Nope, S7ill not Secure: Stealing Private Keys From S7 PLCs

Nadav Adir Eli Biham Sara Bitan Alon Dankner Ron Freudenthal
Or Keret

Faculty of Computer Science, Technion, Haifa 3200003, Israel
{nadav.adir,biham,sarab,dankner,ronf,or.keret}@cs.technion.ac.il

Abstract. Recent years have seen a growing trend within industrial control systems (ICS) towards adopting secure protocols over unencrypted ones. The last versions of Siemens' S7 PLCs use TLS 1.3 to secure the S7 protocol, which is used for communication with the engineering stations and the SCADA systems. Unfortunately, while trying to improve its security, S7 became highly vulnerable. In this work, we introduce a series of six new attacks against the S7 protocol. We reverse-engineered the S7 protocol and analyzed it. We found a severe new vulnerability that surprisingly enables the attacker to steal the PLC private key, by politely asking the PLC itself. We then leveraged the extracted private key to design and implement various attacks, such as PLC impersonation and man-in-the-middle. The attacker can, for example, inject a malicious PLC program while reporting legitimately-looking values to the users, reincarnating the OT features of Stuxnet on the SIMATIC product line newest version. Furthermore, we present attacks against the S7 PKI, exploiting the certificate provisioning. The combination of the presented attacks constitutes a powerful toolset, allowing an attacker to completely takeover the ICS system, while hiding the attack from the users. The impact of this study is far-reaching, as it introduces a direct and immediate threat to ICS customers and critical infrastructures, exemplifying stealth attacks on a secure ICS.

1 Introduction

Industrial Control Systems (ICS) are vital to our modern life, ensuring the reliable and continuous operation of critical infrastructures such as power plants, water treatment facilities, and transportation systems. The core of the ICS is the Programmable Logic Controller (PLC), communicating with engineering stations and SCADA HMIs on one side, and controlling industrial systems on the other side. The security of ICS systems has received much attention in recent years due to the criticality of these systems and the potential damage of a cyber attack. The outcomes range from physical harm to economic losses, and in severe situations, even loss of human life and risks to public safety. However, as ICS systems have become more interconnected and integrated with corporate networks, they have also become exposed to cybersecurity threats.

Over the past few years, many ICS vendors have incorporated standard security protocols into their industrial products. In the recent releases of the S7 PLCs, Siemens deployed TLS 1.3 [14] in the S7 protocol, which is used by the S7 PLCs [17,18], to address numerous weaknesses in the protocol. Ultimately, while trying to improve the security of the protocol, the S7 protocol became highly vulnerable.

In this work, we introduce six attacks that exploit new vulnerabilities in the latest version of S7. We unveil the security mechanisms of the new S7 protocol by reverse engineering the protocol and the PKI. We found a severe new vulnerability in the S7 protocol, in the PLC configuration, that surprisingly allows an attacker to retrieve the private key directly from the PLC. Utilizing the extracted private key, we proceeded to design and implement

various attacks, including a PLC impersonation and a stealth man-in-the-middle attack. Our stealth man-in-the-middle attack tampers with the PLC logic to damage production, while modifying values of read/write HMI variables, in order to deceive the operators and camouflage the attack. The uniqueness of this attack is that it is capable of reincarnating the OT features of Stuxnet, even against the newest version of the SIMATIC product line. Furthermore, we show that the initial provisioning of the PLC's keys and certificates is insecure, providing an attacker with another way to steal the private key, and even a password that should protect it.

1.1 Our Contributions

We introduce six new cyber attacks that exploit new vulnerabilities in the latest version of Siemens S7 protocol. Siemens PLCs are estimated to have 31% of the global PLC market [4]. The presented attacks include:

1. Attack #1: Extracting the PLC's Secrets During the Initial Key Provisioning: exploits the initial key provisioning to steal the private key, and even the password that should protect it.
2. Attack #2: Rogue Client: an impersonation of a client that enables the attacker to send arbitrary control commands to the PLC.
3. Attack #3: PLC Private Key Retrieval Attack: a severe new vulnerability in the S7 upload configuration feature, meant for retrieval of information from the PLC, enables the attacker to steal the PLC's private key.
4. Attack #4: Malicious Control Program Injection: exploits the program encryption to inject a malicious control program to the PLC.
5. Attack #5: Rogue PLC: utilizes the stolen private key to perform impersonation of a PLC (an attack that was not possible in the previous S7 version).
6. Attack #6: Stealth Man-in-the-Middle Attack: combines client and server impersonation, enabling the attacker to read and manipulate values inside the TLS 1.3 payload on live S7 sessions.

The combination of these attacks forms a powerful toolset that allows an attacker to completely take over the ICS system, while hiding the attack from the operators and engineers.

Another contribution of this work is an open-source TLS sniffer tailored for the S7 protocol. We developed this tool to research the protocol and analyze the traffic, as existing sniffing/proxy tools were incompatible with the S7 protocol's stack.

Our threat model assumes an attacker with access to the ICS network of the TIA workstations and the S7 PLCs. We implemented our attacks using Python scripts and C code, and successfully tested them against the S7-1500 PLC with firmware v2.9 and TIA v17. The S7-1200 PLCs employ the same protocol from firmware v4.5. Prior to this paper submission we disclosed our findings to Siemens.

1.2 Structure of this Paper

The remaining part of the paper proceeds as follows: We review related work and the current state of ICS security in Section 2. Section 3 provides necessary background on key concepts. In

Section 4, we explain the new S7 PKI in details. Section 5 describes our research methodology and tools. Throughout Sections 6 to 8, we introduce our six new attacks and explain the vulnerabilities they exploit. Section 9 addresses mitigations, and lastly, Section 10 concludes the paper.

2 Related Work

Several research efforts have contributed to the understanding of security challenges associated with Siemens SIMATIC S7 and identified vulnerabilities in their protocol and system.

Stuxnet worm [11,6] discovered in 2010 targeted Siemens S7 PLCs to cause physical damage to nuclear centrifuges. It maliciously reprogrammed PLCs to alter the industrial process, while concealing its activity by displaying to operators counterfeit data and the original program (through Windows, not through attack on the PLC).

Beresford [1] exposed protocol vulnerabilities in older PLC versions (S7-300 v2.3.4, S7-400, and S7-1200 v2.2). This research highlighted the lack of cryptographic protection in the protocol, leading to various weaknesses, including replay attacks and authentication bypass.

In [12], Cheng et al. analyzed the S7-1200 (v4.0) and S7-1500 PLCs. The authors found proprietary encryption algorithms that did not use private keys, enabling a replay attack against the protocol.

Biham et al. [2] examined the S7-1200 and S7-1500 (v2.1) PLCs and found the use of hard-coded keys and server-only (PLC) authentication. The paper presents a rogue engineering station (rogue client) attack that reprograms the PLC with a malicious control logic, while causing the PLC to report a false and legitimately-looking logic upon request.

Another publication [19] uncovered a memory vulnerability in the S7-1200 (v4.5.0) and S7-1500 (versions preceding v2.9.2) PLCs, allowing an attacker to gain a remote read/write access to the PLC memory, and to extract the above hard-coded keys.

Jian [8] investigated the S7-1500 PLCs (v2.9.2) that incorporate TLS 1.3 in the S7 protocol. The authors presented a rogue client attack implemented a fuzzer to bypass the access protection (password) and execure denial-of-service attack.

3 Background

The SIMATIC S7 ecosystem consists of several key components. The S7 PLCs orchestrate complex sequences of operations with precision and reliability according to a programmable logic (the control program). They monitor inputs from sensors and control a wide array of industrial equipment, including conveyor belts, robotic arms, and more.

Engineers use the engineering stations to program the PLCs, whereas the operators monitor and control the industrial process in real time through SCADA (Supervisory Control And Data Acquisition) stations. The TIA (Totally Integrated Automation) Portal is Siemens' engineering software. Communication between the system components occurs over TCP/IP.

3.1 The S7CommPlus Protocol

The S7CommPlus protocol (commonly referred to as the S7 protocol) is an industrial communication protocol developed by Siemens for its SIMATIC product line. S7 is a proprietary

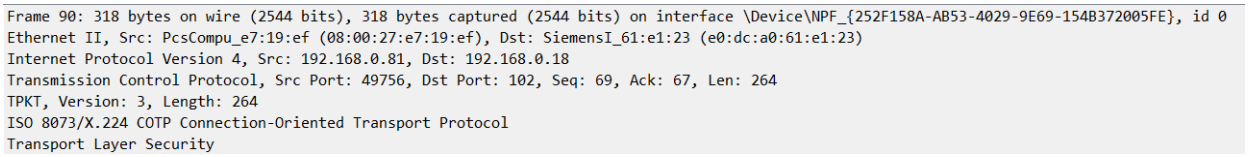


Fig. 1: An Example of an S7 Packet as Shown in Wireshark

protocol and does not have any official documentation available. The protocol enables the exchange of data, control commands, and system information necessary for real-time monitoring, control, and automation of industrial processes.

The S7 protocol follows a client-server architecture, where the client initiates requests and the server responds accordingly. In this context, the client can be an HMI, an engineering station, or another PLC, while the server typically represents the PLC.

Communication is established over the network, utilizing TCP/IP connections. The S7 protocol relies on ITOT [15] and operates through the well-known TCP port 102. It operates within a layered protocol stack, which includes Connection-Oriented Transport Protocol (COTP) [7,13] and TPKT [15]. The latest version of the S7 protocol incorporates TLS 1.3 [14] as the underlying security mechanism [17,18]. Fig. 1 shows an example of an S7 packet captured in Wireshark [22] and the protocol stack.

4 The New S7 Public Key Infrastructure

TLS utilizes digital certificates to authenticate devices, ensuring that communication occurs between trusted entities. Devices on the Internet commonly rely on trusted third-party entities known as Certificate Authorities (CAs) to sign certificates. To secure the SIMATIC ecosystem, Siemens has developed its own PKI mechanisms to address the specific requirements and challenges of ICS systems (e.g., most networks are air-gapped, with no external connections).

The latest versions of the TIA Portal (starting from v17) introduce a new tool called certificate manager [17,18]. The certificate manager facilitates the generation, import, export, and management of digital certificates for S7 PLCs. Users can leverage the certificate manager to view and manage existing certificates. This includes actions like enabling or disabling certificates, modifying certificate properties, and revoking certificates when necessary.

TIA offers three alternatives for the user to set the certificates of the PLCs:

1. Self-signed certificates generated by the TIA.
2. Certificates issued and signed by the TIA, where TIA acts as the CA.
3. External certificates imported by the user. For example, certificates issued by trusted third-party CAs or certificates generated within an organization's internal CA infrastructure.

By default, TIA uses the first option, while the second and third options require manual configuration by the user.

In all of these scenarios, the TIA either generates or receives the public and private keys for the PLC, making it the initial holder of these keys. Consequently, these keys must be

securely provisioned to the PLC over the network. One objective of this work is to gain a thorough understanding of the key provisioning process within the SIMATIC ecosystem. However, since the S7 protocol is proprietary and lacks official documentation, this process presents significant challenges.

4.1 Certificate Pinning

For each PLC, the TIA assigns a specific certificate from the Certificate Manager that the PLC is expected to use, effectively implementing a certificate pinning mechanism.

At the beginning of an S7 session, TIA checks the certificate presented by the PLC in the TLS handshake. If the presented certificate differs from the pinned certificate, the TIA alerts the user, who can then review the certificate and decide whether to trust it and proceed with the connection.

This alert message commonly appears when a new PLC is added to a TIA project or when an existing PLC switches to a new certificate. This mechanism protects the TIA, as it may indicate an attacker's certificate and expose a potential security breach. In the attack descriptions later in this paper, we demonstrate how an attacker bypasses this alert message to avoid detection.

5 Traffic Interception and Decryption

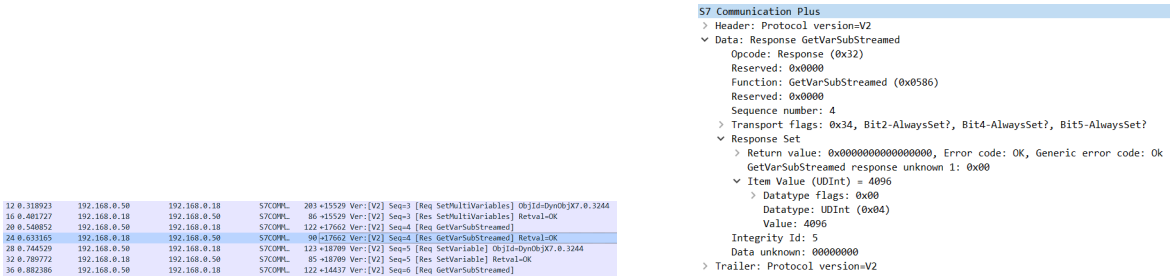
To effectively design and implement the attacks described in this work, it is crucial to gain visibility into the traffic exchanged between the TIA Portal and S7 PLCs. This investigation involves packet analysis to gain a thorough understanding of the S7 protocol's underlying structure. S7 messages can be parsed using the S7CommPlus dissector plugin for Wireshark [21], but the plugin does not work on encrypted communication.

Analysis of the S7 protocol is challenging due to several factors: the S7 messages are encrypted with TLS, the PLC is difficult to debug, and the S7 protocol is proprietary. Our initial approach was to use a standard TLS proxy, but this proved ineffective because standard TLS proxy tools do not support the COTP and TPKT protocols.

To address this, we implemented a dedicated TLS sniffer for the S7 protocol. The primary goal of this sniffer is to allow the viewing of the S7 message content. We used a TLS server to handle the encryption and decryption of communications with the TIA, and a TLS client to handle the communications with the PLC. For the purpose of the study, we used self-signed certificates in our lab environment and imported them into the TIA. However, the TLS client/server also do not support the COTP and TPKT protocols. To resolve this, we implemented encapsulation and decapsulation of COTP and TPKT within our S7 sniffer.

In total, our sniffer is capable of receiving S7 messages that are encrypted by TLS and encapsulated by COTP/TPKT, decrypting and displaying them to the user, and then forwarding them to the intended destination. The sniffer provides three methods to view the S7 content:

1. Directly within the console.
2. By exporting the TLS keys to a file, referred to as SSLKEYLOGFILE, which can be fed into Wireshark. Wireshark decrypts the TLS messages and shows the plaintext, but unfortunately does not parse the S7 messages.



(a) The Transmitted S7 Messages

(b) The Content of a Decrypted S7 Message

Fig. 2: The Generated PCAP File In Wireshark

3. By exporting the plaintext S7 messages to a new PCAP file, which can then be opened in Wireshark. The S7CommPlus dissector plugin successfully parses the S7 messages using this method.

Fig. 2 provides an example of a generated PCAP file and a plaintext S7 message.

The S7 sniffer is implemented in Python. It is a valuable tool for researchers/security professionals, enabling them to analyze encrypted traffic, study the underlying protocol, conduct security testing, and ultimately advance the understanding and defense against cyber threats.

6 Extracting the Private Key During Initial Provisioning

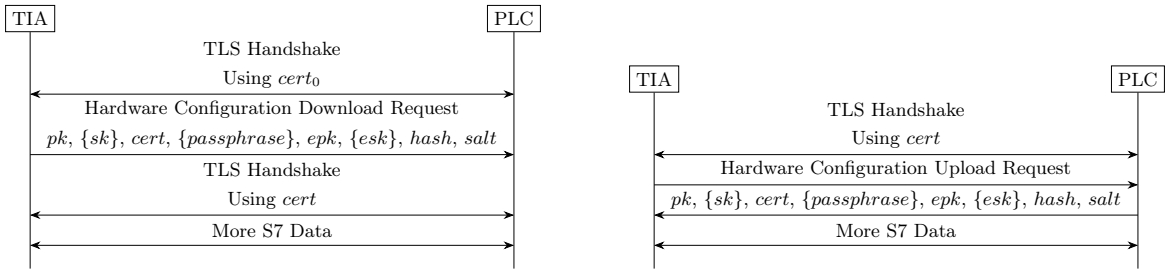
Through rigorous testing and experimentation, we designed six attacks that exploit weaknesses within the S7 protocol stack, revealing the complex security landscape surrounding ICS systems. The following sections detail our attacks, aiming to uncover the root causes of the vulnerabilities they exploit and to explore the underlying technical aspects.

6.1 Attack #1: Extracting the PLC's Secrets During the Initial Key Provisioning

Our first attack exploits the initial key provisioning from the TIA to the PLC. As previously described, the TIA either generates or receives keys from the user and provisions them to the PLC. During the initial setup, the TIA provisions the private key, public key, and certificate to the PLC in a request called hardware configuration download. The PLC configuration encompasses the settings of the PLC, such as its IP address and I/O modules, as well as the above cryptographic fields.

Suppose that the attacker successfully establishes a man-in-the-middle position prior to the initial key provisioning. During this stage, the attacker has the opportunity to capture the TLS keys intended for the PLC. This is achieved by intercepting the keys from the configuration download message, thereby compromising the security of the entire communication with the PLC.

The attack is feasible due to two primary factors: first, the PLC uses a self-signed certificate for initial key provisioning, and second, this certificate is not yet pinned to the PLC



(a) The S7 Configuration Download Protocol

(b) The S7 Configuration Upload Protocol

Fig. 3: Outline of Hardware Configuration Protocols

within the TIA, preventing the TIA from authenticating the PLC. Note that some S7 PLCs are initialized with pre-installed vendor certificates.

The outcome of this attack is unauthorized access to the TLS keys, which enables the attacker to decrypt, manipulate, or replay the communication between the PLC and connected devices. This jeopardizes the integrity and confidentiality of the control processes, potentially leading to unauthorized control of industrial operations, data breaches, or even sabotage of critical infrastructure. The compromise of TLS keys essentially nullifies the security protections that TLS is meant to provide, exposing the system to various attacks.

6.1.1 Reverse Engineering the Key Provisioning The TLS keys of the PLC are provisioned from the TIA to the PLC using the download hardware configuration protocol. These keys can also be uploaded back to the TIA (in the opposite direction) by the upload hardware configuration protocol (Attack #1 focuses solely on the download protocol).

The configuration download and upload protocols are outlined in Fig. 3. Square brackets denote an optional field, and curly brackets denote an encrypted field. The left figure shows how the TIA downloads a new configuration to the PLC and replaces the TLS private key of the PLC (during key initialization/rotation). The right figure shows how the TIA requests the existing configuration from the PLC.

The configuration message includes the PLC's certificate, along with its public and private keys (*cert*, *pk*, and *sk*, respectively). The private key is delivered in an encrypted form [10] and must be decrypted before usage. The message contains several additional fields, some of which are encrypted, that are used to decrypt the private key (*passphrase*, *epk*, *esk*, *salt*). We reverse-engineered the decryption used by TIA and implemented it in our attack tool. Appendix A provides details on the private key decryption process and the hardware configuration message fields.

The TIA allows provisioning multiple keypairs to the PLC. All keys in the certificate manager are concatenated and sent to the PLC (the figure shows only a single key), with the TIA specifying which key the PLC should use. When a CA is used (either the TIA itself or an external CA), the CA (root) certificates are transferred to the PLC along with the PLC's keypair. Multiple CA certificates are concatenated before transmission.

We identified two relevant .NET DLLs and used dnSpy [5] to reverse engineer them. The DLLs are `Siemens.Simatic.Hwcn.Basics.CsiInterop` and `Siemens.Simatic.Hwcn.`

`BusinessLogic`. They allowed us to gain insights into the internal workings and structure of the upload configuration message.

6.1.2 Fetching the PLC Hardware Configuration Password This attack compsomises another security mechanism: the configuration password, which is designed to protect the PLC configuration, and the private key in particular. When a user creates a new project in the TIA, they have the option to set up a configuration password for the PLC. If specified, this password is used in the encryption process of the PLC's private key, safeguarding the key in transit and at rest. The password must adhere to specific complexity requirements, including a minimum of eight characters with at least one lowercase letter, one uppercase letter, and one number. If the user does not set a password, a null string is automatically used as the default password, making the key easily extractable.

The configuration password is transmitted to the PLC during the initial hardware configuration download, similar to the private key provisioning. However, this transmission occurs in plaintext over a TLS session. Thus, even if a configuration password is set, a man-in-the-middle attacker can intercept it during the provisioning protocol and use it to decrypt the private key.

7 Retrieving the Private Key of a Production PLC

In this section, we describe the PLC Key Retrieval attack, which is designed to steal private keys from S7 PLCs. This attack introduces an alternative method to fetch the keys, even if the attacker misses the initial key provisioning (i.e., completely independent of the first attack). The description starts with rogue client (Attack #2) and then moves to the PLC Key Retrieval attack (Attack #3).

7.1 Attack #2: Rogue Client

The rogue client exploits vulnerabilities in the authentication process to gain unauthorized access and potentially manipulate industrial processes. This attack vector leverages the absence of client certificates, which are used to verify the authenticity of clients in TLS communications.

The rogue client initiates a connection with the PLC, pretending to be a legitimate station. During the TLS handshake process, the PLC does not request the client to present a certificate for authentication. Without client certificate verification, the PLC cannot distinguish between legitimate clients and malicious actors, allowing the attacker to complete the handshake successfully.

Once a secure channel is established, the attacker can send malicious requests to the PLC, impersonating a legitimate client. These requests can include unauthorized actions, requests for sensitive data, or any other operation supported by the PLC. As the attacker controls the PLC, the impact of this attack can be severe, including production shutdowns, environmental hazards, safety risks, or even injury to personnel.

Fig. 4: The Key Provisioning Protocol in Wireshark

7.1.1 Attack Implementation The rogue client attack is implemented as an S7 client. It sends to the PLC a sequence of predefined S7 messages of the requested action and parses the PLC's responses.

For this purpose, we recorded in advance S7 sessions in our lab, between a TIA station and an S7 PLC. We used our S7 sniffer to get the plaintext S7 messages (i.e., the TLS data after decryption). These messages are then sent one by one to the PLC via a TLS client, which completes the TLS handshake and encapsulates the messages within the TLS protocol.

A key distinction between the latest, TLS-secured protocol and its predecessor, which relied on Siemens' proprietary cryptographic protocol, is the openness of the cryptographic framework. Unlike the proprietary protocol, TLS benefits from publicly available source code, allowing the community to review it and identify vulnerabilities. However, this transparency is a double-edged sword; it also grants potential attackers access to the same code, potentially facilitating the development of attack tools.

7.2 Attack #3: PLC Private Key Retrieval Attack

The PLC Key Retrieval attack is capable of extracting private keys directly from the PLC. This attack exploits a new and severe vulnerability in the S7 protocol in an S7 feature used for remote PLC configuration. As seen before, the TIA can ask the PLC for its hardware configuration, a request called configuration upload.

We reverse-engineered the S7 protocol and analyzed the S7 messages exchanged during a configuration upload from the PLC to the TIA. Surprisingly, we found that in the new version of the S7 protocol, the configuration contains the public key, certificate, and crucially, the private key. Fig. 4 shows the keys in transit as seen with Wireshark. Hence, when the TIA asks the PLC for its hardware configuration, the PLC sends its private key as part of the configuration in a TLS-protected message.

The private key is encrypted [10], as shown in the figure, and protected by an ephemeral key (as in Attack #1). However, we exposed the private key, and cracked the encryption, as described in more details in Appendix A.

This vulnerability, combined with the lack of client authentication, enables an attacker to impersonate the TIA and retrieve the PLC's private key. By leveraging the capability to issue commands to the PLC, the attacker sends a request to the PLC for its hardware configuration. Since the PLC does not verify the client's identity, as seen in the previous attack, the PLC treats the communication as if it were with a legitimate TIA station. Therefore, in response, the PLC sends its hardware configuration, which contains the private key. The attacker bypasses the TLS protection and fetches the private key of the PLC, from the PLC itself. All you have to do is ask.

If a configuration password is set to protect the TLS private key, our attack tool requires this password to decrypt the private key. This limitation is addressed in Section 6 that shows how an attacker can steal the password.

Exploiting this vulnerability, an attacker could gain unauthorized access to the PLC's TLS keys, compromising the integrity, confidentiality, and availability of critical processes.

8 A Stealth Man-in-the-Middle Attack

Having unveiled the vulnerabilities and methods to obtain the PLC's private keys, this section delves into the details of our stealth man-in-the-middle attack. This section details three attacks: malicious control program injection (Attack #4), rogue PLC (Attack #5), and stealth man-in-the-middle attack (Attack #6).

8.1 Attack #4: Malicious Control Program Injection

This control logic program is a set of instructions written to control and automate machinery or processes. It dictates how the PLC manages various tasks and operations based on input from the field (sensor data). It is typically developed using dedicated programming languages such as ladder logic (LD), function block diagram (FBD), and structured text (ST). The program is downloaded to the PLC's memory and runs continuously to monitor and control the connected equipment according to the programmed logic. It is critical for ensuring the safe and efficient operation of automated systems.

Siemens encrypts the control programs transmitted from the TIA to the PLC over the network. Control programs are encrypted by AES128-CTR with counter incrementation of LFSR [20], using Siemens' own implementation (rather than a standard library).

Through reverse engineering, we discovered that TIA derives the AES key from a random seed and then encrypts this seed under a common hard-coded key, which is shared between PLC models. This is a highly simplified description of the process; additional technical details have been omitted for clarity. In program download, TIA sends the encrypted key along with the encrypted program to the PLC over the TLS session. We analyzed a DLL of TIA called `Siemens.Simatic.Lang.MC7Codegenerator` using IDA Pro, WinDBG, and dnSpy.

Siemens' encryption scheme used in Siemens PLCs is vulnerable to a replay attack. Hence, a program that was originally compiled and encrypted for one PLC is accepted by a second PLC. This vulnerability, the use of a common hard-coded key, allows an attacker to reuse a previously compiled and encrypted program. This enables an attacker to create in advance a malicious control program, encrypt it as if it were intended for a PLC in the attacker's lab, and then replay it to any other victim PLC. The attacker injects unauthorized code into the PLC, potentially compromising the entire industrial control process.

8.2 Attack #5: Rogue PLC

This attack uses the PLC's TLS keys, achieved by either Attack #1 or Attack #3. The attacker sets up a rogue PLC to intercept communication destined for the legitimate PLC. The rogue PLC mimics the legitimate PLC, establishing a seemingly-secure connection with clients by presenting valid certificates and keys during the TLS handshake.

Recall that TIA uses certificate pinning to check whether a PLC certificate is authentic or not. If the presented certificate does not match the pinned certificate, TIA alerts the user. The rogue PLC bypasses the certificate pinning and the alert message, since it uses the correct certificate of the PLC and the corresponding private key in the TLS handshake.

As a result, the rogue PLC can successfully forge responses for the client, potentially causing SCADA systems to display false values. Furthermore, a rogue PLC may access sensitive data transmitted to it, including intellectual property (production data, proprietary algorithms, or industrial secrets), credentials, and cryptographic keys.

8.3 Attack #6: Stealth Man-in-the-Middle Attack

Combining all the attacks we have seen so far, we design a stealth man-in-the-middle attack that impersonates both the client and the server. The man-in-the-middle attack intercepts and manipulates protocol messages between the TIA and the PLC.

The attacker hijacks messages sent from the TIA to the PLC and acts as a relay to deliver them to the PLC. Likewise, messages sent from the PLC to the TIA are intercepted and sent to the TIA by the attacker. As a result, the attacker can read and manipulate the intercepted S7 messages. This manipulation may involve modifying commands, altering data values, injecting malicious code or commands, or even fabricating false responses to deceive both ends of the communication.

The S7 communication is protected with TLS, i.e., the S7 messages are encrypted and authenticated. Therefore, the attacker acts as a TLS server (as explained in Attack #5) to handle the communication with the TIA, and a TLS client (as explained in Attack #2) to handle the communication with the PLC. To work successfully, the TLS server uses the private key, public key, and certificate, which can be acquired by Attack #1 or Attack #3.

The man-in-the-middle attack operates as follows:

1. Steal the PLC's keys and certificate (Attack #1 or Attack #3).
2. Wait for a connection initiation from the TIA.
3. Complete the TLS handshake with the TIA using the stolen keys and certificate.
4. Initiate a connection with the PLC and complete the TLS handshake (client certificates are not required).
5. Intercept a TLS-encrypted message from the TIA containing an S7 request.
6. Decapsulate TPKT and COTP headers.
7. Authenticate and decrypt the message (by a TLS server, Attack #5), resulting in an S7 request in plaintext.
8. Read and/or modify the content of the S7 message according to the attacker's logic and intentions.
9. Re-encrypt and re-authenticate the message (by a TLS client, Attack #2), ensuring that the attack remains undetected.
10. Encapsulate the TLS message with COTP and TPKT.
11. Send the message to the PLC, and wait for the PLC's response.
12. Repeat the process for the PLC's response.

By leveraging the man-in-the-middle attack, the attacker can modify any bit inside the TLS payload, and the results can be disastrous. The attacker can revive the ferocious OT

characteristics of Stuxnet against the newest version of the S7 protocol. It can modify the control program of the PLC, leading the PLC to execute a malicious program and operate the industrial equipment in an unintended and potentially harmful way (Attack #4). At the same time, the attacker manipulates the HMI values sent to the TIA or SCADA stations, leading to legitimatly-looking but wrong displays for the operators. This technique can be used to damage the industrial system and camouflage malicious activity.

The attack is resilient to legitimate key replacements performed by the TIA; the attacker can detect and adapt to new keys, ensuring continued control and stealth.

9 Mitigations

To counter the vulnerabilities highlighted in this study, several essential measures can be implemented.

1. Security of Private Keys: Transmissions of private keys expose them to potential interception and unauthorized access. The PLC should not send its TLS private key over the network.
2. Full Mutual Authentication: ICS systems must implement full mutual authentication. This involves the PLC enforcing client authentication, e.g., by validating the client certificate. This ensures that only trusted entities establish a connection with the PLC and prevents unauthorized access.
3. Avoiding Self-Signed Certificates for Key Initialization: PLCs should have pre-installed certificates issued by a trusted CA (e.g., the vendor) for the initial key provisioning, rather than relying on self-signed certificates.
4. Use Cryptographic Key Exchange Protocols: Instead of sending encryption keys or passwords with ciphertext (a practice considered insecure), key exchange protocols should be employed to securely create shared secrets between parties.

While the vendors play a critical role in enhancing the security, ICS customers can also take immediate actions to prevent the presented attacks:

1. Set the PLC configuration password.
2. Provision keys and passwords in a secure environment, i.e., utilizing dedicated and hardened systems for configuring PLCs, separated from the regular network.
3. Set CPU access protection password [16], which restricts access to certain privileged operations.

Despite these measures, several challenges make this process impractical in complex ICS deployments with a large number of PLCS, TIA stations, and users:

1. Configuring all PLCs with keys, certificates, and passwords in a secure environment is resource-intensive and may not scale effectively in large-scale ICS deployments.
2. The customers are required to establish three distinct passwords on their end: a PLC configuration password, a TIA user password (a user password defined in the project) if the TIA is the CA, and a CPU access protection password.
3. The selected passwords must be complex and unique for each PLC/user, making password management highly difficult.

4. Securing the control program with know-how protection [16] adds another password to manage.

Given these challenges, it is conceivable that the vulnerabilities identified in this study could be exploited in existing ICS systems.

To best address the presented attacks, we recommend a redesign of the security architecture. The K7 protocol [3] offers a suitable solution, where a dedicated server stores and manages device credentials and provides authentication and authorization mechanisms.

10 Discussion

Prior studies investigated the security of the Siemens S7 protocol. The Rogue7 attack [2] showed that remote attackers can issue malicious commands to PLCs, and that all PLCs (in the studied versions) shared a hard-coded private key. Despite numerous attempts by researchers worldwide, extracting this key proved challenging until a remote code execution exploit enabled its extraction [19]. These weaknesses drove Siemens to secure S7 with TLS 1.3 [17,18] and replace the shared hard-coded key with unique private keys for each PLC.

However, our findings suggest that the introduction of TLS is not a cure-all. Poor key management design can introduce significant security vulnerabilities. The aim of the present research is to examine the resiliency of the SIMATIC ecosystem to network-based attacks.

We designed six attacks targeting the most recent version of Siemens S7 PLCs and discovered a critical vulnerability: the PLC surprisingly sends its private key in response to a configuration upload request, without verifying the requestor's identity. This design flaw enables an attacker to simply ask for the key and retrieve it. This is a unique ability that rogue client attacks [2,8] have not presented so far. These attacks were limited to a client impersonation, and did not find the private key nor show PLC impersonation/man-in-the-middle. It is unexpected that such an attack is possible in the most recent version of the system, especially for the most critical systems. These vulnerabilities may lead to operational downtime, damage to equipment and raw materials, and in some cases, pose risks to public safety and national security.

This is the first study to demonstrate the extraction of TLS keys from the S7 PLCs and to examine the new S7 public key infrastructure (PKI). Contrary to expectations, upgrading to a new version may actually downgrade the overall security level. Our research shows that flawed deployment of TLS 1.3 exposes Siemens' ICS to attacks that were not feasible in previous versions that do not use TLS (see Table 1 for a comparison of the discussed protocols).

On a positive note, some limitations are worth noting. Customers can prevent these attacks by setting configuration passwords and provisioning keys in secure environments. However, in large-scale deployments with a vast array of PLCs, this practice is challenging to accomplish. A more robust and scalable solution would involve redesigning the security architecture, incorporating mechanisms such as those proposed in [3], which align with the unique needs of ICS systems.

#	Attack	S7 without TLS	S7 over TLS
1	Extracting PLC secrets during initial download	×	✓
2	Rogue client	✓	✓
3	PLC private key retrieval	×	✓
4	Malicious control program injection	✓	✓
5	Rogue PLC	×	✓
6	Stealth MITM attack	×	✓

Table 1: S7 Versions Comparison

Acknowledgments

We would like to acknowledge Michael Blum and Evyatar Ziv for their contributions to this work and their collaborative efforts.

References

1. Dillon Beresford. Exploiting siemens SIMATIC s7 PLCs. *Black Hat USA*, 16(2):723–733, 2011.
2. Eli Biham, Sara Bitan, Aviad Carmel, Alon Dankner, Uriel Malin, and Avishai Wool. Rogue7: Rogue engineering-station attacks on s7 simatic plcs. *Black Hat USA*, 2019, 2019.
3. Eli Biham, Sara Bitan, and Alon Dankner. K7: A protected protocol for industrial control systems that fits large organizations. In *Sixth Annual Industrial Control System Security (ICSS) Workshop*, pages 1–12, 2020.
4. Statista Research Department. Global PLC market share as of 2017, by manufacturer. `https://www.statista.com/statistics/897201/global-plc-market-share-by-manufacturer/`, September 2023.
5. dnSpy. dnspy. `https://github.com/dnSpy/dnSpy/releases`, 2020.
6. Nicolas Falliere, Liam O Murchu, and Eric Chien. W32. stuxnet dossier. *White paper, Symantec Corp., Security Response*, 5(6):29, 2011.
7. International Organization for Standardization. ISO 8073: Information processing systems – open systems interconnection – connection oriented transport protocol specification. Standard, August 1986.
8. Gao Jian. Fuzzing and breaking security functions of SIMATIC PLCs. *Black Hat Europe*, 2022, 2022.
9. Burt Kaliski. PKCS #5: Password-based cryptography specification, version 2.0. RFC 2898, September 2000.
10. Burt Kaliski. Public-Key Cryptography Standards (PKCS) #8: Private-Key Information Syntax Specification Version 1.2. RFC 5208, May 2008.
11. Ralph Langner. Stuxnet: Dissecting a cyberwarfare weapon. *IEEE Security & Privacy*, 9(3):49–51, 2011.
12. Cheng Lei, Li Donghong, and Ma Liang. The spear to break the security wall of s7commplus. *Blackhat USA*, 2017.
13. Alex McKenzie. ISO transport protocol specification ISO DP 8073. RFC 905, 1984.
14. Eric Rescorla. The transport layer security (TLS) protocol version 1.3. Technical report, 2018.
15. Marshall T Rose and Dwight E Cass. ISO transport service on top of the TCP, version: 3. RFC 1006, 1987.
16. Siemens. Security with SIMATIC controller. `https://support.industry.siemens.com/cs/attachments/90885010/77431846_Security_SIMATIC_DOKU_V20_en.pdf`, March 2016.
17. Siemens. Using certificates with TIA portal. `https://support.industry.siemens.com/cs/document/109769068/using-certificates-with-tia-portal`, September 2019.
18. Siemens. Configuration of TLS-based PG/HMI communication and the protection of confidential PLC configuration data. `https://support.industry.siemens.com/cs/ww/en/view/109798583`, November 2021.
19. Team82. The race to native code execution in PLCs: Using RCE to uncover siemens SIMATIC s7-1200/1500 hardcoded cryptographic keys. `https://claroty.com/team82/research/the-race-to-native-code-execution-in-plcs-using-rce-to-uncover-siemens-simatic-s7-1200-1500-hardcoded-cryptog` October 2022.
20. Roy Ward and Tim Molteno. Table of linear feedback shift registers. *Datasheet, Department of Physics, University of Otago*, 2007.
21. Thomas Wiens. S7comm wireshark dissector plugin, 2014.
22. Wireshark. `https://www.wireshark.org/`.

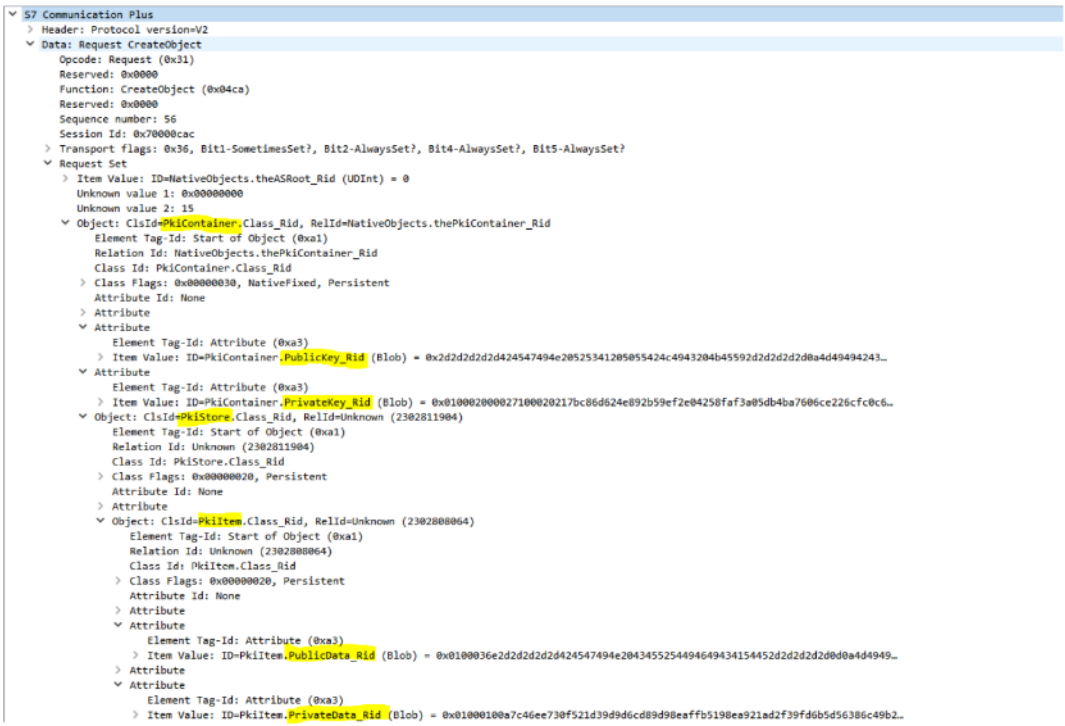


Fig. 5: The Hardware Configuration Message, as Seen in Wireshark

A Details of the Private Key Decryption

In the following section, we present a detailed description of the decryption process to obtain the PLC private key. We also explain the content of the hardware configuration message and the fields that are used in the decryption.

The configuration message is generated by the TIA during the hardware configuration download and is then sent to the PLC. Upon hardware configuration upload, the PLC sends the configuration message to the client. The message comprises multiple fields, facilitating the decryption process for the client to acquire the private key:

- *pk*, *sk* the encrypted private key of the PLC and its corresponding public key.
- *cert* the public key certificate of the PLC.
- *passphrase* an encrypted random string.
- *epk*, *esk* an ephemeral public-private RSA keypair. *esk* is also in an encrypted form.
- *hash* a SHA-1 digest calculated on the protocol version (0x0002), the public key, and the certificate fingerprint.
- *salt* a random salt.
- *pwd* the configuration password. If the configuration password is disabled, its default value is an empty string “”. The password is included in the configuration message when downloading the configuration to the PLC, but is not present when uploading the configuration from the PLC to the client.

See Fig. 5 for a screenshot of the message from Wireshark.

Upon receiving the configuration message, the client follows these steps to decrypt *sk*:

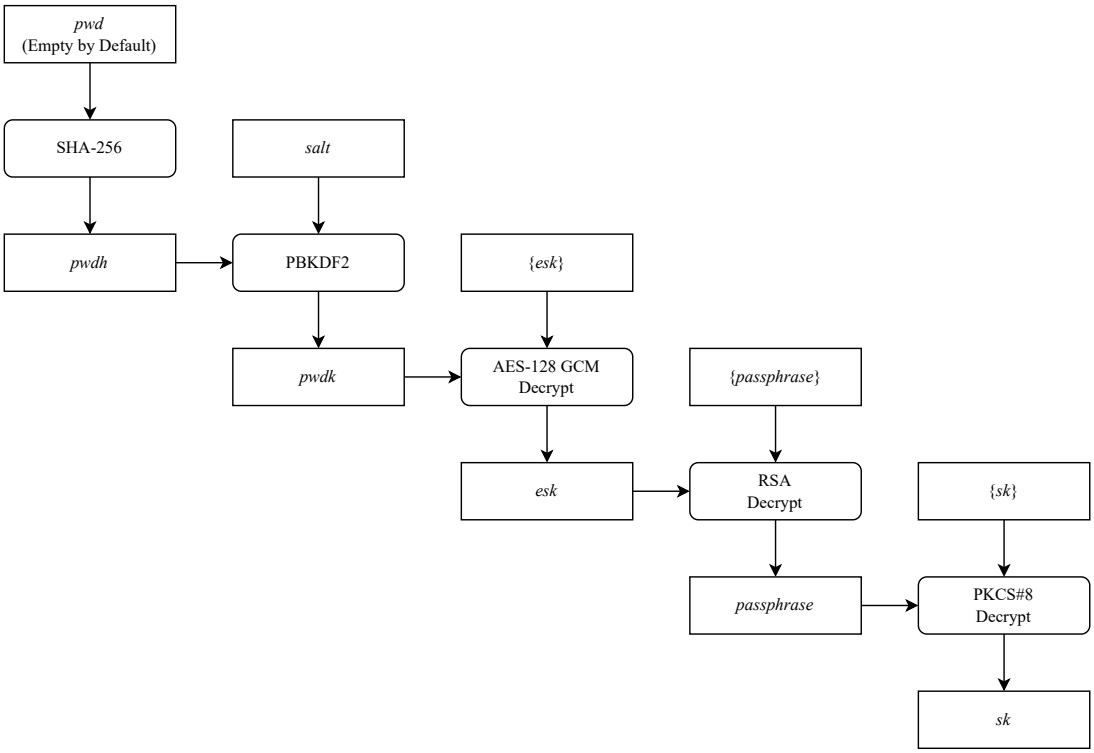


Fig. 6: The Decryption Process of the PLC Private Key

1. Deriving an AES-128 key from *salt* and *pwd*. The key derivation is done using PBKDF2 [9] with HMAC-SHA-256 as follows:

$$\text{PBKDF2}(\text{HMAC-SHA-256}, \text{SHA-256}(pwd), salt, 10{,}000, 0\text{x}2\text{c}).$$

2. Decrypting *esk* using the AES-128 key in GCM mode.
3. Decrypting *passphrase* using the decrypted *esk*.
4. Using the decrypted *passphrase* to decrypt *sk* under PKCS#8 [10].

Fig. 6 illustrates the client's decryption process. In a download hardware configuration, the TIA generates the above fields and encrypts the keys in a symmetric manner.

B Control Program Decryption

As described before, the encrypted seed and the encrypted program are sent to the PLC from the TIA, and the encrypted key is secured by a hard-coded common key (that is shared between PLC models). Since the encrypted seed is sent along with the encrypted program, a man-in-the-middle attacker that previously obtained the common key (for example, as demonstrated in [19]) can extract the AES key and decrypt the PLC program, despite the program encryption.

With the decrypted control program in hand, the attacker gains insight into the logic and instructions intended for the PLC, leading to unauthorized access and data leakage.

Depending on the attacker's intentions, they may read the encrypted control program, or even alter the program to introduce malicious commands. Hence, the attacker compromises the confidentiality and integrity of the program.

We emphasize that this attack has been found to be effective on older versions of the S7 protocol, such as TIA Portal 12, where the protocol lacks TLS encryption. As a result, encrypted programs are transmitted with their keys in the plain. Hence, this attack is even more dangerous against previous versions of the S7 protocol, as attackers in the network can decrypt the control program.