P4Ward: Fine-Grained Behavioral Policy Enforcement for Industrial Networks

Ina Berenice Fink*, William Köhler*, Martin Serror§, and Klaus Wehrle*
*Communication and Distributed Systems, RWTH Aachen University, Germany
§Fraunhofer FKIE, Germany
{fink, koehler, wehrle}@comsys.rwth-aachen.de · martin.serror@fkie.fraunhofer.de

***Abstract*—Industrial Control Systems (ICS) are increasingly targeted by cyber-attacks, yet often lack cryptographic protections due to legacy devices and protocols. Network-based defenses—particularly the enforcement of behavioral policies—offer an effective means of reducing the attack surface. However, such approaches must support industrial protocols and payload inspection while maintaining scalability and low-latency overhead. In this paper, we present P4Ward, a framework that leverages Manufacturer Usage Description (MUD) for scalable, device-specific policy specification and uses P4-programmable hardware switches to enforce these policies at line rate, in conjunction with authentication where feasible. We implement a proof-of-concept supporting multiple industrial protocols, including Modbus and OPC UA, and evaluate its performance and security impact. Our results show that P4Ward achieves latency comparable to industrial-grade switches while enabling precise enforcement of flexible access control rules—such as validating application layer fields (e.g., function codes) and explicitly restricting write operations to authorized endpoints.**

***Index Terms*—P4, MUD, ICS, access control, behavioral policy**

I. Introduction

Industrial Control Systems (ICS) were originally air-gapped and not designed with cybersecurity in mind, but they are increasingly interconnected and exposed to external networks. Still, running systems often operate legacy devices and protocols that lack encryption and introduce vulnerabilities exploitable by attackers. Consequently, there is an urgent need for non-intrusive security solutions that mitigate these inherent weaknesses and ensure the safe operation of ICS environments [1], [2]. One promising direction is the use of network security controls, such as firewalls and behavioral policies tailored to the unique characteristics of ICS and industrial protocols [3]. Indeed, the predictable and well-defined communication patterns of ICS [4] allow the narrow and precise definition of access control policies [5]. However, their effective use also requires performant and scalable methods for policy distribution and enforcement.

The IETF's Manufacturer Usage Description (MUD) standard [6] aims to restrict connections of Internet of Things (IoT) devices to their designated behavior as defined in device-specific MUD files. The similar characteristics and deterministic behavior of industrial devices suggest that MUD is equally promising for the use in ICS [2], which is highlighted by recent publications [7]–[10]. In particular, these works demonstrate the potential of Software-Defined Networking (SDN) to allow for flexible and scalable enforcement of MUD. However, they are based on OpenFlow [11], which limits their processing capabilities, and restricted to basic policies without consideration of industrial protocols. While P4-programmable hardware switches allow for flexible, protocol-independent, and performant policy enforcement, including (limited) payload inspection [12]–[14], respective solutions are missing.

To close this gap, we extend MUD to support industrial protocols, including application layer information (e.g., function codes), and propose **P4Ward**: a framework that combines MUD and authentication with P4-programmable hardware switches to allow for flexible enforcement at line rate, satisfying the stringent communication latency requirements in industry and critical infrastructure. We demonstrate the feasibility of our solution by implementing P4Ward on an Intel Tofino 1, with support for Modbus, OPC UA, Ethernet/IP, and GOOSE. We evaluate its effectiveness against network reconnaissance, Denial of Service (DoS), sniffing, and tampering attacks through theoretical analysis and proof-of-concept implementations.

Contributions. Our main contributions are as follows:

- We assess the characteristics and attack vectors in ICS, the potential of network-based security controls to mitigate them, and derive requirements for respective solutions in ICS.
- We extend MUD files to support industrial protocols and design P4Ward to meet the identified requirements.
- We show the P4Ward's feasibility through a proof-of-concept, and thoroughly evaluate its performance and effectiveness.

Open Science Statement. We will open-source our implementation [15] under the GPLv3 license.

II. Architecture and Security of ICS

In this section, we provide an overview of communication networks in ICS, including typical architectures, performance requirements, and protocols. We then examine prevalent security vulnerabilities and network-based security in ICS.

A. Overview of Industrial Control Systems

ICS enable the automation, monitoring, and control of industrial processes in domains such as manufacturing plants, water treatment plants, and smart grids. These systems rely heavily on communication networks to connect components and facilitate the exchange of control and status information [16].

Network Architecture. Despite differences in scale and implementation across facilities, ICS operation principles

are generally similar [16] and follow a layered architecture defined by the ISA-95 standard [17]: At the lowest layers (L0–L1) field devices such as sensors and actuators interact with control devices such as Programmable Logic Controllers (PLCs) or Intelligent Electronic Devices (IEDs) via fieldbuses, increasingly based on Ethernet. These control devices exchange data over local networks and connect to Human-Machine Interfaces (HMIs) and Supervisory Control and Data Acquisition (SCADA) systems (L2) for operator supervision. In remote deployments, Remote Terminal Units (RTUs) may relay communication to central systems via wide-area networks [18]. Higher layers (L3–L4) manage operation and business processes.

Communication Characteristics. Depending on their application, communication networks in ICS must transmit specific data while meeting stringent performance requirements [3], [19]. Unlike traditional IT networks and the Internet, ICS communication is typically time-critical and demands low latency, high reliability, and deterministic behavior. The transmitted data often consists of sensor values and function or error codes, rather than large files or multimedia content. Consequently, communication patterns are usually predictable and repetitive—such as periodic reading and writing of values or codes—and rely on specialized network protocols.

Network Protocols. Industrial network protocols are specifically designed for enabling efficient and reliable automation and control in ICS [3]. These protocols typically include well-defined code fields (e.g., to indicate read or write operations, or status and error codes), and data fields for transmitting values. However, they operate at different ICS layers. For instance, Modbus/TCP is used for communication at L0–L2, i.e., between PLCs, RTUs, HMIs, and SCADA systems, while OPC UA primarily operates at L2–L4, but is also supported by modern PLCs and RTUs.

Protocol Security. ICS were originally designed as closed, isolated networks, and security was overlooked in early protocol design. With increasing digitalization, many ICS protocols were adapted to support Ethernet and IP-based communication, thereby increasing their exposure to security threats [3]. While newer protocols—or updated versions of existing ones—support encryption and authentication, they are often not deployed in practice due to compatibility issues with legacy devices or increased communication overhead, which compromises stringent latency requirements. Moreover, these security features do not necessarily prevent lower-layer attacks, such as reconnaissance and DoS, requiring complementary security measures. Finally, the effectiveness of protocol-based security relies on proper implementation and configuration, which is frequently lacking in real-world deployments [20].

To understand the concrete security risks faced by ICS, we subsequently examine their most common attack vectors.

B. Attack Vectors in ICS

Attackers typically begin with reconnaissance to gather information about the target network and its topology, which they can later exploit to disrupt services, extract sensitive data, or tamper with system functionality.

Network Reconnaissance Attacks. Reconnaissance commonly involves IP and port scanning. Tools such as ARP-scan and ICMP ECHO requests (e.g., via Nmap [21]) are used to discover active hosts and their IP addresses. TCP SYN and UDP scans reveal open ports and the services running on target devices. Although reconnaissance itself has no immediate consequences, it lays the foundation for more impactful follow-up attacks.

DoS. DoS attacks can target any layer of the network stack but are most often executed at the transport layer or above. Attackers may use flooding techniques with TCP, UDP, or HTTP to exhaust bandwidth or processing resources, disrupting device or service availability. Vulnerabilities in specific protocols can also be exploited—for example, malformed packets that trigger crashes. Historical examples include zero-length messages that crash DNP3 protocol drivers [22], or malicious Modbus function codes (e.g., 0x5A) that cause PLCs to freeze [23]. DoS attacks can result in production downtime, equipment damage, or even pose risks to human safety [24], [25].

Sniffing and Tampering. To guess passwords and gain access to devices, attackers may use dictionary or brute-force techniques. Man-In-The-Middle (MITM) attacks—such as MAC, ARP, or DNS spoofing—enable them to eavesdrop on communications and intercept credentials. With captured or guessed credentials, attackers can gain unauthorized control over devices and manipulate industrial processes. Without credentials, MITM attacks still enable disruption or manipulation of communication by injecting, modifying, or replaying packets to alter function codes or sensor data. Such tampering can have severe consequences [25]—as demonstrated in the 2021 Florida water treatment plant incident, where attackers temporarily altered the chemical composition of the water supply [26].

Due to the prevalence of legacy devices and protocols lacking built-in security in ICS, alternative mitigation mechanisms are urgently needed. In particular, defending against DoS and tampering attacks requires security solutions that support industrial protocols—enabling, e.g., validation of message lengths, data values, or function codes. Network-based security controls offer a promising approach to address these challenges [27].

C. Network-based Security Controls

Network-based security controls are promising and widely used for complementing and retrofitting security in ICS as they can be deployed independently of legacy field and control devices. Common examples include firewalls, access control, and Intrusion Detection and Prevention Systems (IDPS) [28]. Firewalls restrict external traffic at network boundaries, while access control can limit both user and device communication to predefined policies, e.g., based on their roles, identities, or functional behavior. In ICS, behavioral policies are especially effective [5] as they reflect the deterministic and predictable communication patterns of field and control devices. Thus, they help reduce the attack surface without impairing functionality. In addition, IDPS offer deeper inspection and anomaly detection but may introduce unacceptable latency or false positives.

State-of-the-Art. While software-based solutions often lack the required performance for ICS, dedicated commercial appliances like Belden's Tofino Security Appliance already offer protocol-specific access control [29]–[32]. However, they are limited in throughput (1000 pps depending on the protocol [30], [31]), require per-connection deployment, and rely on manual rule configuration, making them less scalable. Also, protocol support is currently limited to GOOSE, Modbus, and Ethernet/IP [29]. This reveals a gap in flexible (ideally protocol-independent), efficient solutions for scalable access control enforcement in ICS—an issue we address in the remainder of this paper.

Takeaway: *ICS depend on specialized devices and protocols to operate critical processes. Yet, the limited deployment of secure protocols leaves them exposed to potentially severe attacks. While network-based controls offer mitigation, current solutions fall short in performance and flexibility.*

III. Towards Network-based Security with Programmable Switches and MUD

Subsequently, we present the considered attacker model, and derive three key requirements for network-based security deployments in ICS. We then examine the potential of programmable switches and MUD, and review related work.

A. Attacker Model

Besides increasing exposure to the Internet and other external systems, ICS are sometimes deployed in environments with Bring-Your-Own-Device (BYOD) policies. Additionally, insiders with access to critical components may act with malicious intent or fall victim to social engineering attacks such as phishing. Given these circumstances, we assume that attackers can gain initial access to the network and connected devices, either remotely or through physical means. Based on the Dolev-Yao attacker model [33], such adversaries are capable of intercepting, modifying, and injecting messages within the compromised environment, enabling both passive and active attacks on the local network. While the Dolev-Yao model assumes that attackers cannot break modern cryptographic methods, we additionally consider that attackers may specifically target deployed systems and protocols, exploiting known and unknown vulnerabilities alike.

B. Requirements

The following principles are recommended to strengthen ICS security [16], [19]:

- **Isolation**: Segment networks to separate services and reduce cross-segment access.
- **Zero Trust**: Deny all communication by default unless explicitly permitted by policy.
- **Least Privilege**: Grant users and services only the permissions necessary for their tasks.
- **Least Route**: Limit network access of nodes strictly to what is required for their function.
- **Defense in Depth**: Layer different security controls to enhance protection.

Combining above recommendations with the significance of industrial protocols and the stringent performance requirements of ICS (cf. Sec. II-A), industrial network-based security solutions should fulfill:

RQ1) Fine-grained filtering of communication patterns and payloads, including industrial protocol support.
RQ2) Scalable deployment and policy management.
RQ3) Transparent, low-latency operation to preserve system availability.

Recent works [34], [35] suggest that P4-programmable, off-the-shelf switches can offer flexible and efficient access control. In parallel, IETF's MUD standard provides a structured way to define and manage behavioral policies. We explore both approaches as potential enablers for practical ICS network security, then investigate related work.

C. The Potential of Programmable Switches and MUD

Programmable Switches. Switches with programmable Application Specific Integrated Circuits (ASICs), such as Intel's Tofino series [36], enable flexible, high-speed packet processing using the P4 language [37]. P4's architecture [38] comprises a programmable data plane (on the ASIC) that processes packets, while a control plane (on a CPU) configures the data plane. Thanks to TCAM, which allows for fast lookups, programmable ASICs are well-suited for high-performance packet matching, such as access control enforcement. However, to maintain deterministic processing, P4 introduces strict constraints (e.g., no loops or division [38]), and Tofino imposes additional limitations compared to virtual P4 switches [39] to sustain line-rate speed. Nonetheless, Tofino has proven capable of advanced tasks, including limited payload inspection and stateful connection tracking [14], [40].

MUD. MUD was standardized by the Internet Engineering Task Force (IETF) [6] and defines device-specific, whitelist-based behavioral profiles (MUD files). MUD files are fetched from servers via corresponding MUD URLs and contain Access Control Entries (ACEs) to restrict network communication of IoT devices to only what is necessary for device function. While native support is limited to basic protocols (IP, TCP, UDP, ICMP), MUD allows for extensions to accommodate custom use cases. Notably, MUD does not prescribe how policies should be enforced, making it compatible with SDN as demonstrated by related work (cf. Sec. III-D).

D. Related Work

We review existing efforts in tailoring network-based security to ICS and enforcing MUD, then discuss current gaps.

1) Retrofittable Network-based Security for ICS: Several researchers recognized the security gaps in ICS and implemented specific solutions. Ndonda et al. [41] implemented a two-staged Modbus-aware Intrusion Detection System (IDS) in P4, while Hu et al. [34] focused on DNP3 inspection in combination with hashing, encryption, and filtering. These works highlight the importance and potential of considering industrial protocol payload in security applications. However, they lack policy-based, protocol-independent access control and

rely on virtual BMv2 switches, which are not compatible with P4-programmable hardware switches (cf. Sec. III-C). Other works address security via behavior-based IDS [42] or hard-coded filters [43], but lack flexibility and, as implemented in software, performance. Last, different authentication and access control schemes were developed for smart grids [44]–[46] and resource-constrained devices [47] as well as other non-ICS-specific P4-based solutions for authentication [48] and link protection [49]. However, all of them lack support for behavioral policies and industrial protocols.

2) Manufacturer Usage Description: MUD has been widely explored for IoT security, with several works demonstrating scalable enforcement via programmable switches [12], [35], [50]–[53]. In the industrial domain, Krishnan et al. [7] developed attestation and authentication of industrial devices with MUD. García et al. [8] proposed secure retrieval of MUD files for resource-restricted devices in industrial networks, while Matheu et al. [9] added enforcement of channel protection and through MUD files and authentication. García et al. [10] further allowed the configuration of security settings and vulnerabilities. However, these works rely on OpenFlow and do not address industrial protocols or payloads.

Discussion. While prior work demonstrates the potential of programmable switches for securing ICS and efficiently enforcing MUD files, no solution yet combines both to enforce behavioral policies on programmable hardware. In particular, it is still unclear whether complex MUD files that account for industrial protocols and payloads can be enforced on programmable hardware switches. In the following, we explore how such an approach could look and perform in practice.

Takeaway: *Solutions for behavioral access control in ICS must fulfill stringent performance and functional requirements, which are not met by existing approaches.*

IV. Scalable Access Control for ICS using P4-programmable Switches and MUD

We propose combining the flexible and performant processing capabilities of P4-programmable switches with the MUD standard to create **P4Ward**—a scalable framework for fine-grained access control in ICS, which satisfies RQ1-RQ3. Below, we overview P4Ward's key components and features.

A. Overview

First, we extend MUD files with access control rules that allow matching against industrial protocol headers and significant payload fields (cf. RQ1). We then design P4Ward, a framework that automatically manages and processes MUD files (cf. RQ2) and enforces the rules on the data plane of P4-programmable switches (cf. RQ1, RQ3). To follow the *defense in depth* paradigm (cf. Sec. III-B) and minimize the attack surface, P4Ward also includes device authentication and enforcement of secure protocol configurations, e.g., of OPC UA. By integrating both into P4Ward, we can also reflect authentication and encryption in the ACEs of our MUD files—e.g., allowing only authenticated and encrypted communication between capable endpoints. Intrusion detection is out of the scope for our design but could be integrated into the control plane in future work.

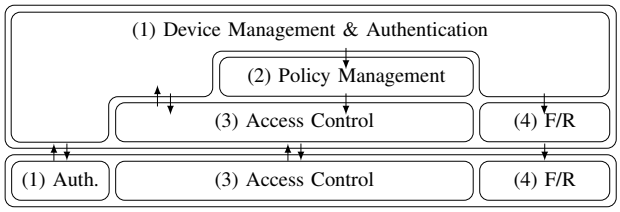

Fig. 1: Modular Architecture of P4Ward

P4Ward Architecture. Due to P4's architecture, we divide P4Ward into data and control plane modules (cf. Sec. III-C), assigning time-critical functionality to the data plane to meet RQ3. Overall, P4Ward integrates four modules:

1. **Device Management and Authentication:** Tracks (manually) registered devices, and authenticates and re-authenticates them periodically if supported by the device.
2. **Policy Management:** Parses MUD files to extract device information and ACEs for management and access control.
3. **Access Control:** Validates and filters incoming packets.
4. **Forwarding and Routing:** Handles packet forwarding within and routing across networks.

Subsequently, we describe the individual modules in detail.

B. Device Management and Authentication

P4Ward enforces a *zero trust* approach, i.e., all endpoints are denied access to the network by default. To gain access, devices must either authenticate themselves or, to support legacy devices, be manually registered by an administrator. The MUD URL is then either provided by the authentication server or statically configured by the administrator during registration. In either case, the MUD URL is passed to the policy management module to fetch the MUD file from a local server and parse it.

Device Management. Device management operates on the control plane and tracks registered (i.e., whitelisted) devices, including their MAC and IP addresses, and hardware ports. It additionally assigns each device a set of identifiers derived from its MUD URL and MUD file, i.e., unique device, group, manufacturer, and model IDs, which are used for abstract addressing in MUD ACEs. The device information remains static from the point of authentication/registration and is used to configure the access control module to validate source and destination information in packets, restricting spoofing attacks.

Authentication. P4Ward enables authentication via standard protocols, e.g., EAP, supporting security features in modern devices and protocols. This allows tight integration with the enforcement of MUD files, which may use authentication as a validity feature. While EAP-based authentication architectures typically include an authenticator that collects authenticating data—e.g., using EAPOL—and sends it to an authentication server for validation [54], our design integrates the authenticator into the switch. Consequently, to avoid blocking authentication, our access control module must support the EAPOL protocol and permit EAPOL exchange between endpoints and authenticator. The server can be integrated as well or

hosted externally, serving multiple switches. Implementing standard cryptographic algorithms like AES-256 on the data plane of modern programmable hardware switches remains underexplored and poses challenges [55] due to processing restrictions (cf. Sec. III-C). Thus, complex methods like EAP-TLS are not trivial on the data plane and likely require control plane or external server implementation. However, simpler mechanisms, e.g., using pre-calculated hashes such as EAP-OTP [56], may be feasible on the data plane and offer a practical compromise for ICS environments needing low latency.

Guest Devices. We assume that local devices are well known, permanently deployed, and statically configured, making them easy to whitelist. However, to support BYOD policies (e.g., in less critical network segments), P4Ward also supports unknown guest devices, which would otherwise be blocked by default. Administrators can define a default MUD file with restrictive policies (e.g., communication limited to specific subnets) to be applied automatically to guest devices. To reference guest devices in MUD files of registered devices, they can also be assigned with a shared guest group ID. External endpoints are not considered guest devices and are referenced directly by IP address or domain name in the MUD files of local devices to allow valid communication.

C. Policy Management

The policy management runs on the control plane and parses MUD files to extract device information and ACEs. Extracted device information is returned to the device management module, while the ACEs are passed to the access control modules. Our extended MUD files support two types of ACEs:

- **Connection-based ACEs** specify which connections are generally allowed. MUD allows endpoint specification via abstract identifiers (e.g., manufacturer IDs), MAC, or IP addresses. At the transport layer, MUD supports TCP/UDP port matching and can restrict the direction of TCP connection initiation. Extensions also allow defining flow rate limits (e.g., packets per second or burst size).
- **Protocol-based ACEs** enabled via MUD extensions, impose further restrictions based on protocol metadata and payloads—e.g., allowing only specific TCP flags or function codes. Furthermore, they may specify the use of encryption for capable protocols, e.g., (D)TLS, for which an official MUD extension exists [57], or OPC UA.

D. Access Control

To enforce *least privilege* and *least route* principles, every packet must be validated against device information and MUD ACEs. However, this must not introduce noticeable latency in ICS, which could disrupt time-sensitive operations. To address this, the access control module uses the control plane to generate match-action entries and install them on the data plane for fast filtering, following a three-step process:

1) **Device Validation:** Ensures the source and destination of a packet belong to authenticated devices. Validates MAC and IP addresses and hardware port to prevent spoofing. Per-device rate limiting is also applied.
2) **Connection Validation:** Matches source and destination information and connection initiation direction against the ACEs from the MUD file.
3) **Protocol Validation:** Checks protocol-specific attributes such as message and field lengths, handshake metadata, or function codes to enhance security, block invalid access, and mitigate known attacks.

E. Forwarding and Routing

The forwarding and routing module processes packets that have passed access control and ensures they reach their destination with minimal delay. Again, the control plane creates match-action rules that are installed in data plane lookup tables, enabling line-rate processing. Since ICS environments utilize L2 and L3 protocols, P4Ward supports local MAC-based forwarding and IP-based routing between networks. ARP is also integrated to resolve IP addresses to MAC addresses.

Takeaway: *We propose extending MUD with ACEs that specify valid values of industrial protocol fields and introduce P4Ward, a framework for managing and enforcing MUD files on the data plane of programmable switches.*

V. Proof-of-Concept Implementation

To demonstrate the feasibility of P4Ward and enable practical evaluation, we developed a proof-of-concept [15] targeting the Intel Tofino 1. We implemented the data plane in P4 and the control plane in Python3. Although Intel has discontinued production of its Tofino ASICs, they have open-sourced its architecture and development tools [58]. Furthermore, Intel's Tofino remains the gold standard in programmable switching hardware, making it a strong platform for research and prototyping. However, the modular design of P4Ward can be easily adapted to other programmable switch architectures.

Device and policy management are largely handled in the control plane, while forwarding and routing in the data plane pose few challenges. Therefore, we omit these aspects in the following and instead focus on the implementation of access control and authentication. We then discuss current limitations.

A. Implementing Access Control and Authentication

The most demanding aspects of our access control module involve its support for custom industrial protocols and payload field validation in the data plane. To date, we have implemented support for Modbus, OPC UA, Ethernet/IP, and GOOSE, along with corresponding MUD extensions. We also implemented a partially data plane-based authentication with EAP-OTP to demonstrate the feasibility and advantages of tightly coupling authentication with P4Ward.

1) Access Control: We detail the implementation of device, connection, and protocol validation, as well as string parsing.

Device Validation. We use match-action tables to verify source and destination information, distinguishing between local (MAC-based) and external (IP-based) destinations. To enforce rate limits, we employ Intel Tofino's two-rate three-color meter, which drops packets on threshold excession.

Connection Validation. Additional match-action tables validate source and destination addresses, ports, protocol, and (for TCP)

the initiation direction. To block packets from connections with invalid initiation direction, we use register-based Bloom filters to track active and valid TCP connections. TCP packets lacking the SYN flag are checked against these filters and dropped if the connection is unrecognized. Bloom filters are cleared through the control plane when connections close via FIN flags or timeouts, i.e., operations that are not time-critical.

Protocol Validation. Protocols up to the transport layer are parsed in the ingress. If applications use only default or hardcoded TCP/UDP port numbers, the corresponding application layer protocol can also be parsed in the ingress. However, to support greater flexibility, we use custom port numbers defined in the MUD ACEs. To this end, we identify the corresponding protocol in the ingress and parse the protocol fields in the egress. Consequently, validation of application layer fields must also be performed in the egress. To achieve this, flags for valid behaviors (e.g., read-only operations) are set in the ingress, as specified by the ACEs. These flags are transferred via metadata headers to the egress, where they are used to validate the application layer protocol fields. Since flags consume valuable space, we group multiple behaviors and assign one flag per group. However, traditional MUD files lack application layer protocol definitions, resulting in no flags being set in the ingress, which by default leads to failed matches and packet drops. Therefore, we also implemented a relaxed variant based on blacklisting, i.e., only invalid values are defined, while all others are accepted by default.

String Parsing. Many application layer protocols use text-based fields that are challenging to parse in the data plane [14]. For example, we need to parse the OPC UA security policy field to enforce specific security requirements. To do so, we parse all possible strings using a step-by-step approach: we begin with the shortest expected string and progress through longer ones. All parsed strings are hashed and compared against a list of allowed values using a simple table lookup.

2) Authentication: The authentication module is largely implemented in the control plane, as we have to keep track of sessions, generate challenges, and register authenticated devices, which we could not realize on the data plane of our Intel Tofino 1. However, we included EAP-OTP as an example of an authentication method that can be partially implemented in the data plane to speed up the authentication process. In this scheme, One-Time Password (OTP) chains are derived from a device's password using SHA-1, SHA-2, and SHA-3 hash functions. The device, using the same password, generates identical OTPs. Precomputing and storing OTPs on the data plane allows the data plane to validate challenge–response pairs without involving the control plane, thereby minimizing latency. Only the final authentication result is forwarded to the control plane for device registration.

B. Current Limitations

Despite its capabilities, our proof-of-concept has limitations:

- **Parsing limitations**: Only simple, fixed-length headers or payload fields are currently supported. Complex strings or nested structures cannot yet be parsed.
- **Matching**: While P4 supports exact, ternary, and range matching, our implementation does not currently support range matching due to its high storage demands.
- **Authentication support**: Advanced authentication mechanisms are not feasible on the data plane, and offloading them to external systems may introduce unacceptable latency for real-time ICS requirements.
- **Capacity**: Due to memory restrictions, our current implementation supports a maximum of 512 registered devices, 2048 ACEs, and 65k precomputed OTPs.
- **Protocol Validation**: Metadata from ingress to egress is limited to 16 bits, restricting validation to 16 flags (i.e., 16 distinct values or behavior groups).

We expect that future programmable hardware will increase storage and processing capabilities, enabling more advanced implementations. Nevertheless, our proof-of-concept demonstrates the feasibility of implementing fine-grained and performant access control for ICS on programmable switches.

Takeaway: *Despite challenges due to processing and hardware constraints, P4Ward runs successfully on Intel Tofino 1, proving its feasibility.*

VI. Performance and Security Evaluation

We conduct a thorough performance evaluation of our proof-of-concept, then evaluate its effectiveness to mitigate the attacks presented in Sec. II-B theoretically and practically.

A. Performance Evaluation

We assess three aspects: packet processing, MUD file handling, and authentication.

1) Processing Speed: Intel Tofino 1 offers up to 3.2 Tbps throughput, which our implementation matches when authentication is disabled, as all core processing is done in the data plane. To assess the viability of programmable switches in industrial settings, we compare Tofino 1's latency with that of a dedicated industrial-grade switch from a reputable manufacturer. Since the industrial device lacks equivalent access control functionality to that provided by P4Ward, we compare the Round-Trip-Time (RTT) between two hosts connected to both switches using basic forwarding and P4Ward on the Tofino.

Results. The industrial switch achieves an average RTT of 0.26 ms across 200 measurements, while Tofino 1 achieves around 0.43 ms—both for simple forwarding and P4Ward. Given the minimal difference and industrial tolerance for millisecond latencies [59], the results confirm the viability of programmable switches in industrial use cases.

2) Installation of MUD files: The control plane retrieves and parses MUD files, then installs match-action entries in the data plane and potentially disables old ones to apply updates. We measured the latency of installing/disabling policies derived from MUD files containing between 2 and 2000 ACEs.

Results. As shown in Fig. 2, installation and removal scales nearly linearly with the number of ACEs. Downscaled, installing one ACE takes between 3.2 ms to 3.3 ms; disabling one takes between 2.2 ms to 2.3 ms. Thus, 100 ACEs can be installed in around 330 ms, which is likely enough to cover

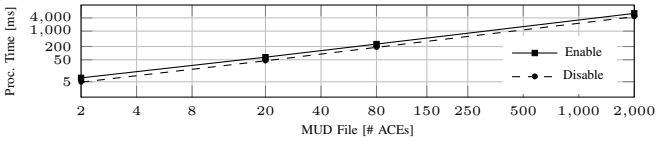

Fig. 2: Processing time of enabling/disabling MUD files depending on their number of ACEs (with logarithmic scales).

a device's communication patterns. Furthermore, installation generally takes place upon network join of a device, i.e., before time-critical operations begin. New ACEs from updated MUD files are enforced before disabling outdated ones, avoiding gaps, while changes that stem from major functional or architectural updates would disrupt operations regardless of P4Ward.

3) Authentication: We evaluated our EAP-OTP implementation using SHA1, SHA2, and SHA3 with 50 OTPs per authentication and 40 OTPs pre-stored on the data plane.

Results. The results are provided in Tab. I and show that the initial authentication takes 165 ms to 170 ms depending on the hash algorithm. Re-authentication takes only around 17.9 ms with pre-generated OTPs, but up to 150 ms if computation of new OTPs is required. In time-sensitive systems, this may still be too slow and should be omitted—leaving only the initial authentication (ideally via stronger protocols like EAP-TLS) to occur before process engagement.

Our performance evaluation highlights P4Ward's ability to meet the strict performance requirements of ICS. Next, we examine its effectiveness.

B. Security Evaluation

In worst-case scenarios, attackers compromise legitimate devices allowed to perform attack-specific communication patterns. Then, P4Ward reaches its functional limitations. However, in all other cases, tightly defined MUD files allow P4Ward to reliably mitigate attacks as demonstrated below. P4Ward blocks devices without MUD file by default, but we assume the support of guest devices with default MUD files—i.e., some restricted network access—for our evaluation.

Setup. We used the Intel Tofino 1 running P4Ward, a virtualized L2 switch, three hosts (D1–D3), and an attacker device (AD). D1–D3 represent legitimate devices while AD is a guest or compromised legitimate device. All devices were governed by MUD files. We connected both switches to each other, and D2 and D3 to the Tofino 1 (cf. Fig. 3). AD and D1 were directly connected through the L2 switch, sharing a network segment with D2, while D3 was isolated. The connection between AD and D1 was never filtered and served as a baseline.

In the following, we provide theoretical analyses followed by practical proof-of-concept evaluations of P4Ward's effectiveness against the attacks presented in Sec. II-B.

TABLE I: EAP-OTP authentication and re-authentication latencies with pre-calculated OTPs [ms].

Algorithm	Auth. (SHA1)	Auth. (SHA2)	Auth. (SHA3)	Re-Auth. (SHA2) (pre-calc.)	Re-Auth. (SHA2) (fresh)
EAP-OTP	170	168	170	17.9	150

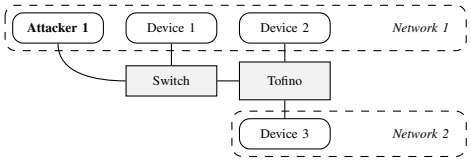

Fig. 3: Setup used for security evaluation.

1) Network Reconnaissance: Completely preventing network reconnaissance is not feasible when protocols like ARP or ICMP are required for operation. However, P4Ward limits reconnaissance by restricting communication partners (by IDs and IP/MAC addresses) and services (by ports and protocols), i.e., scans only reveal devices and services that attackers are already authorized to access instead of the whole network.

Proof-of-Concept. We evaluated the effectiveness of P4Ward using IP- and ARP-based scans via Nmap [21] within the following scenarios: (i) P4Ward disabled, allowing all traffic to/from any endpoint, (ii) only IPv4 traffic allowed, and (iii) IPv4 traffic and ARP responses only allowed between D1-D3; ARP requests allowed to/from any endpoint. We display the observed results, i.e., whether the attacker received responses from D1-D3, in Tab. II. Blocking ARP traffic (policy (ii)) prevented MAC address discovery, halting local communication, but affected also D1 and D2. Under policy (iii), AD's scans failed (except for D1) even if AD was not restricted itself, while pre-defined communication between D1-D3 was maintained. Overall, the results confirm P4Ward's ability to enforce *least privilege/route* (cf. III-B) policies that significantly reduce reconnaissance success.

2) Volumetric DoS: Extended MUD files allow to define rate limits for each connection. This feature can protect devices from high-volume traffic generated by a single attacker device. However, Distributed DoS (DDoS) attacks, where multiple devices send malicious traffic simultaneously, are not mitigated if not exceeding the per-connection rate limits. An additional collective rate limit can be applied to a service's total incoming traffic, helping to prevent resource exhaustion of the device and allowing non-targeted services to remain operational. However, collective limits also throttle traffic from legitimate devices, which may be unacceptable in some scenarios.

Proof-of-concept. We setup a Modbus server on D2 and extended the MUD specification to include rate limits for AD, D1, and D2. We evaluated their effectiveness by monitoring the stability of a Modbus connection between D1 and D2 during a DoS attack conducted against the Modbus server on

TABLE II: Success of IP- and ARP-based reconnaissance attacks for three scenarios with and without restricting MUD ACEs. Wildcards are marked with *, i.e., no restrictions apply.

Scenario	P4Ward	Allowing	Between	Attack	D1	D2	D3
i)	✗	*	*	IP	✓	✓	✓
				ARP	✓	✓	-
ii)	✓	IPv4	*	IP	✓	✗	✓
				ARP	✓	✗	-
iii)	✓	IPv4	D1, D2, D3	IP	✓	✗	✗
		ARP resp.	D1, D2, D3	ARP	✓	✗	-
		ARP req.	*				

TABLE III: Connection state between D1 and D2 during a DoS attacks against D2 with different flood rates and limits.

Scenario	Rate-limit	Flood rate [pps] 1k	2k	4k	8k	20k
i)	D2 (in)	✓	✓	✓	packet loss	conn. drop
ii)	D2 (in), AD (out)	✓	✓	✓	✓	✓

D2 via `hping3` [60] with TCP SYN floods in two scenarios: i) a collective rate limit applied to all incoming traffic of the Modbus server on D2, and ii) additionally, a collective rate limit applied to all outgoing traffic of AD. In both scenarios, the limits were set to 10k pps with a burst size of 40 packets. The DoS attack was executed with increasing intensity from 1k to 20k packets per second sent from AD to D2. The results are presented in Tab. III. In scenario (i), the rate limit stopped the DoS attack but the legitimate Modbus connection between D1 and D2 was affected as well since the overall limit of the Modbus server was exceeded—first dropping packets, then breaking the connection. However, in scenario (ii), the legitimate connection remained stable, since the limit on outgoing traffic of AD prevented exhaustion of D2's limit. From this, we conclude that rate limiting should also be applied to the outgoing traffic of devices, ideally using lower limits than applied to the incoming traffic of their communication partners. This helps preserving legitimate communications even under attack conditions.

3) Sniffing and Spoofing: P4Ward mitigates brute-force-based sniffing by limiting the communication partners and enforcing rate limits similar to its defenses against reconnaissance and volumetric DoS attacks. In addition, P4Ward can effectively prevent spoofing, e.g., ARP spoofing, by enforcing behavioral policies and using static MAC address mappings.

Proof-of-concept. We perform a typical MITM attack with ARP spoofing, executed via `arpspoof` [61] running on AD. We evaluated two scenarios with and without P4Ward to restrict ARP responses to connections between D1 and D2: (i) spoofing between D1 and D2, and (ii) spoofing between D1 and the Intel Tofino switch. The results are shown in Tab. IV. As expected, ARP spoofing was always successful for D1, but also for D2 when P4Ward was not active, i.e., D1 and D2 had spoofed ARP entries with AD's MAC address. However, when active, P4Ward successfully blocked ARP spoofing for D2. Also, spoofing targeting the Intel Tofino switch consistently failed due to P4Ward's reliance on static MAC address assignment (cf. IV-B), which even prevented traffic redirection when spoofing between D1 and D2 was successful. Although this also disrupts communication between D1 and D2, it is usually preferable to silent compromise and easier to notice.

TABLE IV: Success of spoofing attacks with and without P4Wardthat allowed ARP responses only between D1 and D2.

Scenario	P4Ward	D1	D2	Tofino
i)	✗	✓	✓	-
	✓	✓	✗	-
ii)	✗	✓	-	✗
	✓	✓	-	✗

4) Functional DoS & Tampering: P4Ward can validate message and field lengths to block malformed packets, and restrict header and payload fields to pre-defined values, e.g., preventing misuse of harmful function codes that trigger software bugs, or manipulation of sensor values.

Proof-of-concept. We configured a Modbus server on D2 and sent a write operation with a respective value from AD in two scenarios: (i): all function codes allowed, and (ii): only read-only function codes allowed from AD to D2. In scenario (i), AD successfully modified the server's register values. In scenario (ii), however, P4Ward immediately dropped the packet with the unauthorized function code, resulting in a connection failure, and confirming P4Ward's ability to enforce narrow application-specific security policies in ICS.

C. Discussion

Our evaluation demonstrates that P4Ward effectively reduces the attack surface in ICS, while offering adequate performance. Combined with sensitive and device-specific MUD files, P4Ward enforces *zero trust*, *least privilege*, *least route*, and *defense in depth* while supporting logical device *isolation*. However, P4Ward is no panacea and design limitations remain: *i*) attacks within allowed patterns cannot be blocked, *ii*) DDoS attacks within per-device rate limits may still succeed, and *iii*) spoofing attacks may disrupt communication even if sniffing and manipulation is prevented. Ultimately, effectiveness depends on the precise design of MUD files and handling of guest devices: overly permissive policies weaken protection. Also, current hardware (e.g., memory) constraints restrict the complexity of MUD ACEs (Sec. V), but can be overcome with future hardware advancements.

Takeaway: *Our evaluations confirm P4Ward's ability to enforce fine-grained, layered access control with low latency, offering a practical foundation for securing ICS.*

VII. Conclusion

Network-based security—particularly, (behavioral) access control—is critical for reducing the attack surface of ICS. It can either complement encryption-based mechanisms or serve as an effective alternative when legacy devices and protocols lack cryptographic support. However, there remains a gap in performant and flexible access control solutions that are aware of industrial protocols and payloads. To address this gap, we propose P4Ward, a framework that leverages programmable hardware switches to enforce extended MUD policies. Our proof-of-concept supports multiple industrial protocols, and we demonstrate its feasibility through evaluation. While the current implementation comes with limitations such as restricted parsing capabilities and scalability constraints, these are likely to be mitigated with future hardware advances. Crucially, P4Ward enables efficient and fine-grained access control directly on the data plane of off-the-shelf hardware, allowing enforcement of tailored behavioral policies without compromising the strict latency and throughput requirements of ICS. Therefore, P4Ward provides a practical and cost-efficient path to minimizing attack surfaces in operational networks.

References

[1] E. D. Knapp, "3 - Industrial Cybersecurity History and Trends," in *Industrial Network Security (Third Edition)* (E. D. Knapp, ed.), pp. 45–64, Syngress, Third Edition ed., 2024.

[2] M. Serror, S. Hack, M. Henze, M. Schuba, and K. Wehrle, "Challenges and opportunities in securing the industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 5, pp. 2985–2996, 2021.

[3] E. D. Knapp, "6 - Industrial Network Protocols," in *Industrial Network Security (Third Edition)* (E. D. Knapp, ed.), pp. 129–179, Syngress, Third Edition ed., 2024.

[4] S. V. B. Rakas, M. D. Stojanović, and J. D. Marković-Petrović, "A review of research work on network-based scada intrusion detection systems," *IEEE Access*, vol. 8, pp. 93083–93108, 2020.

[5] E. D. Knapp, "12 - Exception, Anomaly, and Threat Detection," in *Industrial Network Security (Third Edition)* (E. D. Knapp, ed.), pp. 383–408, Syngress, Third Edition ed., 2024.

[6] E. Lear, R. Droms, and D. Romascanu, "Manufacturer Usage Description Specification." RFC 8520, Mar. 2019.

[7] P. Krishnan, K. Jain, K. Achuthan, and R. Buyya, "Software-defined security-by-contract for blockchain-enabled mud-aware industrial iot edge networks," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 10, pp. 7068–7076, 2022.

[8] S. N. M. García, A. Molina Zarca, J. L. Hernández-Ramos, J. B. Bernabé, and A. S. Gómez, "Enforcing behavioral profiles through software-defined networks in the industrial internet of things," *Applied Sciences*, vol. 9, no. 21, 2019.

[9] S. N. Matheu, A. Robles Enciso, A. Molina Zarca, D. Garcia-Carrillo, J. L. Hernández-Ramos, J. Bernal Bernabe, and A. F. Skarmeta, "Security architecture for defining and enforcing security profiles in dlt/sdn-based iot systems," *Sensors*, vol. 20, no. 7, 2020.

[10] S. N. Matheu García, A. Sánchez-Cabrera, E. Schiavone, and A. Skarmeta, "Integrating the manufacturer usage description standard in the modelling of cyber–physical systems," *Comput. Stand. Interfaces*, vol. 87, oct 2023.

[11] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, pp. 69–74, Mar. 2008.

[12] H. S. A, H. Kothapalli, S. Lahoti, K. Kataoka, and P. Tammana, "Iot mud enforcement in the edge cloud using programmable switch," in *Proceedings of the ACM SIGCOMM Workshop on Formal Foundations and Security of Programmable Network Infrastructures*, FFSPIN '22, (New York, NY, USA), pp. 1–7, Association for Computing Machinery, 2022.

[13] V. Venugopal, J. Alves-Foss, and S. G. Ravindrababu, "Use of an SDN switch in support of NIST ICS security recommendations and least privilege networking," in *Proceedings of the Fifth Annual Industrial Control System Security (ICSS) Workshop*, (New York, NY, USA), ACM, Dec. 2019.

[14] A. Mazloum, A. AlSabeh, E. Kfoury, and J. Crichigno, "Security applications in P4: Implementation and lessons learned," *Comput. Netw.*, vol. 257, p. 111011, Feb. 2025.

[15] I. B. Fink, W. Koehler, M. Serror, and K. Wehrle, "Prototype Implementation of P4Ward." https://github.com/COMSYS/P4Ward, 2025.

[16] E. D. Knapp, "2 - About Industrial Networks," in *Industrial Network Security (Third Edition)* (E. D. Knapp, ed.), pp. 11–43, Syngress, Third Edition ed., 2024.

[17] "Ansi/isa-95.00.01-2010 (iec 62264-1 mod) enterprise-control system integration - part 1: Models and terminology."

[18] E. D. Knapp, "4 - Introduction to Industrial Control Systems and Operations," in *Industrial Network Security (Third Edition)* (E. D. Knapp, ed.), pp. 65–90, Syngress, Third Edition ed., 2024.

[19] E. D. Knapp, "5 - Industrial Network Design and Architecture," in *Industrial Network Security (Third Edition)* (E. D. Knapp, ed.), pp. 91–128, Syngress, Third Edition ed., 2024.

[20] M. Dahlmanns, J. Lohmöller, I. B. Fink, J. Pennekamp, K. Wehrle, and M. Henze, "Easing the conscience with opc ua: An internet-wide study on insecure deployments," in *Proceedings of the ACM Internet Measurement Conference*, IMC '20, (New York, NY, USA), pp. 101–110, Association for Computing Machinery, 2020.

[21] "Nmap." https://nmap.org, 2025 (accessed April 20, 2025).

[22] CISA, "ICSA-14-289-01: IOServer Resource Exhaustion Vulnerability." https://www.cisa.gov/news-events/ics-advisories/icsa-14-289-01, 2018 (accessed April 20, 2025).

[23] CISA, "ICSA-17-054-03: Schneider Electric Modicon M340 PLC (Update A)." https://www.cisa.gov/news-events/ics-advisories/icsa-17-054-03, 2019 (accessed April 20, 2025).

[24] N. Tripathi and N. Hubballi, "Application layer denial-of-service attacks and defense mechanisms: A survey," *ACM Comput. Surv.*, vol. 54, May 2021.

[25] M. Lehto, "Cyber-Attacks Against Critical Infrastructure," in *Cyber Security: Critical Infrastructure Protection* (M. Lehto and P. Neittaanmäki, eds.), pp. 3–42, Cham: Springer International Publishing, 2022.

[26] A. Greenberg, "A Hacker Tried to Poison a Florida City's Water Supply, Officials Say." https://www.wired.com/story/oldsmar-florida-water-utility-hack/, February 2021 (accessed April 20, 2025).

[27] E. D. Knapp, "11 - Implementing Security and Access Controls," in *Industrial Network Security (Third Edition)* (E. D. Knapp, ed.), pp. 331–381, Syngress, Third Edition ed., 2024.

[28] E. D. Knapp, "9 - Establishing Zones and Conduits," in *Industrial Network Security (Third Edition)* (E. D. Knapp, ed.), pp. 293–314, Syngress, Third Edition ed., 2024.

[29] Belden, "Tofino Xenon Industrial Security Appliance." https://www.belden.com/products/industrial-networking-cybersecurity/cybersecurity/firewalls/tofino-xenon, (accessed April 20, 2025).

[30] Belden, "Tofino™ EtherNet/IP Enforcer LSM." https://assets.belden.com/m/22bc2efd658a1b4b/original/Tofino-NetConnect-LSM_PB1091_INET_TOF_514_A_AG_051114LONG_LR_Original_54400.pdf, (accessed April 20, 2025).

[31] Belden, "Tofino™ Modbus TCP Enforcer LSM." https://assets.belden.com/m/6a0f66be102e114f/original/DS-MBT-LSM-v6.pdf, (accessed April 20, 2025).

[32] Belden, "Tofino Xenon GOOSE Enforcer Loadable Security Module (LSM)." https://assets.belden.com/m/615c861b760fe146/original/GOOSE-ENFORCER-LSM_PB00157_INIT_TOF_1217_ENG_LowRes_Original_122801.pdf, (accessed April 20, 2025).

[33] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 198–208, 1983.

[34] Z. Hu, H. Lin, L. Waind, Y. Qu, G. Chen, and D. Jin, "Industrial network protocol security enhancement using programmable switches," in *2023 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pp. 1–7, IEEE, Oct. 2023.

[35] S. A. Harish, S. Datta, H. Kothapalli, P. Tammana, A. Basuki, K. Kataoka, S. Manickam, U. Venkanna, and Y.-W. Chong, "Scaling iot mud enforcement using programmable data planes," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–9, 2023.

[36] Intel, "Intel® Tofino™." https://www.intel.de/content/www/de/de/products/details/ethernet/programmable-ethernet-switch/tofino-series.html, (accessed April 20, 2025).

[37] P4, "P4 Open Source Programming Language." https://p4.org, (accessed April 20, 2025).

[38] P4, "P4 Language and Related Specifications." https://p4.org/specs/, (accessed April 20, 2025).

[39] "Tools or Resources to Convert P4 BMv2 Code to TNA Architecture." https://forum.p4.org/t/tools-or-resources-to-convert-p4-bmv2-code-to-tna-architecture/1214/3, 2025 (accessed April 20, 2025).

[40] I. B. Fink, I. Kunze, P. Hein, J. Pennekamp, B. Standaert, K. Wehrle, and J. Rüth, "Advancing network monitoring with packet-level records and selective flow aggregation," in *Proceedings of the 2025 IEEE/IFIP Network Operations and Management Symposium (NOMS '25)*, 2025.

[41] G. K. Ndonda and R. Sadre, "A two-level intrusion detection system for industrial control system networks using P4," in *Electronic Workshops in Computing*, pp. 31–40, BCS Learning & Development, Aug. 2018.

[42] M. T. Khan, D. Serpanos, and H. Shrobe, "ARMET: Behavior-based secure and resilient industrial control systems," *Proc. IEEE Inst. Electr. Electron. Eng.*, vol. 106, pp. 129–143, Jan. 2018.

[43] G. Corbò, C. Foglietta, C. Palazzo, and S. Panzieri, "Smart behavioural filter for industrial internet of things: A security extension for PLC," *Mob. Netw. Appl.*, vol. 23, pp. 809–816, May 2017.

[44] S. Ruj and A. Nayak, "A decentralized security framework for data aggregation and access control in smart grids," *IEEE Trans. Smart Grid*, vol. 4, pp. 196–205, Mar. 2013.

[45] D. Rosic, U. Novak, and S. Vukmirovic, "Role-based access control model supporting regional division in smart grid system," in *2013 Fifth International Conference on Computational Intelligence, Communication Systems and Networks*, pp. 197–201, IEEE, June 2013.

[46] G. Karmakar, R. Naha, R. Shah, J. Kamruzzaman, and R. Das, "Software-defined access control in smart grids," in *2023 33rd Australasian Universities Power Engineering Conference (AUPEC)*, pp. 1–6, IEEE, Sept. 2023.

[47] P. P. Pereira, J. Eliasson, and J. Delsing, "An authentication and access control framework for CoAP-based internet of things," in *IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society*, pp. 5293–5299, IEEE, Oct. 2014.

[48] A. Bhattacharya, R. Rana, S. Datta, and Venkanna, "P4-sKnock: A two level host authentication and access control mechanism in P4 based SDN," in *2022 27th Asia Pacific Conference on Communications (APCC)*, pp. 278–283, IEEE, Oct. 2022.

[49] F. Hauser, M. Schmidt, M. Haberle, and M. Menth, "P4-MACsec: Dynamic topology monitoring and data layer protection with MACsec in P4-based SDN," *IEEE Access*, vol. 8, pp. 58845–58858, 2020.

[50] A. Hamza, H. H. Gharakheili, and V. Sivaraman, "Combining mud policies with sdn for iot intrusion detection," in *Proceedings of the 2018 Workshop on IoT Security and Privacy*, IoT S&P '18, (New York, NY, USA), pp. 1–7, Association for Computing Machinery, 2018.

[51] M. Al-Shaboti, I. Welch, A. Chen, and M. A. Mahmood, "Towards Secure Smart Home IoT: Manufacturer and User Network Access Control Framework," in *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, pp. 892–899, 2018.

[52] M. Ranganathan, "Soft mud: Implementing manufacturer usage descriptions on openflow sdn switches," 2019.

[53] I. B. Fink, M. Serror, and K. Wehrle, "Demons: Extended manufacturer usage description to restrain malicious smartphone apps," in *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, pp. 463–470, 2021.

[54] W. Stallings, *Cryptography and Network Security: Principles and Practice*. Pearson, 2022.

[55] X. Chen, "Implementing AES Encryption on Programmable Switches via Scrambled Lookup Tables," in *Proceedings of the Workshop on Secure Programmable Network Infrastructure*, SPIN '20, (New York, NY, USA), pp. 8–14, Association for Computing Machinery, 2020.

[56] J. Vollbrecht, J. D. Carlson, L. Blunk, D. B. D. Aboba, and H. Levkowetz, "Extensible Authentication Protocol (EAP)." RFC 3748, June 2004.

[57] T. Reddy.K, D. Wing, and B. Anderson, "Manufacturer Usage Description (MUD) (D)TLS Profiles for IoT Devices," Internet-Draft draft-ietf-opsawg-mud-tls-12, Internet Engineering Task Force, Jan. 2023. Work in Progress.

[58] P4, "Intel's Tofino P4 Software is Now Open Source." https://p4.org/intels-tofino-p4-software-is-now-open-source/, (accessed April 20, 2025).

[59] J. Hiller, M. Henze, M. Serror, E. Wagner, J. N. Richter, and K. Wehrle, "Secure low latency communication for constrained industrial iot scenarios," in *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, pp. 614–622, 2018.

[60] "hping." https://www.hping.org, 2025 (accessed April 20, 2025).

[61] "Dsniff arpspoof." https://www.kali.org/tools/dsniff/, 2025 (accessed April 20, 2025).