A Comparison of Security ICS Communication Protocols

Daniel Clark✉[0009−0001−6983−3110] and Tom Chothia[0000−0002−9381−1368]

University of Birmingham, United Kingdom
{d.e.clark, t.chothia}@bham.ac.uk

Abstract. This paper presents a summary and comparison of leading secure Industrial Control System (ICS) communication protocols. We find two categories of ICS protocols: those that run an insecure ICS protocol over TLS and those that use a bespoke security protocol. We assess the key properties of bespoke ICS protocols using formal modelling, for OPC-UA and DNP3-SAv5 we use existing models, and we build formal models for S7Comm-Plus and SSP-21; this lets us make a full comparison between these protocols. For ICS protocols based on TLS, we compare the versions and configurations of TLS specified, and any access control add-ons. This leads to a detailed picture of the security provided and best use cases for each protocol.

Keywords: ICS protocols · Formal analysis · TLS

1 Introduction

Industrial Control Systems (ICS) manage most aspects of national critical infrastructure, therefore the security of the protocols is of the utmost importance. Traditionally ICS protocols such as Modbus and PROFINET had no cryptographic protections, relying on firewalled networks for security, however many recent attacks have shown that this makes it possible for an attacker to penetrate the network and send commands to PLCs. Examples of such attacks include Industroyer/CrashOverride [16], Triton [27] and FrostyGoop [22]. In response to this some ICS protocols, such as Siemens S7Comm or OPC, have added encryption and authentication. Other protocols, such as Modbus TLS or BACnet/SC, add security by running the older insecure version of the protocol over a particular flavour of TLS.

While many of these protocols have been looked at individually (e.g., [8,15]), little work has been done on comparing the security they offer. This paper provides a comprehensive analysis and comparison of the leading security ICS protocols. To make a direct comparison of the security offered by the protocols, we build and test models of S7Comm-Plus, and SSP-21 [35] in the Tamarin protocol checker, and we use existing models of OPC-UA and DNP3-SAv5. This allows us to uniformly test and compare the security properties offered by these protocols.

Past work has found attacks against Siemens S7Comm-Plus protocol [12,29]; our formal modelling confirms these attacks and finds more. Our modelling of

SSP-21 finds that it has the strongest security properties. However, this protocol is not supported by any of the devices we have looked at. This leaves OPC-UA as the best option from the non-TLS protocols.

Next we look at protocols that use TLS, i.e., Modbus TLS, BACnet/SC, CIP Security, IEC 62351 and Siemens Logo. We review the documents for these to find the particular kinds of TLS used. The most secure use TLS 1.3 with client certificates and a PKI, leading to a very high level of security. However we identify a major issue with the use of TLS for ICS protocols. Best practice in ICS networks calls for an Intrusion Detection System (IDS) to monitor traffic and a data historian to record network behaviour. This only works with plaintext traffic, but TLS 1.3 has been designed not to allow authenticated plaintext. Hence the use of TLS 1.3 may present an issue for some ICS owners. CIP Security addresses this by specifying the use of TLS 1.2 that allows authenticated plaintext traffic. However TLS 1.2 will become outdated and lack support in the future. Therefore we conjecture that the use of TLS will pose a problem for ICS protocols in the future.

In summary the contributions of this paper are:

- Formal modelling in the Tamarin Prover of S7Comm-Plus and SSP-21.
- A detailed review of how TLS is used in Modbus TLS, BACnet/SC, CIP Security, IEC 62351 and Siemens Logo.
- A comparison of the security offered by these protocols and discussion of future directions.

Our Tamarin models and protocol diagrams can be found in our GitLab repo: https://gitlab.com/ICSProtocolSecurityComparison/ICSProtocolSecurity

2 Background

Industrial Control Systems (ICS) are an example of Operational Technology (OT); they will typically control physical processes such as factory machinery and industrial processes. A key component of ICS networks are Programmable Logic Controllers (PLCs) that will read data from sensors and run a program to control actuators. These PLCs are usually programmed and controlled by proprietorial desktop controller software provided by the manufacturer of the PLC, and the PLC may be set to communicate with other PLCs and Human Machine Interface (HMI)s. Importantly, the ICS owner will often have very little visibility of the software running on the PLC and the ways in which the PLC communicates with other devices; hence the need to understand the security of ICS communication protocols.

ICS protocols have historically not been designed with security in mind, despite controlling significant swathes of critical infrastructure the world over, instead focusing primarily on reliability and compatibility. There are a huge number of varying ICS protocols, but some of the most common are: Modbus, HART, DNP3, CIP, PROFINET and BACnet, the vast majority of which put little to no consideration towards security in the protocols themselves. These

systems were normally designed explicitly to be isolated from all remotely accessible networks, often through the use of air gaps, meaning the attack vectors were much more limited, so the risks of cyber attacks were considered negligible.

More recently however, these assumptions have been disproven in two significant ways. The first is that attacks have been discovered on these air-gapped networks via sophisticated malware designed to bridge the air gap. The second is that in order to introduce novel functionality, like more extensive monitoring and operation between multiple industrial sites, these systems have been connected to wider networks, including the internet.

With the risks associated with added connectivity, and the increased sophistication of attacks, designers of these protocols have sought to either add security to protocols that already exist, or introduce new standard protocols which have security incorporated from the start. One common approach to creating these new secure protocols, has been to wrap an already existing protocol within TLS in various ways (which we refer to as the TLS protocols), whilst others have introduced their own bespoke solutions either by adapting existing protocols or creating a new protocol from scratch (which we label as the Non-TLS/Bespoke protocols).

IEC 62351 outlines broad protocol definitions for various environments in parts 3, 4, 5 and 6:

Part 3 outlines recommendations for protocols using TLS.
Part 4 details an approach for adding cryptographic security features to protocols based on the Manufacturing Message Specification (MMS).
Part 5 describes applying security features to protocols based on IEC 60870-5, and is the basis of DNP3-SAv5 which will be discussed in section 4.4.
Part 6 proposes security additions to protocols based on IEC 61850.

There is a proposal for a version of PROFINET to run over TLS [37], however a full version has not been released. We do not look at the secure version of the Wireless-HART protocol, because this does not seem to be widely supported by PLCs and there is no public specification. We do not include more general protocols that may be used in an ICS environment, such as the Wi-Fi protocol WPA or the IoT device protocol MQTT. While some ICS equipment do use such protocols their security has been well studied outside the ICS domain.

2.1 Related work

Formal Verification of ICS Protocols In [13], Cremers, Dehnel-Wild and Milner performed an evaluation of the DNP3-SAv5 protocol, using the Tamarin Prover, and specifically tested it against the security claims made by the protocol specification. DNP3-SAv5 is a notably complex protocol, primarily because it is made up of multiple interacting sub-protocols which use the same shared state, and the model reflects this complexity. Through this model the researchers were able to prove a number of security properties against the protocol, and notably disprove a previously claimed attack outlined in [8].

The OPC-UA protocol has been modelled in the ProVerif protocol checker by Puys et al. [33] and more recently by Diemunsch et al. [15]. This second paper includes a very large model of OPC-UA and finds a range of security issues, the authors worked with the OPC standards group to fix some of the issues, but others remain and we discuss these below.

Non-Formal Analysis of ICS Protocols Many authors have described attacks on ICS protocols without security (e.g., [1,7,11,20,28]), and other papers have pointed out these insecure protocols often have unsafe password protection for access control, e.g., [38].

Attacks have been discovered during previous analysis of S7Comm-Plus, for instance, [12,29,26] found that the Siemens PLCs using this protocol use common certificates and therefore traffic can be intercepted and decrypted. However, there has been no formal modelling of this protocol to provide a complete picture of exactly what security it does and does not provide.

Barbieri et al. [10] carry out a measurement study of ICS protocols used over the internet and find wide spread use of insecure ICS protocols. Dahlmanns et al. [14] look at the uptake of TLS based ICS protocol, but do not compare the security they offer. Holasova et al. [25] train a neural network to recognise some of the ICS protocols we look at, but they do not compare their security. [19] presents a comparison of OPC-UA and CIP Security; they conclude that both approaches have their benefits and drawbacks.

On the more practical side, Erba et al. [17] investigate devices that use OPC-UA and find that many do not handle certificate security. Tychalas and Maniatakos [36] look at side channel attacks on PLCs including some using OPC-UA, and [24] Hildebrandt et al. present a supply chain attack against PLCs using OPC-UA. All of these are due to the implementations of OPC-UA, rather than the protocol designs, which we analyse in this paper.

Work has also been done on exploiting implementation weaknesses in PLCs using other protocols. For example, Alsabbagh and Langendöerfer discover code injection attacks against various versions of Siemens S7 PLCs [5,4,6,3], and an OpenPLC environment [2]. These attacks highlight further the need for secure cryptographic protocols which would eliminate a key attack vector needed to exploit them.

2.2 Tamarin Prover

The Tamarin Prover is a tool for verifying the security of protocols using symbolic models. A Tamarin model is formed of two parts, the description of the protocol itself, and a set of security properties to be proven.

Protocol descriptions are built as a set of rules, which take a particular state as input, and output a new state. A state takes the form of a set of facts, which each declare some term exists in that state. By consuming and creating facts, rules are therefore the transitions between states. Agents are typically defined by an identifier, which is then stored in all facts which relate to them, and so are effectively restricted to only being accessed by that agent. Special In and

Out facts exist to model the transfer of messages across the network, only data which has been output to the *Out* fact may be taken as input with the In fact.

Tamarin uses the Dolev-Yao attacker model to define the properties of the network. In this model the attacker is in full control of the network, they can intercept, modify, block, delay or create any message. The attacker can only create messages from their own knowledge, which can be derived from any information sent across the network and any public function. This model can be extended to allow malicious agents, that is, genuine agents which have been taken over by the attacker.

Cryptographic functions are defined symbolically and perfectly, e.g., all symmetric encryption schemes are modelled as the same functions (*senc* and *sdec*) and cannot be brute forced or otherwise broken by the attacker.

Security properties are defined as lemmas which reason about terms at specific points in the protocol. Action facts are used to mark terms in the protocol at particular points, which will be referenced by the lemmas. Lemmas are either defined as Exists or For-All lemmas. Exists lemmas simply state that there is at least one possible protocol run where a specific set of actions happen (or do not happen). For-All lemmas state that, in all cases where some set of actions happen, then this other set of actions must always happen.

For example, a possible secrecy for-all lemma could say that for all runs of the protocol where an agent has a secrecy key, it is never the case that the attacker knows that key. To prove this lemma, Tamarin will try to find a counter example. It will start from the assumption that both the agent and the attacker have the key, and work backwards, applying rules in reverse until it finds a valid run of the protocol. If no valid protocol run exists, it considers the lemma proven. If Tamarin does find a valid protocol run, the lemma is disproven, and Tamarin can present the counter example as a possible attack.

3 Security Properties and Attacker Model

In this paper we investigate each of the non-TLS/bespoke protocols against a number of security properties. To give a more complete picture and comparison of each protocol and each protocol mode, we go beyond checking basic secrecy and authenticity and check each of the following properties:

1. Authenticity
   (a) Non-Injective Agreement: If an entity believes it has finished a transaction with a particular party, then they have done so, and they agree on the protocol parameters, e.g., session keys or a message. This is the crucial security property for ICS; if a PLC believes it has received a specific command from a specific controller, then it is critical that this is actually the case.
   (b) Injective Agreement: As above, but the two parties must be in a one-to-one agreement regarding the sending and receiving of a message, e.g., messages cannot be replayed.

2. Secrecy - We consider this both in cases where agents external to the session may be compromised, and where they are not.
   (a) Secrecy: can an attacker learn the keys or messages of the protocol? Keeping long term and session keys secret is an important goal for any protocol. We note that some protocol (modes) deliberately do not keep messages secret so that they can be analysed by, e.g., IDSs.
   (b) Perfect Forward Secrecy (PFS): If a long term key is leaked, can previous messages be decrypted? We follow the definition of PFS from the Tamarin manual that allows an active attacker. PFS is a less important property for ICS than secrecy, but can still be important in some situations, e.g., if an encrypted control program was captured, and the key for this later learned from a compromised device.
3. Key Compromise Impersonation (KCI): If the long term key of an entity has been compromised, can an attacker impersonate other entities to them.
4. Key Uniqueness: Every session key should be different. If the attacker can force two sessions to have the same key they could for instance redirect encrypted messages from one device to another, leading to machinery behaving in dangerous ways.

Attacker Model: We consider an attacker that has gained access to the network used by the ICS equipment. Many ICS owners will aim to keep the ICS network secure and would consider the presence of an attacker on this network to already be a failure of security; this is the rationale behind the continued use of insecure ICS protocols. However, as many ICS attacks show, attackers do find their way onto these networks, and as Barbieri [10] shows, there is a strikingly large amount of ICS protocol traffic sent across the public internet.

All agents not directly involved in the current run of the protocol are considered malicious. For PFS, agents become malicious after the protocol run has completed, and for KCI, the agent which is the target of the attack is also considered malicious.

4 Non-TLS/Bespoke Protocols

In this section we will outline a number of protocols developed for communication between devices in ICS networks, specifically those which are not built around the TLS protocol, as these will be discussed later in this paper.

For each, we will give a brief background to the protocol, give an outline of the protocol itself if required, outline the security claims made by the protocol specifications and describe our contribution to the analysis of each protocol. After describing all the protocols we will then describe the results of testing the formal models and conclude with a summary of the results of each of the analyses. Protocol diagrams and Tamarin models for each protocol we model can be found on our website.

4.1 Siemens S7Comm-Plus

S7Comm-Plus is a proprietary standard developed by Siemens as an iteration of their earlier S7Comm protocol which has been used since 1995. Many forms of S7Comm-Plus have been released since 2012, primarily with the aim of adding encryption, authentication and integrity protection to the S7Comm protocol. Past work [29,12] reverse engineered the S7Comm-Plus protocol, and discovered a number of vulnerabilities. In this paper we focus on protocol version 3, as described in [12].

Protocol Description A full protocol diagram for S7Comm-Plus can be found on our website, but we will provide a simplified description here based on the work of Biham et al. [12]. S7Comm-Plus sessions run between a Totally Integrated Automation Portal (TIA), which acts as the client, and a PLC which acts as the server. The PLC firmware is pre-configured with a public/private key pair, and that public key is also stored in the TIA firmware. Biham et al. [12] found that all PLCs of the same model and firmware version, used the same public/private key pair.

The protocol begins with a TIA sending a cleartext "Hello" message, to which the PLC responds with a random $serverSessionChallenge$. The TIA hashes and MACs this challenge with a freshly generated $KeyDerivationKey(KDK)$ to create the $sessionKey$. A fresh $preKey$ is then generated, which is used to derive the $KeyEncryptionKey(KEK)$ and other keys for error checking.

The TIA then sends a $SecurityKeyEncryptedKey$ message to the PLC that includes the $preKey$ encrypted with the public key, the $serverSessionChallenge$ and KDK both encrypted with the KEK, and other values for randomness and error checking. The PLC decrypts this message and hashes the newly acquired KDK with the $serverSessionChallenge$ to derive the $sessionKey$ in the same way as the TIA above. From this point on, every message in both directions is authenticated using a MAC keyed with the $sessionKey$.

Modelling In S7Comm-Plus, TIAs have no long-term secrets, identifiers or private functions. This means that an attacker can trivially impersonate a valid TIA with no prior knowledge whatsoever. PLCs do have a long-term secret, a private key, however, in practice it has been seen that the same PLC private key is shared amongst all PLCs with the same model and firmware version [12]. The protocol itself does not depend on this beyond the TIAs needing to know the PLC's public key in advance. As such, we model two scenarios, one where the PLC private key was shared amongst all PLCs, and one more idealised version where the private key is unique for each PLC. These are both contained within the same model, and are switched via the -DNoSharedKey Tamarin flag.

PLC agents are restricted via a Once() restriction whereby each PLC may only have a single private key. TIA agents also have Once() restrictions on their initialisation, in order to stop the same combination of PLC and TIA from creating multiple identical !TIA facts.

The majority of the cryptographic algorithms used in S7Comm-Plus are custom and proprietary, and where standard algorithms are used, they are often used incorrectly [12]. The result of this is that there are significant cryptographic weaknesses in the the protocol brought about by these algorithms. Due to the symbolic nature of Tamarin models, all cryptographic primitives are considered perfect, with the focus on flaws in the protocol rather than the cryptographic algorithms themselves.

During key establishment, a $preKey$ is used with a Key Derivation Function (KDF) to derive 3 sub-keys, the Key Encryption Key (KEK), the Checksum Encryption Key (CEK), and the Checksum Seed (CS). The CS is then used to derive the Look Up Table (LUT), which is then used as an input to a checksum algorithm. In our model, we remove the step of deriving the LUT from the CS, and simply consider the Look Up Table to be an output from the KDF instead of the CS. The KDF is abstracted to a 1-way function with two inputs, the key name, and the $preKey$, and the checksum algorithm is also abstracted to a 1-way function with two inputs, the checksum input, and the LUT.

The Elliptic-Curve-ElGamal-Like encryption algorithm [12] used to encrypt the $preKey$ is modelled using a custom asymmetric encryption equation: $adec\_s7(aenc\_s7(plaintext, nonce, pk(sk)), nonce, sk) = plaintext$, where the $preKey$ and random y value are the plaintext. Both the $Nonce$ and the PLC's public key are needed to encrypt, then the PLC's private key X and the $Nonce$ are needed to decrypt.

The AES-CTR encryption of the Challenge and KDK is modelled as two separate symmetric encryptions, with counter values '0' and '1' respectively, where the IV, counter and KEK are used together as the key. The HMAC-SHA256 used to authenticate final protocol messages is modelled as a 1-way function with two parameters, the message data, and the the $sessionKey$.

4.2 SSP-21

The Secure SCADA Protocol for the 21st Century (SSP-21) [35,23] started development in 2016, with the latest version being published in May 2020 as part of the California Energy Systems for the 21st Century (CES-21) project by the California Public Utilities Commission. It is inspired by the Noise protocol [32], especially the static-ephemeral triple Diffie-Hellman Key Exchange. This was a major effort to design a better ICS protocol, but is has not been widely adopted.

Protocol Description In this section we will outline a simplified version of the SSP-21 protocol. A full protocol description for SSP-21 can be found in [35], and protocol diagrams can be found on our website.

SSP-21 can be run in one of 4 handshake modes: Shared Secret (SS), Public Keys (PK), Industrial Certificates (IC) and Quantum Key Distribution (QKD), each with their own requirements for any prior knowledge, values for EphemeralData and ModeData, and some Input Key Material (IKM). These are outlined in table 1.

Table 1. SSP-21 Handshake Modes

Mode	Prior Knowledge	Ephemeral Data	Mode Data	Input Key Material (IKM)
SS	*SharedSecret*	Initiator: Random *INonce* Responder: Random *RNonce*	-	$SharedSecret \| \| INonce \| \| RNonce$
PK	Initiator: *IStaticPrivate* *RStaticPublic* Responder: *RStaticPrivate* *IStaticPublic*	Initiator: $IEphemeralPublic^{*}$ Responder: $REphemeralPublic^{*}$	-	Initiator: $REphemeralPublic^{IEphemeralPrivate} \| \| REphemeralPublic^{IStaticPrivate} \| \| RStaticPublic^{IEphemeralPrivate}$ Responder: $IEphemeralPublic^{REphemeralPrivate} \| \| IStaticPublic^{REphemeralPrivate} \| \| IEphemeralPublic^{RStaticPrivate}$
IC	Initiator: *TAPublic* *IStaticPrivate* *ICert* Responder: *TAPublic* *RStaticPrivate* *RCert*	Initiator: $IEphemeralPublic^{*}$ Responder: $REphemeralPublic^{*}$	Initiator: ICert Responder: RCert	Initiator: $REphemeralPublic^{IEphemeralPrivate} \| \| REphemeralPublic^{IStaticPrivate} \| \| RStaticPublic^{IEphemeralPrivate}$ Responder: $IEphemeralPublic^{REphemeralPrivate} \| \| IStaticPublic^{REphemeralPrivate} \| \| IEphemeralPublic^{RStaticPrivate}$
QKD	Depends on QKD method and is not defined in the standard	-	KeyIdentifier	The QuantumKey identified using the KeyIdentifier

* Derived from a fresh ephemeral private key generated at the start of each session.

The *Initiator* sends the first message, *RequestHandshakeBegin*, which contains some configuration values (e.g what algorithms are supported, the supported handshake mode(s), the required $NonceMode$ and $CryptoMode$), the $EphemeralData$ and the $ModeData$. The $NonceMode$ simply defines if the counters in all $SessionData$ messages must be strictly incrementing, or just needs to be higher than the last valid counter. The $CryptoMode$ determines if the $SessionData$ messages should be encrypted and MACed, or just MACed. The Responder then replies with the *ReplyHandshakeBegin* message, containing its $EphemeralData$ and $ModeData$.

Each side then computes the hash of the two $HandshakeBegin$ messages, as well as calculating their mode-specific IKM. A Key Derivation Function uses the hash and the IKM to generate two symmetric session keys on each side, one for sending ($SessionKeyTX$), and one for receiving ($SessionKeyRX$), where the TX key on each side matches the RX key on the other. Data transfer can then begin using $SessionData$ messages, the first of which in each direction are used to authenticate both sides using the newly calculated session keys, and are nicknamed the *SessionAuthRequest* and *SessionAuthReply*.

All $SessionData$ messages contain an $AuthMetadata$ field which contains a $Nonce$ (a counter which increments for each message sent in a session starting at 0) and the expiry time of the message $ValidUntilMS$. $SessionData$ messages also included the length of the User Data, the User Data itself, and a MAC over the contents of the message using the $SessionKeyTX$. The *SessionAuthRequest* and *SessionAuthReply* messages may contain User Data, or may be empty with a User Data length of 0.

Once the *Responder* receives and verifies the *SessionAuthRequest* using its $SessionKeyRX$, it sends a *SessionAuthReply*. The *Initiator* receives and verifies the *SessionAuthReply* using its $SessionKeyRX$, after which authenticated communication can begin using standard $SessionData$ messages. Authentication is considered complete by the *Responder* and *Initiator* once they have received the *SessionAuthRequest* and *SessionAuthReply* messages respectively.

Modelling We constructed a separate model for each Crypto Mode (with and without encryption), which differ only in that the messages in the encrypted model are encrypted, decrypted and verified using the Authenticated Encryption with Associated Data (AEAD) equations from [21].

The SSP-21 standard defines 4 Handshake Modes, however the details of the Quantum Key Distribution (QKD) mode are not specified in sufficient detail to be implemented or modelled effectively, so we omitted this handshake mode from our models. The protocol is identical in every mode beyond the two *HandshakeBegin* messages, therefore, the 3 modes were implemented into the one shared model, with Tamarin preprocessor flags -DSS, -DPK and -DIC used to determine which modes to include.

SSP-21 uses a Key Derivation Function (KDF) function to generate session keys, which we model as a one-way function with three inputs, some IKM (based on the Handshake Mode), the Handshake Hash (generated from the two HandshakeBegin messages), and a key number, so that two keys can be generated. The certificates used in IndustrialCertificates mode are modelled as the Identity of the agent, the agent's static public key and the CA's signature of the identity and key. We use a NotEqual(NEq) restriction to ensure that incoming public keys do not have invalid values, as per the protocol specification, and we use a *Once* restriction to ensure only one Trusted Authority (TA) can exist.

In order to limit model complexity, we have only modelled the $STRICT\_INCREMENT$ $NonceMode$, and have hard-coded the message counter into each message. This is a valid optimisation as the two $SessionAuth$ messages must always have their counter set to 0, and we only model one further message which should therefore always have the counter 1.

4.3 OPC-Unified Architecture (OPC-UA)

OPC-UA is an open standard developed by the OPC Foundation, a large consortium of of industry players, which since its inception has been designed to allow interoperability and a layer of abstraction between devices using the plethora of legacy SCADA communications protocols. The first version of OPC-UA was released in 2006, and although developed by the same group, bares little resemblance to the original OPC protocol.

Protocol Description Throughout the OPC-UA specification it often refers to "signing", however due to their symmetric nature, they should actually be more considered as Message Authentication Code (MAC)s rather than signatures in most cases. For consistency, we will continue to use the terminology set by the specification.

OPC-UA can be used in one of three Security Modes:

None - All messages are neither signed nor encrypted.
Sign - Final messages are signed but not encrypted. Open Secure Channel, Create Session and Activate Session messages are all both signed and encrypted.
SignEncrypt - All messages are both signed and encrypted.

The OPC-UA standard outlines a number of Security Policies, which define which algorithms are to be used for signing, encryption and key derivation.

Notably, OPC-UA authenticates Users separately from Clients and Servers. Clients/Servers are authenticated using TA-signed certificates, whilst Users can be authenticated using one of four User Identity Token Modes: **Anonymous** which provides no authentication, **UserName** which authenticates users via a Username and Password, **X509** which uses a self-signed User certificate, or **Issued** which uses an OAuth2 Security Token issued by an Authorisation Service.

The address of a Discovery Endpoint is pre-configured into each client. From there, the protocol is split into four logical sub-protocols:

Get Endpoints - A Client uses this to retrieve a list of available Servers from the Discovery Endpoint. It also includes each Server's certificate and the Security Modes and Policies it supports.

Open Secure Channel - Sends the Client's certificate to the Server, and establishes shared symmetric Session Keys (except in Security Mode: None). This can either use RSA or Elliptic-Curve Cryptography (ECC) with Elliptic-Curve Diffie–Hellman (ECDH), depending on the Security Policy.

Create Session - Both sides are authenticated using their certificates and a challenge-response exchange. The Server also shares a Session Authentication Token, which the client then includes in future requests to associate them with a session (and once activated, a User).

Activate Session - Uses the User Identity Token to authenticate the user, this depends on the User Identity Token Mode.

Modelling For OPC-UA we use the detailed ProVerif model presented in [15].

4.4 DNP3-SAv5

The original 1993 version of Distributed Network Protocol 3 (DNP3) was derived from IEC 60870-5 to provide interoperability between different SCADA systems from different manufacturers, and was eventually adopted into IEEE 1815-2010. In 2013, IEC 62351-5 was published to standardise security additions to IEC 60870-5-derived protocols. It added support for asymmetric cryptography and a CA, and with the release of IEEE 1815-2012, these changes were introduced into the DNP3 specification in the form of Secure Authentication Version 5 (SAv5).

DNP3 is primarily used in the control of power grids and related infrastructure, and is used in the majority of grids worldwide. Its use has been growing in other parts of critical infrastructure such as water and gas supplies, but has found limited adoption outside of this.

DNP3-SAv5 has been analysed previously by [8], who claimed a vulnerability, however this was disproven by an in-depth investigation using the Tamarin Prover in [13]. They built a comprehensive model of DNP3-SAv5 and proved its security against the security guarantees claimed by the specification.

Modelling We use Cremers et al.'s Tamarin model of DNP3-SAv5 from [13].

4.5 Analysis

In this subsection we present and describe the results of the formal analysis. These results are summarised in Table 2. Below we summarise notable findings and their impact, for each class of security property.

Table 2. Results of the formal analysis of bespoke ICS protocols

		Secrecy With External Compromise			Secrecy No External Compromise			Authentication				KCI		Uniqueness
		Secrecy		PFS	Secrecy		PFS	Client → Server		Server → Client		Client	Server	
		LTKs	Messages	Messages	LTKs	Messages	Messages	Non Injective	Injective	Non Injective	Injective			
S7	Shared SK Private PK	$X_{[S1]}$	-	-	✓	-	-	$X_{[A1]}$	$X_{[A1]}$	$X_{[A1]}$	$X_{[A1]}$	-	$X_{[K1]}$	✓
	Shared SK Public PK	$X_{[S1]}$	-	-	✓	-	-	$X_{[A1]}$	$X_{[A1]}$	$X_{[A1]}$	$X_{[A1]}$	-	$X_{[K1]}$	✓
	Individual SKs Private PKs	✓	-	-	✓	-	-	✓	✓	✓	✓	-	$X_{[K1]}$	✓
	Individual SKs Public PKs	✓	-	-	✓	-	-	$X_{[A2]}$	$X_{[A2]}$	✓	✓	-	$X_{[K1]}$	✓
SSP	SS - MAC	✓	-	-	✓	-	-	✓	✓	✓	✓	$X_{[K2]}$	$X_{[K2]}$	✓
	PK - MAC	✓	-	-	✓	-	-	✓	✓	✓	✓	✓	✓	✓
	IC - MAC	✓	-	-	✓	-	-	✓	✓	✓	✓	✓	✓	✓
	SS - AEAD	✓	✓	$X_{[P1]}$	✓	✓	$X_{[P1]}$	✓	✓	✓	✓	$X_{[K2]}$	$X_{[K2]}$	✓
	PK - AEAD	✓	✓	$X_{[P2]}$	✓	✓	$X_{[P2]}$	✓	✓	✓	✓	✓	✓	✓
	IC - AEAD	✓	✓	$X_{[P2]}$	✓	✓	$X_{[P2]}$	✓	✓	✓	✓	✓	✓	✓
OPC	Sign - RSA	✓	-	-	✓	-	-	✓	$X_{[A3]}$	✓	✓	✓	$X_{[K3]}$	*
	SignEnc - RSA	✓	✓	$X_{[P3]}$	✓	✓	$X_{[P3]}$	✓	✓	✓	✓	✓	$X_{[K4]}$	*
	Sign - ECC	✓	-	-	✓	-	-	✓	$X_{[A3]}$	✓	✓	✓	$X_{[K3]}$	*
	SignEnc - ECC	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	$X_{[K4]}$	*
DNP3		✓	✓	*	✓	✓	*	✓	✓	✓	✓	*	*	*

- Not Applicable, e.g. Testing secrecy of an unencrypted message
* Not Checked, i.e. No equivalent lemma is found in the original model

Secrecy

S7Comm-Plus

*S*1 **- Description** All PLCs with the same model and firmware version share the same long-term secret key, consequently an attacker can recover it by compromising any PLC, even one which is not involved in the session.

*S*1 **- Impact** Since TIAs have no long-term secrets, with access to the shared PLC secret key, an attacker can perform any action on the network.

Perfect Forward Secrecy

SSP-21

*P*1 **- Description** In SharedSecret mode both agents share a single SharedSecret, and there are no ephemeral public keys exchanged. Once an attacker has extracted the SharedSecret, they can generate the session keys using public data from the two HandshakeBegin messages, and decrypt the session.

*P*1 **- Impact** Data exchanged in SharedSecret mode can be decrypted by any agent who witnessed the two HandshakeBegin messages and has possession of the SharedSecret. This weakness is acknowledged in the specification.

*P*2 - **Description** In PublicKeys and IndustrialCertificates modes, an attacker can masquerade as an existing Responder until it is required to send the SessionAuthResponse. At this stage the attacker only possesses 3 of the 4 public/private keys needed to derive the session keys and forge the SessionAuthResponse. If using AEAD, the attacker also cannot decrypt the previous message, the SessionAuthRequest, until the genuine Responder's Static Private Key is revealed after session termination - at which point the attacker has all 4 keys required to derive the session keys and decrypt the message.

*P*2 - **Impact** The SessionAuthRequest can optionally include session data, which the attacker is now able to decrypt. This session data could potentially include application configuration, commands, or even a partial application binary, which could lead to possible implementation attacks. Although the specification briefly alludes to "certain implementations" not leaving this field empty, but it provides no further details.

OPC-UA

*P*3 - **Description** When using an RSA Security Policy rather than an ECC one, OPC-UA session keys are generated only from Client and Server nonces exchanged during secure channel establishment. These nonces are shared encrypted only with their counterpart's public keys, therefore an attacker with access to both private keys is able to decrypt both nonces and generate the session keys.

*P*3 - **Impact** With the session keys, the attacker can decrypt all messages shared over the life of the secure channel. ECC support was only introduced to the OPC-UA specification in version 1.05, and of the 565 products listed on the OPC Foundation marketplace [31], only 43 claim any support for version 1.05, of which only 3 list support for any ECC Security Policies.

Authentication

S7Comm-Plus

*A*1 - **Description** After compromising the shared PLC long-term secret key as per S1, an attacker can impersonate any PLC to any TIA. TIAs have no long-term secrets of their own, and are authenticated only by possession of the PLC public key, consequently an attacker can simply derive the public key from the compromised secret key and impersonate any TIA to any PLC.

*A*1 - **Impact** If an attacker can compromise any PLC and recover the shared secret key, it can break all authentication and perform any action on the network.

*A*2 - **Description** The PLC public key is never made public as part of the protocol, but is pre-programmed onto each TIA with the correct firmware version. If an attacker can compromise any TIA and recover the public key, it can forge the unauthenticated $SecurityKeyEncryptedKey$ message sent

to the PLC. This includes a forged Key Derivation Key, which is used to MAC the *serverSessionChallenge* (sent in an earlier cleartext message), resulting in a forged session key established with the PLC.

$A2$ **- Impact** An attacker only need compromise any TIA (with the correct firmware version), to be able to send messages to any PLC, without needing to compromise any PLCs. This still holds in a hypothetical case where each PLC has an individual long-term secret key, however the attacker would specifically need to compromise a TIA which was set up to work with the target PLC.

OPC-UA

$A3$ **- Description** Diemunsch, V., et. al. [15] previously found the following attack, which they refer to as 'Session Hijack by Reopening or Switching', and remains unpatched. In Sign Only mode, the Session Authentication Token is sent in clear text, and once received by the Client, the User activates the session as normal. If an attacker is able to then recover the Client's secret key, they can use that secret key and the Session Authentication Token to hijack the original (still activated) session, and impersonate the User.

$A3$ **- Impact** With control over an activated session, an attacker can send arbitrary commands to the Server. Human User sessions on HMI systems are typically relatively short lived, however automated clients with fixed user credentials may have sessions lasting weeks or even indefinitely. A Client may have multiple sessions active with multiple Servers at the same time, therefore this attack could lead to an attacker gaining a significant amount of control from a single vulnerable Client.

Key Compromise Impersonation

S7Comm-Plus

$K1$ **- Description** If an attacker is able to obtain a target PLC's secret key (which is trivial in the realistic case where they are shared amongst PLC's of the same model and firmware version), the attacker can decrypt the $SecurityKeyEncryptedKey$ message sent by a TIA, and derive the session key.

$K1$ **- Impact** An attacker can therefore transparently impersonate any TIA with which a session has been established. This enables an attacker to take control of the PLC without interrupting the existing session.

SSP-21

$K2$ **- Description** In SharedSecret mode both agents share a single SharedSecret, and use no other cryptographic identities, consequently if an attacker can extract that SharedSecret from either agent, they can impersonate both agents to each other.

$K2$ **- Impact** In this mode, an attacker need only to compromise a single device, to be able to authenticate to, and create malicious sessions with, every other device that the compromised device has been paired with.

OPC-UA

$K3$ **- Description** A KCI attack on user authentication is trivial when in Password mode, however [15] also finds a KCI attack when using X509 certificates for User authentication in Sign Mode. If an attacker has compromised a Server's secret key, it can use that key to impersonate the Server to a Client, and collect the User's signature of the Server certificate and nonce, which the attacker can then use to impersonate the User to the Server.

$K3$ **- Impact** With a compromised Server secret key, an attacker on an ICS network could already cause significant damage to the system, so the additional damage this attack could enable would be, in most cases, minor.

$K4$ **- Description** [15] also discovered that when in SignEncrypt mode and using x509 certificates for User authentication, an attacker that has compromised a Server's secret key may wait for a Client to renew its session keys and then impersonate the Server and learn the Session Authentication Token. The attacker can then use this token to impersonate the Client to the Server.

$K4$ **- Impact** With a compromised Server secret key an attacker on an ICS network could already cause damage to an ICS system so the additional damage this attack could enable would be, in most cases, minor.

4.6 Summary

Reviewing Table 2 we see that, based on the work of [13], the only protocol not to fail any tests was DNP3-SAv5 (although we note that KCI and key uniqueness were not tested, and we plan to do this as future work). DNP3-SAv5 is only used in the power domain, therefore for electricity systems it is the clear choice.

S7Comm-Plus fails most tests, this provides one more example of the failure of security through obscurity. However our formal model does show a new result that if S7Comm-Plus devices were configured with individual secret keys (and secure crypto), then the protocol would be secure against most attacks. While we note that Siemens does not sell PLCs in this configuration, this perhaps shows a path to making these widely deployed devices more secure.

SSP-21 provides the best authentication properties and is based on the state of the art Noise protocol [32]. We only find edge case weaknesses in forward secrecy and when a shared secret has been leaked. Unfortunately this protocol is not supported by any of the main PLC manufacturers; this is perhaps a missed opportunity of ICS security.

OPC-UA is widely deployed and the analysis of [15] shows that it provides most security properties. So in the case where an ICS owner can have some confidence that none of their devices have been compromised, OPC-UA seems to be the best option from the non-TLS protocols.

5 TLS Protocols

Many unsecured ICS protocols have added security by wrapping the protocol in TLS. While TLS is widely regarded as providing a high-level of security, it has many possible configurations which affect this in practice. In this section we review ICS protocols that make use of TLS and assess the security they provide.

5.1 Protocol Overviews

The Modbus/TCP Security specification [34] describes how the TCP version of Modbus can be run over TLS to make it secure. The specification requires TLS 1.2 or higher and use of mutual TLS (mTLS, i.e., a certificate for both the server and the client, hence authenticating both parties). The only required cipher suites are TLS_RSA_WITH_AES_128_GCM_SHA256 and TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 and the only forbidden cipher suite is TLS_NULL_WITH_NULL_NULL, which means that cipher suites with no encryption, but with authentication, are allowed.

The specification recommends the use of a PKI to provision the certificates, and let each device validate the certificate of the device it is talking to. As a novel point, Modbus TLS extends the standard TLS X509 certificates with a "RoleOID" field, this can be used to allow role-based access control for devices.

BACnet/SC is the secure version of the BACnet building automation protocol, which uses TLS 1.3 [9]. Devices must be provisioned with a certificate, the related private key, and the CA certificates. The devices use mutual TLS and will use the CA certificate to validate the certificate of any party it connects to. Devices may have multiple CA certificates, in which case it will connect to devices with certificates from any of the CAs.

The need to set up and manage the CA certificate and the certificates on every device, seems well suited to networks the size of a building, and BACnet typically uses a hub-and-spoke network topology [18], which could make configuration easier. The requirement of TLS 1.3 rules out the use of any insecure cyber suites, however it also disallows any authenticated plaintext cipher suites.

CIP Security is a standard managed by ODVA, for running EtherNet/IP (Industrial Protocol) over TLS [30]. EtherNet/IP is a complex protocol with no security, that can use TCP for messaging and UDP data streams.

The CIP Security specification, adds mTLS 1.2 for the TCP traffic and DTLS for the UDP traffic. TLS 1.2 is explicitly specified to support authenticated, plaintext traffic, and devices are required to support a plaintext cipher suite. Users may add other cipher suites, including insecure ones. While it is an open standard, it is mainly used by Rockwell Automation.

The (D)TLS connections may use X509 Certificates or Pre-Shared Keys (PSK). Devices may be used with a PKI, and the specification defines a Certificate Management Object interface to interact with a local PKI. This supports pushing certificates onto devices and also devices pulling any that they need.

The CIP Security specification also defines "Security Profiles" for access control, which can be exchanged after the TLS handshake.

IEC 62351-3 is not a protocol in itself, but standardises recommendations for power system communication protocols that use TLS. The standard mandates support for TLS 1.2, and optionally for TLS 1.3. It provides as set of mandatory cipher suites for both TLS versions, and for TLS 1.2 it explicitly disallows the TLS_NULL_WITH_NULL_NULL suite, as well as those using DES or MD5. When using TLS 1.2, additional cipher suites can be used, which it recommends follow the latest guidance from the IETF, but this it not mandatory.

Authentication-only operation is only permitted in cases where the connection is already encrypted by other means, or if it is limited to within the same organisation or administrative domain. If using TLS 1.2, the standard allows the use of the TLS_RSA_WITH_NULL_SHA256 cipher suite, and despite TLS 1.3 not officially supporting any authentication-only cipher suites, the standard allows use of two suites defined in RFC 9150: TLS_SHA256_SHA256 and TLS_SHA384_SHA384. IEC 62351-8 describes the additional requirements for adding role based access control support to IEC 62351-3 certificates.

The Siemens Logo series of PLCs communicate with their configuration interface using a proprietary protocol which is optionally wrapped in TLS 1.2 via HTTPS, and has been partially reverse engineered[1].

We inspected the traffic of a Siemens Logo device and found that it uses TLS 1.2 with the strong cipher suite ECDHE-ECDSA-AES128-GCM-SHA256 using a certificate pre installed on the device and desktop controller. It does not use mutual TLS; the controller authenticates itself to the PLC using a password.

The reverse engineering of the protocol, mentioned above, found that an obfuscated CRC of the password is send to the PLC, which only provides a very week level of protection. Certificates could be removed from any device and reused by an attacker, and it would be straightforward for an attacker that MITMed the connections to reuse the password CRC.

Siemens Logo TLS traffic is always encrypted, however the TLS protection can be disabled to allow traffic analysis, though this removes all security. It does not provide an authenticated plaintext mode.

5.2 Analysis of TLS protocols

We summarise the main aspects of each of the TLS protocols in Table 3. With the exception of Siemens Logo, all the TLS protocols we looked at provide strong security guarantees based on mutual TLS that authenticates both the client and server based on a certificate or pre-shared key. However, this security requires the additional effort of provisioning devices with certificates. The access control extensions and possible use of PKIs in Modbus TLS and CIP Security would make management of large deployments easier. BACnet/SC's requirement to manually add certificates to each device may be better for smaller deployments the size of a building management system, which is the target domain for BACnet.

IEC 62351-3 specifies a similar level of security and features to Modbus TLS and CIP Security, but is targeted just at power control systems.

[1] https://github.com/jankeymeulen/siemens-logo-rest

Table 3. Comparison of TLS Protocols. mTLS: Is mutual TLS supported? Plain text: Is an authenticated plain text mode available? User certs: Can a user install their own certificates? PKI: Is there support for a PKI? PSK: Can a user install pre-shared keys? No weak ciphers: Is it impossible to use weak ciphers? PFS: Do all ciphers provide Perfect Forward Secrecy? Access Control: Is there support for profiles and permissions?

Protocol	TLS Version	mTLS	Plain text	User certs	PKI	PSK	No Weak ciphers	PFS	Access Control	Usage
Modbus TLS	1.2 +	Y	Y	Y	Y	N	N	N	Y	General
BACnet/SC	1.3	Y	N	Y	N	N	Y	Y	N	Buildings
CIP Security	1.2	Y	Y	Y	Y	Y	N	N	Y	General
IEC 62351-3	1.2+	Y	Y	Y	Y	Y	N	N	Y	Power
Siemens Logo	1.2	N	N	N	N	N	Y	N	N	General

One common issue across all the TLS protocols is around the supported TLS version. Either the protocol is restricted to using TLS 1.2, which is rapidly becoming obsolete, or it uses the latest TLS 1.3, which requires encryption and therefore blocks the use of IDS and Data Historians, which are required by a significant number of industrial applications.

ODVA and Cisco have authored RFC 9150 as an extension to TLS 1.3 to support integrity-only cipher suites, however this has not yet been widely adopted. IDSes and Data Historians could still be used if they were given the Certificate Authority private key and traffic was directed through them such that they could in effect perform a MITM attack on the TLS communications and recover the data. This opens up a number of other issues, including the risk of leaking the CA private key, significantly increased latency and increased system complexity.

This conflict may be a significant barrier to use of any of these protocols going forward, and given that the issue is only set to get worse as TLS 1.2 is further phased out, the concept of using TLS to build ICS protocols is problematic.

6 Conclusion

In this paper we provide the first direct comparison of leading, secured ICS protocols. We analyse non-TLS/bespoke ICS protocols using the Tamarin Prover. We identify OPC-UA as one of the most useful, secure protocols but highlight some minor security issues, such as lack of forward secrecy and key compromise impersonation attacks, however these are unlikely to be major issues in most ICS deployments.

We summarise the leading TLS-based ICS proposals and show that their designs promise a high level of security, with better protection than OPC-UA. However, there may be issues with the use of TLS 1.3 meaning that all data is encrypted, so cannot be used for IDS and ICS analysis. Protocols based on TLS 1.2 risk becoming outdated and unsupported, and as the proposal for plaintext TLS 1.3 has yet to gain traction, TLS based ICS protocols may face obstacles in the future, making OPC-UA one of the best options for ICS operators.

References

1. Alsabbagh, W., Amogbonjaye, S., Urrego, D., Langendörfer, P.: A stealthy false command injection attack on modbus based scada systems. In: 2023 IEEE 20th Consumer Communications & Networking Conference (CCNC) (2023)
2. Alsabbagh, W., Kim, C., Langendörfer, P.: Good night, and good luck: A control logic injection attack on openplc. In: IECON 2023-49th Annual Conference of the IEEE Industrial Electronics Society (2023)
3. Alsabbagh, W., Langendöerfer, P.: A new injection threat on s7-1500 plcs-disrupting the physical process offline. IEEE Open Journal of the Industrial Electronics Society (2022)
4. Alsabbagh, W., Langendörfer, P.: A control injection attack against s7 plcs-manipulating the decompiled code. In: IECON 2021–47th Annual Conference of the IEEE Industrial Electronics Society (2021)
5. Alsabbagh, W., Langendörfer, P.: A stealth program injection attack against s7-300 plcs. In: 2021 22nd IEEE International Conference on Industrial Technology (ICIT) (2021)
6. Alsabbagh, W., Langendörfer, P.: No need to be online to attack-exploiting s7-1500 plcs by time-of-day block. In: 2022 XXVIII International Conference on Information, Communication and Automation Technologies (ICAT) (2022)
7. Alsabbagh, W., Langendörfer, P.: Patch now and attack later - exploiting s7 plcs by time-of-day block. In: International Conference on Industrial Cyber-Physical Systems (ICPS) (2021)
8. Amoah, R., Çamtepe, S., Foo, E.: Formal modelling and analysis of dnp3 secure authentication. In: Journal of Network and Computer Applications (2016)
9. ANSI/ASHRAE: A data communication protocol for building automation and control networks- addendum bj to standard 135-2016. In: https://bacnet.org/wp-content/uploads/sites/4/2022/08/Add-135-2016bj.pdf (2019)
10. Barbieri, G., Conti, M., Tippenhauer, N.O., Turrin, F.: Assessing the use of insecure ics protocols via ixp network traffic analysis. In: Proceedings of International Conference on Computer Communications and Networks (ICCCN) (2021)
11. Beresford, D.: Exploiting siemens simatic s7 plcs. Black Hat USA (2011)
12. Biham, E., Bitan, S., Carmel, A., Dankner, A., Malin, U., Wool, A.: Rogue7: Rogue engineering-station attacks on s7 simatic plcs. In: BlackHat (2019)
13. Cremers, C., Dehnel-Wild, M., Milner, K.: Secure authentication in the grid: A formal analysis of dnp3: Sav5. In: ESORICS (2017)
14. Dahlmanns, M., Lohmöller, J., Pennekamp, J., Bodenhausen, J., Wehrle, K., Henze, M.: Missed opportunities: Measuring the untapped tls support in the industrial internet of things. In: Proceedings of Asia Conference on Computer and Communications Security (2022)
15. Diemunsch, V., Hirschi, L., Kremer, S.: A comprehensive formal security analysis of opc ua. In: USENIX Security Symposium (2025)
16. Dragos, Inc.: Crashoverride: Analysis of the threat to electric grid operations. Tech. rep., Dragos, Inc. (Jun 2017), https://dragos.com/blog/crashoverride/, accessed: 2025-06-19
17. Erba, A., Müller, A., Tippenhauer, N.O.: Resting on feet of clay: Securely bootstrapping opc ua deployments. In: BlackHat Europe (2021)
18. Fisher, D., Isler, B., Osborne, M.: BACnet secure connect a secure infrastructure for building automation. In: https://modbus.org/docs/MB-TCP-Security-v36_2021-07-30.pdf (2020)

19. Gebhard, A., Perouli, D.: Comparing the security approaches of CIP and OPC UA. In: Re-Design Industrial Control Systems with Security (2024)
20. Ghaleb, A., Zhioua, S., Almulhem, A.: On plc network security. International Journal of Critical Infrastructure Protection (2018)
21. Girol, G., Hirschi, L., Sasse, R., Jackson, D., Cremers, C., Basin, D.: A spectral analysis of noise: A comprehensive, automated, formal analysis of Diffie-Hellman protocols. In: USENIX Security Symposium (2020)
22. Greenberg, A.: How russia-linked malware cut heat to 600 ukrainian buildings in deep winter. Wired (Jul 2024), https://www.wired.com/story/russia-ukraine-frostygoop-malware-heating-utility/, accessed: 2025-06-19
23. Helms, J., Van Randwyk, J., Taymaz, S., Weed, J., McClelland, C.: California energy systems for the 21st century (ces-21) program (final report). Tech. rep., Lawrence Livermore National Lab (2022), https://www.osti.gov/biblio/1880939
24. Hildebrandt, M., Lamshöft, K., Dittmann, J., Neubert, T., Vielhauer, C.: Information hiding in industrial control systems: An opc ua based supply chain attack and its detection. In: Information Hiding and Multimedia Security (2020)
25. Holasova, E., Blazek, P., Fujdiak, R., andJiri Misurec, J.M.: Exploring the power of convolutional neural networks for encrypted industrial protocols recognition. In: Sustainable Energy, Grids and Networks (2024)
26. Hui, H., McLaughlin, K., Sezer, S.: Vulnerability analysis of s7 plcs: Manipulating the security mechanism. International Journal of Critical Infrastructure Protection (2021)
27. Johnson, B., Caban, D., Krotofil, M., Scali, D., Brubaker, N., Glyer, C.: Attackers deploy new ics attack framework "triton" and cause operational disruption to critical infrastructure. Tech. rep., Mandiant (Dec 2017), https://cloud.google.com/blog/topics/threat-intelligence/attackers-deploy-new-ics-attack-framework-triton, accessed: 2025-06-19
28. Krotofil, M., Derbyshire, R.: Greetings from the '90s: Exploiting the design of industrial controllers in modern settings. In: BlackHat Europe (2021)
29. Lei, C., Donghong, L., NS-Focus, M.L.: The spear to break the security wall of s7commplus. In: BlackHat Europe (2017)
30. ODVA: Overview of CIP security. In: https://www.odva.org/wp-content/uploads/2023/07/PUB00319R2_CIP-Security-At-a-Glance.pdf (2020)
31. OPC Foundation: Opc foundation marketplace, https://opcfoundation.org/products/, accessed: 2025-06-19
32. Perrin, T.: The noise protocol framework. noiseprotocol, Protocol Revision (2018)
33. Puys, M., Potet, M., Lafourcade, P.: Formal analysis of security properties on the OPC-UA SCADA protocol. In: Computer Safety, Reliability, and Security SAFECOMP (2016)
34. Schneider Electric USA, I.: Modbus/tcp security protocol specification. In: https://modbus.org/docs/MB-TCP-Security-v36_2021-07-30.pdf (2018)
35. Secure scada protocol for the 21st century (ssp21) (2020), https://ssp21.github.io/spec/latest/pdf/ssp21.pdf
36. Tychalas, D., Maniatakos, M.: Special session: Potentially leaky controller: Examining cache side-channel attacks in programmable logic controllers. In: International Conference on Computer Design (ICCD) (2020)
37. Walz, A., Niemann, K.H., Göppert, J., Fischer, K., Merklin, S., Ziegler, D., Sikora, A.: Profinet security: A look on selected concepts for secure communication in the automation domain. In: Conference on Industrial Informatics (INDIN) (2023)
38. Wardak, H., Zhioua, S., Almulhem, A.: Plc access control: A security analysis. In: World Congress on Industrial Control Systems Security (WCICSS) (2016)