LASARUS: Lightweight Attack Surface Reduction for Legacy Industrial Control Systems

Anhtuan Le, Utz Roedig, and Awais Rashid

Security Lancaster Institute, Lancaster University, UK

Abstract. Many operational Industrial Control Systems (ICSs) were designed and deployed years ago with little or no consideration of security issues arising from an interconnected world. It is well-known that attackers can read and write sensor and actuator data from Programmable Logic Controllers (PLCs) as legacy ICS offer little means of protection. Replacing such legacy ICS is expensive, requires extensive planning and a major programme of updates often spanning several years. Yet augmenting deployed ICS with established security mechanisms is rarely possible. Legacy PLCs cannot support computationally expensive (i.e., cryptographic) operations while maintaining real-time control. Intrusion Detection Systems (IDSs) have been employed to improve security of legacy ICS. However, attackers can avoid detection by learning acceptable system behaviour from observed data. In this paper, we present LASARUS, a lightweight approach that can be implemented on legacy PLCs to reduce their attack surface, making it harder for an attacker to learn system behaviour and craft useful attacks. Our approach involves applying obfuscation to PLC data whenever it is stored or accessed which leads to a continuous change of the target surface. Obfuscation keys can be refreshed depending on the threat situation, striking a balance between system performance and protection level. Using real-world and simulated ICS data sets, we demonstrate that LASARUS is able to prevent a set of well-known attacks like random or replay injection, by reducing their passing rate significantly—up to a 100 times.

1 Introduction

Industrial Control Systems (ICSs) are used to monitor and control systems such as power plants, power grids, water treatment plants, oil refineries and gas distribution networks. Many ICS are legacy systems and offer little means of protection. Specifically, attackers are able to read and write sensor and actuator data of Programmable Logic Controllers (PLCs) in much the same way as the operator of the plant does (see, e.g., [1, 2]). Thus, an attacker is able to inject false command instructions to control the physical process (managed by the PLCs), while at the same time injecting false response packets to mislead remote monitoring stations and operators. Such Man-In-The-Middle (MitM) attacks can enable attackers to: steal physical resources (oil, gas, water); alter production processes or bypass safety procedures, all the while leaving remote operators blind to such actions [3]. Attackers may also hide their activities from operators for a long time by falsifying data carefully, based on long-term observation of normal behaviour before starting an attack. The security vulnerabilities of ICS have been demonstrated by a number of high profile attacks such as Maroochy water services (2000), Stuxnet (2007), Turkish pipeline (2008), and the German steel mill (2014) [4].

Many ICS services need to be operated in real time, which restricts the performance latency within a strong bound. For example, according to IEC 61850 (reference architecture for electric power systems), the latency bound for fault isolation and protection services is 3ms [4]. On the other hand, applying access control and cryptographic solutions can create significant delay, for instance, the latency of applying a 1024 bit DSA algorithm in an average system is 14.9ms while that of applying the 2048 bit RSA is 61.04ms [4]. The delay created by cryptographic solutions may exceed largely from the required bounds, which will disable certain ICS real-time operations. Therefore, augmenting legacy ICS with well known access control and cryptographic protocols to address the aforementioned issues is rarely possible as the required major system upgrades are not feasible to satisfy such performance requirements. In many cases, ICS are in operation for more than 20 years and it is not possible to augment these systems as necessary. Replacement of the entire control infrastructure is also often not possible as it is too costly and would require system shutdown for lengthy periods. Finally, in many cases, real-time control requirements make it impossible to use time consuming cryptographic procedures.

A number of Intrusion Detection Systems (IDSs) have been proposed, e.g., [1, 5, 6]. However, capable attackers can study system behaviour over long periods and manipulate the system state such that it still falls within acceptable bounds not detected by the IDS [7]. In a worst-case scenario, the attacker may have all past system data and the same detection algorithm available and can craft attacks that are undetectable.

As current legacy ICS give an attacker too much information, in this paper we focus on reducing their attack surface in order to make it harder for an adversary to carry out a successful attack. Our proposed approach, LASARUS, is designed to operate under real-world constraints in ICS environments, namely:

- Deployment of sophisticated access control and cryptographic solutions is not possible due to architectural and resource constraints.
- The approach must be implementable using available functionality in legacy PLCs.
- The approach must be lightweight so as not to compromise the real-time properties of an ICS.

LASARUS involves the use of an obfuscation key K_d to obfuscate data deposited in the data block of a PLC, with the obfuscation pattern changing on a per session basis. The obfuscation key K_d is shared between all components that have to access the data block (i.e. shared between operator and PLC). The obfuscation key is updated periodically whereby the frequency of key updates is determined by system needs. The obfuscation function is a simple XOR operation, making it possible to be implemented by existing legacy PLC programming frameworks while fast and lightweight enough to limit the latency and processing overhead so as not to compromise real-time PLC operations.

The novel contributions of this paper are as follows:

- We propose a lightweight approach to reduce the attack surface of legacy ICS that can be implemented on legacy PLCs.
- We demonstrate the effectiveness of the approach by evaluating it against three types of attacks: (i) injecting random values; (ii) replaying eavesdropped values; (iii) injecting estimated values. We show that adding LASARUS to a legacy ICS can help to reduce the probability of these attacks to bypass an IDS (even one using a simple algorithm) significantly, up to a 100 times.
- We provide an analysis of security and usability trade-offs of LASARUS, supporting system configuration decisions so that the additional security features introduced by LASARUS do not compromise operation.

The structure of the paper is as follows. Section 2 presents the background on ICS and related work on ICS security. Section 3 describes the Man-In-The-Middle (MitM) attacks that are the focus of our approach. Section 4 introduces the proposed approach while Section 5 shows the evaluation and presents a discussion of the insights. Finally, Section 6 concludes the paper.

2 Background and Related Work

2.1 Industrial Control Systems

ICSs are used to control physical processes in a range of critical infrastructure such as, power plants, water works, oil refineries and gas distribution. Programmable Logic Controllers (PLCs) are used at the heart of any ICS to implement control logic, interacting with sensors and actuators. PLCs store data computed by the control process, sent to actuators and obtained from sensors locally, often referred to as "data block"[1]. Data in the data block is used to control the PLC behaviour and, thus, the physical process. The data block is also accessed externally by other ICS equipment, including management systems and other PLCs. For example, an operator console will fetch data from the PLC's data block to visualise system state. A historian, a database to store long-term process data, will fetch data periodically from the data block. In a Supervisory Control and Data Acquisition (SCADA) system other PLCs may obtain data from the data block to facilitate complex distributed control tasks.

Although ICSs are used to control many critical infrastructures, their security is rarely considered properly. A reason can be found in their historic development. In the past, ICS were air-gapped systems using specialist system parts, including software and hardware, with which only few people were familiar. Thus, attacks were unlikely as physical presence of highly skilled personnel was required. Today this situation has changed. To improve operational efficiency, ICS are now connected to enterprise systems and, in many cases, also to the

[1] This term is used for Siemens PLC equipment but a similar data construct is used by other vendors too.

Internet [8]. Also, the systems are easier to use and to program, and knowledge to interact with these computer systems is much more widespread. Attacks are, therefore, now more likely and ICS security must step up.

2.2 Security of ICS

Apart from real world reported ICS attacks, recent research has also highlighted various potential threats to ICS. For instance, Stouffer et al. [9] present a wide range of probable adversaries – from inexperienced attackers utilising prewritten scripts through to complex and organised Advanced Persistent Threats (APT), which are crafted carefully aiming at system manipulation over long period of time. Morris and Gao [10] describe 17 attacks on ICS, grouping them into four main types, namely: reconnaissance attacks, response and measurement injection attacks, command injection attacks, and denial of service attacks. MitM attacks in ICSs are also discussed widely in the literature, and shown to be feasible in testbed environments, for instance, see [11, 12]

Although security mechanisms are well developed for other IT systems, these cannot be simply transferred to the ICS domain [2, 13]. A number of intrusion detection systems (IDS), specific to ICS, have been proposed. For instance, Sainz et al. [6] apply modal interval analysis to derive suitable envelopes, including upper and lower bounds for values according to time, in order to check the veracity of the data. Hadziosmanovic et al. [5] model logged data using an autoregressive model to predict the next value to detect potential tampering. Almalawi et al. [1] introduce a detection approach, which first clusters the process parameters to identify the normal and critical states of the system. They then apply data mining techniques to extract the proximity-based detection rules from such states. Given that the MitM attacker may have access to the logged data as much as the operator does, the attacker can also predict the next value with similar accuracy. Therefore, such attacks may not be readily detected by IDSs.

As discussed earlier, augmenting deployed ICS with well known access control and cryptographic protocols is not feasible due to the cost of updating the infrastructure or constraints arising from real-time properties. Therefore, any security solution needs to be lightweight and must require little change to be compatible with existing parts of the system. Existing research on Moving Target Denfense (MTD) has considered the problem of reducing the attack surface of systems. Davidson and Andel [14] are among the first to point out the feasibility and effectiveness of applying MTD to ICS security. However, to the best of our knowledge, there has been no attempt to design such a system so far. LASARUS can thus be seen as a concrete instantiation of such an MTD approach.

3 Problem Definition

3.1 Threat Model

Figure 1 shows a simplified MitM attack in ICS. As shown, the physical system communicates with the PLC through its actuators and sensors. The PLC controls the physical side based on parameters received from the system operator. The state of the physical system is reported to and recorded by the PLC. State information is also transmitted periodically to the operator who observes the system and may tune parameters in response. In practice the control side may be connected to multiple PLCs and multiple physical systems at the same time. However, for simplification we consider here a single PLC and operator.

The MitM attacker is able to interact with the PLC in a fashion similar to that of the operator. He can request recorded physical system state logged by the PLC and can inject operation parameters into the PLC. He may also simply modify data passed between operator and PLC without direct manipulation of the PLC's data block. Such attacks can be implemented using a variety of techniques. For example, by uploading malicious control logic to gain control of devices that the PLC manages [15]; compromising Address Resolution in the network to manipulate communication between operator and PLC [9]; or eavesdropping on the communication followed by analysing and altering its contents [16].

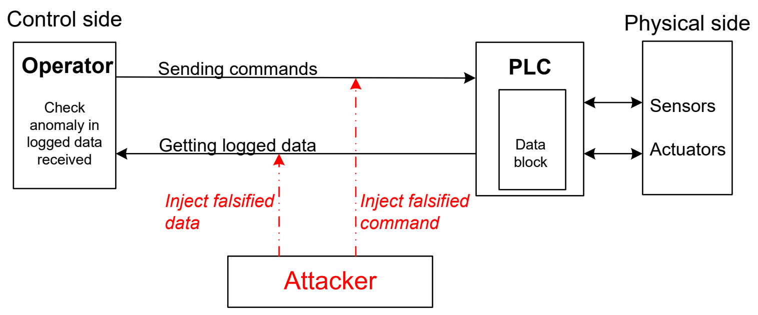


Fig. 1: ICS Man-In-The-Middle (MitM) attack model.

3.2 Example ICS

We consider an ICS used to control a water tank as typical example to illustrate MitM attacks. We also use this example application later for our evaluation. The physical system in this particular example is a water system, which consists of: a tank, a pump to fill the tank and a sensor to record water level. The PLC controls the pump according to the following parameters set by the operator: HH, H, LL, L [17]. HH and LL are two alarm values. If the water level rises higher than HH or falls lower than LL, the operator will raise an alarm. H and L are the two values to control the pump. If the water level is higher than H, the pump will need to be turned off. If it falls lower than L, the pump will need to be turned on. These operation parameters submitted to the PLC are stored in the data block. The filling level is read periodically and these readings are stored as well in the PLC's data block. Filling levels are requested periodically by the operator from the data block to monitor the process.

The system can be attacked by injecting false values into the data block. For example, Morris et al. [18] have shown that simple replay attacks (storing a previous value as the next value in the data block) are only detected in 12% of cases. Improved attacks by Morris et al. [10] simulate the value trend and estimate the next value, making the detection accuracy even worse. If the attacker can collect a significantly large sample of data and apply robust techniques to learn the pattern, the chances to bypass the detection system are likely to be even higher. In the worst case, when the attacker knows and applies the algorithm that the system uses to check the data, he can always inject a legal value into the system and the attack will go undetected. The operator will loose control of the system while thinking that everything is in order.

Obviously, anomaly detection can be improved but the attacker can adjust accordingly. As operational parameters and sensor readings are available to the operator and attacker via the data block, the problem remains.

4 Lightweight Attack Surface Reduction with LASARUS

As the problem arises from the large attack surface easily accessible to the attacker, we propose an approach whereby data maintained in a PLC's data block is obfuscated, with the obfuscation pattern changing on a per session basis. An obfuscation key K_d is used to obfuscate data deposited in the data block. Likewise, data obtained from the data block must be de-obfuscated using K_d. The obfuscation key K_d is shared between all components that have to access the data block (i.e. shared between operator and PLC). The obfuscation key is updated periodically whereby the frequency of key updates is determined by system needs (we discuss this aspect in detail in our evaluation for a specific application case). The obfuscation function is a simple XOR operation. We chose this simple approach as obfuscation can then be implemented by existing legacy PLC programming frameworks and processing overhead is limited as required for real-time PLC operations.

We aim at manipulating the data block in a PLC such that an attacker has a limited view on system behaviour. The attacker should not be able to learn acceptable value ranges in order to craft sophisticated attacks by injecting seemingly acceptable values. In order to do so, we assume that there is a shared secret key K between the PLC and other system components (such as the operator). This key can be stored at a specific memory address in the PLC which cannot be written to or read by any other party. We assume that this key is valid over a long time. The issue of key distribution is beyond the scope of this paper. However, other research has tackled this question, e.g., [19].

As shown in Figure 2, the shared secret K is used to calculate a session key K_d for each session d. K_d is used within each session to obfuscate data in the PLC's data block. A session is active for an agreed duration T_s and, after this time elapses, a new session key will be used. Note that we can generate this new key right after the last logging cycle of the session instead of waiting until session ends. As the logging frequency is considerably longer (e.g., 10 seconds) compared to the time needed to generate a key (e.g., under a millisecond using hash), generating a new session key right after the last logging cycle will eliminate the impact of any delay it may cause. Session keys are generated based on a time stamp T_d and the shared secret K. Therefore, session keys do not need to be distributed; they are computed by all involved ICS components based on the same time base. It is feasible to follow this approach in an ICS context as components require tight time synchronisation to facilitate real-time control operations. An attacker can disrupt time synchronisation; however, this will generally lead to system faults which are immediately obvious. Also, manipulating time synchronisation will lead to de-synchronisation of obfuscation keys which will lead to parts of the ICS not being able to access data which is also immediately detectable. The timestamp for session d is calculated as $T_s \cdot (d-1)$. The session key K_d for session d is calculated as: $K_d = hash(K \oplus T_d)$.

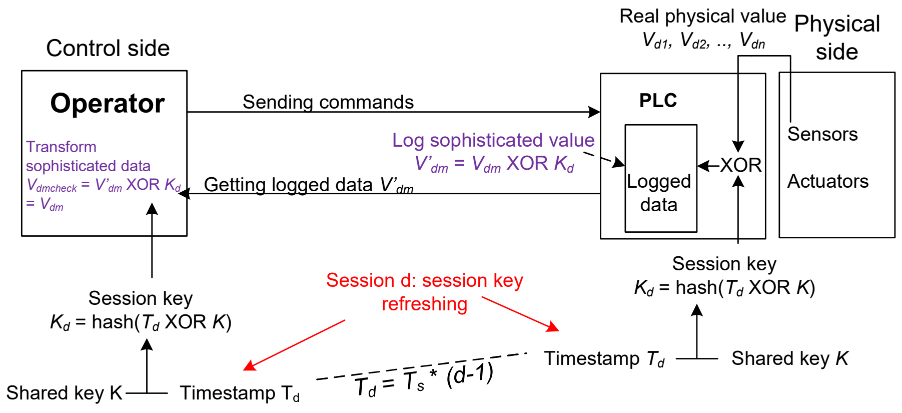

Control side
Operator
Transform sophisticated data $V_{dmcheck} = V'_{dm}$ XOR K_d = V_{dm}
Sending commands
Log sophisticated value $V'_{dm} = V_{dm}$ XOR K_d
Getting logged data V'_{dm}
Real physical value $V_{d1}, V_{d2}, .., V_{dn}$
Physical side
PLC
Logged data
XOR
Sensors
Actuators
Session key K_d = hash(T_d XOR K)
Session d: session key refreshing
Session key K_d = hash(T_d XOR K)
Shared key K
Timestamp T_d
$T_d = T_s * (d-1)$
Timestamp T_d
Shared key K

Fig. 2: Obfuscating the logged values using the session key to prevent MitM attacks.

The obfuscation pattern changes in every session, though the real physical values may remain the same. A Man-In-The-Middle who does not know the shared key and time reference will not be able to generate useful values for an attack.

Assume that for session d, the physical values are V_{d1}, V_{d2}, V_{d3}, ..., V_{dn}. The PLC will store the obfuscated representation of these values, V'_{d1}, V'_{d2}, V'_{d3}, ..., V'_{dn}, in which:

$$V'_{dm} = V_{dm} \oplus K_d, \; m = 1, 2, ..., n$$

When the control side receives data from session d, it will use the session key K_d to obtain the original values:

$$V_{dmcheck} = V'_{dm} \oplus K_d = V_{dm} \oplus K_d \oplus K_d = V_{dm}, \; m = 1, 2, ..., n$$

These original values may be used to detect if any anomaly is present in the physical system. For an attacker, it will be challenging to predict which obfuscated values to inject such that an anomaly detection system (even a very simple one) does not trigger an alarm.

5 Evaluation

We evaluate the effectiveness of LASARUS on three types of attacks: (i) injecting random values; (ii) replaying eavesdropped values; (iii) injecting estimated values. These three attack scenarios are the best options for a MiTM attacker to bypass the defense system as shown in other work [10, 17].

5.1 Evaluation Scenario and Datasets

We use the water storage tank dataset from [18]. The dataset was recorded from a water tank ICS testbed under normal and attack conditions. It contains attributes such as pump states, the automatic thresholds for turning the pump on/off or raising alarm, the physical water level, the response time of the PLC, and so on. We use only the water level attribute as test value in our experiment which is sampled every 11.5 seconds. We pre-processed the data to exclude values recorded under attack, only using data from normal operation conditions. We use 10044 water level values collected over a duration of approximately 32 hours for our experiments. We refer to this real-world dataset as *Set-Real*.

We assume that the attackers have the ability to learn patterns from the eavesdropped data, so we are also interested in seeing whether the obfuscated data contains patterns that can be manipulated. Therefore, we generate three artificial datasets with clear patterns: *Set-Linear:* the water level is a function of the time and pump velocity; *Set-Repeated:* a range of the Set-Real is chosen and repeated continuously; *Set-Constant:* all values are constant, e.g., when the system is inactive and the logged water values are not changed.

5.2 Experiment Setup

Session Duration Obfuscation keys K_d change after a time period (the session duration T_s). In our experiments we use the following configurations for T_s: T_s = 10s; T_s = 250s and T_s = 500s. The PLC will use the session key to obfuscate the sampling water level and record to a data block. The operator will use the session key to transform values back to the real physical water level.

Intrusion Detection We assume that the control side (the operator) employs a means to detect intrusions. We assume the following two detection rules:

- Rule-Simple (R-S): This rule checks whether the water level is within a believable range. We assume the range to be between 20% and 80% of tank capacity. This rule is named simple because it does not involve any particular data learning algorithm.
- Rule-Advanced (R-A): This rule checks whether the water level falls into a legitimate envelope, which is not 10% higher or 10% lower than the true value. We assume here a prediction algorithm is in place which can estimate true values. This rule is called advanced because it emulates a more sophisticated intrusion detection mechanism.

These rules are used to demonstrate how LASARUS supports an existing IDS.

Random Attack We assume two attack conditions which differ regarding the attacker's knowledge of data obfuscation being in place or not:

- Normal Range Random (NRR) attack: The attacker does not know that obfuscation is used. Random value generation assumes values should fall in the normal data range of water values.

– Obfuscated Range Random (ORR) attack: The attacker knows that obfuscation is used. The attacker generates values falling in the range of the valid obfuscated data range.

Replay Attack The attacker can choose to replay data differently to bypass the system. For Set-Real, the attacker is able to capture a value (obfuscated or not) and use this value for replay. For Set-Linear, Set-Repeated, Set-Constant, the attacker can replay a set of values in the pattern that he learnt.

5.3 Results and Discussions

Dataset Observation Figure 3 illustrates the first 250 values of the dataset S-Real. Figure 3a shows the real water level, while Figures 3b, c, d shows the obfuscated values with session duration T_s = 10s, 250s, and 500s respectively. As can be seen, when values are similar, their obfuscated counterparts will also be similar if transformed with the same key. When the keys are updated for every value ($T_s = 10s$), the transformed values fluctuated strongly as shown in Figure 3b, which makes it hard for an attacker to predict future values. On the other hand, all the obfuscated datasets have a large standard deviation at about 1200 (compared with 8.48 of the original dataset), with the values spread over a large range from about -2100 to 2100, which indicates such datasets are not stable. The initial observations of the dataset shows it would be much more difficult for the attackers to inject a legal value into the obfuscated datasets compared with injecting into the original dataset.

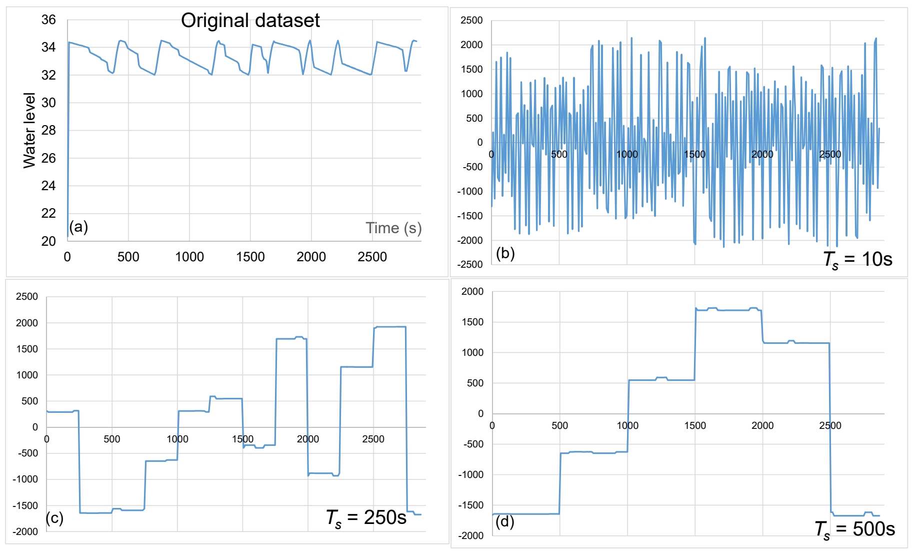


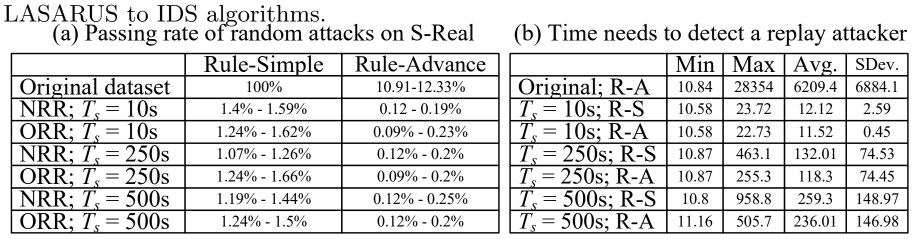
Fig. 3: The first 250 values of the water level dataset S-Real and the obfuscated datasets with $T_s = 10s$, $250s$, and $500s$.

Random Attack We now investigate the effectiveness of the two types of random attacks previously described. For each type of the random attack (NRR, ORR), we generate random values in the [Min Max] range for every entry in the dataset and check if these are detected by the two detection rules (R-S, R-A).

Table 1a shows the passing rates of the two types of random attacks. We first apply the random attacks on the original dataset without obfuscation in place. The results show that R-S cannot detect either of the random attacks because all the injected values fall in the legitimate data range [20, 80]. R-A improves the detection ability significantly, which lets the passing rates fall to values between 10.91% and 12.33%. LASARUS, however, reduces the passing rate even more, which is less than 2% if using R-S, and less than 0.25% if using R-A to detect the attacker. With the same detection technique, the obfuscated solutions are about 100 times more effective when compared with the non-obfuscated solution.

The length of the session key does not affect the passing rate. This is because after being obfuscated, the range and the standard deviation of the dataset values are similar, which create similar results when dealing with the random attacks.

Table 1: Improvements of reducing passing rates and detection time when combining LASARUS to IDS algorithms.



(a) Passing rate of random attacks on S-Real

	Rule-Simple	Rule-Advance
Original dataset	100%	10.91-12.33%
NRR; T_s = 10s	1.4% - 1.59%	0.12 - 0.19%
ORR; T_s = 10s	1.24% - 1.62%	0.09% - 0.23%
NRR; T_s = 250s	1.07% - 1.26%	0.12% - 0.2%
ORR; T_s = 250s	1.24% - 1.66%	0.09% - 0.2%
NRR; T_s = 500s	1.19% - 1.44%	0.12% - 0.25%
ORR; T_s = 500s	1.24% - 1.5%	0.12% - 0.2%

(b) Time needs to detect a replay attacker

	Min	Max	Avg.	SDev.
Original; R-A	10.84	28354	6209.4	6884.1
T_s = 10s; R-S	10.58	23.72	12.12	2.59
T_s = 10s; R-A	10.58	22.73	11.52	0.45
T_s = 250s; R-S	10.87	463.1	132.01	74.53
T_s = 250s; R-A	10.87	255.3	118.3	74.45
T_s = 500s; R-S	10.8	958.8	259.3	148.97
T_s = 500s; R-A	11.16	505.7	236.01	146.98

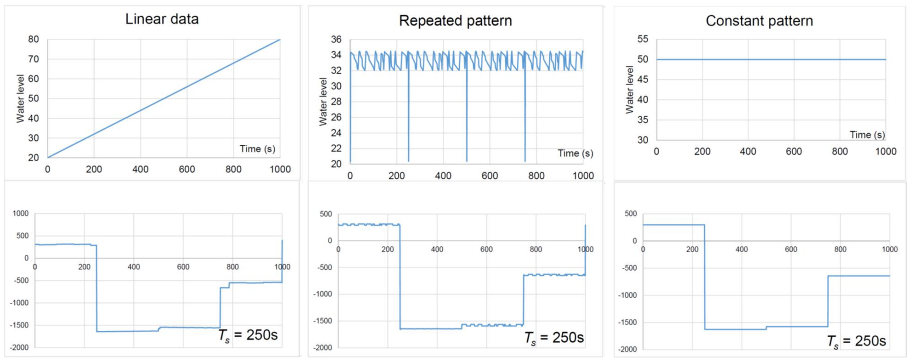
Replay Attack Next we investigate the effectiveness of replay attacks. First, we select a random time at which to initiate the attack. From that time till the end of the dataset, we replace the water level value with the selected replay value. We then check the modified dataset using the two detection rules to measure how long it takes to detect the attacker. We run 1000 attacks for each experiment.

Table 1b shows the time to detect the replay attacker for each setting. For the original dataset (no obfuscation), R-S cannot detect the replay attacker because any replayed value will fall into the legitimate value range of [20, 80]. Using R-A the system can catch the replay attacker after 6209 seconds (approx. 2 hours), on average. In the worst-case scenario, it will take 28354 seconds (almost 8 hours).

Table 1b shows that the shorter the time to change the session key, the quicker the defense system can point out the anomaly on the dataset. The time to detect a replay attacker depends on when the attack is initiated. If the attack is initiated at the beginning of a session, it is likely to be undetected until a new key is replaced. On the other hand, when the attack is initiated at the end of a session, it will be spotted quickly when a new key is used.

Table 1b also shows the maximum time that is required to detect a replay attacker. If using R-S, the maximum time is approx. twice the session length. This is because the XOR function applied on the value will likely jump out of the legal range [20, 80] after two key changes. While using R-A, the maximum time to detect a replay attacker is only approx. the length of the session. This is due to the checking range of R-A being much narrower compared with R-S, which helps to detect the attacker right after changing the session key.

Attacks on Data with Patterns Once the attackers find the pattern, the best chance they have to bypass the system is to do the replay attack which inject data following the found pattern. Figure 4 illustrates the differences between the original and obfuscated datasets of the three most common patterns.



(a) Obfuscation with S-Linear; $T_s = 250$s (b) Obfuscation with S-Repeated; $T_s = 250$s (c) Obfuscation with S-Constant; $T_s = 250$s

Fig. 4: Patterns of the three data sets.

Figure 4a compares the pattern of a linear rising water level in Set-Linear with its obfuscated versions. The figure clearly shows that a linear original value will not lead to similar pattern on the transformed value. The flat lines present in the obfuscated figures are due to the similarity of the original values.

Figure 4b presents the repeated pattern in Set-Repeated and its obfuscations. When the session length is smaller than the length of the pattern, the obfuscated datasets are not repeated like the original pattern. This is because different keys were applied in the same dataset. However, when the session is longer than the pattern duration, the repeated real values XORing with the same session key create repeated obfuscated values. Nevertheless, unlike in the original Set-Repeated, the pattern in the obfuscated set does not repeat over more than one session due to the session key change.

Figure 4c shows the comparisons between the original patterned dataset Set-Constant and the obfuscated values when the original data is constant. As can be seen, a transformed constant value will also be constant given the same obfuscation key. When the session key is not changed, a replay attack will be successful for any detection algorithm. However, at the moment when the session key is changed (or the starting of a new session), the attackers will likely fail as they cannot predict the next value.

The experiments show that replay attacks on patterned dataset will not be successful in multiple sessions. In Figure 5, we illustrate the passing possibility of such attack within one session. The linear pattern of Set-Linear does not help the attacker to increase the chance to bypass the system. On the other hand, repeated pattern of Set-Repeated can create repeated patterns of obfuscated dataset, which helps the replay attack to be successful after the first cycle of the pattern. The worst case for the detection system happens with Set-Constant, where the attacker only fails in the first sampling time of the session. However, protecting the constant dataset from the replay attack is also a challenge for any other defending mechanisms. If the system operator knows that their dataset contains patterns, s/he can adjust the session duration to interrupt the pattern in the obfuscated set. For example, if the data is a repeated pattern (not constant), setting the session duration smaller than the pattern's length eliminates the chance of replay attacks. If the system operator cannot do so, s/he will face the risk of not detecting the attackers at some points in the session. However, the attacker will always fail at the beginning of a new session.

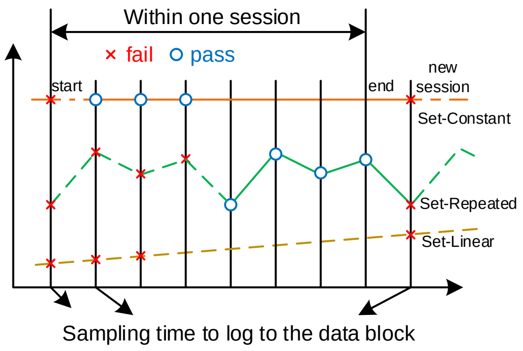
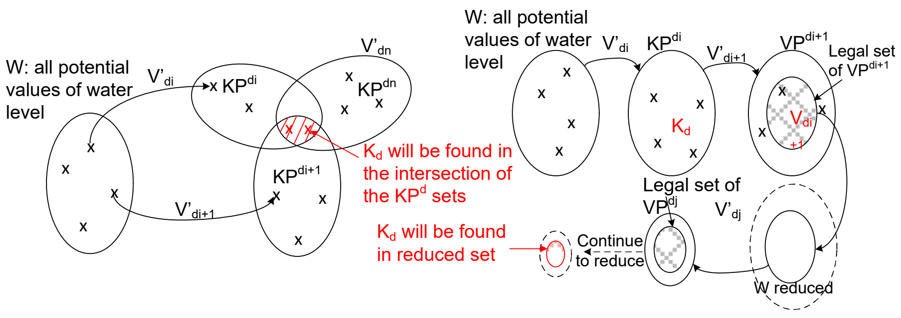


Fig. 5: Comparison of passing rate in one session for the three pattern sets.

5.4 Possible Attacks on Obfuscation Keys

The previous sections show that common injection attacks on the obfuscated dataset are not effective. However, attackers may aim to expose the session and, ultimately, secret keys instead. Once the key is revealed, an attacker can transform between obfuscated and normal values to inject accurate data values. We are aware of two fundamental approaches (see Figure 6) to obtain session keys with we detail here. The first method does not rely on any information about the data values; it is agnostic to the data obfuscated with the session key. The second method, which converges much faster, uses known information about obfuscated values. For example, in an ICS context the range of valid values may be known which limits the search space to find the session key.

Session Key Extraction Without Data Insight. Assume that in session d, an attacker has knowledge of the set of all potential real water values W. He also obtains a sequence of obfuscated values $V'_{di}, V'_{di+1}, ..., V'_{dn}$. The first method to expose the session key is to compute a set $\mathrm{KP}^{\mathrm{dj}}$ for each $V'_{dj}, j = i, i+1, .., n$, in which $\mathrm{KP}^{\mathrm{dj}} = \{\mathrm{KP}_k{}^{\mathrm{dj}} | \mathrm{KP}_k{}^{\mathrm{dj}} = V'_{dj} \oplus W_k \ , \forall k \in W\}$. As W contains all possible values, $\exists W_m \in W : W_m = V_{dj}$ as a result $\mathrm{KP}_m{}^{\mathrm{dj}} = W_m \oplus V'_{dj} = V_{dj} \oplus V_{dj} \oplus K_d = K_d$. The attacker knows that K_d is contained in every $\mathrm{KP}^{\mathrm{dj}}$ set. An attacker can derive K_d from the intersection of all these sets (see Figure 6a). The number of



W: all potential values of water level
V'di
KPdi
V'dn
KPdn
KPdi+1
V'di+1
Kd will be found in the intersection of the KPd sets
W: all potential values of water level
V'di
KPdi
Kd
V'di+1
VPdi+1
Legal set of VPdi+1
Vdi+1
Legal set of VPdj
V'dj
Kd will be found in reduced set
Continue to reduce
W reduced

Fig. 6: Examples of methods to expose K_d

candidate session keys in the intersection reduces with the number of collected obfuscated values. The intersection will contain a single element (the session key) when enough obfuscated values are observed by the attacker.

Session Key Extraction With Data Insight The second method relies on knowledge of the physical process behind the data. For example, it might be known that a value will be in a particular range (e.g., the filling level of a tank). More detailed information, such as process behaviour over time, might also be available. This knowledge can be used to further reduce the search space for the session key leading to faster exposure of the obfuscation key. The search space of potential keys is reduced recursively (see Figure 6b), using the following steps.

Step 1. Using the first obfuscated value V'_{di}, compute $\mathrm{KP}^{\mathrm{di}}$. Now the attacker has $\mathrm{KP} = \mathrm{KP}^{\mathrm{di}}$, which contains all potential K_d.

Step 2. Compute $V^{\mathrm{di+1}}$ from KP and V'_{di+1}, in which $V^{\mathrm{di+1}} = \{{V_k}^{\mathrm{di+1}} | {V_k}^{\mathrm{di+1}} = V'_{di+1} \oplus \mathrm{KP}_k, \forall k \in \mathrm{KP}\ \}$. KP contains all the potential K_d and $\exists m \in \mathrm{KP} : \mathrm{KP}_m = K_d$. We have ${V_m}^{\mathrm{di+1}} = V'_{di+1} \oplus \mathrm{KP}_m = V_{di+1} \oplus K_d \oplus K_d = V_{di+1}$. So attackers know that $V^{\mathrm{di+1}}$ contains V_{di+1}.

Step 3. V_{di+1} represents the real value. Now constraints using process knowledge (e.g., legitiate value ranges) can be applied. The attacker can use these constrains to eliminating all the unsatisfied values in the $V^{\mathrm{di+1}}$ set. The size of KP will also be reduced by tracing back from the reduced $V^{\mathrm{di+1}}$.

Step 4. Step 2 is repeated with the reduced KP and the obfuscated V'_{di+2}.

Discussion. Once the session key is exposed, an attacker can perform a dictionary attack to retrieve the secret key. To do so, T_d must also be exposed. The attacker can XOR all the dictionary values with T_d and hash these. The hashed results are then compared with K_d and a match will reveal the secret key.

Thus, attacks with the aim to expose session keys are reasonable, but note that such attacks cannot bypass the system longer than one session duration. This is because in the session interchange time, the attackers do not have time for collecting enough data to expose the next session key, while they still need to inject the falsified values. If they do not inject anything, the real states of the physical system will be revealed to the system operators, who will spot the abnormal changes and detect the attacks. If they inject without knowing the session key, the chance of success is very low. Therefore, the session interchange time is likely the time that the attackers will be detected. Besides, exposed session keys should not lead to an exposed secret key if it is complex enough. Secret keys may also be renewed from time to time.

5.5 System Configuration Options

We have shown that the shorter the T_s, the smaller the attack surface and the impact of attacks on patterned data or session keys. However, such reductions commonly come with cost. We next consider the trade-off between security and usability to ensure that the security does not compromise operation.

Cost Consideration. The overheads created by LASARUS, C, are the combination of the obfuscating cost, C_o, and the key refreshing cost, C_k. We will consider such overhead in a time T. The period to log the physical value is T_l; the cost for running each XOR operator is x; the period to refresh the session key is T_s; the cost for generating a session key is y. The system will process approximately T/T_l obfuscating operating, so we will have:

$$C_o = \frac{T}{T_l} \cdot x$$

The key will be refreshed T/T_s over time T, so we have:

$$C_k = \frac{T}{T_s} \cdot y$$

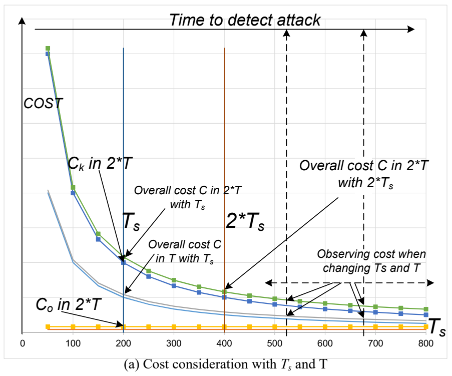
Overall, the total overhead will be:

$$C = C_o + C_k = \frac{T}{T_l} \cdot x + \frac{T}{T_s} \cdot y \tag{1}$$

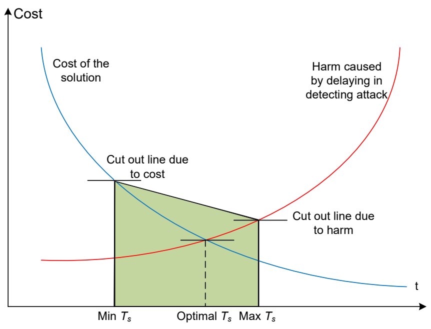
Note that costs x and y are computable and constant for a specific PLC. Besides, T_l is set according to specific ICS applications so we will not consider varying T_l for security purposes. As a result, from Equation (1), only T and T_s will have an effect on C.

We illustrate the relations between the factors in Figure 7a, which shows that the shorter the T_s, the higher the overhead. If we know the specific values of x, y, T and T_l, and given the cut out point of cost, which means the maximum cost we can afford, we can derive the value of T_s. Such values are the intersections between the cost lines and the constant vertical lines, which represent T_s. The overall time T to run the system also affects the cost. The longer it runs, the more overhead it will create. That suggests the operator to stop LASARUS when the resource is limited, or to run it only when there is a risk.

System Configuration Consideration The main factor that can be configured to control the trade-off is T_s. We illustrate the way that an operator can configure T_s in Figure 7b. The red line indicates the harm that the attacker can create over time while the blue line shows the overhead that LASARUS adds. Assume that the operators have a maximum of cost that they can afford, represented by the cost cut out line. On the other hand, they need to protect their system from damage by the undetected attacker, represented by the harm cut out line. In order to satisfy these cost-harm trade-offs, T_s should be larger than $T_{sMinCost}$ to save cost, and smaller than $T_{sMaxHarm}$ to prevent harm.



(a) Cost consideration with T_s and T



(b) Configuration of T_s considering cost and harm

Fig. 7: Cost consideration with T_s and T

Note that cost and harm are not necessary correlated in the system, so $T_{sMinCost}$ may not always be smaller than $T_{sMaxHarm}$. Figure 7b illustrates the case when $T_{sMinCost} < T_{sMaxHarm}$. In this case, when T_s falls into the [$T_{sMinCost}$, $T_{sMaxHarm}$] range, the conditions of cost and harm are always satisfied. As a result, operator can always configure T_s to have a value between these two points. The intersection point of the cost (blue line) and harm (red line) is the optimal value of T_s to balance the cost and security need.

There may be cases where $T_{sMinCost} > T_{sMaxHarm}$, whereby any implementation of T_s will exceed either cost or harm. When that happens, the operator can consider other configurations such as lowering T to save cost, or using stronger detection mechanisms, to detect the attackers faster and lower the harm.

6 CONCLUSION

We have designed and experimented with LASARUS, a system to continuously obfuscate data stored or accessed from a PLC in order to reduce the attack surface of legacy ICS. We show that LASARUS represents a significant challenge for attackers, decreasing the passing rate of attacks by up to 100 times with even simple detection systems. We also show that our obfuscated solution hides the patterns in the original data effectively, which stops the attackers from learning and predicting the next values successfully. We have also analysed the trade-off between security and cost by adjusting the length of the session, which adds more flexibility for the system operator. In future work, we plan to implement and evaluate LASARUS on a real-world ICS testbed available to us [20].

Acknowledgements. Supported by EPSRC/Chist-Era grant: EP/N021657/1.

References

1. Abdulmohsen Almalawi, Adil Fahad, Zahir Tari, Abdullah Alamri, Rayed Al-Ghamdi, and Albert Y Zomaya. An efficient data-driven clustering technique to detect attacks in scada systems. *IEEE Transactions on Information Forensics and Security*, 11(5):893–906, 2016.
2. Rob Antrobus, Sylvain Frey, Benjamin Green, and Awais Rashid. Simaticscan: Towards a specialised vulnerability scanner for industrial control systems. In *Proceedings of the 4th International Symposium on ICS & SCADA Cyber Security Research (ICS-CSR 2016)*, ICS-CSR 2016, 2016.
3. William Jardine, Sylvain Frey, Benjamin Green, and Awais Rashid. SENAMI: selective non-invasive active monitoring for ICS intrusion detection. In *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy, CPS-SPC@CCS 2016, Vienna, Austria, October 28, 2016*, pages 23–34, 2016.
4. E.J.M. Colbert and A. Kott. *Cyber-security of SCADA and Other Industrial Control Systems*. Advances in Information Security. Springer International Publishing, 2016.
5. Dina Hadziosmanovic, Robin Sommer, Emmanuele Zambon, and Pieter H. Hartel. Through the eye of the PLC: semantic security monitoring for industrial processes. In *Proceedings of the 30th Annual Computer Security Applications Conference, ACSAC 2014, New Orleans, LA, USA, December 8-12, 2014*, pages 126–135, 2014.
6. Miguel Sainz, Joaquim Armengol, and Josep Vehi. Fault detection and isolation of the three-tank system using the modal interval analysis. *Journal of process control*, 12(2):325–338, 2002.
7. David Evans, Anh Nguyen-Tuong, and John Knight. Effectiveness of moving target defenses. In *Moving Target Defense*, pages 29–48. Springer, 2011.
8. Infracritical. Project SHINE findings report. http://www.slideshare.net/BobRadvanovsky/project-shine-findings-report-dated-1oct2014, 2014. Last accessed 12 April 2016.
9. Keith Stouffer, Joe Falco, and Karen Scarfone. Guide to industrial control systems (ICS) security. *NIST special publication*, 800(82):16–16, 2011.
10. Thomas Morris and Wei Gao. Industrial control system cyber attacks. In *Proceedings of the 1st International Symposium on ICS & SCADA Cyber Security Research 2013*, pages 22–29. BCS.
11. Peter Maynard, Kieran McLaughlin, and Berthold Haberler. Towards understanding man-in-the-middle attacks on iec 60870-5-104 scada networks. In *Proceedings of the 2nd International Symposium on ICS & SCADA Cyber Security Research 2014*, pages 30–42. BCS, 2014.
12. Y Yang, HT Jiang, K McLaughlin, L Gao, YB Yuan, W Huang, and S Sezer. Cybersecurity test-bed for iec 61850 based smart substations. In *Power & Energy Society General Meeting, 2015 IEEE*, pages 1–5. IEEE, 2015.
13. R. E. Mahan, J. D. Fluckiger, S. L. Clements, C. Tews, J. R. Burnette, C. A. Goranson, and H. Kirkham. Secure data transfer guidance for industrial control and SCADA systems. Pacific Northwest National Lab (PNNL) Report, http://www.pnnl.gov/main/publications/external/technical_reports/PNNL-20776.pdf, 2011. Last accessed 4 January 2016.
14. Cordell Davidson and Todd Andel. Feasibility of applying moving target defensive techniques in a scada system. In *11th International Conference on Cyber Warfare and Security: ICCWS2016*, page 363. Academic Conferences and publishing limited.

15. Stephen McLaughlin and Patrick McDaniel. SABOT: specification-based payload generation for programmable logic controllers. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 439–449. ACM.
16. Ronald L Krutz. *Securing SCADA systems.* John Wiley & Sons, 2005.
17. Wei Gao, Thomas Morris, Bradley Reaves, and Drew Richey. On SCADA control system command and response injection and intrusion detection. In *eCrime Researchers Summit (eCrime), 2010*, pages 1–9. IEEE.
18. Thomas H Morris, Zach Thornton, and Ian Turnipseed. Industrial control system simulation and data logging for intrusion detection system research.
19. Abdalhossein Rezai, Parviz Keshavarzi, and Zahra Moravej. Key management issue in scada networks: A review. *Engineering Science and Technology, an International Journal*, 2016.
20. Benjamin Green, David Hutchison, Sylvain Andre Francis Frey, and Awais Rashid. Testbed diversity as a fundamental principle for effective ics security research. *SERECIN*, 2016.