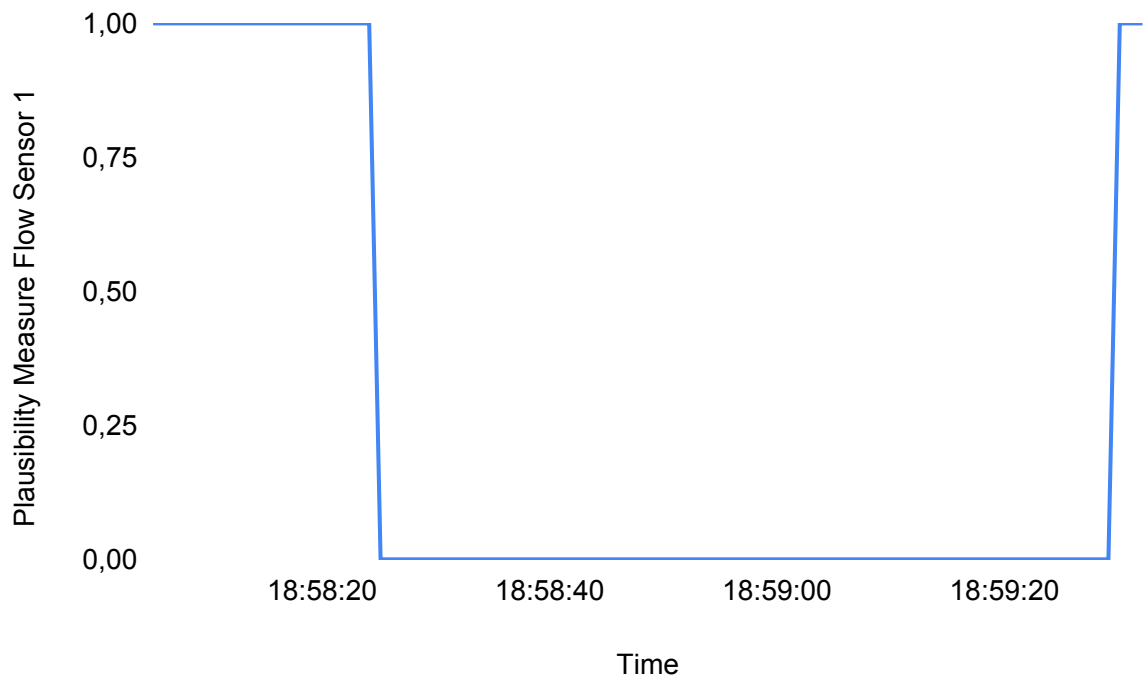


Figure 12 – Physical Plausibility Measure during MITM Attack from Scenario 1.7

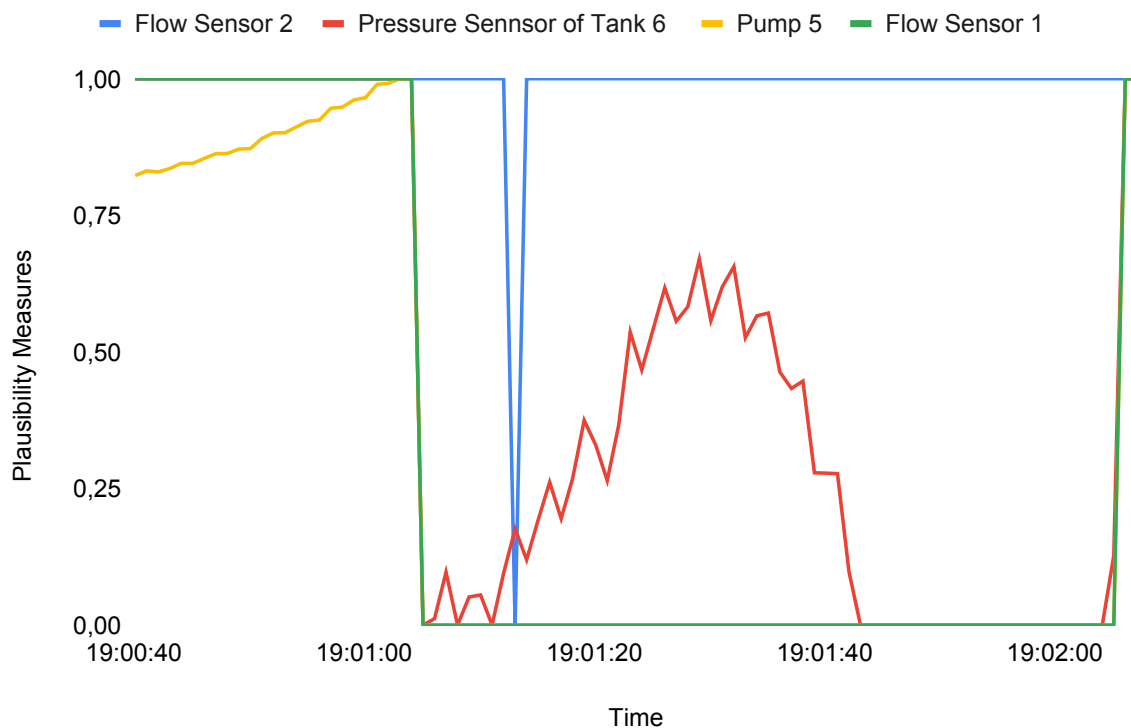


Figure 13 – Physical Plausibility Measure during MITM Attack from Scenario 1.8

partly explained by the fact that the plausibility functions are NN models that approximate the relationships between variables. An alternative approach for modeling the plausibility functions would be defining association rules that specify explicit actuation

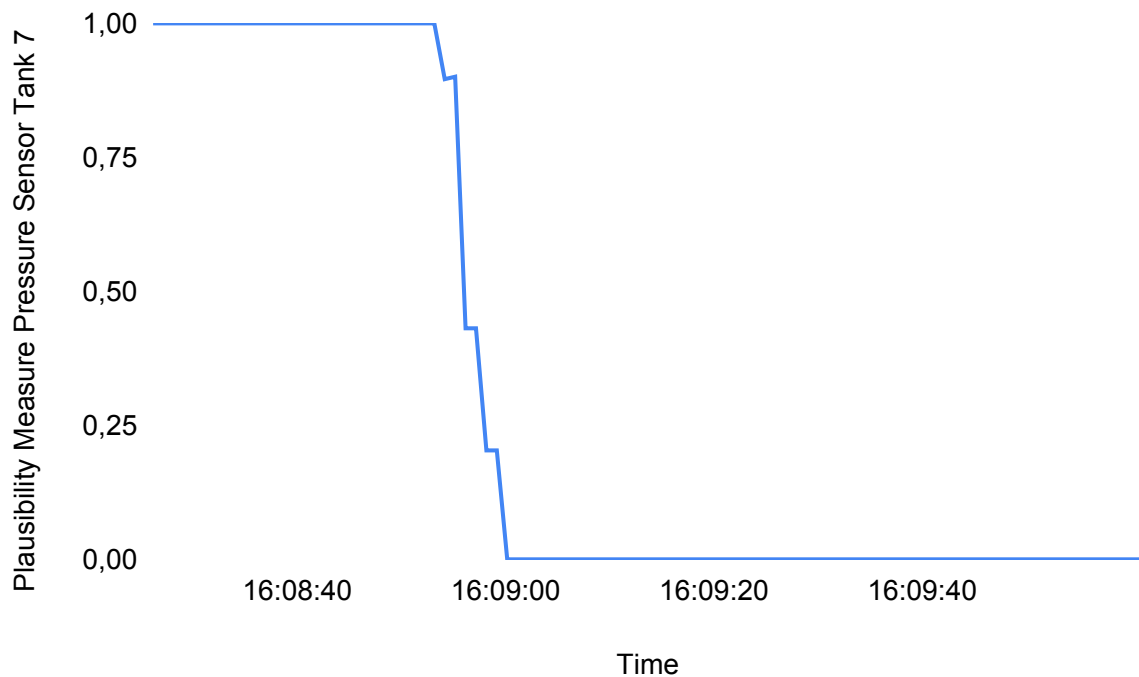


Figure 14 – Physical Plausibility Measure during MITM Attack from Scenario 2.12

triggers. However, automatically defining such triggers is not always feasible for complex systems. While the learned relationship model may introduce temporal latency as it accumulates evidence of physical inconsistency, it provides a more flexible and comprehensive validation of the system's operation. This delay effectively represents the time required for the manipulated cyber-input to manifest as a statistically significant deviation in the physical domain.

Another interesting observation is that the accuracy of the AIS drops in between attacks. This is explained by the fact that once an attack is over, the cybernetic components that control the physical components are operating with an inconsistent state representation. This is corroborated by Figures 9, 10, 11, 14, 15, where the measure of plausibility remains very low even as the attack ends. In fact, when considering the nature of MITM attacks, which, in this case, fix the value of readings of the sensors, Figures 12 and 13, in which the measure of plausibility presents a high value that will potentially change as the cybernetic components get an updated state representation. The prolonged low measures of plausibility, as expected, is not true in moments before attacks, which is corroborated by Figures 9, 11, 12, 13, 15, and 16.

Despite representing an eventual drop in the accuracy of the classification, this phenomenon is only observed because the proposed solution is detecting an existing inconsistency between the system state as modeled by the cybernetic processes and the actual state of the physical components of the CPS. This could be solved by loosening the detection parameters either of the AIS or on the confidence attribution equation,

which is being used to calculate the plausibility measure. Nevertheless, by doing so, one is actively decreasing the IDS's ability to detect intrusions and abnormal behavior.

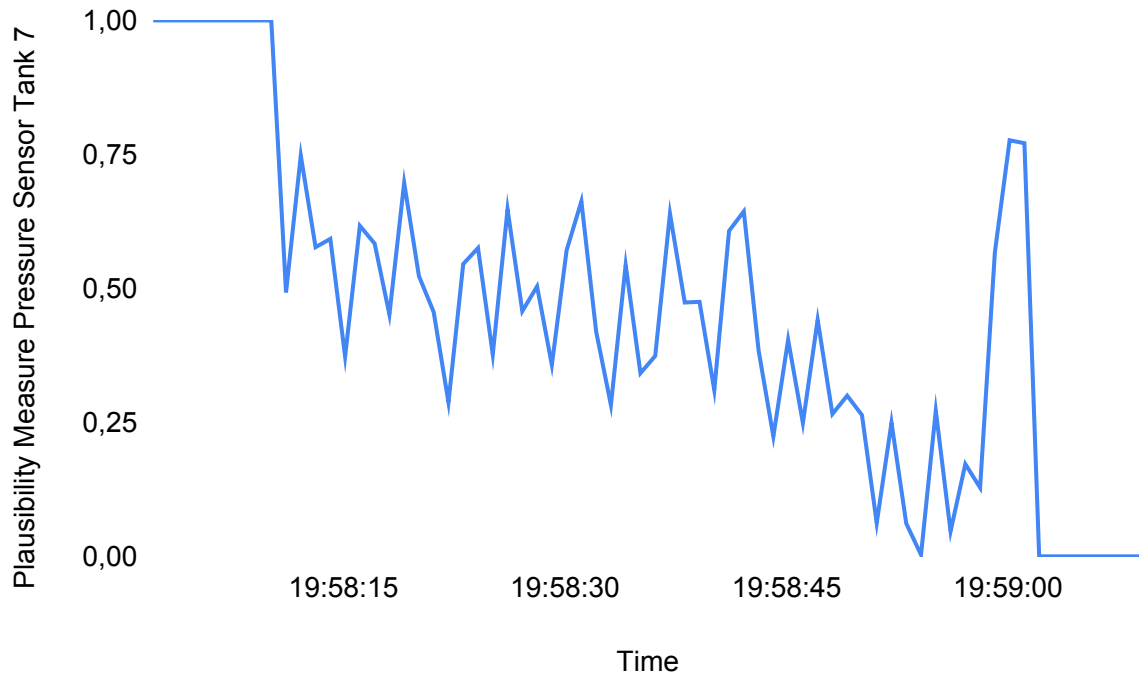


Figure 15 – Physical Plausibility Measure during MITM Attack from Scenario 3.6

It is important to notice that this would not be the case for normal CPS operation with no attacks disrupting the variables that compose the system state representation used by the cybernetic processes to model the CPS physical processes. This is demonstrated by average plausibility scores for normal operation without attacks, depicted in Table 11.

5.3.5 Discussion

The False Positive Rate for AIS-based models is mainly controlled by the minimum number of members for a cluster to not be censored when finalizing training. In other words, a very small threshold tends to generate more clusters. In a worst-case scenario, all of those clusters will represent a very small part of the CPS behavior.

Considering the implementation proposed in Chapter 4 for the AIS, once we reduce the binding threshold used for training to a value smaller than the final threshold of the current models, set by the end of their training phase, we will force fewer samples to be accepted on existing clusters, potentially generating new clusters.

In this way, reducing the binding threshold can increase the number of clusters generated, which in turn may reduce the number of false negatives, increasing the intrusion detection capabilities. However, such a change might result in reducing

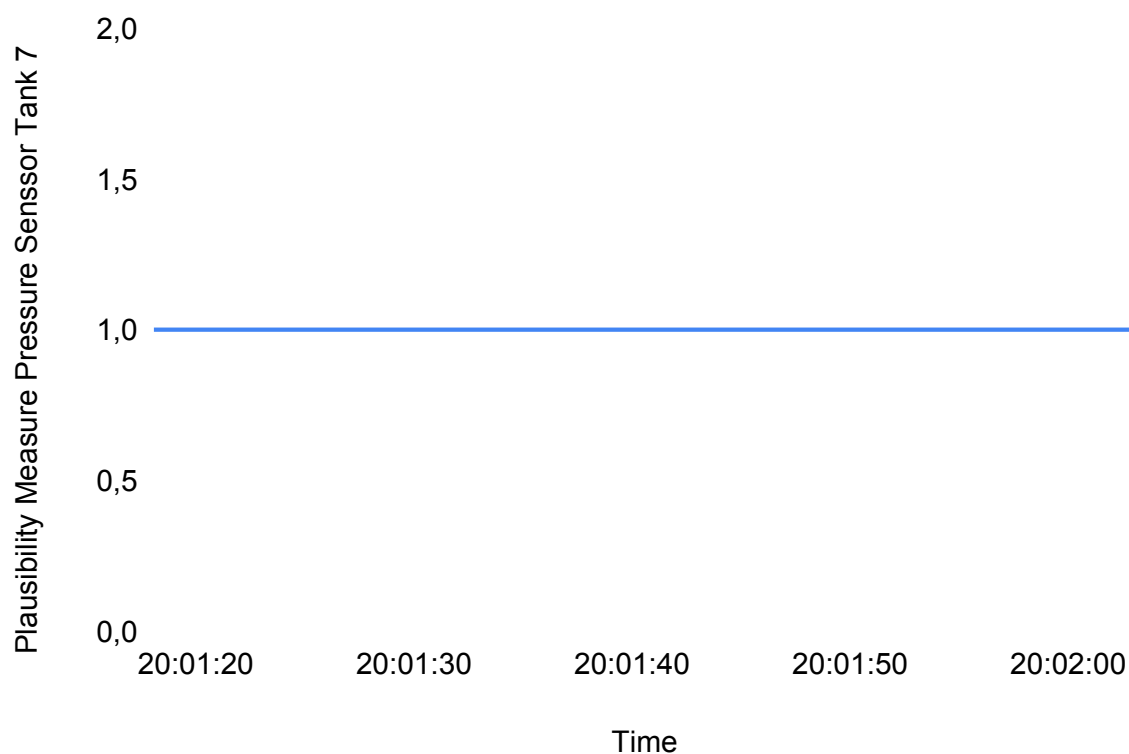


Figure 16 – Physical Plausibility Measure during MITM Attack from Scenario 3.7

the accuracy for self classification, increasing the number of false positives. Such a phenomenon is demonstrated in the results presented in this work.

On the other hand, despite reducing false positives, the relaxation of the binding threshold can result in generating fewer clusters, reducing the specificity of the model. This would reduce the usage of computational resources at the cost of reducing the ability to detect intrusions.

Increasing the minimum number of members parameter can present a negative impact on the self detection, especially for less frequent behaviors. Meanwhile, a higher minimum number of members can positively impact the number of T cells on the T set and the detection of nonself. If considering a lower value for the minimum number of members, however, more clusters can be accepted as part of the CPS model. This would increase the utilization of computational resources and potentially increase the accuracy for detecting self behavior. However, this can potentially increase the number of false negatives, i.e., reduce the ability to detect anomalies.

In general, the selection of the final AIS configuration is a tradeoff between tolerance and intrusion detection capabilities, which in turn represent a direct tradeoff between a smaller number of false positives and a smaller number of false negatives. Therefore, such a decision should take into consideration the consequences of both scenarios regarding the CPS application.

The integration of a plausibility function demonstrated great potential to improve the results obtained by AIS. Both integration mechanisms (i.e., threshold manipulation and complementary verification) significantly increased the ability to detect cyber attacks that impact values on sensors and actuators, as corroborated by the obtained results.

The choice of plausibility function, however, should be carefully considered, as well as its application. When dealing with a model that evaluates data plausibility based solely on the learning capabilities of an AI technique, one can introduce or propagate errors. An example of that comes from the fact that, after an attack is detected, the plausibility measure values related to the component impacted by said attack take time to stabilize. The same effect was noticed in the charts presented in the previous section. From the moment an attack begins to the moment its change impacts the confidence model used to calculate the plausibility measure enough so it detects it, a few samples are considered to be perfectly normal.

It is important to notice that, for adopting the self-nonsel self discrimination process considering a threshold-based integration of the AIS and the plausibility measure of the data representing the state of the dynamical system, one should be able to determine a threshold based on how precisely the functions that estimate the plausibility are able to model the system behavior. While this could increase the detection capabilities and potentially reduce the possibility of generating false positives, the definition of an optimal threshold is a challenging task. Another possible solution would be to provide a hybrid solution, where a threshold is assigned to determine whether we should consider the plausibility function as a factor in the AIS classification process.

A manual definition of the plausibility functions, taking into consideration the data relationships established between the components and actual physical properties that govern the operation of each physical process, may be more effective in terms of reducing both the IDS classification errors and the delay for detection, as previously discussed. The potential tradeoff would be requiring a domain engineering process to define the operation rules, which may not be possible for complex CPS.

The training methodology for the proposed IDS is structured as a systematic, automatable pipeline encompassing data acquisition, temporal feature extraction, feature selection, model training, and hyperparameter optimization. Nevertheless, its effective application to complex CPSs, such as autonomous vehicles, may be challenging due to the scale, diversity, and overall complexity of the target environments.

Fundamentally, relying on temporal features presupposes architectural support for precise, synchronized timestamping, which may be absent in legacy designs. Without the required support, the system's temporal behavior model becomes unreliable. Obtaining representative physical data for complex applications may also be challenging. For instance, capturing the continuous, context-dependent spectrum of vehicle

dynamics (e.g., variable friction, aerodynamics) requires extensive, costly real-world or Hardware-in-the-Loop campaigns to prevent the model from overfitting to specific environmental conditions. Even with sufficient data, modeling these physical plausibility functions may present a tradeoff between lightweight linear algorithms, which often fail to capture the necessary non-linear dynamics of modern control systems, and high-fidelity models (e.g., PINNs), which impose computational overheads that can introduce detection delays.

5.4 OVERHEAD EVALUATION

This section outlines the platform setup for measuring the overhead imposed by executing the AIS-based IDS mechanism on an embedded platform. As aforementioned, the present solution assumes the computing platform on which software components of a CPS are executed to provide means for resource partitioning mechanisms. To mimic such conditions, we first configured a hardware platform to provide the executing components with resource isolation thanks to a hypervisor solution. After that, we describe the proposed evaluation scenarios for measuring performance alongside the selected metrics.

For the evaluation of the AIS overhead, we considered a CPS platform configured with three VMs. The VMs are meant to operate isolated from each other, i.e., without incurring into shared-resource contention, which would avoid disrupting the behavior of any of the software components. For the performance measurement estimation, no inter-VM communication was considered. Nevertheless, communication is possible either through shared-memory or internet ports.

For the hardware platform to execute our experiments, we selected a Raspberry Pi 4B board (RASPBERY PI FOUNDATION, 2025), which features a 4-core ARM Cortex A72 CPU 64-bit SoC 1.5 GHz, with 8GB LPDDR4-2400 SDRAM, 32kB data + 48kB instruction L1 cache per core, and 1MB L2 cache.

As a means to provide isolation, we selected the Bao (MARTINS et al., 2020) hypervisor, which is a lightweight, open-source embedded hypervisor designed mainly for targeting mixed-criticality systems that aims at providing strong isolation and real-time guarantees. Bao provides a minimal, from-scratch implementation of the partitioning hypervisor architecture.

Bao implementation comprises only a minimal, thin-layer of privileged software leveraging ISA virtualization support to implement the static partitioning hypervisor architecture: resources are statically partitioned and assigned at VM instantiation time; memory is statically assigned using 2-stage translation; IO is pass-through only; virtual interrupts are directly mapped to physical ones; and it implements a 1-1 mapping of virtual to physical CPUs, with no need for a scheduler.

Three VMs were configured on this experiment. The first VM executed a version

of Linux, the second one executed a workload typical of an autonomous vehicle CPS applications. Finally, the third VM was configured to execute the AIS. Linux is set to run solely as a means to stimulate the architectural utilization and will receive most of the resources in the platform. Both the AIS and the CPS workload were configured as applications inside FreeRTOS (REAL TIME ENGINEERS LTD, 2025), a RTOS for microcontrollers and small microprocessors that is distributed freely under the MIT open source license. The goal of running the secondary application is to guarantee no impacts are caused by AIS operation despite the isolation mechanisms.

As previously mentioned, Bao is a static partitioning hypervisor. This kind of hypervisor assigns hardware resources individually to each VM prior to the system execution. On Bao, this setup is executed according to the settings specified in a configuration file to be defined according to the application and platform. In the following, we outline the configuration adopted for the experiments.

The first step of the configuration of Bao is to define the images to be used in each VM to be instantiated. The code snippet in the Listing 5.1 depicts this step of the configuration.

The VM images defined are Linux, FreeRTOS, and the AIS. The Linux VM represents a version of the General Purpose OS based on Linux. FreeRTOS VM image will hold the critical application hosted by FreeRTOS. Lastly, AIS VM image holds an application consisting of repeating the AIS classification, which is also hosted by FreeRTOS.

```

1 #include <config.h>
2
3 VM_IMAGE(linux_image, XSTR(BAO_DEMOS_WRKDIR_IMGS/linux.bin));
4 VM_IMAGE(freertos_image, XSTR(BAO_DEMOS_WRKDIR_IMGS/freertos.bin));
5 VM_IMAGE(ais_image, XSTR(BAO_DEMOS_WRKDIR_IMGS/ais.bin));

```

Listing 5.1 – Configuration Code - Part 1

The second step of the setup of Bao is to declare the configuration structure which will hold the settings for each of the VMs. The respective code snippet is presented in the Listing 5.2. In case a shared memory region would be used for inter-VM communication (i.e., inter-VM interruptions, and data sharing), a list of the shared memory regions would be declared in this structure.

```

7 struct config config = {
8     CONFIG_HEADER
9
10     .vmlist_size = 3,
11     .vmlist = {

```

Listing 5.2 – Configuration Code - Part 2

The next step of the configuration, depicted by Listing 5.3, is responsible for outlining the settings for the Linux VM. The configuration for the Linux VM is based on the Linux+FreeRTOS Demo which features a dual-guest VM configuration (BAO-PROJECT, 2025).

This configuration assigns two CPUS (physical cores) for the Linux VM, defines its memory region, sets up its Generic Interrupt Controller (GIC) address and the interrupt id to which its timer interrupt will be mapped. Finally, the chosen platform (Raspberry Pi 4B) has a total of 4 available color schemes for page coloring. Therefore, one way (color) of the available coloring schemes was assigned for the Linux VM.

```

12     {
13         .image = {
14             .base_addr = 0x20000000,
15             .load_addr = VM_IMAGE_OFFSET(linux_image),
16             .size = VM_IMAGE_SIZE(linux_image)
17         },
18         .entry = 0x20000000,
19         .colors = 0b0001,
20         .platform = {
21             .cpu_num = 2,
22             .region_num = 1,
23             .regions = (struct vm_mem_region[]) {
24                 {
25                     .base = 0x20000000,
26                     .size = 0x40000000,
27                     .place_phys = true,
28                     .phys = 0x20000000
29                 }
30             },
31             .dev_num = 2,
32             .devs = (struct vm_dev_region[]) {
33                 {
34                     /* GENET */
35                     .pa = 0xfd580000,
36                     .va = 0xfd580000,
37                     .size = 0x10000,
38                     .interrupt_num = 2,
39                     .interrupts = (irqid_t[]) {189, 190}
40                 },
41                 {
42                     /* Arch timer interrupt */

```

```

43         .interrupt_num = 1,
44         .interrupts = (irqid_t[]) {27}
45     }
46 },
47     .arch = {
48         .gic = {
49             .gicd_addr = 0xff841000,
50             .gicc_addr = 0xff842000,
51         }
52     }
53 },
54 },

```

Listing 5.3 – Configuration Code - Part 3

Next, we declare the two FreeRTOS based VMs. For the AIS VM setup, one CPU (hardware core) was assigned, a timer interrupt id was set just like the Linux VM, and a Generic Interrupt Controller (GIC) was configured. The major difference here is the definition of a UART component to be used for the AIS to print its statistics by the end of the execution.

```

55 {
56     .image = {
57         .base_addr = 0x0,
58         .load_addr = VM_IMAGE_OFFSET(ais_image),
59         .size = VM_IMAGE_SIZE(ais_image)
60     },
61     .entry = 0x0,
62     .colors = 0b0010,
63     .platform = {
64         .cpu_num = 1,
65         .region_num = 1,
66         .regions = (struct vm_mem_region[]) {
67             { .base = 0x0, .size = 0x8000000 }
68         },
69         .dev_num = 2,
70         .devs = (struct vm_dev_region[]) {
71             {
72                 /* UART1 */
73                 .pa = 0xfe215000,
74                 .va = 0xff000000,
75                 .size = 0x1000,
76                 .interrupt_num = 1,
77                 .interrupts = (irqid_t[]) {125}
78             },

```

```

79         {
80             /* Arch timer interrupt */
81             .interrupt_num = 1,
82             .interrupts = (irqid_t[]) {27}
83         }
84     },
85     .arch = {
86         .gic = {
87             .gicd_addr = 0xF9010000,
88             .gicc_addr = 0xF9020000,
89         }
90     }
91 },
92 },

```

Listing 5.4 – Configuration Code - Part 4

The final VM to be configured stands for an application mimicking a critical CPS software component. For this CPS software component VM setup, one CPU (hardware core) was assigned, a timer interrupt ID was set just like the previous FreeRTOS-based VM, and a Generic Interrupt Controller (GIC) is configured. The major difference in here is the definition of a UART component to be used for the AIS to print its statistics by the end of the execution.

```

93     {
94         .image = {
95             .base_addr = 0x0,
96             .load_addr = VM_IMAGE_OFFSET(freertos_image),
97             .size = VM_IMAGE_SIZE(freertos_image)
98         },
99         .entry = 0x0,
100        .colors = 0b1100,
101        .platform = {
102            .cpu_num = 1,
103            .region_num = 1,
104            .regions = (struct vm_mem_region[]) {
105                { .base = 0x0, .size = 0x8000000 }
106            },
107            .dev_num = 2,
108            .devs = (struct vm_dev_region[]) {
109                {
110                    /* UART1 */
111                    .pa = 0xfe215000,
112                    .va = 0xff000000,
113                    .size = 0x1000,

```

```

114         .interrupt_num = 1,
115         .interrupts = (irqid_t[]) {126}
116     },
117     {
118         /* Arch timer interrupt */
119         .interrupt_num = 1,
120         .interrupts = (irqid_t[]) {27}
121     }
122 },
123 .arch = {
124     .gic = {
125         .gicd_addr = 0xF9010000,
126         .gicc_addr = 0xF9020000,
127     }
128 }
129 },
130 },
131 },
132 };

```

Listing 5.5 – Configuration Code - Part 4

As one can notice, the VM designated to run a typical CPS software component received two of the four colors available for page coloring. The configuration of the page coloring mechanism was solely adopted to guarantee isolation and avoid shared-resource contention between VMs. No technical or scientific claim is made related to the choice of the color assignment in the context of this work.

Other than that, the UART device was mapped for both FreeRTOS-based VMs. Nevertheless, Bao does not control the utilization of the hardware resources. Once mapped, the VM accesses the resource by simply reading or writing. No queue or buffer control operation is performed. Therefore, no extra overhead is incurred.

5.4.1 Evaluation Scenarios

The evaluation of the overhead considered two main scenarios, with or without the AIS execution, with different configurations of AIS. The metric for evaluating the impact of the AIS over the platform considers the overhead introduced by the AIS execution over the CPS software component, which is set as typical CPS workloads, in this case represented by the Disparity Map task, from the San Diego Vision Benchmark Suite (VENKATA et al., 2009). Table 13 depicts the configuration scenarios.

The number of features for the AIS cluster will be defined based on the resulting clusters for the classification scenario described later in this chapter. Furthermore, 1000 iterations of the inference process are executed to collect the total and the average exe-

Table 13 – Experiment Scenarios

Scenario	Description
Experiment Type 1: No AIS	
1.1	CPS Software Component Without AIS
Experiment Type 2: AIS Overhead	
2.1	CPS Software Component With AIS (10 nodes).
2.2	CPS Software Component With AIS (20 nodes).
2.3	CPS Software Component With AIS (50 nodes).
2.4	CPS Software Component With AIS (100 nodes).
2.5	CPS Software Component With AIS (250 nodes).
2.6	CPS Software Component With AIS (500 nodes).
2.7	CPS Software Component With AIS (1000 nodes).

cution time. Finally, beyond measuring the execution time and overhead, the necessary memory for each of the scenarios will be estimated.

5.4.2 Computational Performance, Memory Utilization, and Overhead Results

This section presents the results obtained for the overhead evaluation scenarios defined in Section 5.4. The analysis encompasses the aspects of execution time and memory utilization for different configurations of the AIS. Furthermore, an analysis of the overhead incurred on the CPS software components by the impacts of the AIS execution is also presented. Taking into consideration the results of the data preprocessing 5.1, the AIS is configured in both evaluations to hold a total of 6 input variables.

For the overhead estimation, the execution consisted of creating a set of N random T cells, where N is defined according to Table 13 in Section 5.4.1. For the sake of optimization and simplicity, all samples are converted to integers by a simple scaling process that consisted of multiplying and truncating the value of each variable.

The overhead results calculated according to the described scenarios are depicted in Table 14. The presented results demonstrate that the average inference time for the AIS is practically linear to the number of T cells. This is corroborated by Figure 17, which focuses on the relationship between inference time and number of T cells. Such behavior also reflects the computational complexity evaluation provided in Chapter 4.

The other interesting aspect of the observed performance results is the fact that the execution time of the task configured in the VM parallel to this one in the Bao Hypervisor, the Disparity Map, was not negatively impacted by the execution of the AIS in isolation.

To analyze memory utilization, we must recall the data allocated by the AIS. According to the Chapter 4, the AIS holds two pieces of information, a list of T cells and

Table 14 – Performance measurements with varying numbers of T cells.

Scenario	# T cells	AIS Inference (ms)	Disparity Map Exec. (ms)
1.1	-	-	9,12
2.1	10	0,002	9,1
2.2	20	0,005	9,1
2.3	50	0,01	9,1
2.4	100	0,03	9,1
2.5	250	0,07	9,1
2.6	500	0,14	9,1
2.7	1000	0,28	9,1

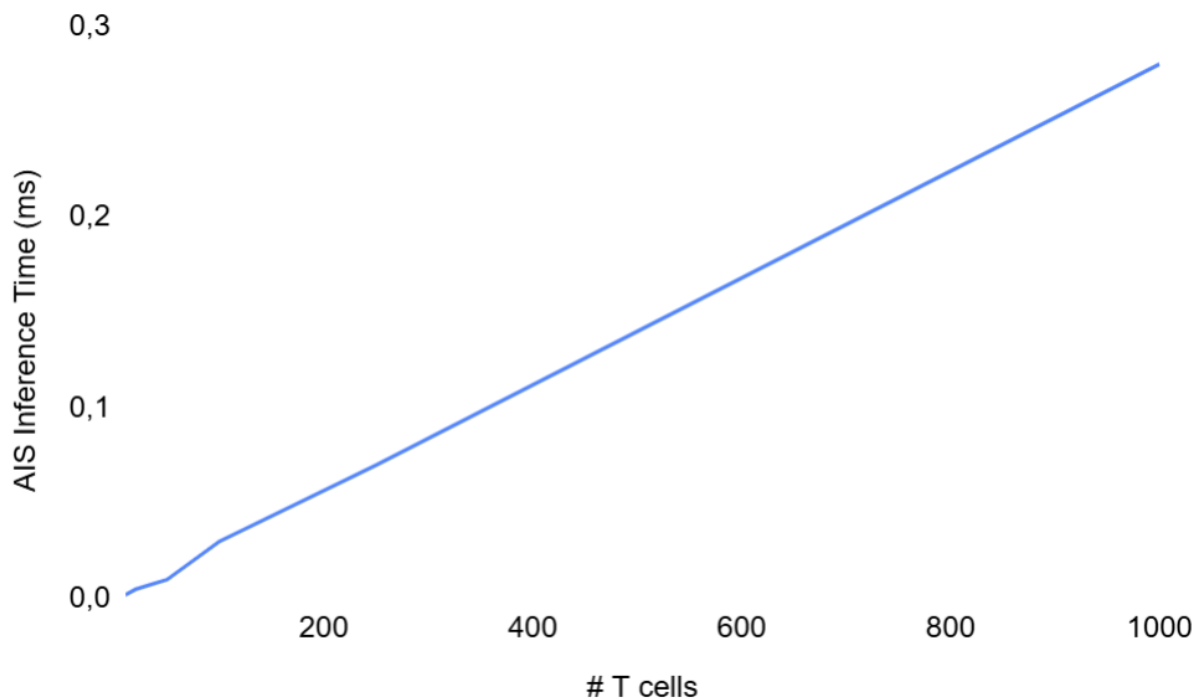


Figure 17 – AIS Inference Time vs Number of T cells

a threshold for detection. The models considered for the evaluation of the AIS overhead, as depicted in Section 5.1, model the T cells as 6-dimensional data points.

Let us consider each variable as a 64-bit (8-byte) variable. Taking into consideration the scenarios presented in Section 5.4.1, the memory utilization for each of the scenarios is depicted in Table 15.

Other than the space necessary for the T cells and the binding threshold, another data structure is stored for the normalization (scaling) of the data. This structure consists of minimum and maximum values for all components being monitored, which in this case are assumed to be equal to 6. Therefore, another 96 bytes of memory are expected to be used.

Table 15 – AIS Memory Consumption

Scenario	Number of Clusters	Total Memory
2.1	10	488 B
2.2	20	968 B
2.3	50	2.408 KB
2.4	100	4.808 KB
2.5	250	12.008 KB
2.6	500	24.008 KB
2.7	1000	48.008 KB

5.4.3 Discussion

In terms of the training and classification processes, the iterative clustering proposed in this work differs from typical clustering methods. In fact, we do consider the present method to be the more suitable method when taking into account both runtime training and classification.

When classifying, the KNN method verifies the distance between the incoming data sample and all the training samples in order to classify it, which yields an inference computational complexity and memory utilization significantly higher, since by using centroids to represent the clusters, we expect the number of clusters to be much smaller than the training dataset size.

K-Means clustering defines a fixed number of clusters for training, rather than specifying a minimum number of members for each cluster. The K-Means centroids are constantly updated during training. The update of centroids during training can yield a more accurate representation of a part of the CPS self behavior. Nevertheless, during the training phase, all data must be processed for each incoming sample in order to update the cluster centroids. Such a policy increases computational complexity and may hinder online training.

In a much different approach, DBSCAN does not define a cluster centroid. Instead, each DBSCAN cluster is represented by a set of core points according to cluster density and the minimum number of points. In fact, in the clustering process used by DBSCAN, every incoming sample can change a previous sample that was classified as a border point (i.e., a member of a cluster that is not a core point) into a core point. Similarly, a noise point can be transformed into a border point or even a core point. For this to work, the entire dataset must be processed during the training phase, which implies a higher computational complexity and memory utilization when compared to the proposed solution, since we assume the number of clusters will be lower than the number of samples in the dataset.

The technique proposed in this work assumes a centroid that is static from the

moment it is defined. Thus, the time complexity of the training phase and classification depends on the number of centroids currently modeling the CPS behavior. Despite increasing computational efficiency, fixing centroids at the act of creation risks producing centroids that are worse representations of a behavior when compared to centroids that are updated throughout the entire model training. A typical example of this is the possibility of T cells that are close enough to recognize overlapping segments of the same behavior, but are just far enough so that they result in two clusters in the model. Although enforcing a minimum membership threshold during the censoring phase can prune weak detectors, these border centroids often lead to the over-representation of dominant behaviors at the expense of model granularity. Therefore, when compared to the aforementioned clustering-based approaches, this solution leads to a possible trade-off between the quality of the generated clusters and the time complexity. Future work can investigate alternatives for mitigating this problem. A possible solution would be merging overlapping T cells based on their representativeness (how many members they recognize).

The performance comparison discussed in this section considers the physical plausibility function to be defined before the system execution. In fact, the complexity of the physical plausibility function should be accounted for when considering its adoption. Despite not impacting the training of the AIS model, high-complexity plausibility functions would also hinder the adoption of the proposed solution for runtime classification.

6 FINAL REMARKS

The research presented in this work introduces an Artificial Immune System-based intrusion detection mechanism tailored for modeling, monitoring, and distinguishing temporal properties of software components of Cyber-Physical Systems. The proposed solution monitors temporal properties through the messages exchanged by each component and increases reliability by accounting for the physical plausibility of the data exchanged in said messages. The experimental validation, considering real-world applications and a high-relevance CPS dataset for IDS, demonstrates the effectiveness of the proposed approach and its suitability for CPS applications. In this way, the contributions detailed in this thesis pave the way for developments in safe and secure CPS.

6.1 FUNDAMENTAL STEPS ACHIEVED TOWARDS INTRUSION DETECTION FOR CPS WITH TEMPORAL BEHAVIOR AND PHYSICAL PLAUSIBILITY

This work provides a solution for detecting intrusion on Cyber-Physical Systems with low false-positive rates based on the concept of Artificial Immune Systems first introduced by the author in Horstmann et al (HORSTMANN, Leonardo Passig, 2020; HORSTMANN, Leonardo Passig; FROHLICH, Antonio Augusto, 2022) and extended in this thesis to incorporate aspects of temporal behavior and physical plausibility of the monitored Cyber-Physical Systems. First, we reviewed and updated the original version of the AIS (HOFFMANN; FROHLICH, Antonio Augusto, 2022), removing its ties to the EPOS monitoring framework (HORSTMANN, Leonardo Passing et al., 2019) and promoting it as an individual software component. Next, the AIS sampling strategy was modified to accommodate messages and samples from typical network datasets, which required different strategies for preprocessing data, namely, data encoding and handling variables with unsynchronized sampling. Next, we proposed and incorporated temporal features following the data-driven model of the CPS components and their data dependencies.

Once the AIS supported handling messages, dealing with unsynchronized sampling, and temporal features, we reviewed alternatives for verifying the plausibility of the physical process related to the CPS cyber components. We presented physics-related and data-intensive approaches taking advantage of the data-driven model of CPS subsystems for both the identification and elaboration of plausibility functions and the definition of the applicability of a plausibility measure to the AIS model.

Finally, experimental validation demonstrates both the feasibility of incorporating the proposed solution for runtime operation on CPS and the improvement potential of considering both cyber and physical worlds when addressing IDS. On the one hand, the system was evaluated in terms of performance, memory utilization, and overhead in

a complex platform, operating on a VM configured with resource isolation and sharing the platform with a secondary typical CPS workload and a general-purpose OS. On the other hand, the solution was evaluated for its intrusion detection capability using a dataset collected from a relevant CPS focused on IDS development.

We will now review the key milestones achieved in this thesis, in line with the specific goals defined in Section 1.2.

1. **Verification of temporal features on AIS-based IDS:** The original AIS design was enhanced to support the inclusion of temporal features observed on the messages exchanged by CPS components. The extraction of temporal features was introduced into the AIS training process in order to make the AIS classification aware of the system's temporal behavior. As the temporal behavior is a critical factor in the correct operation of a CPS, this represents an improvement over the original AIS model, which solely considered absolute values for the monitored variables as available features. The temporal features introduced in the training process represent the intervals between samples and between events, the communication latency and jitter between CPS components, event frequency, and the number of occurrences on a given interval. The most meaningful features are then selected according to their relevance for the detection of intrusions on the target application through a multi-factor feature selection process, resulting in a consistent model of the CPS operation.

Considering this, we have addressed the first specific goal of this work: **Adapt the AIS-based IDS design proposed by (HORSTMANN, Leonardo Passig, 2020; HORSTMANN, Leonardo Passig; FROHLICH, Antonio Augusto, 2022) to model temporal properties observed on messages exchanged by CPS software components.**

2. **Physical Plausibility Measure:** The AIS-based IDS solution was further enriched by the addition of a plausibility measure of the data that compose the representation of the state of the system. The present work proposes the utilization of ML-based mechanisms to identify and model by approximation the functions that govern the physical operation of the CPS components. The operational models are generated for the variables composing the representation of the system state and provide a means to calculate a plausibility measure of the data used by cybernetic processes to model the physical processes related to the CPS. The plausibility measure is integrated into the IDS classification as a tolerance factor for the deviation between observed and modeled behavior by actively controlling the threshold for anomaly detection. Furthermore, we define the process for selecting the plausibility measures to be applied for intrusion detection by considering the data-driven relationships

between CPS components.

In this way, we have achieved the second specific goal of this work: **Adapt the AIS-based IDS solution to encompass the physical plausibility of the data exchanged by the CPS software components based on properties extracted by their respective SmartData model.**

3. **Proof-of-Concept Implementation:** The proposed approach was implemented for two validation scenarios. The first validation scenario considers its ability to model the CPS behavior through the packets exchanged by CPS components by modeling aspects of their temporal behavior as well as modeling the plausibility of the data used to model the operation of the CPS. We first observed the network traffic data, samples were collected, temporal features were extracted, the features that better represent normal traffic were selected through the analysis of their correlation to the presence of attacks, and the AIS models were generated. The physical data was also analyzed, and a ML-based approximation of the functions and properties that govern the CPS operation was generated for each feature composing the model of the CPS operation state. First, for each of the features, we selected a subset with the most correlated features to train NN-based predictors that were used as plausibility functions and used to extract plausibility measures for each sample. Both mechanisms were then integrated for the final evaluation of the network traffic. The second validation scenario accounted for the computational performance and overhead introduced by the AIS-based IDS operation on a platform equipped with a resource isolation mechanism based on virtualization techniques and on the presence of two other VMs, one with a typical workload of CPS applications in the realm of autonomous vehicles operating on a RTOS, and another considering a general purpose OS.

Thus, we have accomplished our third specific goal for this work: **Define evaluation scenarios and build the models of safe (i.e., normal, expected) behavior. The evaluation scenario definition will include the specification of technologies applied for the resource isolation of software components and the selection of a relevant dataset for IDS application on CPS.**

4. **Classification Performance, Importance Physical Plausibility, and Overhead:** The results obtained through the first evaluation scenario demonstrated the ability of the present implementation of our AIS-based intrusion detection mechanism to model the normal behavior of the CPS components based on the messages they exchange. In fact, with a total of 6 features, two of which were temporal, and 129 T cells, the present AIS-based solution was

able to capture 99% of the system behavior. Despite reducing the detection latency, the policy to detect abnormal behavior based on point anomalies (analyzing individual data points), however, demonstrated inefficiency in detecting a window of MITM attacks. In fact, such attacks were only noticed in individual moments throughout the attacking window. Nevertheless, the incorporation of a physical plausibility measure enhanced the proposed IDS mechanism to detect the effects of such an attack, increasing our recall rates to nearly 90%. Furthermore, the results obtained through the second evaluation scenario demonstrated the computational efficiency of the proposed IDS classification mechanism as the inference time did not exceed 0.28 ms for 1000 T cells. Furthermore, no noticeable impact was observed on the execution of the secondary CPS application running on the same platform, which demonstrates that the AIS overhead did not exceed the limits of the isolation provided by the virtualization mechanism that was adopted.

Thus, we have accomplished the final specific goal of this work: **Evaluate the performance of the solution in terms of overhead, latency, and misbehavior detection accuracy.**

In conclusion, the proposed solution provides a computationally efficient, robust IDS technique that relies on an AIS-based mechanism to model the normal behavior of a CPS and on ML-based mechanisms to learn the rules and functions that govern the CPS operation. The AIS-based mechanism models the system behavior through the messages exchanged by CPS components, extracting and observing temporal features alongside other communication variables. Meanwhile, ML-based mechanisms are used to approximate and model the functions and properties that govern the CPS operation for extracting a measure of the physical plausibility of the data used to model the state of the operation of such dynamical systems. Finally, both mechanisms are integrated for a final decision on the system's normal or abnormal behavior. The combination of such techniques allows for the construction of IDS mechanisms without the dependency on domain expertise to determine aspects of the correct operation of the CPS. Future work may consider investigating the applicability and scalability of the present solution on different CPS scenarios, such as autonomous vehicles. Another significant opportunity lies in investigating alternatives to navigate the tradeoff between model quality and performance optimizations due to the iterative training mechanism as described in previous sections. A typical example would be the merging of overlapping T cells based on their spatial proximity and affinity radius. By consolidating detectors that represent the same behavioral manifold, the model could achieve higher granularity and a more compact representation of the "self" space, ultimately improving the precision of the IDS in complex CPS environments.

REFERENCES

ABDULGANIYU, Oluwadamilare Harazeem; AIT TCHAKOUCHE, Taha; SAHEED, Yakub Kayode. A systematic literature review for network intrusion detection system (IDS). **International Journal of Information Security**, Springer Science and Business Media LLC, v. 22, n. 5, p. 1125–1162, Mar. 2023. ISSN 1615-5270.

AGRAWAL, Rakesh; IMIELIŃSKI, Tomasz; SWAMI, Arun. Mining Association Rules between Sets of Items in Large Databases. **SIGMOD Rec.**, Association for Computing Machinery, New York, NY, USA, v. 22, n. 2, p. 207–216, June 1993. ISSN 0163-5808.

ALI, Jokha. Intrusion Detection Systems Trends to Counteract Growing Cyber-Attacks on Cyber-Physical Systems. In: 2021 22nd International Arab Conference on Information Technology (ACIT). [S.l.: s.n.], 2021. P. 1–6.

ALMUQREN, Latifah; MAASHI, Mashael S.; ALAMGEER, Mohammad; MOHSEN, Heba; HAMZA, Manar Ahmed; ABDELMAGEED, Amgad Atta. Explainable Artificial Intelligence Enabled Intrusion Detection Technique for Secure Cyber-Physical Systems. **Applied Sciences**, MDPI AG, v. 13, n. 5, p. 3081, Feb. 2023. ISSN 2076-3417.

AMORIM, Lucas B.V. de; CAVALCANTI, George D.C.; CRUZ, Rafael M.O. The choice of scaling technique matters for classification performance. **Applied Soft Computing**, Elsevier BV, v. 133, p. 109924, Jan. 2023. ISSN 1568-4946.

AUTOSAR. **Overview of Functional Safety Measures in AUTOSAR.**

Hörgertshausen, Germany: AUTOSAR, 2016. Available from:

https://www.autosar.org/fileadmin/standards/R22-11/CP/AUTOSAR_EXP_FunctionalSafetyMeasures.pdf.

BADILLO, Solveig; BANFAI, Balazs; BIRZELE, Fabian; DAVYDOV, Iakov I.; HUTCHINSON, Lucy; KAM-THONG, Tony; SIEBOURG-POLSTER, Julianne; STEIERT, Bernhard; ZHANG, Jitao David. An Introduction to Machine Learning. **Clinical Pharmacology & Therapeutics**, Wiley, v. 107, n. 4, p. 871–885, Mar. 2020. ISSN 1532-6535.

BAO-PROJECT. **linux+freertos**. [S.l.: s.n.].

<https://github.com/bao-project/bao-demos/tree/master/demos/linux+freertos>. Visited on: 9 Aug. 2025.

BENMALEK, Mourad. Ransomware on cyber-physical systems: Taxonomies, case studies, security gaps, and open challenges. **Internet of Things and Cyber-Physical Systems**, v. 4, p. 186–202, 2024. ISSN 2667-3452.

BOLÓN-CANEDO, Verónica; SÁNCHEZ-MAROÑO, Noelia;
ALONSO-BETANZOS, Amparo. A review of feature selection methods on synthetic data. **Knowledge and Information Systems**, Springer Science and Business Media LLC, v. 34, n. 3, p. 483–519, Mar. 2012.

BOOIJ, Tim M.; CHISCOP, Irina; MEEUWISSEN, Erik; MOUSTAFA, Nour;
HARTOG, Frank T. H. den. ToN_IoT: The Role of Heterogeneity and the Need for Standardization of Features and Attack Types in IoT Network Intrusion Data Sets. **IEEE Internet of Things Journal**, v. 9, n. 1, p. 485–496, 2022.

BORACCHI, Giacomo; ROVERI, Manuel. Exploiting self-similarity for change detection. In: 2014 International Joint Conference on Neural Networks (IJCNN). [S.l.]: IEEE, July 2014.

BORGES HINK, Raymond C.; BEAVER, Justin M.; BUCKNER, Mark A.;
MORRIS, Tommy; ADHIKARI, Uttam; PAN, Shengyi. Machine learning for power system disturbance and cyber-attack discrimination. In: 2014 7th International Symposium on Resilient Control Systems (ISRCs). [S.l.]: IEEE, Aug. 2014. P. 1–8.

BURNS, Alan; DAVIS, Robert I. A Survey of Research into Mixed Criticality Systems. **ACM Computing Surveys**, Association for Computing Machinery (ACM), v. 50, n. 6, p. 1–37, Jan. 2017.

CANADIAN INSTITUTE FOR CYBERSECURITY (CIC). **CSE-CIC-IDS2018) on AWS**. [S.l.: s.n.], 2018. Accessed: 2025-11-18. Available from:
<https://www.unb.ca/cic/datasets/ids-2018.html>.

CATILLO, Marta; PECCHIA, Antonio; VILLANO, Umberto. CPS-GUARD: Intrusion detection for cyber-physical systems and IoT devices using outlier-aware deep autoencoders. **Computers & Security**, Elsevier BV, v. 129, p. 103210, June 2023. ISSN 0167-4048.

CATILLO, Marta; PECCHIA, Antonio; VILLANO, Umberto. MultiCIDS: Anomaly-based collective intrusion detection by deep learning on IoT/CPS multivariate time series. **Internet of Things**, Elsevier BV, v. 30, p. 101519, Mar. 2025. ISSN 2542-6605.

CHANDOLA, Varun; BANERJEE, Arindam; KUMAR, Vipin. Anomaly Detection: A Survey. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 41, n. 3, July 2009. ISSN 0360-0300.

CHEN, Xiaohan; ZHANG, Beike; GAO, Dong. Bearing fault diagnosis base on multi-scale CNN and LSTM model. **Journal of Intelligent Manufacturing**, v. 32, n. 4, p. 971–987, Apr. 2021. ISSN 1572-8145.

CHO, Kyong-Tak; SHIN, Kang G. Fingerprinting electronic control units for vehicle intrusion detection. In: PROCEEDINGS of the 25th USENIX Conference on Security Symposium. Austin, TX, USA: USENIX Association, 2016. (SEC'16), p. 911–927.

CONRADI HOFFMANN, Jose Luis; FROHLICH, Antonio Augusto. Online Machine Learning for Energy-Aware Multicore Real-Time Embedded Systems. **IEEE Transactions on Computers**, p. 1–1, 2021.

CORERO NETWORK SECURITY. **DDOS Trends and Analysis**. [S.l.: s.n.]. <https://www.alpha-gen.co.uk/wp-content/uploads/2015/12/corero-ddos-trends-and-analysis-2015-mid-year-report.pdf>. Accessed: 2020-03-11.

COX, Joseph. Millions of Vehicles Could Be Hacked and Tracked Thanks to a Simple Website Bug. **WIRED**, 2024.

COX, Joseph. Subaru Security Flaws Exposed Its System for Tracking Millions of Cars. **WIRED**, 2025.

DASGUPTA, D.; GONZALEZ, F. An immunity-based technique to characterize intrusions in computer networks. **IEEE Transactions on Evolutionary Computation**, Institute of Electrical and Electronics Engineers (IEEE), v. 6, n. 3, p. 281–291, June 2002.

DINA, Ayesha S.; MANIVANNAN, D. Intrusion detection based on Machine Learning techniques in computer networks. **Internet of Things**, Elsevier BV, v. 16, p. 100462, Dec. 2021. ISSN 2542-6605.

DIRO, Abebe Abeshu; CHILAMKURTI, Naveen; KUMAR, Neeraj. Lightweight Cybersecurity Schemes Using Elliptic Curve Cryptography in Publish-Subscribe fog Computing. **Mobile Networks and Applications**, Springer Science and Business Media LLC, v. 22, n. 5, p. 848–858, Apr. 2017.

DONYANAVARD, Bryan; MÜCK, Tiago; SARMA, Santanu; DUTT, Nikil. SPARTA: Runtime task allocation for energy efficient heterogeneous manycores. **2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)**, IEEE, p. 1–10, 2016.

DUQUE, Solane; OMAR, Mohd. Nizam bin. Using Data Mining Algorithms for Developing a Model for Intrusion Detection System (IDS). **Procedia Computer Science**, Elsevier BV, v. 61, p. 46–51, 2015. ISSN 1877-0509.

ERICSON, Amanda; THAR, Kyi; FORSSTRÖM, Stefan. Enhancing Intrusion Detection in CPS and IIoT with Lightweight Explainable AI Models. In: 2025 IEEE 21st International Conference on Factory Communication Systems (WFCS). [S.l.: s.n.], 2025. P. 1–8.

EZEME, M. O.; MAHMOUD, Q. H.; AZIM, A. Hierarchical Attention-Based Anomaly Detection Model for Embedded Operating Systems. In: 2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA). [S.l.: s.n.], Aug. 2018. P. 225–231.

FARAMONDI, L.; FLAMMINI, F.; GUARINO, S.; SETOLA, R. A Hardware-in-the-Loop Water Distribution Testbed Dataset for Cyber-Physical Security Testing. **IEEE Access**, Institute of Electrical and Electronics Engineers (IEEE), v. 9, p. 122385–122396, 2021. ISSN 2169-3536.

FERNANDES, Diogo A.B.; FREIRE, Mário M.; FAZENDEIRO, Paulo A.P.; INÁCIO, Pedro R.M. Applications of artificial immune systems to computer security: A survey. **Journal of Information Security and Applications**, Elsevier BV, v. 35, p. 138–159, Aug. 2017.

FORREST, S.; PERELSON, A.S.; ALLEN, L.; CHERUKURI, R. Self-nonself discrimination in a computer. In: PROCEEDINGS of 1994 IEEE Computer Society Symposium on Research in Security and Privacy. [S.l.]: IEEE Comput. Soc. Press, 1994.

FROHLICH, Antonio A.; RESNER, Davi. DATA-CENTRIC CYBER-PHYSICAL SYSTEMS DESIGN WITH SMARTDATA. In: 2018 Winter Simulation Conference (WSC). [S.l.]: IEEE, Dec. 2018. P. 1274–1285.

FRÖHLICH, Antônio Augusto. SmartData: an IoT-ready API for sensor networks. **International Journal of Sensor Networks**, Inderscience Publishers, v. 28, n. 3, p. 202, 2018.

GHARAEI, Hossein; HOSSEINVAND, Hamid. A new feature selection IDS based on genetic algorithm and SVM. In: 2016 8th International Symposium on Telecommunications (IST). [S.l.: s.n.], 2016. P. 139–144.

GHOSH, Anup K.; SCHWARTZBARD, Aaron. A Study in Using Neural Networks for Anomaly and Misuse Detection. In: PROCEEDINGS of the 8th Conference on USENIX Security Symposium - Volume 8. Washington, D.C.: USENIX Association, 1999. (SSYM'99), p. 12.

GOH, Jonathan; ADEPU, Sridhar; JUNEJO, Khurum Nazir; MATHUR, Aditya P. A Dataset to Support Research in the Design of Secure Water Treatment Systems. In: CRITICAL Information Infrastructures Security. [S.l.: s.n.], 2016.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron; BENGIO, Yoshua. **Deep learning**. [S.l.]: MIT press Cambridge, 2016. v. 1.

GREENBERG, Andy. Hackers Remotely Kill a Jeep on the Highway—with Me in It. **WIRED**, 2015.

GREENBERG, Andy. Hackers Remotely Kill a Tesla: China Researchers Take Over a Model S. **WIRED**, 2016.

HERN, Alex. Hyundai Facing Legal Action Over Car That Can Be Stolen 'Effortlessly in Seconds'. **The Guardian**, 2025.

HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long Short-Term Memory. **Neural Computation**, v. 9, n. 8, p. 1735–1780, 1997.

HOFFMANN, José Luis Conradi. **On the Specification and Verification of Safety Properties in Data-Driven Critical Systems**. 2025. Ph.D. thesis – Federal University of Santa Catarina and Università di Pisa.

HOFFMANN, José Luis Conradi; FROHLICH, Antonio Augusto. Embedding Anomaly Detection Autoencoders for Wind Turbines. In: 2022 IEEE 27th International

Conference on Emerging Technologies and Factory Automation (ETFA). [S.l.]: IEEE, Sept. 2022.

HOFFMANN, José Luis Conradi; FRÖHLICH, Antônio Augusto. SmartData Safety: Online Safety Models for Data-Driven Cyber-Physical Systems. In: IECON 2022 – 48th Annual Conference of the IEEE Industrial Electronics Society. [S.l.: s.n.], 2022. P. 1–6.

HOORNAERT, Denis; GHAEMI, Golsana; BASTONI, Andrea; MANCUSO, Renato; CACCAMO, Marco; CORRADI, Giulio. Mcti: mixed-criticality task-based isolation. **Real-Time Systems**, Springer Science and Business Media LLC, v. 60, n. 2, p. 328–365, June 2024. ISSN 1573-1383.

HORNIK, Kurt. Approximation capabilities of multilayer feedforward networks. **Neural networks**, Elsevier, v. 4, n. 2, p. 251–257, 1991.

HORSTMANN, Leonardo Passig. **Online intrusion detection in multicore embedded systems based on artificial immune systems**. 2020. S. 120. MA thesis – Federal University of Santa Catarina, Florianópolis, Santa Catarina, Brazil. M.Sc. Thesis.

HORSTMANN, Leonardo Passig; FROHLICH, Antonio Augusto. Intrusion Detection in Multicore Embedded Systems based on Artificial Immune Systems. In: 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA). [S.l.]: IEEE, Sept. 2022. P. 1–8.

HORSTMANN, Leonardo Passig; FRÖHLICH, Antônio Augusto; VÖLP, Marcus. On the Impacts of Shared-Resource Contention on Intrusion Detection Systems based on Performance Monitoring. In: 2024 IEEE 27th International Symposium on Real-Time Distributed Computing (ISORC). [S.l.: s.n.], 2024. P. 1–8.

HORSTMANN, Leonardo Passing; HOFFMANN, José Luis Conradi; FRÖHLICH, Antônio Augusto. A Framework to Design and Implement Real-time Multicore Schedulers using Machine Learning. In: 24TH IEEE Conference on Emerging Technologies and Factory Automation (ETFA). Zaragoza, Spain: [s.n.], Sept. 2019. P. 8.

HUMAYED, Abdulmalik; LIN, Jingqiang; LI, Fengjun; LUO, Bo. Cyber-Physical Systems Security—A Survey. **IEEE Internet of Things Journal**, v. 4, n. 6, p. 1802–1831, 2017.

ISLAM, Shareeful; JAVEED, Danish; SAEED, Muhammad Shahid; KUMAR, Prabhat; JOLFAEI, Alireza; ISLAM, A. K. M. Najmul. Generative AI and Cognitive Computing-Driven Intrusion Detection System in Industrial CPS. **Cognitive Computation**, Springer Science and Business Media LLC, v. 16, n. 5, p. 2611–2625, June 2024. ISSN 1866-9964.

JEFFREY, Nicholas; TAN, Qing; VILLAR, José R. A Review of Anomaly Detection Strategies to Detect Threats to Cyber-Physical Systems. **Electronics**, MDPI AG, v. 12, n. 15, p. 3283, July 2023. ISSN 2079-9292.

JIANG, Yuchen; YIN, Shen; KAYNAK, Okyay. Data-Driven Monitoring and Safety Control of Industrial Cyber-Physical Systems: Basics and Beyond. **IEEE Access**, v. 6, p. 47374–47384, 2018.

JUN, Chen; CHI, Chen. Design of Complex Event-Processing IDS in Internet of Things. In: 2014 Sixth International Conference on Measuring Technology and Mechatronics Automation. [S.l.: s.n.], 2014. P. 226–229.

KADAR, Marine. **Integration Methods for Host Intrusion Detection into Embedded Mixed-Criticality Systems**. 2022. PhD thesis – Kaiserslautern University of Technology, Germany.

KIM, AeChan; PARK, Mohyun; LEE, Dong Hoon. AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection. **IEEE Access**, v. 8, p. 70245–70261, 2020.

KOLIAS, Constantinos; KAMBOURAKIS, Georgios; STAVROU, Angelos; VOAS, Jeffrey. DDoS in the IoT: Mirai and Other Botnets. **Computer**, Institute of Electrical and Electronics Engineers (IEEE), v. 50, n. 7, p. 80–84, 2017.

KOPETZ, H.; BAUER, G. The time-triggered architecture. **Proceedings of the IEEE**, Institute of Electrical and Electronics Engineers (IEEE), v. 91, n. 1, p. 112–126, Jan. 2003.

KUNAL; DUA, Mohit. Machine Learning Approach to IDS: A Comprehensive Review. In: 2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA). [S.l.: s.n.], 2019. P. 117–121.

LANSKY, Jan; ALI, Saqib; MOHAMMADI, Mokhtar; MAJEED, Mohammed Kamal; KARIM, Sarkhel H. Taher; RASHIDI, Shima; HOSSEINZADEH, Mehdi;

RAHMANI, Amir Masoud. Deep Learning-Based Intrusion Detection Systems: A Systematic Review. **IEEE Access**, v. 9, p. 101574–101599, 2021.

LEE, E. A.; MESSERSCHMITT, D. G. Static Scheduling of Synchronous Data Flow Programs for Digital Signal Processing. **IEEE Transactions on Computers**, v. C-36, n. 1, p. 24–35, 1987.

LEE, Jaewoo; KOH, Keumseok. Scheduling Complex Cyber-Physical Systems with Mixed-Criticality Components. **Systems**, MDPI AG, v. 11, n. 6, p. 281, June 2023. ISSN 2079-8954.

LENG, Elena Woo Lai; ZWOLINSKI, Mark; HALAK, Basel. Hardware performance counters for system reliability monitoring. **2017 IEEE 2nd International Verification and Security Workshop (IVSW)**, IEEE, p. 76–81, 2017.

LOYOLA-GONZÁLEZ, Octavio; MEDINA-PÉREZ, Miguel Angel; CHOO, Kim-Kwang Raymond. A Review of Supervised Classification based on Contrast Patterns: Applications, Trends, and Challenges. **Journal of Grid Computing**, Springer Science and Business Media LLC, v. 18, n. 4, p. 797–845, Oct. 2020.

LYU, Minzhao; SHERRATT, Dainel; SIVANATHAN, Arunan; GHARAKHEILI, Hassan Habibi; RADFORD, Adam; SIVARAMAN, Vijay. Quantifying the reflective DDoS attack capability of household IoT devices. In: **PROCEEDINGS of the 10th ACM Conference on Security and Privacy in Wireless and Mobile Networks - WiSec '17**. [S.l.]: ACM Press, 2017.

MAJUMDER, Shibarchi; NIELSEN, Jens Frederik Dalsgaard; BAK, Thomas. PaRTAA: A Real-Time Multiprocessor for Mixed-Criticality Airborne Systems. **IEEE Transactions on Computers**, v. 69, n. 8, p. 1221–1232, 2020.

MARTINS, José; TAVARES, Adriano; SOLIERI, Marco; BERTOOGNA, Marko; PINTO, Sandro. BAO: A Lightweight Static Partitioning Hypervisor for Modern Multi-Core Embedded Systems. In: **WORKSHOP on Next Generation Real-Time Embedded Systems (NG-RES 2020)**. [S.l.]: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020. v. 77. (Open Access Series in Informatics (OASIs)), 3:1–3:14. Open Access.

- MCCULLOCH, Warren S; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. **Bulletin of mathematical biology**, Bulletin of Mathematical Biophysics, v. 5, p. 115–133, 1943.
- MISHRA, Sourav; CHAURASIYA, Vijay Kumar. Secure Transactions in <scp>IoT</scp> Network Using <scp>PINN</scp>-Based Intrusion Detection System and <scp>HDPoA</scp> Blockchain Protocol. **Transactions on Emerging Telecommunications Technologies**, Wiley, v. 36, n. 4, Apr. 2025. ISSN 2161-3915.
- MITCHELL, Robert; CHEN, Ing-Ray. A survey of intrusion detection techniques for cyber-physical systems. **ACM Comput. Surv.**, Association for Computing Machinery, New York, NY, USA, v. 46, n. 4, Mar. 2014. ISSN 0360-0300.
- MITCHELL, Robert; CHEN, Ing-Ray. Behavior Rule Specification-Based Intrusion Detection for Safety Critical Medical Cyber Physical Systems. **IEEE Transactions on Dependable and Secure Computing**, Institute of Electrical and Electronics Engineers (IEEE), v. 12, n. 1, p. 16–30, Jan. 2015. ISSN 1545-5971.
- MITCHELL, Thomas M. **Machine Learning**. 1. ed. USA: McGraw-Hill, Inc., 1997. ISBN 0070428077.
- MOLINA, L.C.; BELANCHE, L.; NEBOT, A. Feature selection algorithms: a survey and experimental evaluation. In: 2002 IEEE International Conference on Data Mining, 2002. Proceedings. [S.l.]: IEEE Comput. Soc.
- MONDRAGON, Jose Carlos; BRANCO, Paula; JOURDAN, Guy-Vincent; GUTIERREZ-RODRIGUEZ, Andres Eduardo; BISWAL, Rajesh Roshan. Advanced IDS: a comparative study of datasets and machine learning algorithms for network flow-based intrusion detection systems. **Applied Intelligence**, Springer Science and Business Media LLC, v. 55, n. 7, Apr. 2025. ISSN 1573-7497.
- MORRIS, Thomas; GAO, Wei. Industrial Control System Traffic Data Sets for Intrusion Detection Research. In: PROGRESS in Pattern Recognition, Image Analysis, Computer Vision, and Applications. [S.l.]: Springer International Publishing, 2014. P. 65–78. ISBN 9783319125688.
- MORRIS, Thomas H.; THORNTON, Zach; TURNIPSEED, Ian P. Industrial Control System Simulation and Data Logging for Intrusion Detection System Research. In.

MOUSTAFA, Nour. **ToN_IoT datasets**. [S.l.]: IEEE Dataport, 2019.

NEUPANE, Subash; ABLES, Jesse; ANDERSON, William; MITTAL, Sudip; RAHIMI, Shahram; BANICESCU, Ioana; SEALE, Maria. Explainable Intrusion Detection Systems (X-IDS): A Survey of Current Methods, Challenges, and Opportunities. **IEEE Access**, v. 10, p. 112392–112415, 2022.

NISSEN, Steffen. **Large Scale Reinforcement Learning using Q-SARSA(λ) and Cascading Neural Networks**. 2007. S. 264. MA thesis – University of Copenhagen, Denmark. M.Sc. Thesis.

OLIVEIRA, Afonso; COSTA, Diogo; MOREIRA, Gonalo; MARTINS, Jos ; PINTO, Sandro. **H-MBR: Hypervisor-level Memory Bandwidth Reservation for Mixed Criticality Systems**. [S.l.]: arXiv, 2025. Available from: <https://arxiv.org/abs/2502.02437>.

OTOUM, Yazan; NAYAK, Amiya. AS-IDS: Anomaly and Signature Based IDS for the Internet of Things. **Journal of Network and Systems Management**, Springer Science and Business Media LLC, v. 29, n. 3, Mar. 2021. ISSN 1573-7705.

OTT, Karl; MAHAPATRA, Rabi. Hardware Performance Counters for Embedded Software Anomaly Detection. In: 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech). [S.l.]: IEEE, Aug. 2018.

PARK, Jiho; SURABHI, Virinchi Roy; KRISHNAMURTHY, Prashanth; GARG, Siddharth; KARRI, Ramesh; KHORRAMI, Farshad. Anomaly Detection in Embedded Systems Using Power and Memory Side Channels. In: 2020 IEEE European Test Symposium (ETS). [S.l.]: IEEE, May 2020.

PRATIBHA; VASHIST, Dr. Prem Chand. A Detail Survey on Wireless Sensor Networks (WSNs) Security Issues. In.

QUINCOZES, Silvio E.; ALBUQUERQUE, C lio; PASSOS, Diego; MOSS , Daniel. ERENO: An Extensible Tool For Generating Realistic IEC-61850 Intrusion Detection Datasets. In: ANAIS Estendidos do XXII Simp sio Brasileiro de Segurana da Informa o e de Sistemas Computacionais (SBSeg Estendido 2022). [S.l.]: Sociedade Brasileira de Computa o - SBC, Sept. 2022. (SBSeg Estendido 2022).

RAISSI, M.; PERDIKARIS, P.; KARNIADAKIS, G.E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. **Journal of Computational Physics**, Elsevier BV, v. 378, p. 686–707, Feb. 2019. ISSN 0021-9991.

RASPBERRY PI FOUNDATION. **BCM2711 — Raspberry Pi Documentation**.

[S.l.: s.n.], 2025. [https:](https://www.raspberrypi.com/documentation/computers/processors.html#bcm2711)

[//www.raspberrypi.com/documentation/computers/processors.html#bcm2711](https://www.raspberrypi.com/documentation/computers/processors.html#bcm2711).

Accessed: 2025-08-04.

REAL TIME ENGINEERS LTD. **FreeRTOS - Real-Time Operating System**. [S.l.: s.n.], 2025. <https://www.freertos.org/>. Accessed: 2025-08-04.

REHMAN, Shafiq ur; ALHULAYYIL, Hisham; ALZAHIRANI, Taher; ALSAGRI, Hatoon; KHALID, Muhammad U.; GRUHN, Volker. Intrusion detection system framework for cyber-physical systems. **Egyptian Informatics Journal**, v. 30, p. 100600, 2025. ISSN 1110-8665.

SANTOS, Vinícius F.; ALBUQUERQUE, Célio; PASSOS, Diego; QUINCOZES, Silvio E.; MOSSÉ, Daniel. Assessing Machine Learning Techniques for Intrusion Detection in Cyber-Physical Systems. **Energies**, MDPI AG, v. 16, n. 16, p. 6058, Aug. 2023. ISSN 1996-1073.

SARKER, Iqbal H. Data Science and Analytics: An Overview from Data-Driven Smart Computing, Decision-Making and Applications Perspective. **SN Computer Science**, v. 2, n. 5, p. 377, July 2021. ISSN 2661-8907.

SATPATHY, Swati P; MOHANTY, Sujata; KUMAR, Rakesh. A Lightweight DoS and DDoS Attack Detection Mechanism-Based on Deep Learning. In: 2022 5th International Conference on Computational Intelligence and Networks (CINE). [S.l.: s.n.], 2022. P. 1–6.

SCHEFFEL, R. M.; FRÖHLICH, A. A. WSN Data Confidence Attribution Using Predictors. In: 2018 Eighth Latin-American Symposium on Dependable Computing (LADC). [S.l.: s.n.], 2018. P. 145–154.

SCHEFFEL, Roberto M.; FRÖHLICH, Antônio A. Increasing sensor reliability through confidence attribution. **Journal of the Brazilian Computer Society**, v. 25, n. 1, p. 13, Dec. 2019. ISSN 1678-4804.

SCHEFFEL, Roberto Milton; HOFFMANN, Jose Luis Conradi; HORSTMANN, Leonardo Passig; ARAUJO, Gustavo Medeiros de; FROHLICH, Antonio Augusto; MATSUO, Tiago Kaoru; POHLENZ, Vitor; NISHIOKA, Marcos Hisashi Napoli. Data Confidence Applied to Wind Turbine Power Curves. In: 2020 X Brazilian Symposium on Computing Systems Engineering (SBESC). [S.l.]: IEEE, Nov. 2020.

SHAHRIAR, Md Hasan; HAQUE, Nur Imtiazul; RAHMAN, Mohammad Ashiqur; ALONSO, Miguel. G-IDS: Generative Adversarial Networks Assisted Intrusion Detection System. In: 2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC). [S.l.: s.n.], 2020. P. 376–385.

SHARAFALDIN, Iman; HABIBI LASHKARI, Arash; GHORBANI, Ali A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In: PROCEEDINGS of the 4th International Conference on Information Systems Security and Privacy. [S.l.]: SCITEPRESS - Science and Technology Publications, 2018. P. 108–116.

SHARAFALDIN, Iman; LASHKARI, Arash Habibi; HAKAK, Saqib; GHORBANI, Ali A. Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy. In: 2019 International Carnahan Conference on Security Technology (ICCST). [S.l.: s.n.], 2019. P. 1–8.

SHURMAN, Mohammad; KHRAIS, Rami; YATEEM, Abdulrahman. DoS and DDoS Attack Detection Using Deep Learning and IDS. **The International Arab Journal of Information Technology**, Zarqa University, v. 17, 4A, p. 655–661, July 2020. ISSN 1683-3198.

SINGH, Shio; SINGH, M; SINGH, Dharmendra. A Survey on Network Security and Attack Defense Mechanism For Wireless Sensor Networks. v. 1, Jan. 2011.

TAORMINA, Riccardo et al. Battle of the Attack Detection Algorithms: Disclosing Cyber Attacks on Water Distribution Networks. **Journal of Water Resources Planning and Management**, American Society of Civil Engineers (ASCE), v. 144, n. 8, Aug. 2018. ISSN 1943-5452.

TAVALLAEE, Mahbod; BAGHERI, Ebrahim; LU, Wei; GHORBANI, Ali A. A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications. [S.l.: s.n.], 2009. P. 1–6.

UCI KDD ARCHIVE. **KDD Cup 1999 Data**. [S.l.: s.n.], 1999. Accessed: 2025-11-18. Available from: <https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>.

VENKATA, Sravanthi Kota; AHN, Ikkjin; JEON, Donghwan; GUPTA, Anshuman; LOUIE, Christopher; GARCIA, Saturnino; BELONGIE, Serge; TAYLOR, Michael Bedford. SD-VBS: The San Diego Vision Benchmark Suite. In: 2009 IEEE International Symposium on Workload Characterization (IISWC). [S.l.]: IEEE, Oct. 2009. P. 55–64.

VIEGAS, E.; SANTIN, A. O.; FRANÇA, A.; JASINSKI, R.; PEDRONI, V. A.; OLIVEIRA, L. S. Towards an Energy-Efficient Anomaly-Based Intrusion Detection Engine for Embedded Systems. **IEEE Transactions on Computers**, v. 66, n. 1, p. 163–177, Jan. 2017. ISSN 0018-9340.

VISALAKSHI, S.; RADHA, V. A literature review of feature selection techniques and applications: Review of feature selection in data mining. In: 2014 IEEE International Conference on Computational Intelligence and Computing Research. [S.l.]: IEEE, Dec. 2014.

WERBOS, Paul John. **Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences**. 1974. PhD thesis – Harvard University.

WEST, Richard; LI, Ye; MISSIMER, Eric; DANISH, Matthew. A Virtualized Separation Kernel for Mixed-Criticality Systems. **ACM Transactions on Computer Systems**, Association for Computing Machinery (ACM), v. 34, n. 3, p. 1–41, Sept. 2016.

WUYTS, Roel; DUCASSE, Stéphane; NIERSTRASZ, Oscar. A data-centric approach to composing embedded, real-time software components. **Journal of Systems and Software**, Elsevier BV, v. 74, n. 1, p. 25–34, Jan. 2005. ISSN 0164-1212.

XING, Wei; SHEN, Jun. Security Control of Cyber–Physical Systems under Cyber Attacks: A Survey. **Sensors**, MDPI AG, v. 24, n. 12, p. 3815, June 2024. ISSN 1424-8220.

ZADEH, Mohammad Mehdi Zeinali; SALEM, Mahmoud; KUMAR, Neeraj; CUTULENCO, Greta; FISCHMEISTER, Sebastian. SiPTA. In: PROCEEDINGS of the 14th International Conference on Embedded Software - EMSOFT 14. [S.l.]: ACM Press, 2014.

ZHANG, Junping; WANG, Fei-Yue; WANG, Kunfeng; LIN, Wei-Hua; XU, Xin; CHEN, Cheng. Data-Driven Intelligent Transportation Systems: A Survey. **IEEE Transactions on Intelligent Transportation Systems**, v. 12, n. 4, p. 1624–1639, 2011.

ZHANG, Lichen. Designing big data driven cyber physical systems based on AADL. In: 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC). [S.l.: s.n.], 2014. P. 3072–3077.

ZHONG, Meihui; LIN, Mingwei; ZHANG, Chao; XU, Zeshui. A survey on graph neural networks for intrusion detection systems: Methods, trends and challenges. **Computers & Security**, Elsevier BV, v. 141, p. 103821, June 2024. ISSN 0167-4048.

ZHOU, C.; HUANG, S.; XIONG, N.; YANG, S.; LI, H.; QIN, Y.; LI, X. Design and Analysis of Multimodel-Based Anomaly Intrusion Detection Systems in Industrial Process Automation. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, v. 45, n. 10, p. 1345–1360, Oct. 2015. ISSN 2168-2216.

ZHU, Lingzi; ZHAO, Bo; LI, Weidong; WANG, Yixuan; AN, Yang. TICPS: A trustworthy collaborative intrusion detection framework for industrial cyber–physical systems. **Ad Hoc Networks**, Elsevier BV, v. 160, p. 103517, July 2024. ISSN 1570-8705.