Legacy-Compliant Data Authentication for Industrial Control System Traffic

John Henry Castellanos( ), Daniele Antonioli, Nils Ole Tippenhauer, and Martín Ochoa

Singapore University of Technology and Design, Singapore, Singapore
{john_castellanos,daniele_antonioli}@mymail.sutd.edu.sg,
{nils_tippenhauer,martin_ochoa}@sutd.edu.sg

Abstract. Industrial Control Systems (ICS) commonly rely on unencrypted and unauthenticated communication between devices such as Programmable Logic Controllers, Human-Machine-Interfaces, sensors, and actuators. In this work, we discuss solutions to extend such environments with established cryptographic authentication schemes. In particular, we consider schemes that are legacy compliant in the sense that authentication data is embedded as additional payload for domain specific protocols, for example the industrial EtherNet/IP protocol. To that end, we propose a selective protocol (that signs every critical packet sent) and a protocol that aggregates groups of packets based on real-time requirements and the available throughput, for various realistic hardware configurations. We evaluate our analysis by implementing an authenticated channel in a realistic Water Treatment testbed.

Keywords: Industrial Control Systems · Authentication · Network security

1 Introduction

Industrial Control Systems (ICS) commonly rely on unencrypted and unauthenticated communication between the industrial devices such as Programmable Logic Controllers (PLC), Human-Machine-Interfaces (HMI), sensors, and actuators. The use of cryptographic schemes in such devices is often hindered by their long lifetime, compatibility issues, low processing power of the embedded devices, and real-time requirements in the communication [9]. Most common industrial communication protocols do not feature any built-in capabilities for authentication (e.g., Modbus/TCP, EtherNet/IP). In the past, critical infrastructure control networks' were isolated from the office and corporate networks, and thus malware and other advanced attacks did not pose a realistic threat to such control networks.

Nowadays, with the increased connectivity to general IT infrastructures such as a LAN, and the Internet itself, attacks such as the well-known Man-in-the-Middle between ICS devices are more realistic [29]. For ICS communications,

D. Gollmann et al. (Eds.): ACNS 2017, LNCS 10355, pp. 665–685, 2017.
DOI: 10.1007/978-3-319-61204-1_33

message *integrity* is paramount, while *confidentiality* of messages exchanged is less important. In particular, if an attacker can alter the values of the sensors or the commands being sent to the actuators, he could effectively alter the control of the ICS and potentially cause the malfunctioning of the physical system.

A number of works (e.g., [2,3,5,23,26,31] to name a few) highlight the importance of securing the networks of ICS. Most of them also remark that by using cryptography a non negligible overhead can be introduced, and that such systems usually have strict timing constraints. However, to the best of our knowledge, no detailed analysis on cryptography-enabled authentication has been reported so far for ICS under realistic constraints. In particular, data in ICS is also often passing through intermediate gateways and other industrial network appliances. For that reason, the solution must be legacy-compliant (which can prevent solutions such as TLS encapsulation).

In this paper, we explore the application of well-known cryptographic primitives to verify authenticity of communication between devices in modern ICS, embedded as payload in legacy industrial protocols. In particular, we are interested in solutions that would either be feasible to implement in constrained legacy devices, or could be provided by low-cost additional devices. The focus of the paper is *to detect* manipulations of legitimate traffic by the attacker, and the presence of new messages introduced by the attacker. If confidentiality is required, an additional suitable encryption scheme should be used. In addition, the discussion of consequences of successful detection of message manipulation attacks is out of scope of this work. The mitigation to such an attack is a subject on its own, since in general one cannot simply drop incorrectly signed packets: on one hand, even a single missing packet could have unforeseen consequences in controlling physical processes, on the other hand if packets are dropped attackers could easily implement denial of service attacks [15].

In order to maximize the efficiency of legacy-compliant authentication while preserving security, we propose a basic authentication protocol that signs a subset of *critical* packets. This is in contrast to closely related work, that usually focuses on authenticating *all* of the communication between nodes in ICS [18,25,28,33]. In addition, we propose a second protocol that aggregates groups of packets based on the real-time requirements, network throughput, and the processor capacity for various realistic hardware configurations. We evaluate our analysis experimentally in an industrial Water Treatment testbed [11] (Secure Water Treatment, SWaT) by means of additional network components. However our protocols can be deployed by controller manufacturers as a firmware modification of existing Ethernet modules.

We summarize our contributions as follows. In this work we: **(a)** discuss design options for legacy-compliant basic authentication protocols in the context of ICS networks; **(b)** perform an experimental performance evaluation of a number of cryptographic primitives on several hardware platforms; **(c)** propose a novel aggregated authentication protocol, adapted to requirements of ICS; **(d)** empirically evaluate the proposed protocol in a Water Treatment testbed.

2 Preliminaries

In the following we introduce some fundamental notions on Industrial Control Systems, and introduce the attacker model and the expected security guarantees of our solution.

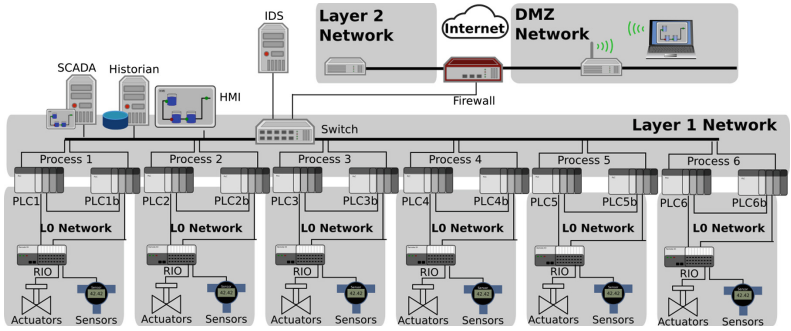

Fig. 1. SWaT network architecture.

2.1 Industrial Control Systems

ICSs are a subset of Cyber-Physical Systems that monitor industrial infrastructures such as Water Treatment systems, nuclear power plants, smart grids, and electric power distribution systems [22]. Securing a safety-critical ICS against malicious attackers is crucial to avoid catastrophic events that may result in natural disasters, economical crises, and loss of human life. The threat model of ICS often assumes a strong attacker, the system under attack provides a large attack surface because an ICS may be vulnerable both to *cyber-attacks* and *physical attacks*. Unfortunately, a combination of cyber and physical attacks can result in a severe damage of the system even without physical access to the system, e.g., [2,32].

Figure 1 shows an example of the architecture of an ICS network. The network is layered to logically separate devices and monitored processes. Compared to a standard corporate network, an ICS network includes a wider range of devices. The ICS has to connect several older legacy hardware and new hardware, and it has to manage software with different capabilities and interfaces. In addition, a traditional ICS network is expected to have a long life time (e.g., twenty years), and many of its components are unlikely to change or be upgraded over the years. That is why the protocols we propose in the remainder of the paper target *both* high-end ICS devices (able to tolerate the computation overhead), and low-end ICS devices (with the introduction of an external module that is able to tap into the ICS network).

Different industrial protocols have been used in ICS. They evolved from serial communication networks (e.g., RS-485, RS-232) to bus systems (e.g.,

Fieldbus), and then to Ethernet-based communications such as EtherNet/IP (ENIP) [6]. ENIP is a modern, object-oriented application layer industrial protocol, that implements the Common Industrial Protocol (CIP) specifications [24] over the TCP/IP protocol stack. ENIP can be extended to support custom commands and device profiles, and it provides a native compatibility with traditional TCP/IP based IT corporate network. It is important to notice that our authentication scheme *does not* depend on the underlying industrial protocol. We use ENIP as an example protocol as we have a local Water Treatment testbed that uses ENIP. However, the same scheme should easily translate to other modern industrial protocols.

2.2 Security Guarantees

In this work, we discuss solutions to provide authenticity of network traffic for ICS. The solution is not designed to provide confidentiality, and does not prevent denial of service attacks. The attacker is assumed to be able to eavesdrop, insert, drop, and manipulate messages on the network. The attacker does not have access to pre-shared keys, and he is constrained by polynomial computational power on the size of the key, where the computational hardness assumption are simular to the ones proposed in [27].

Moreover, one key observation of this work is that not all packets being transmitted by nodes in ICS need to be authenticated: in the following we will discuss (using ENIP as an example) how to select a subset of critical packets that need to be protected, whereas we argue that other packets are less critical and do not necessarily need to be authenticated since their manipulation does not pose a threat to ICS.

3 Traffic Authentication for ICS: The SPA Protocol

In this section we introduce a first efficient protocol to guarantee authenticity of critical communication in ICS. In addition to providing authenticity, the solution also needs to integrate well into existing systems. In particular, the solution needs to fulfill *real-time* requirements and *legacy-compliance* integration. Note that as stated in the previous section, we will use ENIP as a running example to illustrate our protocol, since it is the protocol used in our evaluation setting. Our idea can be applied to other industrial network protocols (i.e. Modbus TCP [19] and PROFINET [10]), however the implementation details will vary, essentially because a signature must be appended to certain packets, which is easily accomplished in ENIP as we will discuss later.

Our solution must be computationally efficient enough to allow resource-constrained devices (such as PLCs) to sign and verify packets fast enough. In particular, the solution should be able to handle high volume traffic loads, without introducing high queuing, and processing delay. We start by proposing a first solution we called SPA (Selective Packet Authentication). The SPA protocol relies on a simple algorithm: we assume a setting where two devices, that

want to communicate, possess a common pre-shared key, and we propose to sign only selected outgoing packets using well-known cryptographic algorithms, such as authenticated signatures schemes [12,14].

This solution is conceptually simple, however it potentially conflicts with the real-time constraints outlined earlier. In Sect. 4, we discuss our refinement of this idea, we present a mathematical analysis able to capture the constraints, in order to guarantee the normal operation of the ICS.

3.1 Legacy-Compliance

ICS often integrate devices that cannot easily be replaced or updated. As a result, a number of legacy industrial protocols are established that are widely supported, but do not feature any security capabilities by design. In general, such protocols allow the reading of distinct memory locations (e.g., in Modbus/TCP) or *tags* (in EtherNet/IP) that represent sensor values or similar. We argue that an upgrade of all such devices in an ICS is costly; therefore an authentication system needs to impact the existing system as little as possible. In particular, the use of TLS tunnels to transmit data would not only incur computational overhead, but could also fail to pass through industrial network appliances or intermediate gateways. Therefore, we propose to embed the authentication data as additional payload in the existing industrial protocols. This will ensure that receivers that are not aware of the authentication scheme can at least process the normal payload without benefiting from the authentication data. Intermediate devices that are unaware of the authentication scheme could also just pass along the authentication data as normal payload. As result, our authentication solution can be integrated in legacy systems, e.g., by introducing external modules to data sources, or through firmware modification of Network modules.

3.2 SPA Protocol Description

The intuition behind the SPA protocol is that (a) only a subset of transmitted messages is security relevant, and (b) selective signing of that subset is more efficient than signing all messages in the stream. For simplicity, we will refer to sign as the process of generating a Message Authentication Code using a pre-shared key for the case of symmetric key cryptography or, a signature using the counterpart's public key for the asymmetric case.

Let p be a critical data packet (we discuss how to identify them later in the context of ENIP). The SPA protocol, shown in Fig. 2, calculates a signature $Sig_k\{p\}$ of the packet p, generates a new packet $p' = (p, Sig_k\{p\})$, and sends it to B. By using the inverse function Ver_K (where $K = k$ in the symmetric case or the corresponding public key otherwise), B verifies the authenticity of the message as $p = Ver_k\{Sig_k\{p\}\}$. Note that the signature scheme Sig guarantees that a computationally constrained adversary cannot forge a signature (with high probability).

We note that to prevent replay attacks, the payload should contain a timestamp or a counter to identify the ENIP session or packet. Therefore, if

an active adversary replays the message in a future ENIP session, the application layer will check the nonce and mark it as invalid. On the other hand, since we do not sign non-critical packets in the data stream (nor e.g., TCP synchronisation packets) an attacker could manipulate the content of those messages. Such an attack is easily detected by orthogonal means [7]. By design, ICS systems will trigger alarms in case of problems in network connectivity. In particular, we cannot prevent against denial of service attacks by means of authenticity and such attacks are out of scope.

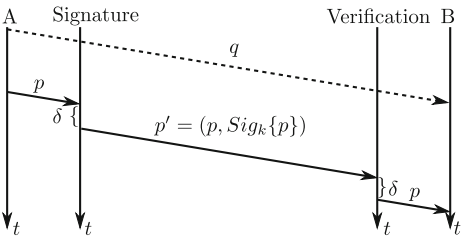

Fig. 2. SPA protocol overview: p is a critical message that is signed by the signature module. q is a non-critical message, which is simply forwarded; δ is the delay introduced by authentication and verification.

Real-Time Requirements and Backlogs. ICS operate under very strict real-time operation conditions, with maximal critical response time, very high availability requirements, and low tolerance for high delay or jitter conditions. As described before, in order to guarantee the authenticity of messages, we authenticate outgoing packets using a *Signature* module, and verify incoming packets using a *Verification* module respectively. We define $g(\Delta)$ as the number of packets generated by a device in an interval of time Δ, and $s(\Delta)$ as the rate at which packets are being signed. As an example, $g(\Delta)$ can be 1000 packets per second if $\Delta = 1$ second, while $s(\Delta)$ generally depends on the processing speed, e.g., 300 packets per second. Similarly we denote $r(\Delta)$ the number of packets received by a node and $v(\Delta)$ the number of verifications performed.

In order to be compliant with the real-time requirements of a given system, essentially one needs to authenticate/verify packets at least as fast at the rate they are being produced/received, that is $g \leq s$ and $r \leq v$ (see Appendix A).

3.3 Application to ENIP-CIP

We now discuss the application of the SPA protocol to Common Industrial Protocol (CIP) traffic to ensure legacy compliance, leveraging its extension possibilities. SPA payload is given by $p = (payload, ENIPsequencenr, ENIPsessionid)$ based on a critical ENIP packet from A to B. The ENIP session ID is a randomly generated 32-bit integer and will serve as a counter as discussed in Sect. 3.2. Note that although 32-bit is a relatively small search space, in this context we do not

rely on it for security, but merely to prevent replay attacks. An attacker still needs to forge a secure MAC or cryptographic signature to bypass verification, as we will illustrate in Sect. 5.

When replacing the original p with p' of SPA, we increase the length of any packet p that we sign. The packet extension must conform to the ENIP standard. The structure is defined as follows: *Type ID*: $0x00c1$. *Length*: size of the signature specified in Bytes. *Data*: the signature $Sig_k\{p\}$.

The device receiving p' will search for the Type ID $0x00c1$, verify the content of the payload, and then remove the signature. In the case of a mismatch, i.e., $p \neq Ver_k\{Sig_k\{p\}\}$ the device will raise an alarm.

Identification of Critical Data. The integrity of messages exchanged in an ICS system can be protected in different layers of the OSI network stack. We identified critical data from the pool of CIP services observed in the traffic captures. In particular, protection is required for data that can affect the normal operation of the control of a physical process.

The identified critical services are: *Read Data* (Service 0x4C), *Write Data* (Service 0x4D), and *Read Tag Fragmented Data* (Service 0x52). In Sect. 5, we discuss in more detail this choice in the context of other CIP services observed in our Water Treatment testbed. By authenticating critical packets only, we increase security and minimise the computation overhead.

3.4 Ad-Hoc Protocols vs TLS

We have chosen to focus on ad-hoc protocols at the application layer, rather than of using TLS [4], for various reasons. Firstly, we want to reduce the computation overhead, by only authenticating packets that contain critical payloads (such as commands to actuators and values from sensors). In comparison, while TLS would sign every packet in a ENIP connection, SPA protocol would only sign and verify a comparatively small amount of those ignoring packets such as TCP handshake, EtherNet/IP communication control messages and CIP non critical service messages, as shown in Fig. 3. We believe that integrity attacks on TCP handshake and ACK messages can be detected by orthogonal methods. On the other hand, we want to be backwards compatible with devices that do not support message authentication, a feature that we achieve by using the extension capabilities of ENIP. Such a feature would not be achievable by using TLS.

We now discuss performance of SPA vs TLS. Let v the speed in packets per second that a given cryptographic signature can provide for a given average packet size. For simplicity we assume that both TCP packets and ENIP packets are about the same size, although in practice TCP will be slightly bigger. Then if the number of critical packets is c, then the actual throughput on the total of generated packets g will be higher, since we only need to sign $c \cdot g$. Thus, effectively we will increase the tolerance on the generated packets g by a factor of c, allowing a maximum of $\frac{v}{c}$ packets per second. This is illustrated in Fig. 3.

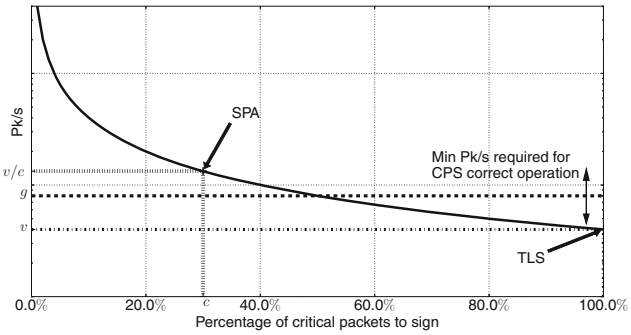

Fig. 3. The y-axis represents the tolerance to a network constraint's in packets/second. The x-axis represents the percentage of critical packets authenticated.

Example. In practice, the amount of sent and received packets is symmetric, as we will discuss in the following sections. In addition, the signature and verification time are similar. Let $g = r$ be 10^3 packets per second, and let $c = 0.4$. Let $s = v$ be 10^5 packets per second. In this case, cryptography-enabled authentication does not create backlogs neither for TLS (signing every packet) nor for SPA since $g + r = 2 \cdot 10^3 < s + v = 2 \cdot 10^5$. This example is based in the real-time communication constraints of SWaT and the signature and verification rates for SHA-256 based HMAC on a virtualized ARM similar to the one used in the SWaT PLCs. A deeper analysis and discussion is presented in Sect. 5.

In sum, the proposed SPA protocol has both advantages and drawbacks in terms of the authentication goals. We briefly summarized them as:

Advantages. The protocol is conceptually simple and easy to implement and to integrate into legacy systems by means of an external component as we will discuss in Sect. 5. Moreover, it gives fine-granularity detection capabilities, since it can be pointed out which packet was altered in transit with almost instantaneous detection times. Additionally, by using the extension capabilities of ENIP, the protocol is backwards compatible with devices that do not implement a verification module.

Disadvantages. The main disadvantage of the SPA protocol is that although it is designed to sign critical packets only, the overheads might still be unacceptable for devices and/or algorithms with low signature/verification rates. This is the main motivation for proposing an extended protocol, as we will discuss in the following section.

4 Extensions: The ASPA Protocol

The SPA protocol presented in the previous section and the constraint analysis, give us guidelines to determine whether the scheme is feasible, depending on

the packet generation rate, packet size, and signature algorithm performance on links that have a reliable connectivity. However, there exist two limitations to the practical application of this protocol: first, we would like the protocol to be used even in scenarios where (for legacy or cost reasons) the underlying hardware is not as fast as necessary (i.e. in the sense of the signature rate s vs generation rate g as discussed in the previous section), and second, even faster hardware might be insufficient for stronger signature algorithms, that are computationally more expensive.

In particular, strong signature algorithms would enable the use of asymmetric cryptography, that has several advantages in terms of key management. For example, the use of public/private keys in asymmetric cryptography allows dynamic addition of new devices to the system if centrally signed certificates are used. In addition, the compromise of a single device will only expose a single private key, instead of exposing a secret key shared between many devices.

Our protocol can be extended to deal with more expensive cryptographic algorithms and slower CPUs. The intuition behind this extension, called *ASPA* (Aggregated Selective Packet Authentication) is the following: typically, authenticated signature schemes are relatively inefficient for short amount of data, but they get more efficient for large amounts of data. Thus, on average, it is usually faster to perform an aggregate signature over multiple packets instead of signing them individually, and as a result the aggregate signature increases the signing rate s.

4.1 The ASPA Protocol

Let $P(T)$ be the sequence of outgoing packets in the time interval T. Let n be the expected number of packets in this time $\mathbf{E}[|P(T)|] = n$ such that the typical set is $P(T) = \{p_1, \ldots, p_n\}$. In the ASPA protocol, shown in Fig. 4, the *Signature* module simply forwards packets p_i to B, and in addition accumulates their payload in a queue. After time T has passed, it signs the accumulated queue, and sends the *aggregated signature* together with the sequence number of the first and the last element of the queue. The *Verification* module will forward all received messages to the original destination (without immediately validating a signature), and in addition store the received messages in a queue

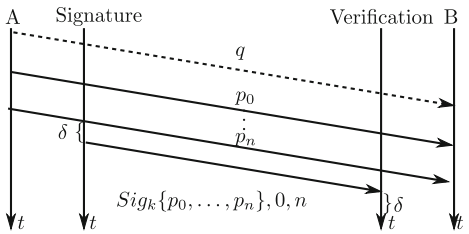

Fig. 4. ASPA overview. p_i are critical messages which are aggregated. q is a non-critical message, thus not authenticated. δ is the delay introduced by authentication.

until the aggregated signature arrives. Then, the verification module can check whether the signature matches the content of the respective packets received. If the Verification module cannot validate the signature, an attack was detected. For example values of T and $P(T)$, we refer to Sect. 5. We note that the ASPA protocol relies on lower layers (i.e., TCP) to ensure that no message gets lost during transmission. That will ensure that the verification and signature modules always have the same view of exchanged messages.

Security Trade-Off. In the first proposed protocol, an attack can be detected as soon as a spoofed packet with an invalid signature is received. In the ASPA protocol, we can only detect an attack after T time has passed, and an invalid signature is received. Depending on the value of T, and the particular ICS system, this could be problematic (or not). This is related with the problem of ICS resilience and reaction time in case of failure, and therefore we see it as orthogonal. For most practical applications (see Sect. 5), the values of T will be small and thus the reaction time will not differ much from the reaction time of the first protocol.

4.2 Performance Advantage

Symmetric Authentication. Let $\delta(size, cpu, alg)$ be the time needed to sign a packet of size *size* with algorithm *alg* on CPU *cpu*. The signature scheme is typically based on HMAC and an underlying hash function (such as SHA-256). In that case, there is a constant b such that $\delta(b', cpu, alg) \approx \delta(b, cpu, alg)$ for $b' \leq b$. However, for $B > b$: $\delta(B, cpu, alg) = \delta(b, cpu, alg) + \left\lceil \frac{B-b}{b} \right\rceil \cdot c$ where $c < \delta(b, cpu, alg)$. Therefore, given an expected packet number of a certain expected size, it is more efficient to sign multiple packets instead of just one, in terms of the rate per interval s. As we will discuss in the next section, the optimization value converges after a certain number of packets, as is to be expected.

Asymmetric Authentication. In the case of signatures based on asymmetric cryptography (such as ECDSA [12]), typically the payload is first hashed and then an operation involving the private key is performed on the digest (such as decryption). Since the cryptographic operation is orders of magnitude slower than the hashing operation (i.e., hundreds of ms vs. μs in some cases), signing multiple packets takes almost as long as signing a single packet.

The above described effects on the signing rate s can be observed in the plotted values for different algorithms and values of n of Fig. 6. Note that a similar discussion about the signing rate s and the packet generation rate g, presented in the previous section, also applies for the ASPA protocol. In practise, a device might have active connections (TCP streams) to m devices at the same time. As result, there will be m queues $Q^S_{\Delta,i}$, possibly signed with different keys. In that case, the constraint becomes $\forall_i \; g_i < s_i$, where $g = \sum_{i=1}^{m} g_i$. In the case of ICS such as SWaT, devices typically communicate only with one or two devices (e.g., $m = 2$). For the sake of simplicity we assume a single queue in the following.

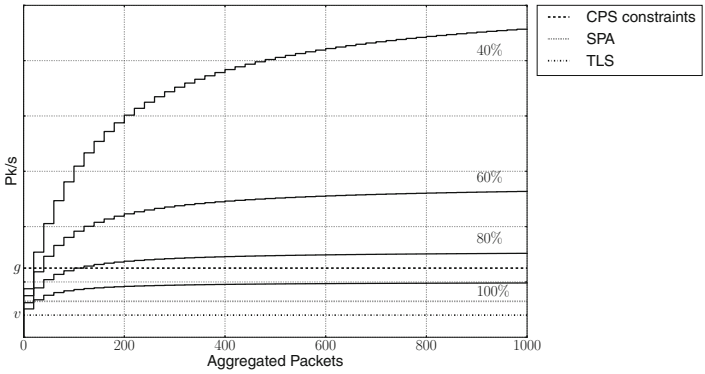

Fig. 5. The x-axis represents chunks of packets, y-axis tolerance in terms of packets per second. The step functions represent different percentages of critical packets, segmented lines various CPS communication constraints. CPU power and packet size are constant.

ASPA vs TLS. In Fig. 5, we illustrate the general case comparison against TLS. Let $v = \frac{1}{\delta}$ the number of packets per second an ideal implementation of TLS could sign. By using ASPA to aggregate n packets we can tolerate approximately: $v' = \frac{n}{\delta+(n-1)\cdot\delta'}$ packets per second, which will be faster than v and will converge to a constant since: $\lim_{n\to\infty} \frac{n}{\delta+(n-1)\cdot\delta'} = \frac{1}{\delta'}$. We will discuss empirically this phenomenon in the evaluation section for various hardware configurations.

Example. Let $p = r$ be 10^3 packets/second as in the previous example (as observed by us in SWaT). Let $s = v$ be 300 packets/second using the SPA protocol of the previous section and ECDSA and a Raspberry Pi3 (for more details see Sect. 5). In that case, cryptography-enabled authentication clearly causes backlogs since $g + r = 2 \cdot 10^3 > s + v = 600$. As discussed above, signing one packet takes almost as much time as signing multiple packets, and then we can implement the ASPA protocol by accumulating chunks of 50 packets (for $T \approx 30\,\text{ms}$), thus augmenting s to about $1,5 \cdot 10^4\,\text{s}$ packets per second. This implies a minimum reaction time to attacks of about 60 ms.

In sum, the ASPA protocol improves the signing and verification rates per packet (s and v), while at the same time providing good reaction time.

Advantages. The ASPA protocol is useful in situations where signing each packet individually is not feasible due to slow hardware or constraints of the ICS network. In particular, it offers a significant advantage when signing multiple packets with ECC based authentication. The ASPA protocol can be used selectively for critical messages (as in the SPA protocol). Alternatively, it could be used to provide delayed authentication for all messages, as the amount of data included in the aggregated signature has a negligible impact on the overall time of creating the signature.

Disadvantages. The coarse granularity of the authentication potentially can delay reaction times to attacks, depending on the size T of the signing window. However in our evaluation we have found that the ideal window for authentication is a few dozens of packets for most signing algorithms/hardware, which allows for a fast reaction time in practice. In our experience, actual attacks require a relatively long interaction with the system in order to influence its physical state. However, we stress that the subject of reaction to attacks is out of the scope of this paper and is left for future work.

5 Validation

In this section, we use a real SWaT [11] to obtain realistic examples of real-time constraints. We investigate the amount of critical traffic shared between devices in the network, in order to approximate the expected number of packets per time interval, and the expected size of those packets. Furthermore, we benchmark symmetric and asymmetric signature algorithms for a variety of hardware platforms, and discuss the feasibility of their implementation with respect to the constraints derived in the previous sections. We use standard algorithms such as SHA (instead of light-weight algorithms specifically designed for embedded systems) to allow better comparison against TLS. When using light-weight algorithms, the presented numbers are expected to improve.

5.1 SWaT

SWaT [11] (see Fig. 1) consists of an ICS with a process made up of six stages. In a nutshell, the process begins by collecting the incoming water in a tank, then it performs a chemical treatment stage, it filters the treated water through an Ultrafiltration (UF) system. Afterwards, the water is de-chlorinated using a combination of chemicals and Ultraviolet lamps, and then fed to a Reverse Osmosis (RO) stage. A backwash process cleans the membranes in UF using the water produced by RO. The cyber portion of SWaT consists of a layered communications network, PLCs, HMIs, SCADA, and a Historian.

The validation focuses on benchmarking integrity, and authenticity controls in the plant network, that connects the PLCs, the HMI and the SCADA system. The network is using the EtherNet/IP industrial protocols on top of an (Ethernet based) TCP/IP network. We performed several network capture by setting up a mirroring port on the plant network industrial switch. From those captures we identified ENIP-CIP communications among 21 hosts, through implicit and explicit messages. Implicit messages (UDP/2222) are used in our plant for keep-alive signals, while explicit messages (TCP/44818) are used for configuring, monitoring and controlling the plant stages. The plant performs its communication to a rate of 16.000 ENIP-CIP messages per second on average over all stages. About 14,3% of ENIP-CIP connections belong to UDP/2222, and the rest 85,7% to TCP/44818. We can split TCP connections between TCP session traffic (42,7%), and CIP explicit messages (42,9%).

We focused on CIP explicit messages, and we tried to extract a subset of CIP services that deals with critical data. Manipulation of those services could affect the state of the controlled physical process. We selected the following services as critical: **Read Data** (Service 0x4C); **Write Data** (Service 0x4D); **Read Tag Fragmented Data** (Service 0x52).

Read Data and Read Tag Fragmented Data. CIP services are classified as critical data since an attacker might rise a fake alarm in the SCADA system or he might hide a safety-related event modifying its data on the fly. The *Write Data* CIP service is classified as critical because an attacker might directly modify the behavior of actuators pushing data into PLCs. By selecting CIP services with critical data, our proposal only signs around 42% of SWaT's traffic (including TCP and UDP), avoiding processing and bandwidth overheads for non-critical data CIP services such as the *Get all attributes* service.

We performed an in-depth analysis of the capture file in order to estimate the frequency, and the size of the packets received per second, by an arbitrary testbed's device. Table 1 shows a summary of the results targeting PLC2. We decided to use PLC2 as an upper bound because we estimated that it is the "busiest" device in our testbed. As you can see from the table, PLC2 sends 1127 packets per second on average, and it receives 1168 packets per second on average.

Table 1. Frequency and size of critical packets shared by host PLC2 to others.

		Sent	Received
ENIP message	Request	561 Pk/s $\mu = 63$B, $\sigma = 3.36$	607 Pk/s $\mu = 69$B, $\sigma = 5.32$
	Response	566 Pk/s $\mu = 75$B, $\sigma = 58.16$	561 Pk/s $\mu = 86$B, $\sigma = 9.42$
Total Pk/s		1127	1168

5.2 Hardware Benchmark

In order to evaluate the efficiency of the underlying primitives, and therefore the packet signing rate $s(t, cpu, alg)$, we used different types of hardware platforms.

Controllino is an open source Hardware PLC, based on Arduino Mega 2560 board, with an ATmega2560 CPU (16 MHz), 256 KB of flash memory, an Ethernet connector, and two serial interfaces. For our experiments, we used the spaniakos cryptographic library [8]. **ARM** We used the QEMU emulator with the following settings: ARM926EJ-S rev 5 processors family at 530MHz, 256MB of RAM, Debian 3.2.51 32 bit Operating System, and the libgcrypt-1.6.5 cryptographic library. **Raspberry Pi** is a single-board computer of credit card-size. It was initially developed for educational purposes, but because of its low energy

consumption and low cost, it has become into a popular multipurpose hardware. We choose it as a possible hardware to implement the authentication and integrity mechanism. The characteristics of the Raspberry Pi model 2 (RPi2) are: Quad-core ARM Cortex-A7 processor at 900 MHz, 1 GB of RAM, 4 USB ports, 40 GPIO pins and an Ethernet port. The cryptographic library used was again libgcrypt-1.6.5 [17]. **PC** We used a workstation with the following specifications: Intel Core i5-5300U processor at 2.30GHz, 3 GB of RAM, Xubuntu 15.10 64 bit OS, and libgcrypt-1.6.5 as cryptographic library.

Table 2. Benchmark of HMAC-SHA256 and ECDSA signature process for different packet sizes over 5 types of hardware (rounded values). Times are in μs.

	HMAC - Time				
Size	Controllino	ARM	RPi2	RPi3	PC
64 B	$2.2 \cdot 10^4$	76	53	15	2
128 B	$3.3 \cdot 10^4$	78	58	16	2
256 B	$5.5 \cdot 10^4$	84	69	18	3
512 B	$1 \cdot 10^5$	117	89	32	4
1 KB	$1.8 \cdot 10^5$	171	130	35	6
2 KB	$3.6 \cdot 10^5$	252	211	58	10
4 KB	$7 \cdot 10^5$	474	374	104	18
	ECDSA - Time				
4 KB	N/A	$1.5 \cdot 10^5$	$1 \cdot 10^5$	$3.2 \cdot 10^4$	$3.1 \cdot 10^3$

Table 3. Benchmark of performance of HMAC-SHA256 and ECDSA authentication for different packet sizes and hardware. Average size per packet 73 Bytes from Table 1.

	HMAC - Average Pkt/s				
Size	Contr.	ARM	RPi2	RPi3	PC
64 B	40	$1.1 \cdot 10^4$	$1.6 \cdot 10^4$	$5.8 \cdot 10^4$	$4.4 \cdot 10^5$
128 B	53	$2.2 \cdot 10^4$	$3 \cdot 10^4$	$1.1 \cdot 10^5$	$8.8 \cdot 10^5$
256 B	64	$4.2 \cdot 10^4$	$5 \cdot 10^4$	$1.9 \cdot 10^5$	$1.2 \cdot 10^6$
512 B	70	$6 \cdot 10^4$	$7.9 \cdot 10^4$	$2.2 \cdot 10^5$	$1.7 \cdot 10^6$
1 KB	78	$8.2 \cdot 10^4$	$1.1 \cdot 10^5$	$4 \cdot 10^5$	$2.3 \cdot 10^6$
2 KB	78	$1.1 \cdot 10^5$	$1.3 \cdot 10^5$	$4.8 \cdot 10^5$	$2.8 \cdot 10^6$
4 KB	80	$1.2 \cdot 10^5$	$1.5 \cdot 10^5$	$5.4 \cdot 10^5$	$3 \cdot 10^6$
	ECDSA - Average Pkt/s				
4 KB	N/A	$3.7 \cdot 10^2$	$5.6 \cdot 10^2$	$1.7 \cdot 10^3$	$1.8 \cdot 10^4$

We also benchmarked the elliptic curve based ECDSA signing standard. As discussed previously, the cryptographic operation dominates the execution time

of the hashing component of the algorithm. This makes the times for signing payloads of up to 4KB almost identical to small payloads of 32B. The results are reported in the last row of Table 2. The resulting s for ASPA for aggregated signatures of about 58 packets (4KB) is reported in Table 3.

5.3 Discussion

We now provide a summary of our empirical results using different hardware platforms and cryptographic algorithms combination over our SWaT. In Table 4, we compare the SPA protocol (first two columns) with the ASPA protocol for $n = 58$ packets. As we can see, with a processor such as the one in the Controllino (16MHz), it is infeasible to cope with the real-time requirements of the SWaT networks for all algorithms considered. As shown in Fig. 6, and taking as reference our plant constraints, a symmetric signature is supported by most hardware. On the other hand, ECC signatures are possible in Raspberry Pi2, Pi3 and PCs thanks to our extension.

From a communications cost perspective, a signature in HMAC would add an overhead of 28% in size for an average ENIP packet, while ECDSA would add about 57%. Since we are only signing critical data, which corresponds to 42% of total traffic, our overhead in bandwidth will be 12% and 24% for HMAC and ECDSA respectively.

In sum, we have shown that in some cases, there is an advantage on using aggregation and selectiveness over traditional authenticated tunnels in terms of efficiency. Although we have shown an example for a concrete testbed and some hardware configurations, the possible configurations in practice could vary

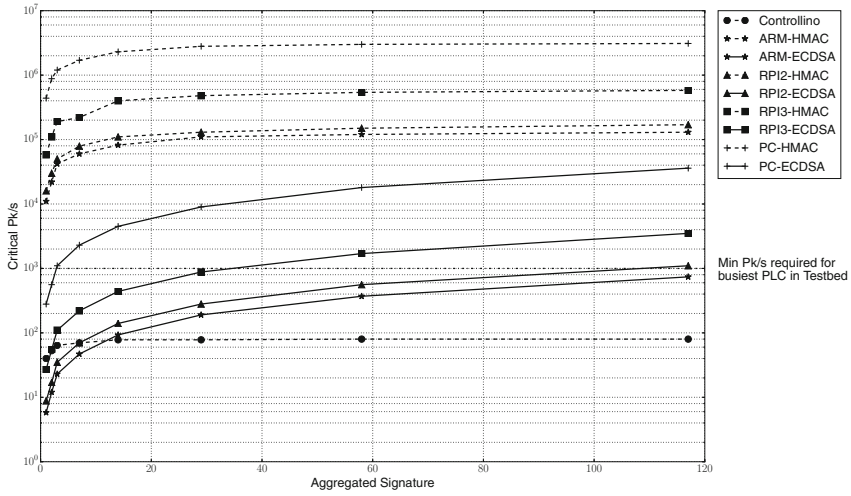

Fig. 6. ASPA performance on various hardware. PLC2, as the busiest, requires 1000 Pk/s for implementing a multi-link authentication process, y-axis is in log scale.

Table 4. Feasibility of algorithms vs. hardware in the SWaT. Slow hardware does not tolerate the minimum required signing rate even for ASPA. However, thanks to ASPA we can use RPi2, RPi3 and PCs to implement ECC based authentication.

	Sequential				Parallel			
	HMAC	ECC	HMAC-ASPA	ECC-ASPA	HMAC	ECC	HMAC-ASPA	ECC-ASPA
Controllino	×	×	×	×	×	×	×	×
ARM	✓	×	✓	×	✓	×	✓	×
RPi2	✓	×	✓	✓	✓	×	✓	✓
RPi3	✓	×	✓	✓	✓	×	✓	✓
PC	✓	×	✓	✓	✓	×	✓	✓

highly. For instance, in a full scale water plant the number of sensors and actuators could be one order of magnitude higher ($g = 10^4$), whereas the available hardware could range between the Controllino (16 MhZ) and the ARM processor (500 MhZ). Moreover, the underlying cryptographic algorithm could also vary, thus affecting the concrete performance.

Implementation in SWaT. We have successfully tested the proposed approach in a link between two PLCs of SWaT using Raspberry PIs and summarize our implementation efforts in Appendix B.

6 Related Work

In [23], the authors analyze different attacks on PLCs and several protection techniques. The authors state that the bulk of the security issues in ICS is the lack of security in the network communications. A public key based authentication protocol is proposed. However, the authors mention that their techniques could degrade the performance of the system, and they do not provide any further analysis.

A detailed analysis on cryptography-enabled authentication is reported in [31]. The authors analyze the security of electric power grids and identify authenticity and integrity as the most important security properties after availability. Some constraints on the message delays are considered during the analysis based on the expected message frequency as discussed in industrial standards.

In [1], the authors discuss integrity, authenticity, and authorization policies of ENIP and CIP. Two security profiles (providing integrity and confidentiality) for the ENIP are described and two for CIP (providing authorization and integrity) are proposed as a future extension. Authors define security profile as a set of well-defined capabilities to facilitate device interoperability, and end-user selection of devices with the appropriate security capability. In particular, the

CIP authorization profile will provide secure communications between CIP endpoints, to ensure device and user authenticity. Authors describe ENIP over TLS (for TCP-based communication) and DTLS (for UDP).

Authors propose to implement confidentiality and integrity checks between paired devices called SCADA Cryptographic Modules (SCM) [33]. They are featured with two ports (plaintext and ciphertext). While it receives plaintext messages from a device through the plaintext port, it encrypts the message using AES and send it to its remote SCM through the ciphertext port. The message includes a MAC code (HMAC-SHA-1 or CBC-MAC) for integrity verification. In [28], authors propose a bump-in-the-wire solution to add security features such as Data privacy and authenticity to ICS communication. The solution suggests adding two devices (each per peer) into the communication channel (YASIR transmitter and YASIR receiver). YASIR Transmitter encrypts the message using AES-CTR, and attach an HMAC-SHA-1 signature. A YASIR Receiver decrypts and checks the message's integrity. In the case of integrity violation, YASIR Receiver adds an error byte to the frame; it guarantees that destination device discards the final message.

In IP-based communications, authors at [18] introduce a DNP protocol modification. They propose to assure confidentiality using DES cryptographic primitive and Integrity-Authenticity with HMAC-MD5-96 or HMAC- SHA-1-96. Their work suggests a change in the DNP's data structure. It could thwart their adoption by industry. Recently, researchers create a new MODBUS security development module (NMSDM) [25] that replaces MODBUS message. They propose to use cryptographic algorithms such as RSA, AES and SHA-2 to guarantee security properties as confidentiality, integrity, authentication and non-repudiation. Their proposal is to add a *cryptographic buffer* on top of PDU, as part of the TCP payload. Like ours, it allows adding security properties to the communication without modifying the protocol by itself. Both works [18,25] search to protect every message in an ICS, and computing overhead of cryptographic implementation are not taken into account.

In automotive applications, authors at [21,30] present authentication techniques called CANAuth and LeiA for CAN bus protocol. Similarly to ICS, CAN bus systems relies on hard real-time constraints. The authors shows that state of the art authentication mechanisms for broadcast networks cannot be used due to time (and storage) constraints. LeiA and CANAuth, however, are specifically designed for CAN bus systems. A low encryption overhead is also of paramount importance in Wireless Sensor Network (WSN), e.g., [13,20]. This is mainly due to bandwidth, latency and energy cost introduced by encryption. Solutions such as [13] provide authentication, but are not analyzed in the context of real-time constraints.

To the best of our knowledge, we are the first to propose the aggregated selective packet authentication protocol in the context of ICS. The idea is inspired by a delayed authentication scheme for GPS proposed in [16].

7 Conclusions

In this work, we discussed the introduction of efficient legacy-compliant authentication in ICS networks. By design, our protocols are backward compatible with devices not implementing them, as they transmit authentication data as payload in legacy industrial protocols.

We have shown that with the advent of inexpensive and fast hardware (such as the Raspberry Pi), it is feasible to enhance legacy plants with constraints similar to the ones of SWaT with authentic channels for strong signature algorithms with simple protocols. However, introducing state of the art asymmetric algorithms (which are advantageous from the key management point of view), or implementing our solution in slower hardware requires careful performance trade-offs. In future work, we plan to compare the real-time constraints of SWaT with constraints found in other ICS test-beds (i.e. smartgrid).

A Appendix: Real-Time Requirements and Backlogs

The delay produced by the signature module can be represented as a queue of packets pending to be signed after a time interval Δ: $Q^S_\Delta = g(\Delta) - s(\Delta)$. We assume an empty queue at the beginning of the time interval Δ. To avoid an avalanche effect on the packet queue Q^S_Δ the queue must remain below a certain constant threshold C: $\forall\ \Delta.\ Q^S_\Delta < C$. In particular, since Δ is arbitrary $C \approx 0$ (i.e., $\forall\ \Delta.\ Q^S_\Delta < 0$) and thus: $\forall\ \Delta.\ g(\Delta) < s(\Delta)$. Similarly, if $r(\Delta)$ is the number of expected packets in a time interval Δ, then the number v of verified packets in this interval must be $v(\Delta, cpu, alg) > r(\Delta)$ to avoid backlogs of the verification queue Q^V_Δ.

$s(\Delta)$ depends not only on the time but also on the algorithm used, the CPU capacity, and the size of the packets. In a fraction of time Δ, the device will produce not only packets at different rates, but will also nondeterministically produce packets of different sizes. To simplify our analysis, and without loss of generality, we thus set $\Delta = 1\,\mathrm{s}$, and $s(cpu, alg)$ the rate at which packets of a certain constant expected size can be signed using a certain *cpu* and a given signature algorithm *alg*. We use s, v, g and r to abbreviate the rates per second of the signing, verification, outgoing (or *generated*) and incoming packets respectively.

Parallel versus Sequential Signature and Verification. We are assuming an architecture where network communications are handled by dedicated hardware. This avoids sharing the main device CPU (such as the main PLC computing unit), which is busy computing other control and communication related tasks. In practice this is usually the case, since communications are often handled by a dedicated network module. For a parallel implementation of the protocol, where the signature and verification components have at least a single core each, then we have effectively two queues that we can consider separate, Q^S_Δ and Q^V_Δ. Otherwise, for a sequential implementation of the protocol, an incoming packet that

needs to be verified has to potentially wait until an outgoing packet is signed; and vice versa, an outgoing packet has to wait until a verification is finished. We have thus the constraints: $\forall\ \Delta.\ Q^S_\Delta < C \wedge Q^V_\Delta < C'$ which we can simplify by considering a single queue $Q_\Delta = Q^S_\Delta \cup Q^V_\Delta = (g(\Delta) - s(\Delta)) + (r(\Delta) - v(\Delta)) < C + C'$. As above, $C + C' \approx 0$ and thus: $g(\Delta) + r(\Delta) < s(\Delta) + v(\Delta)$. In other words, the rate of signature and verification in a time interval has to be faster than the amount of packets sent and received in that same time interval.

B Appendix: Authenticated Link Using Raspberry Pi

We used two (paired) Raspberry Pi3, (as shown in Fig. 7) each with two Ethernet adapters. We choose a link between two PLCs in our SWaT to perform the test. The devices were configured as Ethernet bridges and placed as physical Man-in-the-Middle over the link. This choice was motivated because the source code of the Network Adapter in the SWaT was unavailable to us (as such proprietary code is confidential), and we could not directly implement our solution without additional hardware. However, a vendor could easily deploy our solution.

Once connected, the devices passively listen to packets from and to the PLCs. When a critical-data packet is identified, it is captured and the ENIP payload is signed with HMAC-SHA256 algorithm using a pre-shared key. The concatenation of the captured packet and its signature is injected back into the communication channel.

Fig. 7. Raspberry Pi3 connected between PLCs.

Similarly, the remote Raspberry Pi3 is placed as a verification module in front of the destination PLC. Once the verification module identifies a packet coming from its counterpart, the packet is analysed looking for an attached signature, the signature is extracted and verified against the ENIP payload using HMAC-SHA256 algorithm with the pre-shared key. The packet is converted back to its original version and it is delivered to its destination.

We configured four additional variables in the PLCs to store information about the authentication processes: *Signed messages*: Number of signed messages by its cryptographic module; *Checked messages*: Number of signed message correctly verified by its cryptographic module; *Wrong-signature messages*: Number of signed messages which signature does not correspond to its payload detected by its cryptographic module; *No-signed messages*: Messages with critical data from a peered host with no-attached signature detected by its cryptographic module.

References

1. Batke, B., Wiberg, J., Dubè, D.: CIP security phase 1 secure transport for Ethernet/IP. In: ODVA Industry Conference (2015)
2. Cárdenas, A.A., Amin, S.M., Sinopoli, B., Giani, A., Perrig, A., Sastry, S.S.: Challenges for securing cyber physical systems. In: Workshop on Future Directions in Cyber-physical Systems Security, DHS, July 2009
3. Cárdenas, A.A., Baras, J.S.: Evaluation of classifiers: practical considerations for security applications. In: AAAI Workshop on Evaluation Methods for Machine Learning (2006)
4. Dierks, T.: The transport layer security (TLS) protocol version 1.2 (2008). https://www.ietf.org/rfc/rfc5246.txt
5. Fletcher, K.K., Liu, X.: Security requirements analysis, specification, prioritization and policy development in cyber-physical systems. In: Secure Software Integration Reliability Improvement Companion (SSIRI-C), pp. 106–113 (2011)
6. Galloway, B., Hancke, G.: Introduction to industrial control networks. Commun. Surv. Tutor. **15**(2), 860–880 (2013). IEEE
7. Gomes, N., Mattos, L.: Attacks detection based on IP and TCP protocols violation. Int. J. Forensic Comput. Sci. **1**, 49–56 (2006)
8. Hash libraries for arduino. http://spaniakos.github.io/Cryptosuite/
9. Igure, V.M., Laughter, S.A., Williams, R.D.: Security issues in scada networks. Comput. Secur. **25**(7), 498–506 (2006)
10. P. Inc.: Profinet and it. Technical report, PROFIBUS Nutzerorganisation e.V. (2008)
11. iTrust: Center for Research in Cyber Security. Secure water treatment test-bed. http://itrust.sutd.edu.sg/research/testbeds/secure-water-treatment-swat/
12. Johnson, D., Menezes, A., Vanstone, S.: The elliptic curve digital signature algorithm (ECDSA). Int. J. Inf. Secur. **1**(1), 36–63 (2001)
13. Karlof, C., Sastry, N., Wagner, D.: TinySec: a link layer security architecture for wireless sensor networks. In: Proceedings of the International Conference on Embedded Networked Sensor Systems, SenSys 04, pp. 162–175. ACM (2004)
14. Krawczyk, H., Bellare, M., Canetti, R.: HMAC: keyed-hashing for message authentication (1997). https://www.ietf.org/rfc/rfc2104.txt
15. Krotofil, M., Cárdenas, A.A., Manning, B., Larsen, J.: CPS: driving cyber-physical systems to unsafe operating conditions by timing DoS attacks on sensor signals. In: Proceedings of the Computer Security Applications Conference (ACSAC), pp. 146–155. ACM (2014)
16. Kuhn, M.G.: An asymmetric security mechanism for navigation signals. In: Fridrich, J. (ed.) IH 2004. LNCS, vol. 3200, pp. 239–252. Springer, Heidelberg (2004). doi:10.1007/978-3-540-30114-1_17
17. Gnu cryptographic library. https://www.gnu.org/software/libgcrypt/
18. Majdalawieh, M., Parisi-Presicce, F., Wijesekera, D.: DNPSec: distributed network protocol version 3 (DNP3) security framework. In: Elleithy, K., Sobh, T., Mahmood, A., Iskander, M., Karim, M. (eds.) Advances in Computer, Information, and Systems Sciences, and Engineering, vol. 3, pp. 227–234. Springer, Dordrecht (2007). doi:10.1007/1-4020-5261-8_36
19. Modbus-IDA. Modbus messaging on tcp/ip implementation guide v1.0b. Technical report, Modbus Organization (2006)

20. Nie, P., Vähä-Herttua, J., Aura, T., Gurtov, A.: Performance analysis of HIP diet exchange for wsn security establishment. In: Proceedings of the ACM Symposium on QoS and Security for Wireless and Mobile Networks, Q2SWinet 11, pp. 51–56. ACM (2011)
21. Radu, A.I., Garcia, F.D.: LeiA: a lightweight authentication protocol for CAN. In: Askoxylakis, I., Ioannidis, S., Katsikas, S., Meadows, C. (eds.) ESORICS 2016. LNCS, vol. 9879, pp. 283–300. Springer, Cham (2016). doi:10.1007/978-3-319-45741-3_15
22. Rajkumar, R., Lee, I., Sha, L., Stankovic, J.: Cyber-physical systems: the next computing revolution. In: 2010 47th ACM/IEEE on Design Automation Conference (DAC), pp. 731–736, June 2010
23. Sandaruwan, G., Ranaweera, P., Oleshchuk, V.A.: PLC security and critical infrastructure protection. In: Industrial and Information Systems (ICIIS), pp. 81–85. IEEE (2013)
24. Schiffer, V., Vangompel, D., Voss, R.: The common industrial protocol (CIP) and the family of CIP networks. ODVA, Ann Arbor (2006)
25. Shahzad, A., Lee, M., Lee, Y.-K.K., Kim, S., Xiong, N., Choi, J.-Y.Y., Cho, Y.: Real time MODBUS transmissions and cryptography security designs and enhancements of protocol sensitive information. Symmetry **7**(3), 1176–1210 (2015)
26. Slay, J., Miller, M.: Lessons learned from the maroochy water breach. In: Goetz, E., Shenoi, S. (eds.) ICCIP 2007. IIFIP, vol. 253, pp. 73–82. Springer, Boston, MA (2008). doi:10.1007/978-0-387-75462-8_6
27. Smart, N., Babbage, S., Catalano, D., Cid, C., Weger, B. d., Dunkelman, O., Ward, M.: Ecrypt ii yearly report on algorithms and keysizes (2011–2012). In: European Network of Excellence in Cryptology (ECRYPT II) (2012)
28. Tsang, P.P., Smith, S.W.: YASIR: a low-latency, high-integrity security retrofit for legacy SCADA systems. In: Jajodia, S., Samarati, P., Cimato, S. (eds.) SEC 2008. ITIFIP, vol. 278, pp. 445–459. Springer, Boston, MA (2008). doi:10.1007/978-0-387-09699-5_29
29. Urbina, D., Giraldo, J., Tippenhauer, N.O., Cárdenas, A.: Attacking fieldbus communications in ICS: applications to the SWaT testbed. In: Proceedings of Singapore Cyber Security Conference (SG-CRC), January 2016
30. Van Herrewege, A., Singelee, D., Verbauwhede, I.: CANAuth-a simple, backward compatible broadcast authentication protocol for CAN bus. In: ECRYPT Workshop on Lightweight Cryptography, vol. 2011 (2011)
31. Wang, W., Lu, Z.: Cyber security in the smart grid: survey and challenges. Comput. Netw. **57**(5), 1344–1371 (2013)
32. Weinberger, S.: Computer security: is this the start of cyberwarfare? Nature **174**, 142–145 (2011)
33. Wright, A.K., Kinast, J.A., McCarty, J.: Low-latency cryptographic protection for SCADA communications. Acns **3089**, 263–277 (2004)