Technology
Arts Sciences
TH Köln

Security of Network Protocols in Industrial Automation

M.Eng Automation and IT

Authors: Group 1

Mohamed Shattat	11367046	(HTTPS Security)
Sugin Sukumaran	11368759	(OPC UA Security)
Cheng-Chih Weng	11389611	(SNMP Security)

TH Köln – University of Applied Sciences
November 23, 2025

Contents

Chapter 1

Analysis of HTTPS Security and Its Use in Industrial Automation

1.1 Methodology

This chapter is based on a structured review of primary standards and academic sources relevant to HTTPS security and industrial communication. Core specifications, including RFC 2616, RFC 8446, and RFC 2818, were used to analyse the behaviour of HTTP, TLS, and HTTPS. In addition, industrial research such as TiCS, IIoT protocol comparisons, and the HTTPS security systematic literature review were evaluated to assess practical limitations in operational technology (OT). The methodology consists of three steps: (1) analysing protocol foundations (HTTP, TLS, HTTPS), (2) reviewing industrial use-cases and constraints, and (3) evaluating attack vectors and standards relevant to secure deployment. This ensures both scientific and industrial perspectives are represented.

1.2 Hypertext Transfer Protocol (HTTP)

The Hypertext Transfer Protocol (HTTP) is an application-layer client–server protocol that forms the foundation of data exchange on the World Wide Web. Originally intended for transmitting hypermedia documents, HTTP has evolved into a general-purpose communication protocol widely used in

REST APIs, cloud services, and industrial human–machine interfaces (HMIs) [1, 2].

HTTP follows a stateless request–response model. A client opens a reliable TCP connection—typically on port 80—and sends a request consisting of a request line, headers, and an optional message body. The server responds with a status line (e.g., `HTTP/1.1 200 OK`), headers, and the requested resource. Since HTTP is stateless, no session information is maintained unless explicitly implemented through mechanisms such as cookies or authentication tokens [2].

HTTP defines several request methods used to operate on resources identified by Uniform Resource Identifiers (URIs), including `GET` for retrieval, `POST` for data submission, `PUT` for updating, and `DELETE` for removal. These well-defined semantics have contributed to HTTP's widespread adoption across both IT and industrial automation.

A significant limitation of HTTP is the absence of built-in security. Messages are transmitted in plaintext, making them vulnerable to interception, tampering, and impersonation. This is unacceptable in industrial environments where manipulated data may jeopardise safety or operations. To address this, HTTP is commonly combined with Transport Layer Security (TLS), forming HTTPS [3].

1.3 Transport Layer Security (TLS)

TLS is the cryptographic protocol that ensures confidentiality, integrity, and authentication for HTTPS connections. Positioned between the application and transport layers, it prevents unauthorised reading or modification of HTTP messages. Modern deployments rely on TLS 1.3 [4], which improves both performance and security compared to TLS 1.2.

1.3.1 Security Goals of TLS

TLS provides three core security properties [4]:

- **Confidentiality:** Achieved through strong symmetric encryption.
- **Integrity:** Ensured by message authentication codes or authenticated encryption.

- **Authentication:** Provided through X.509 certificates [5].

TLS 1.3 enforces *Perfect Forward Secrecy* (PFS), meaning that even if a server's long-term private key is compromised, past encrypted sessions cannot be decrypted.

1.3.2 TLS 1.3 Handshake Overview

The TLS 1.3 handshake reduces latency by using a one-round-trip (1-RTT) key-agreement process [4]. A simplified handshake includes:

1. **ClientHello:** Proposes cipher suites and sends an ephemeral Diffie–Hellman key share.
2. **ServerHello:** Selects a cipher suite, provides its key share, and sends a certificate.
3. **Key Derivation:** Both parties derive symmetric encryption keys.
4. **Encrypted Data:** Application data is encrypted for the remainder of the session.

1.3.3 Cryptographic Algorithms and Ciphersuites

TLS uses bundles of algorithms known as *ciphersuites*. TLS 1.3 simplifies and modernises available options by removing insecure algorithms and enforcing strong AEAD encryption [4, 6]. Table 1.1 summarises key differences.

1.3.4 Limitations of TLS 1.3

Despite its robustness, TLS 1.3 does not mitigate all security risks:

- **No DNS protection:** DNS occurs before TLS [7].
- **No protection against application-layer attacks:** XSS, SQL injection, and API vulnerabilities remain [6].
- **No DoS resilience:** TLS cannot prevent availability attacks.
- **Configuration dependency:** Incorrect certificate validation or hostname checks weaken security.

Table 1.1: Comparison of TLS 1.2 and TLS 1.3 Features

Feature	**TLS 1.2**	**TLS 1.3**
Handshake Latency	2–RTT	1–RTT
Forward Secrecy	Optional	Mandatory
Key Exchange Methods	RSA, DH, ECDHE	Only ECDHE
Allowed Suites	Many (incl. insecure)	Only secure AEAD
Deprecated Algorithms	CBC, RC4, SHA–1 allowed	All removed
Handshake Encryption	Partial	Nearly complete
Misconfiguration Risk	High	Lower (simplified)
Zero-RTT Data	Not supported	Supported (with risks)

- **Endpoint compromise:** TLS secures data in transit only.

These limitations highlight the need for additional security controls in OT environments.

1.4 HTTPS: HTTP Over TLS

HTTPS encapsulates HTTP messages inside a TLS-encrypted channel, ensuring confidentiality, integrity, and authentication [3]. Instead of sending plaintext HTTP requests, the client first initiates a TLS handshake on port 443. After key agreement, HTTP messages become encrypted application data [4].

During the handshake, the client validates the server's X.509 certificate, ensuring that it matches the hostname and is issued by a trusted authority. This process prevents man-in-the-middle attacks. Crucially, HTTPS preserves HTTP semantics—only the transport layer is secured—allowing existing HMIs, REST interfaces, and industrial dashboards to be secured without changing application logic.

1.5 Industrial Context of HTTPS

1.5.1 Use of HTTPS in Industrial Automation

Industrial automation increasingly integrates IT-style communication. Frameworks such as Time-Constrained Services (TiCS) use SOAP-based web services to expose PLC functions over HTTP. These SOAP services can be securely transported using HTTPS, making TLS an essential component when operational or control data is exchanged [8].

1.5.2 Deployment Challenges in OT Environments

Deploying HTTPS inside OT environments introduces unique challenges:

Performance Overhead. TLS handshakes, encryption, and certificate validation introduce CPU load and latency. On PLCs with strict scan cycles, even small delays can affect timing guarantees [8].

Network Timing and Determinism. TLS adds jitter and encryption overhead, conflicting with real-time Ethernet protocols such as PROFINET IRT and EtherCAT [9].

Certificate and Key Management. Devices often operate for decades and may lack secure storage or support for modern certificate validation. Legacy equipment may only support outdated TLS versions [10].

Complex Trust Architecture. Industrial networks are typically flat and long-lived, making certificate management and TLS negotiation reliability difficult to maintain.

1.6 HTTPS Threats and Security Challenges

1.6.1 Threat Types

The four classical threat categories—interception, modification, fabrication, and interruption—map directly to confidentiality, integrity, authenticity, and availability. Table 1.2 summarises HTTPS protections.

Table 1.2: Main Types of Security Threats and HTTPS Protection Capabilities

Threat	**Meaning**	**Affects**	**HTTPS Protection**
Interception	Eavesdropping	Confidentiality	Yes
Modification	Tampering	Integrity	Yes
Fabrication	Forged data / impersonation	Authenticity	Partial
Interruption	DoS / failures	Availability	No

1.6.2 Threats Not Fully Addressed by HTTPS

While HTTPS provides strong confidentiality, integrity, and server authentication, it does not mitigate all security risks. HTTPS protects the transport channel using TLS, but it does not secure underlying name resolution, device configuration, application logic, or embedded web interfaces. These limitations are especially relevant in industrial operational technology (OT), where devices often use outdated stacks, lack certificate management, and operate inside flat networks.

Key limitations are:

- **DNS Spoofing:** DNS resolution occurs before the TLS handshake; poisoned DNS can redirect clients to malicious servers.
- **SSL Stripping:** Without strict HTTPS enforcement, attackers can downgrade connections to HTTP.
- **Weak or Misconfigured Certificates:** Self-signed, expired, or reused certificates break authentication.
- **Mixed Content:** HTTPS pages loading insecure HTTP resources allow content injection.
- **Session Weaknesses:** Cookies without `Secure`/`HttpOnly` can be hijacked.
- **Outdated TLS Versions:** Legacy devices supporting TLS 1.0/1.1 remain vulnerable.
- **Compromised Certificate Authorities:** A malicious or compromised CA can issue trusted certificates.

These gaps motivate additional controls such as DNS security, certificate lifecycle management, secure coding practices, and segmentation using industrial firewalls.

Table 1.3: Summary of HTTPS-Resistant Threats and Industrial Impact

Threat Type	**Why HTTPS Cannot Protect**	**Industrial Impact**
DNS Spoofing	DNS occurs before TLS; HTTPS trusts whatever IP is resolved	PLCs / HMIs redirected to malicious servers
SSL Stripping	No protection without HSTS; HTTPS can be blocked	Industrial devices fall back to HTTP; traffic exposed
Weak Certificates	Misconfiguration breaks HTTPS authentication	MITM inside OT networks; fake servers trusted
Mixed Content	Insecure HTTP elements break HTTPS security model	Injected scripts manipulate HMI/SCADA data
Session Hijacking	HTTPS does not secure cookies; flags required	Hijacked operator sessions and unauthorized control
XSS / Malware Delivery	HTTPS does not validate content	Malware distributed over HTTPS from compromised SCADA
Outdated TLS Versions	Legacy devices use weak crypto	Downgrade attacks; decryptable industrial traffic
Compromised CA	HTTPS trust chain can be abused	Fake “trusted” servers impersonate vendors/cloud

1.6.3 Real Case Studies of HTTPS Failures in OT and IT

To illustrate the practical limitations of HTTPS, this section summarises six real case studies where HTTPS or the TLS trust model failed in practice. These real-world events emphasise that secure transport is insufficient without secure DNS, robust certificate management, and hardened endpoints.

DigiNotar Certificate Authority Breach (2011)

The DigiNotar incident demonstrated that a compromised Certificate Authority (CA) can undermine the entire HTTPS ecosystem. Attackers generated more than 500 fraudulent certificates, including for `google.com`, enabling large-scale MITM attacks [11]. Because browsers fully trusted DigiNotar, victims saw a valid HTTPS lock even while traffic was intercepted. This incident highlights the fragility of PKI trust, especially relevant for OT devices that do not inspect certificate issuers.

Equifax Breach and Expired TLS Certificates (2017)

An expired TLS inspection certificate left encrypted traffic unmonitored for 19 months, allowing attackers to exploit an application flaw without detection [12]. This incident demonstrates that improper certificate lifecycle management can disable security monitoring and is a severe risk for industrial gateways and remote access servers that rely on HTTPS.

Siemens PLC Web Server Vulnerability (CVE-2020-15782)

Siemens S7-1200/1500 PLCs were found vulnerable to a memory access flaw that could be exploited even when the connection used HTTPS [13]. HTTPS secured the channel but did not protect against device-level vulnerabilities, demonstrating that encrypted transport does not guarantee secure embedded logic.

Schneider Electric Modicon HTTPS Authentication Bypass

Multiple Modicon PLCs suffered from flaws where the HTTPS web interface could be accessed without proper authentication [14]. Even with HTTPS enabled, weak authentication allowed attackers to change configuration and logic. This highlights that HTTPS does not protect against insecure application design.

DNS Spoofing Enabling HTTPS Bypass

Real attacks in Brazil and other regions showed that DNS poisoning can redirect victims to malicious HTTPS servers with fraudulent certificates [15, 16]. Because DNS occurs before TLS, HTTPS cannot verify the correctness

of the IP address. Industrial HMIs relying on local DNS servers are especially vulnerable.

TLS Downgrade Attacks on IoT and Industrial Devices

Studies have shown that many IoT and OT devices accept obsolete TLS versions, allowing attackers to downgrade connections and intercept traffic [17, 18]. This is common in PLCs and gateways that use outdated firmware with legacy TLS libraries.

1.7 Comparison with Other Industrial Protocols

Industrial communication systems rely on different protocols depending on their timing constraints, reliability requirements, message formats, and deployment environments. Although HTTPS provides strong security based on TLS, it is not originally designed for real-time automation. This section compares HTTPS to three widely used industrial protocols: OPC UA, MQTT over TLS, and PROFINET, with a focus on architectural differences, security models, and implications for use in industrial environments. The analysis is supported by the comparative findings of Jaloudi [9].

1.7.1 HTTPS vs. OPC UA Security

OPC UA is one of the most advanced industrial communication standards, designed specifically for automation, SCADA, MES, and Industry 4.0 systems. While HTTPS provides secure transport, OPC UA provides *application-layer security*, including:

- Object-level access control,
- Method-level authentication,
- Built-in role-based permissions,
- Support for asymmetric encryption, signing, and user tokens.

Unlike HTTPS, which is stateless and document-oriented, OPC UA defines a stateful session model and a rich semantic information structure. According

to Jaloudi's comparison table , OPC UA (listed within 'industrial protocols") uses binary encoding for high performance, supports publish–subscribe and client–server, and maintains millisecond-level latency. Thus, HTTPS can securely transport data, but it cannot replace OPC UA for industrial semantics, deterministic timing, or server-managed address spaces.

1.7.2 HTTPS vs. MQTT over TLS

MQTT is a lightweight publish–subscribe protocol optimized for IoT and cloud integration. Jaloudi [9] highlights several characteristics:

- Event-based asynchronous communication,
- Very small overhead (UTF-8 topic encoding),
- Designed for constrained devices,
- Native support for extremely low bandwidth links.

MQTT over TLS (port 8883) provides confidentiality and integrity similar to HTTPS, but differs fundamentally in messaging behaviour:

- HTTPS = synchronous request–response (one-to-one),
- MQTT = asynchronous publish–subscribe (one-to-many).

In industrial contexts, MQTT is typically used for:

- Telemetry from PLCs,
- Cloud dashboards,
- Remote monitoring and alarms.

Jaloudi's analysis (pp. 9–11) shows that MQTT has:

- Higher overhead than MODBUS but lower than HTTP,
- Lower real-time determinism compared to SCADA protocols,
- Excellent scalability for IIoT integration.

Compared to HTTPS, MQTT is more suitable when data must be pushed periodically or upon events—not polled via HTTP requests.

1.7.3 HTTPS vs. PROFINET Security

PROFINET is a real-time industrial Ethernet protocol widely used in Siemens automation environments. Unlike HTTPS, PROFINET is designed for:

- Hard real-time (IRT) cycle times (<1 ms),
- Deterministic control of drives, sensors, actuators,
- Low-jitter cyclic communication.

Security in PROFINET traditionally depended on network segmentation (VLANs, firewalls, cell protection), but recent versions integrate:

- Certificate-based device authentication (Security Class 3),
- Integrity protection of messages,
- Secure commissioning and configuration.

Compared to PROFINET:

- HTTPS is secure but not deterministic,
- HTTPS sessions introduce high latency (tens of ms),
- TLS handshake makes HTTPS unsuitable for cyclic control.

Therefore, HTTPS is not a replacement for field-level protocols like PROFINET, but may complement them for:

- Secure remote diagnostics,
- Device web interfaces,
- Firmware management,
- Cloud connectivity.

Table 1.4: Summary of HTTPS Compared to Other Industrial Protocols

Protocol	**Strengths Compared to HTTPS**	**Limitations Compared to HTTPS**
OPC UA	Rich industrial semantics; built-in security at application layer; role-based access; low latency; deterministic sessions	More complex; heavier implementation; requires full OPC UA stack
MQTT-over-TLS	Lightweight; efficient publish–subscribe model; ideal for telemetry; low bandwidth usage	Not deterministic; lacks industrial semantics; depends on broker infrastructure
PROFINET	Hard real-time capability; deterministic Ethernet; microsecond-level jitter control	Not suited for Internet use; security depends on extensions and segmentation; not TLS-based
HTTPS (baseline)	Strong TLS encryption; widely supported; firewall-friendly; easy deployment; platform-agnostic	Not real-time; no industrial semantics; stateless; higher latency; handshake overhead

Summary of Comparison

1.8 Regulations and Standards

1.8.1 EU Cyber Resilience Act (CRA)

The CRA mandates that products with digital elements, including industrial devices using HTTPS, must implement secure-by-design principles and maintain vulnerability management throughout their lifecycle.

1.8.2 NIS2 Directive

NIS2 strengthens cybersecurity obligations for operators of essential and important entities, requiring robust risk management and incident reporting.

1.8.3 IEC 62443

IEC 62443 provides comprehensive security guidelines for industrial automation, including authentication, secure communication, and certificate management.

1.8.4 IETF Standards

HTTPS behaviour is defined through core IETF specifications including RFC 2818, RFC 8446, and RFC 6125.

1.9 Summary

HTTPS provides strong confidentiality, integrity, and authentication through TLS 1.3, making it suitable for secure industrial dashboards and cloud connectivity. However, OT environments introduce challenges such as strict timing constraints, legacy devices, certificate management complexity, and application-layer vulnerabilities that HTTPS alone cannot address. Additional security measures—network segmentation, certificate lifecycle management, DNS security, and secure application design—remain essential.

Chapter 2

OPC UA Literature Review and Security Analysis

2.1 Literature Review

2.1.1 OPC UA (Unified Architecture)

In the mid-1990s, the OPC Foundation introduced its initial Microsoft COM-based Data Access (DA) specification, establishing a standardised client–server communication framework supporting read, write, and subscription operations. Additional specifications such as Alarm and Events (AE) and Historical Data Access (HDA) followed, quickly becoming industry standards. However, reliance on Microsoft COM/DCOM introduced several limitations:

- **Platform dependency:** classic OPC was tied to Windows platforms.
- **Internet communication limitations:** DCOM packets were complex and difficult to route across firewalls.
- **Fragmentation:** DA, AE, and HDA servers required separate components, complicating integration.

To address these shortcomings, the OPC Foundation developed XML-DA, offering platform-independent communication via Web Services. However, XML-DA suffered from large XML overhead, poor real-time performance, and weak security.

OPC Unified Architecture (OPC UA) was introduced as a complete redesign [19]. It provides:

- platform independence (binary + XML encodings),
- a unified information model,
- service-oriented architecture (SOA),
- built-in security,
- cross-platform support including embedded devices, Linux, and Windows,
- abstract service definitions decoupled from implementation.

OPC UA addresses earlier limitations by integrating all OPC capabilities—DA, AE, HDA—into a single framework with enhanced modelling and communication flexibility [20].

2.2 OPC UA Security Model

OPC UA supports communication from field devices to enterprise IT. Because it handles sensitive operations, it is subject to threats such as spoofing, replay, flooding, profiling, session hijacking, and credential theft [19].

The security model is divided into three layers [21]:

- **Application Layer:** user authorization, authentication, and management of secure OPC UA sessions.
- **Communication Layer:** establishes a Secure Channel, providing confidentiality (encryption), integrity (signatures), and application-level authentication via X.509 certificates.
- **Transport Layer:** transmits already-protected messages across TCP or HTTPS.

These layers work together: a session is created at the application level, protected via a Secure Channel, and transported securely.

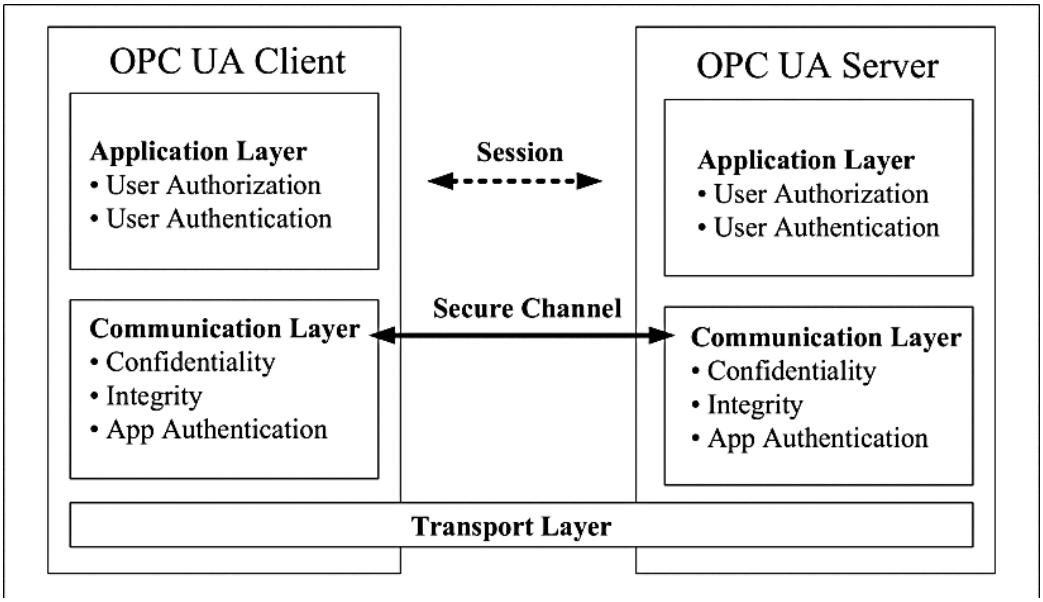

Figure 2.1: OPC UA Security Model

2.3 OPC UA Threat Model

The OPC UA threat model includes three dimensions:

- Threat sources,
- Threats, impacts, and countermeasures,
- OPC UA threats in Industry 4.0 network environments.

2.3.1 Threat Sources

- **Internal user:** legitimate users who may misuse access or accidentally compromise security.
- **Lack of security-by-design:** default passwords, weak configurations, or manufacturer backdoors.
- **External attacker:** unauthorised entities aiming to disrupt operations or steal data.

2.3.2 Threats, Impacts and Countermeasures

OPC UA faces threats including:

1. eavesdropping,
2. session hijacking,
3. rogue servers,
4. credential compromise,
5. message spoofing,
6. message alteration,
7. malformed messages,
8. flooding (DoS),
9. replay attacks,
10. repudiation,
11. server profiling.

Type \ Objective	Confidentiality	Integrity	Availability	Authentication	Authorisation	Auditability	Non-repudiation
Direct	*1, 2, 3, 4*	*2, 5, 6, 7*	*2, 3, 7, 8*	*2, 3*	*2, 3, 4, 5, 6, 7, 9*	*2, 3*	*10*
Indirect	*11*						

Figure 2.2: Security objectives impacted by OPC UA threats

Mitigation strategies include encryption, signatures, secure session management, authentication, auditing, and restricting pre-authentication services.

2.3.3 OPC UA Threats in Industry 4.0 Networks

Figure 2.3 shows a modern Industry 4.0 architecture using OPC UA, fieldbuses, gateways, PLCs, ERP systems, and firewalls. Shop-floor networks contain diverse devices, some supporting OPC UA natively and others connected via OPC UA gateways.

Threats include:

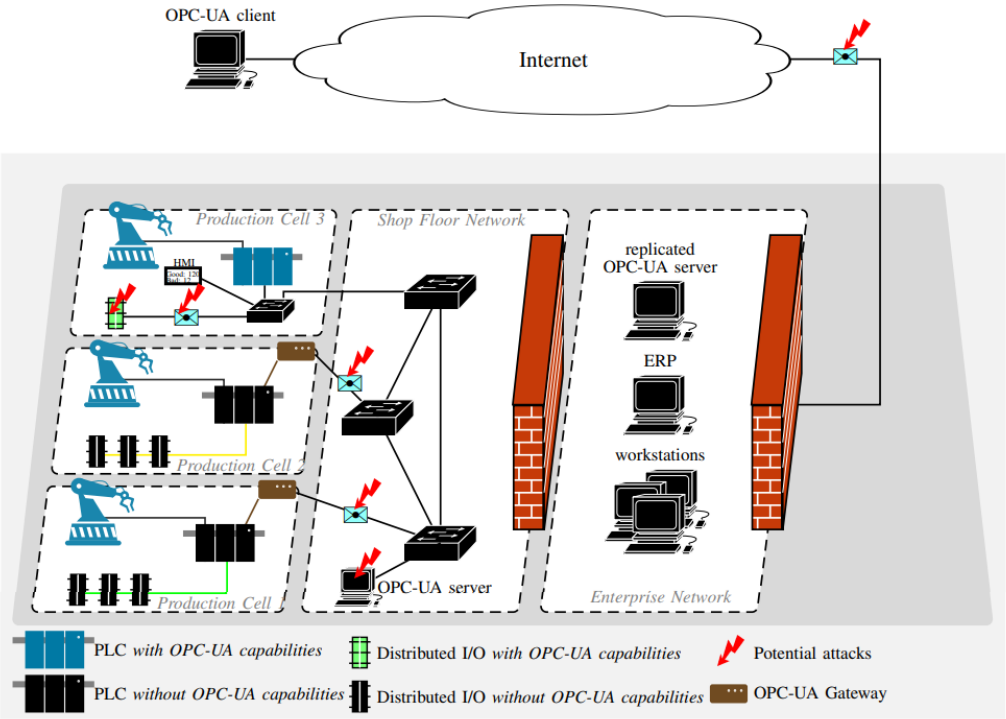

Figure 2.3: OPC UA Threat Model in an Industry 4.0 environment

- eavesdropping on unencrypted traffic,
- message flooding affecting PLCs or gateways,
- MITM attacks via ARP spoofing,
- rogue OPC UA clients/server impersonation,
- modifying payloads or subscriptions,
- replaying captured packets.

OPC UA's built-in security reduces risks, but layered defences such as segmentation, firewalls, and DPI remain necessary.

2.3.4 Example: MITM Attack on OPC UA

When OPC UA uses **SecurityMode: None**, an attacker can:

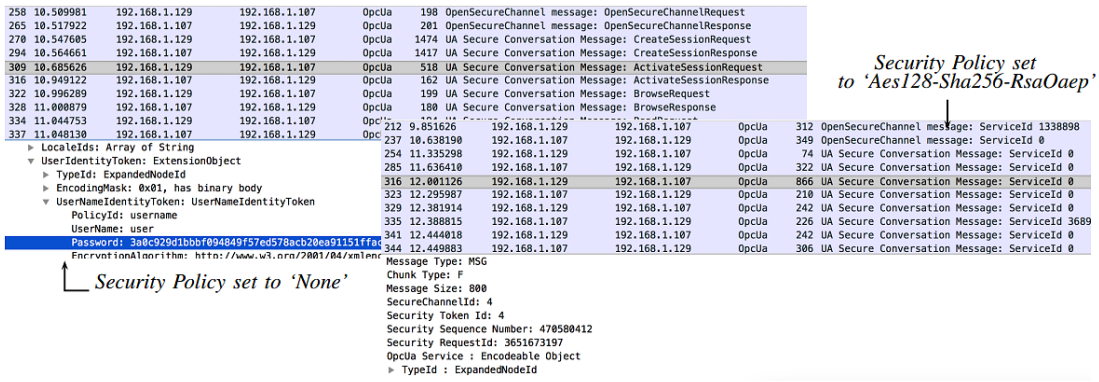

Figure 2.4: Wireshark capture under "None" vs. "Aes128-Sha256-RsaOaep" security policy

- inspect user identity tokens,
- read service requests,
- monitor address-space browsing.

With **Aes128-Sha256-RsaOaep**, all payloads become encrypted, blocking visibility of usernames, node browsing, and service calls.

2.3.5 OPC UA and DoS Attacks

OPC UA servers allocate resources to each client. Attackers may exhaust:

- CPU,
- memory,
- bandwidth,
- service queue capacity.

Built-in defences include:

- restricting pre-authentication services,
- enforcing session tokens,

- rate limiting,
- certificate-based access control.

These reduce attack surface but cannot fully eliminate DoS risks.

2.4 Conclusion

OPC UA provides secure, interoperable communication for Industry 4.0 systems. However, secure deployment requires careful configuration, certificate management, and complementary protection mechanisms such as segmentation and DPI firewalls. As Time-Sensitive Networking (TSN) evolves, new standards (e.g., IEEE 802.1AEcg, IEEE 802.1Qci) will further strengthen industrial communication security [22].

Chapter 3

Scientific and Industrial Background of SNMP

3.1 Simple Network Management Protocol (SNMP)

SNMP stands for simple network management protocol. It is a protocol used for monitoring, managing, and configuring network devices such as routers, switches, and servers. Within SNMP, there are two types of entities: Managers and Agents. Managers are often referred to as Network Management Systems (NMSs)—software applications that communicate with devices running an SNMP agent [23, 24].

There are two primary types of messages.

- A **GET** request retrieves information from an agent's Management Information Base (MIB)—a structured database of monitored objects identified by Object Identifiers (OIDs) [23].
- A **SET** request modifies the values of the OIDs on a monitored device [23].

Another common type of message is a TRAP, which is an unsolicited alert sent by an agent to the manager when a problem or urgent events occur [23].

SNMP currently has three versions. SNMP Version 1 (SNMPv1) is the original version and uses a community string for security. This string acts as a "password" for accessing data. Since the string are transmitted in plain text, anyone who intercepts it can access the sensitive device information, which

is insecure. In SNMP Version 2 (SNMPv2c) more functions and performance improvements are introduced. SNMP Version 3 (SNMPv3) is the latest version of SNMP. It supports authentication and encryption, enhancing security [24].

3.2 Industrial Context of SNMP

Today, the installation and administration of industrial automation systems are becoming more and more complex. We can discover such complexity in an example provided by ABB. These systems contain many heterogeneous devices like (process) controllers, motors, small machines, routers, switches, servers and client PCs, connected by Ethernet and TCP/IP [25]. These systems must be running 24/7 because they control crucial production processes.

With the increasing complexity of industrial automation systems, managing their networks has become more challenging. It is essential to effectively monitor and configure all connected devices to ensure stable operation [25]. Traditionally, management has been done manually by engineers. However, manual management is no longer sufficient due to rapidly evolving technologies nowadays.

Under these circumstances, network management protocols such as SNMP plays a vital role. In ABB's 800xA industrial automation systems , SNMP is used to collect device information and support continuous supervision of the industrial network infrastructure.

SNMP is also essential for ABB's "Network and Device Scanning" (NDS) tool, which automatically discovers and configures industrial devices. NDS uses ICMP and SNMP to scan the network, retrieve MIB files, extract OIDs, and map them to Windows Management Instrumentation (WMI) paths [25].

The primary benefits of this automation include reduced engineering effort, improved configuration data quality, and faster integration of new assets into the control system. SNMP is an important tool in this industrial automation environment, as it is responsible for device discovery and network monitoring.

3.3 SNMP Threats and Security Challenges

3.3.1 General Threats Types Relevant to SNMP

To evaluate the security of SNMP, it is important to understand the general types of threats relevant to the protocol. These threats can be classified as:

- Denial of service (DOS): A system is flooded with requests, causing it to crash or become unresponsive. Users can not access to the service [26].
- Masquerading (Social Engineering): An attacker claims to be an authority figure or trusted person to gain unauthorized access to restricted information [26].
- Data manipulation (Tampering): An unauthorized person can intercept the transmitted message and maliciously modify its content [26].
- Disclosure: Confidential information is being stolen or accessed by someone who do not have permission [26].
- Abuse of privilege: Users with read-write permissions perform actions beyond their authorized access rights [26].
- Traffic pattern analysis: Instead of focusing on message content, Attackers extract information from communication patterns and traffic behavior [26].

3.3.2 SNMP Vulnerabilities and Threats

The Following are vulnerabilities and threats specific to SNMP:

- Lack of encryption : SNMPv1 and SNMPv2c do not support encryption, meaning all data is transmitted in plaintext. Anyone who intercepts packets can read sensitive data.
- Default community strings: In SNMPv1 and SNMPv2c, the community string functions as a “password” for accessing device data. Common default strings such as “public” or “private” are easily guessed by attackers [26].

- Absence of community strings: A system without community strings is even less secure than using "public" because now everyone can access the data [26].
- Access control: Without proper access control mechanisms, unauthorized people can retrieve sensitive information from devices [26].

3.3.3 Security Requirements for SNMP

A qualified secure communication network should provide several secure services as follow [26]:

- Authentication: Verifies the identities of agents and managers.
- Access control: Ensures that only authorized users can perform specific operations.
- Confidentiality: Protects sensitive information from unauthorized users through encryption.
- Integrity: Protecting data from tampering and altering during transmission
- Availability: Guarantees that services remain accessible to authorized users.

3.4 Comparison Between SNMP and Web-Services-Based Management

In terms of Bandwidth consumption, SNMP is remarkably more efficient when retrieving only one or a few objects. However, compressed Web services handle better for larger datasets more than about 70 objects [27]. Regarding CPU load, XML encoding requires 3-7x more CPU time than SNMP's BER encoding [27].

In summary, SNMP is ideal for retrieving smaller datasets, while Web-Service-Based Management is more effective for large data transmissions.

3.5 Regulations and Standards

3.5.1 NIST SP 800-82

NIST describes SNMP as a principal mechanism for network management between a management system and devices such as routers, printers, and PLCs. At the same time, it explicitly warns that SNMPv1 and SNMPv2c offer very weak security because they use unencrypted passwords (community strings) to read and configure devices, and in many cases these passwords are often widely known and unchangeable [28].

Hence, the document recommends that:

- SNMPv1 and SNMPv2c traffic to and from the control network should be prohibited, unless it is restricted to a separate, secured management network.
- SNMPv3 may be used in for Industrial control systems, as it provides built-in security functions for authentication and encryption.

3.5.2 Other Industrial Security Standards (IEC 62443, BSI ICS)

Although IEC 62443 does not explicitly mention SNMP, it defines mandatory security requirements for all managenent interfaces, including authentication, access control, and secure communication. SNMPv1 and SNMPv2c do not meet these requirements, whereas SNMPv3 can satisfy them when configured with authentication and encryption [29].

The BSI ICS Security Compendium states that administrative and configuration protocols must ensure confidentiality and integrity. As a result, insecure management protocols must be replaced by secure alternatives or protected through segmentation. This requirement applies directly to SNMP when used in industrial networks [30].

3.5.3 SNMP Standards(RFC 1157, RFC 1901–1908, RFC 3410–3418)

SNMPv1 is defined in RFC 1157, the initial Internet Standard for SNMP. It introduces the community string security model, where a plaintext community string controls access. The RFC contains no authentication, integrity,

or confidentiality. SNMPv1 is therefore unsuitable for secure or regulated industrial environments.

SNMPv2c is specified in RFC 1901–1908 and provides functionality improvement such as bulk transfers and improved error handling, but preserves the same community-string feature as SNMPv1. Because the security design remains unchanged and community strings are still transmitted in plaintext, SNMPv2c is considered insecure by design ,despite its technical improvements.

SNMPv3, defined in RFC 3410–3418 (STD 62), introduces the first fully secure SNMP framework. It adds:

- Authentication and integrity (User-based Security Model, USM)
- Encryption ("privacy")
- Role-based access control (VACM)

These features enable SNMPv3 to align with modern security requirements. It is the only SNMP version acceptable for use in security-sensitive or regulated industrial networks.

3.6 Real Example: APT28 Attacks Cisco Routers

In 2021, APT28 used infrastructure to impersonate SNMP access into Cisco routers worldwide by exploiting SNMP misconfigurations and a vulnerability [31]. The targeted routers in this case were configured with SNMPv2. SNMPv2 does not support encryption, meaning all data, including community strings, is transmitted in plaintext. In addition, weak of default community strings such as "public" can be easily guessed by attackers. Hence, APT28 was able to gain unauthorized access to router information. After obtaining SNMP access, APT28 exploited a known vulnerability CVE-2017-6742 Cisco announced in 2017 [31]. This vulnerability allowed attackers to send additional SNMP commands via SNMP packets. APT28 used this flaw to deploy the Jaguar Tooth malware. The malware exfiltrated device information through trivial file transfer protocol (TFTP), enabling unauthenticated access through a backdoor [31].

Bibliography

[1] Roy T. Fielding et al. *Hypertext Transfer Protocol – HTTP/1.1*. RFC 2616. June 1999. DOI: 10.17487/RFC2616. URL: https://www.rfc-editor.org/rfc/rfc2616.

[2] Roy T. Fielding, Mark Nottingham, and Julian Reschke. *HTTP Semantics*. RFC 9110. June 2022. DOI: 10.17487/RFC9110. URL: https://www.rfc-editor.org/rfc/rfc9110.

[3] Eric Rescorla. *HTTP Over TLS*. RFC 2818. May 2000. DOI: 10.17487/RFC2818. URL: https://www.rfc-editor.org/rfc/rfc2818.

[4] Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 8446. Aug. 2018. DOI: 10.17487/RFC8446. URL: https://www.rfc-editor.org/rfc/rfc8446.

[5] David Cooper et al. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280. May 2008. DOI: 10.17487/RFC5280. URL: https://www.rfc-editor.org/rfc/rfc5280.

[6] Elaine Barker and Quynh Dang. *Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations*. NIST Special Publication 800-52 Revision 2. Aug. 2019. DOI: 10.6028/NIST.SP.800-52r2. URL: https://doi.org/10.6028/NIST.SP.800-52r2.

[7] Peter Saint-Andre and Jeffrey Hodges. *Representation and Verification of Domain-Based Application Service Identity in PKIX Certificates with TLS*. RFC 6125. Mar. 2011. DOI: 10.17487/RFC6125. URL: https://www.rfc-editor.org/rfc/rfc6125.

[8] Christian Reich and Birgit Vogel-Heuser. "Time-constrained services: a framework for using real-time web services in industrial automation". In: *Journal of Systems Architecture* 58.10 (2012), pp. 412–423. DOI: 10.1016/j.sysarc.2012.05.001.

[9] Oana Iova, Gian Pietro Picco, et al. "A Comparative Review of Industrial IoT Protocols". In: *Future Internet* 10.3 (2018). DOI: 10.3390/fi10030034.

[10] Amandeep Gill. "Enhancing Data Security: Applications and Challenges of Secure Protocols in Key Industries and Public Services". In: *IEEE Access* (2025). DOI: 10.1109/ACCESS.2025.xxxxxx.

[11] *The DigiNotar Incident: Why Digital Safety Fails to Attract Enough Attention from Public Administrators*. Tech. rep. Official investigation report. The Hague, The Netherlands: Dutch Safety Board, 2012.

[12] *The Equifax Cybersecurity Incident*. Tech. rep. Official Equifax post-incident disclosure. Equifax Inc., 2017.

[13] Yasser Shoukry et al. "Physical Security Analysis of PLC Vulnerabilities including CVE-2020-15782". In: *Proceedings of the ACM Workshop on Cyber-Physical Systems Security*. 2021, pp. 45–56. DOI: 10.1145/3485832.3485901.

[14] Huan Yang, Liang Cheng, and Mooi Choo Chuah. "Detecting Payload Attacks on Programmable Logic Controllers (PLCs)". In: *2018 IEEE Conference on Communications and Network Security (CNS)*. Includes analysis of Modicon authentication bypass attacks. 2018, pp. 1–9. DOI: 10.1109/CNS.2018.8433163.

[15] Aline Nascimento and Valter Ferrari. "DNS Spoofing Techniques and Countermeasures". In: *2014 IEEE Latin-American Conference on Communications*. 2014, pp. 1–6. DOI: 10.1109/LatinCom.2014.7041874.

[16] Panagiotis Kintis, Najmeh Miramirkhani, and et al. "Understanding the Role of DNS in Malware Command-and-Control". In: *IEEE Transactions on Dependable and Secure Computing* 15.4 (2017), pp. 622–636. DOI: 10.1109/TDSC.2017.2650918.

[17] Omar Alrawi et al. "SoK: Security Evaluation of IoT Devices". In: *IEEE Symposium on Security and Privacy (SP)*. 2019, pp. 1411–1427. DOI: 10.1109/SP.2019.00012.

[18] Karthikeyan Bhargavan and Gaëtan Leurent. "On the Practical (In)Security of 64-bit Block Ciphers: Collision Attacks on HTTPS and OpenVPN". In: *2016 ACM SIGSAC Conference on Computer and Communications Security*. 2016, pp. 456–467. DOI: 10.1145/2976749.2978423.

[19] Huiming Lu and Zhifeng Yan. "Research on Key Technology of the Address Space for OPC UA Server". In: *2010 2nd International Conference on Advanced Computer Control*. Vol. 3. 2010, pp. 278–281. DOI: 10.1109/ICACC.2010.5486620.

[20] Sebastian Lehnhoff et al. "OPC Unified Architecture: A Service-Oriented Architecture for Smart Grids". In: *2012 First International Workshop on Software Engineering Challenges for the Smart Grid (SE-SmartGrids)*. IEEE. 2012, pp. 1–7.

[21] Nikolas Mühlbauer et al. "Open-Source OPC UA Security and Scalability". In: *2020 IEEE ETFA*. Vol. 1. 2020, pp. 262–269. DOI: 10.1109/ETFA46521.2020.9212091.

[22] Hanwen Wang, Xinge Li, and Xiaoya Hu. "A Vulnerability Mining Method for IEEE 802.1Qbv in TSN Systems Based on Timed Automata". In: *2022 China Automation Congress (CAC)*. 2022, pp. 6644–6649. DOI: 10.1109/CAC57257.2022.10056002.

[23] Nigel R. Lawrence. "Vulnerabilities in SNMPv3". Master's Thesis. Atlanta, GA, USA: Georgia Institute of Technology, 2012.

[24] Douglas Mauro and Kevin Schmidt. *Essential SNMP*. Sebastopol, CA: O'Reilly Media, 2001. ISBN: 0-596-00020-0.

[25] Thomas E. Koch and Esther Gelle. "On Automating the Network Management in Industrial Automation Systems". In: *Proceedings of ABB Corporate Research*. ABB Switzerland Inc. Baden-D"attwil, Switzerland, 2004.

[26] Periklis Chatzimisios. "Security Issues and Vulnerabilities of the SNMP Protocol". In: *1st International Conference on Electrical and Electronics Engineering*. Bournemouth University. Poole, UK, 2004.

[27] Aiko Pras et al. "Comparing the Performance of SNMP and Web Services-Based Management". In: *IEEE Transactions on Network and Service Management* 1.1 (2004), pp. 1–12.

[28] Keith Stouffer et al. *Guide to Industrial Control Systems (ICS) Security*. Tech. rep. NIST SP 800-82 Rev. 2. Gaithersburg, MD: National Institute of Standards and Technology (NIST), 2015. DOI: 10.6028/NIST.SP.800-82r2.

[29] *IEC 62443-4-2: Security for Industrial Automation and Control Systems – Part 4-2: Technical Security Requirements for IACS Components*. Edition 1.0. 2019.

[30] Federal Office for Information Security (BSI). *ICS Security Compendium*. Tech. rep. Version 1.23. Bonn, Germany: Bundesamt für Sicherheit in der Informationstechnik (BSI), 2013.

[31] National Cyber Security Centre (NCSC) et al. *APT28 Exploits Known Vulnerability to Carry Out Reconnaissance and Deploy Malware on Cisco Routers*. Tech. rep. Version 1. UK NCSC, Apr. 2023.