SimProcess: High Fidelity Simulation of Noisy ICS Physical Processes

Denis Donadel
denis.donadel@univr.it
University of Verona
Verona, Italy

Gabriele Crestanello
gabriele.crestanello@studenti.unipd.it
University of Padua
Padova, Italy

Giulio Morandini
giulio.morandini@studenti.unipd.it
University of Padua
Padova, Italy

Daniele Antonioli
daniele.antonioli@eurecom.fr
EURECOM
Sophia Antipolis, France

Mauro Conti
mauro.conti@unipd.it
University of Padua
Padova, Italy

Massimo Merro
massimo.merro@univr.it
University of Verona
Verona, Italy

Abstract

Industrial Control Systems (ICS) manage critical infrastructures like power grids and water treatment plants. Cyberattacks on ICSs can disrupt operations, causing severe economic, environmental, and safety issues. For example, undetected pollution in a water plant can put the lives of thousands at stake. ICS researchers have increasingly turned to honeypots—decoy systems designed to attract attackers, study their behaviors, and eventually improve defensive mechanisms. However, existing ICS honeypots struggle to replicate the ICS physical process, making them susceptible to detection. Accurately simulating the noise in ICS physical processes is challenging because different factors produce it, including sensor imperfections and external interferences.

In this paper, we propose *SimProcess*, a novel framework to rank the fidelity of ICS simulations by evaluating how closely they resemble real-world and noisy physical processes. It measures the simulation distance from a target system by estimating the noise distribution with machine learning models like Random Forest. Unlike existing solutions that require detailed mathematical models or are limited to simple systems, *SimProcess* operates with only a timeseries of measurements from the real system, making it applicable to a broader range of complex dynamic systems. We demonstrate the framework's effectiveness through a case study using real-world power grid data from the EPIC testbed. We compare the performance of various simulation methods, including static and generative noise techniques. Our model correctly classifies real samples with a recall of up to 1.0. It also identifies Gaussian and Gaussian Mixture as the best distribution to simulate our power systems, together with a generative solution provided by an autoencoder, thereby helping developers to improve honeypot fidelity. Additionally, we make our code, dataset, and experimental results publicly available to foster research and collaboration.

CPSS '25, Hanoi, Vietnam

ACM ISBN 979-8-4007-1413-9/2025/08
https://doi.org/10.1145/3709017.3737711

CCS Concepts

• **Computer systems organization → Embedded and cyber-physical systems**.

Keywords

Industrial Control System, Honeypot, Simulation, Physical Process, Noise, Power Grid, EPIC.

ACM Reference Format:
Denis Donadel, Gabriele Crestanello, Giulio Morandini, Daniele Antonioli, Mauro Conti, and Massimo Merro. 2025. SimProcess: High Fidelity Simulation of Noisy ICS Physical Processes . In *11th ACM Cyber-Physical System Security Workshop (CPSS '25), August 25–29, 2025, Hanoi, Vietnam.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3709017.3737711

1 Introduction

Industrial Control Systems (ICSs) are crucial for managing and automating critical infrastructure, including power grids, water treatment facilities, and manufacturing plants. These systems integrate hardware and software components to monitor and control physical processes, ensuring efficiency and safety. ICS interact directly with the physical world, making their security essential to prevent operational disruptions that could lead to economic losses, environmental damage, or safety hazards [28].

The importance of ICS security has been highlighted by real-world attacks. A notable example is Stuxnet [20], a sophisticated malware that targeted nuclear centrifuges by subtly altering their behavior while remaining undetected by monitoring systems. Other attacks followed, demonstrating how adversaries could exploit ICSs vulnerabilities to manipulate physical processes, causing physical damage [51]. This made the research community develop security solutions, such as Machine Learning (ML) based anomaly-detection systems [24, 40], to identify a wide range of attacks.

Recently, ICS researchers and industries have started to deploy decoy systems to attract attackers and learn about their behavior, the so-called ICS honeypots. Low-interaction honeypots are designed to simulate only a limited set of services or behaviors, typically fooling attackers with minimal interaction to gather data on their tactics [25, 43]. High-interaction ICS honeypots emulate a full range of services and behaviors, allowing attackers to engage deeply with the system.

High interaction honepots can provide comprehensive insights about attacking methods and motivations, but at the cost of greater

resource investment and potential risk [30, 31, 61]. Building a high-interaction honeypot is challenging because it must emulate, among others, industrial control devices, industrial protocols, sensors, actuators, and the underlying physical process. For example, an attacker could spot a honeypot by looking at its software details [53].

Motivation. A honeypot developer needs to understand the fidelity that their honeypot can achieve. Fidelity can be increased in every component of the decoy system. For instance, industrial device functionalities can be emulated through high-level computer software, or their firmware can be fully emulated to expose the correct architectures and all the bugs specific to it [27]. Another important step toward a realistic honeypot is related to the physical process that needs to replicate a real scenario.

A *realistic physical process simulation* is essential to prevent attackers from understanding that they are dealing with an ICS honeypot. In fact, attackers, aware of the presence of honeypots, have started to develop solutions to identify simulated environments. Various techniques have been presented over the years [52], but only a few investigated ICS specific techniques looking at changes in the device registers [64] or by employing fuzzy testing [53]. To our knowledge, the fidelity of industrial physical process simulations has never been investigated thoughtfully.

A central issue with ICS physical process simulation is reproducing a *noise* consistent with the actual process's characteristics. Such noise may be intrinsic to the physical process because it is generated by its components, including industrial sensors [19, 35], or it may be due to external factors, like weather conditions, electromagnetic interference, or people working on the industrial plant [2].

However, to our knowledge, no framework exists to assess if a simulated physical process includes realistic noise components. Such a framework would be essential for defenders to build effective ICS honeypots, even without disclosing the details of the protected physical process. In this scenario, the attacker would hardly detect being in a honeypot and infer information about the real physical process.

Contribution. We present ***SimProcess***, a framework for benchmarking ICS physical process simulations and identifying the one that resembles with the highest fidelity a real (noisy) physical process. Unlike prior works, our framework only employs a collection of measurements from the real system and does not require the mathematical model (i.e., differential equations) that govern the underlying physical process, allowing us to deal with complex systems, including nonlinear time-invariant (NLTI) ones, and to reproduce variations of the real system. *SimProcess* extracts selected features from the original signal that are related to the noise, in addition to features extracted from a noise estimation.

We implemented and evaluated *SimProcess* framework on EPIC [1], a power grid testbed. Our empirical results demonstrate its capabilities in identifying the best simulation. In particular, our classification models exhibit a recall of up to 1.0 in determining the actual process from various simulations. *SimProcess* identified Gaussian, Gaussian Mixture Model (GMM), and the autoencoder solutions as the best ways to add noise to resemble the real system and increase the fidelity of the power grid simulation.

We summarize our contributions as follows:

- We introduce *SimProcess*, the first framework designed to identify simulations that most accurately replicate an ICS physical process by focusing on the characterizing noise.
- We implement and evaluate *SimProcess* on real-world power grid data coming from EPIC, a state-of-the-art power grid testbed, and derive simulations of the physical process employing both static additions of noises and generative solutions.
- We show that our models can correctly classify the real physical process with a recall up to 1.0 in a *binary classification* scenario. Moreover, we evaluate the capabilities of *SimProcess* of classifying different noises and its ability to preserve the ranking even when dynamic changes are added to the underlying process, making *SimProcess* applicable in high-interaction simulations.
- We made our code, dataset, and experimental results publicly available: https://github.com/donadelden/SimProcess

Organization. Section 2 briefly introduces important background knowledge, while the system and threat model are presented in Section 3. Section 4 introduces the *SimProcess* framework, while Section 5 presents our case study, the fine-tuning experiments, and final results. Related works are discussed in Section 6 and Section 7 concludes this work.

2 Background

Industrial Control Systems. An Industrial Control System (ICS) is a system in which industrial operations are supervised, coordinated, controlled, and unified by a computing and communication core [44]. These systems constitute a vital element of *critical infrastructures* that deliver essential services, including water supply, electricity generation and distribution, and nuclear power generation. They combine industrial hardware and software.

Industrial processes can be modeled linearly or nonlinearly. With a linear time-invariant (LTI) process, there is a linear relation between an input and the output that does not depend on time. More complex models involve nonlinear terms, time, and (partial) differential equations.

ICSs are highly interconnected. An ICS network is composed of an enterprise zone containing general purpose Information Technology (IT) systems, connected through a demilitarized zone to a Operational Technology (OT) control zone. The OT network operates with the physical process through intelligent devices such as Programmable Logic Controllers (PLCs) connected to I/O devices such as sensors and actuators. Operators can supervise and act on the physical process through control systems such as Human Machine Interfaces (HMIs) [16].

ICS are subject to impactful and large-scale attacks. A notable attack target example is the power grid, where ICS controls electricity transmission. The high number of cyberattacks in Ukraine since 2015 [45, 49], highlighted the vulnerability of these systems and introduced the need for specific security strategies to protect power systems from cyberattacks [12].

ICS Testbeds and Simulations. ICS testbeds are experimental environments designed to test and validate technologies, security systems, and protocols in realistic but controlled conditions. They

combine physical simulations with real network traffic to assess the security and reliability of critical infrastructures. They could be physical, hybrid, or virtualized [16]. While physical testbeds provide high fidelity to the researchers, they are more complex and expensive to develop and maintain. Conversely, a virtualized testbed can also be employed as a decoy system to study attacker's behaviors [30, 31, 46].

There are several tools to create virtual ICS testbed and simulate ICS physical processes. MiniCPS [4] or Factory I/O[1] are designed for generic Cyber-Physical System (CPS) simulation and emulation. Others investigate specific sectors. NEFICS [45], ICSNet [46], and Mashima et al. [37] provide simple power grid simulations but are focused on the emulation of the device more than on the physical process, which usually result in static measurements. Pandapower [56] is a Python-based open-source library designed to enable electric power systems modeling, simulation, and analysis. Key components of the power grid, like buses, power lines, transformers, generators, and loads, are represented as data tables, enabling easy manipulation and customization of network parameters. It provides an element-based modeling approach that allows us to define an electrical network by using nameplate parameters. Mosaik [47] is another popular open-source simulation framework designed for modular and distributed simulations of electrical systems. It enables the coupling and orchestration of diverse simulators, allowing for the coordinated execution and data exchange. Mosaik provides a modular architecture with a central scheduler synchronizing simulators while supporting various APIs and adapters, including Python and Java.

Noise in ICSs. In a real system, physical process measurements are subjected to noise, an unwanted random variation generated by the system implementation itself, or from interferences with external factors. Different forms of noise impact both the physical process [19] and the sensors employed to collect the measurements [2, 3]. Examples of noise sources include thermal fluctuations, electromagnetic interference, mechanical vibrations, and intrinsic limitations of electronic components or sensors, and are also influenced by the generation source [19].

Measurements of electric voltage could be usually characterized as white noise [59], even though some research works proposed other noise models [18]. Different solutions have been proposed to filter noise [33] or reduce its impact in the real system [29]. Such forms of errors and variations should be integrated into ICS simulations to reduce the probability of being detected as a decoy system.

3 System and Attacker Model

System Model. We assume a scenario where a developer (defender) builds a high-interaction honeypot for an ICS. The developer is challenged to create a realistic physical process simulation for the honeypot. The developer does not want to employ a complex model based on differential equations for the ICS physical process as it is unreliable. This is common in complex dynamic systems such as power grids, where many events can impact the process and cannot be reliably described by a system of equations [34].

[1]Factory I/O - Next-Gen PLC Training (https://factoryio.com/)

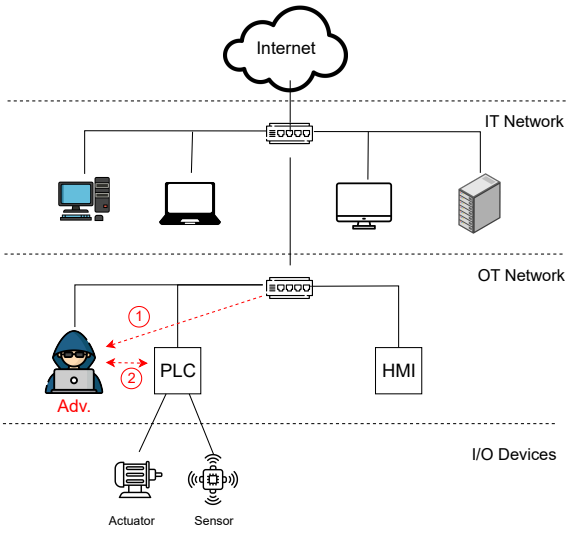


Figure 1: Threat model. ① represents passive collection of data, while ② requires the attacker to request values to a PLC.

The developer can access a timeseries of physical process measurements from the ICS PLC or HMI. Such timeseries could be obtained from one-time access to the plant or by employing a public measurement dataset. However, the developer is not required to access the ICS plant after the data collection, which can happen at different times.

The developer creates a simulation of the real system in its entirety or replicates a small part of it. Small architectural differences are acceptable since the final target does not aim at having a replica of the system but a simulation to be employed in a high-interaction honeypot. Moreover, an attacker without access to the real system cannot employ this kind of imperfections to detect the simulations. The developer may start from digital twins [38] or simulators in the literature [4, 21, 30, 31, 46] and adapt them to resemble the target system.

To prevent identification, the developer follows honeypot anti-fingerprinting best practices, such as the ones proposed by Tay et al. [55]. They ensure that the honeypot network resembles the real one. For example, the attacker can identify industrial devices with valid MAC addresses and see network packets utilizing industrial protocols.

However, the developer is conscious that an attacker may try to detect the simulation using advanced techniques that investigate the physical process's consistency. Therefore, s/he takes additional design steps to enhance the fidelity of the physical process by adding an appropriate form of noise to the simulated plant.

Attacker Model. We assume a *remote* attacker with network access to the OT network of an ICS honeypot. As shown in Figure 1, the attacker can *passively* collect data or *actively* interact with the honeypot devices, including its virtual PLC and HMI and collect sensor reading from the virtual physical process.

The attacker, who cannot access data from the real system, gets sensor measurements from the defender honeypot as timeseries that contain various noises, including the inherent system noise,

the sensor measurement noise, and external noise. The attacker uses such timeseries as a data source to fingerprint the honeypot and categorizes it as a decoy or a real system.

This is a realistic threat model since many ICS devices are exposed on the Internet and exhibit none to low-security measures [7, 39]. Moreover, industrial honeypots generally employ few to no defenses in the external perimeter to get more interactions from attackers [21].

4 *SimProcess*

We present the design (Section 4.1) and the implementation (Section 4.2) of *SimProcess*, a novel framework that allows identifying the simulations with the best fidelity to replicate a real-world physical process. Our system do not require any differential equation, but only a timeseries of measurements from the real process.

4.1 Design

As shown in Figure 2, *SimProcess* has a *seven-stage pipeline*. The raw signals are collected (Section 4.1.1) and noise estimation is computed from each timeseries (Section 4.1.2). Then, a windowing approach is taken to prepare the data for classification (Section 4.1.3) with an integrated filtering mechanism to maintain only relevant windows (Section 4.1.4). Then, features are extracted from each window (Section 4.1.5), which are used in the classification phase (Section 4.1.6) which is used to generate the final scores (Section 4.1.7). Next, we describe each pipeline stage.

4.1.1 Signal. The framework collects timeseries from the real physical process and the simulated one the developer wants to benchmark. The timeseries contains measurements from real and virtual sensors connected to real or virtual PLCs or HMIs. Data are collected using a sampling rate of 1 Hz (i.e., one sample per second).

A timeseries is defined as: $\hat{x}(t) = x(t) + n(t)$, for a certain period of time $t \in (t_0, t_n)$. $x(t)$ represents the values that, in a theoretical scenario, will be obtained by sampling the process at a specific time t, and it follows the equations that regulate the system. However, real physical processes are subject to noises, modeled by $n(t)$, that sum with $x(t)$. In our scenario, only $\hat{x}(t)$ is available because the honeypot developer neither knows the equations that regulate the system $x(t)$, nor the noise $n(t)$, and it is the only signal used as input for *SimProcess*.

4.1.2 Noise Estimation. *SimProcess* is based on a profilation of the noise, but due to the absence of ground truth theoretical process $x(t)$, it relies on a noise estimation $\tilde{n}(t)$. Since the developer has some high-level knowledge about the target physical process s/he is replicating (e.g., s/he knows whether the process is a power system, a water treatment, or a manufacturing process), s/he can choose the most suited filter $f(\cdot)$ to be applied on the collected noisy signal $\hat{x}(t)$ to estimate the signal without noise. Then, *SimProcess* estimate the real noise as the difference between the noisy and the filtered signal, as follows:

$$\tilde{n}(t) = \hat{x}(t) - f(\hat{x}(t)). \tag{1}$$

4.1.3 Windows Extraction. At this point, each timeseries $\hat{x}(t)$ is paired with its noisy estimation $\tilde{n}(t)$. Then, the preprocessing logic applies a sliding window to split the original signal $\hat{x}(t)$ into chunks of size N that overlap by 80% with each other. Dividing signals into windows allows for managing signals with different lengths and prevents the dynamics of the system from being preponderant in the classification.

4.1.4 Windows Filtering. Next, *SimProcess* prunes the generated windows by removing the ones containing peaks and big variations in the signal. This allows for more precise extraction of characterizing features, limiting external influences from automated (e.g., a valve opening because a tank is full) or manual (e.g., a user opening an electric switch to turn on a machine) actions on the physical process. This is an essential step to ensure that only real noise is extracted from the original signal and that events happening to the system are not considered.

The framework performs the pruning by considering the windows W containing values such that:

$$W = \{w_1, \ldots, w_N\}, \quad s.t.\ w_i \in [\mu(1-\epsilon), \mu(1+\epsilon)] \quad \forall i \in [1, n] \tag{2}$$

where μ is the mean value of the window and $\epsilon \in (0, 1)$ represents a tradeoff between the natural variation allowed and the removal of potential outliers. High ϵ alleviates the filter effect and reduces the amount of deleted data. While a low ϵ removes many samples containing variations with respect to the mean. The correct value of ϵ should be chosen based on the variability of the physical process and the length of the data collection.

For every window obtained from the signal $\hat{x}(t)$ which has not been pruned by the filtering in Equation 2, *SimProcess* also collects the corresponding window from the paired noise estimation $\tilde{n}(t)$. At the end of this preprocessing, the system ends up with a set of windows containing both the raw signal and its corresponding noise estimation, which are the inputs for the feature extraction phase.

4.1.5 Feature Extraction. To feed the classification stage, *SimProcess* extract features from each window that are relevant to the variations of the signal due to the presence of the noise, without including features that profile the underlying process. This allows the freedom in choosing features extracted from windows belonging to the noise estimation $\tilde{n}(t)$ since the decoupling from the underlying process signal has been done with Equation 1. Instead, when dealing with windows generated from the raw timeseries $\hat{x}(t)$, *SimProcess* takes extra care to avoid generating biases related to the physical process in our system.

For instance, the system avoids features such as the mean, which is strictly related to the underlying physical process. Instead, it considers features such as the approximate entropy, which measures the complexity or regularity of a time series, and Lempel-Ziv Complexity, which characterizes variations of the system and, in a short period without significant changes in our window, is a good estimation of the noise. Moreover, *SimProcess* considers the standard deviation ratio std_mean_ratio and the variance ratio var_mean_ratio, defined as:

$$std\_mean\_ratio = (\sigma/\mu),$$

$$var\_mean\_ratio = (\sigma^2/\mu),$$

where σ and μ represent the window standard deviation and mean, respectively. These features allow the signal to be decoupled from superimposed noise generated by physical components as much as possible.

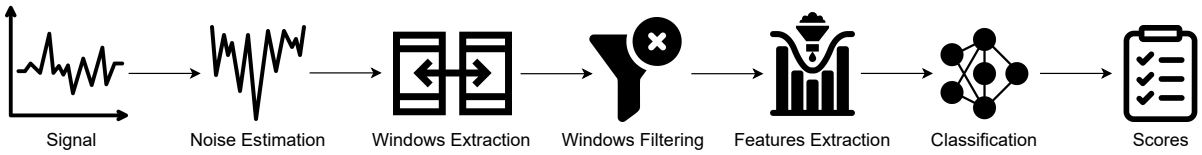


Figure 2: *SimProcess* seven-stage pipeline.

4.1.6 Classification. The features extracted in this pipeline are then fed to a ML model trained to differentiate between real physical processes and simulations. The task can be formulated as a binary classification problem, where data are labeled as *real* or *simulated* based on their origin. The models take into account a feature set related to a single window and output a prediction. Various ML models are suitable for the purpose and may be chosen based on recall scores but also on the final model complexity [8].

4.1.7 Scores. Having a good-performing model is essential to ensure a solid benchmarking process. However, in this case, the framework aims at maximizing the true positives (i.e., real samples correctly classified as real) possibly at the cost of misclassifying some fake system. When this happens it is a suggestion that such a sample is similar enough to the real process that it can be passed as a real system. To enhance this effect, our system is designed to output *the probability for each sample to be classified as real* instead of thresholding the output model probability and directly predicting the class. This allows for the generation of a *fidelity score* for every window provided to the model. The final score for each data source (and, therefore, on each noise) is computed by averaging the probabilities on every window associated with it.

4.2 Implementation

We implemented *SimProcess* using Python. Datasets are saved as CSV files to be easily imported into the code. Noise is estimated by employing a filter that can be easily changed to best suit each scenario. Then, the windowing approach is applied to extract windows of a predefined size. Every window is checked on a simple implementation of Equation 2 and windows not satisfying the equation are dropped, taking care of keeping the sync between different measurements.

To extract relevant features from the windows, we employed the `tsfresh` Python package [13]. The package automatically computes statistical and mathematical characteristics from time series data and selects the most relevant ones for regression or classification tasks. The extracted features include, among others, statistical measures (e.g., mean, variance, skewness, kurtosis), frequency-domain features (e.g., Fourier coefficients, spectral entropy), and time-series properties (e.g., autocorrelation, trend strength, peaks, crossings).

We let `tsfresh` identify the most suitable features for the noise estimations $\tilde{n}(t)$, selecting a set of features we employed for all subsequent extractions. Instead, we manually limit the possible features extracted from the noisy signal $\hat{x}(t)$. We carefully choose 9 features (described in Appendix A.1) that profile the signal variations due to the noise without taking into account the underlying signal, in addition to the two new features introduced in Section 4.1.5.

Finally, we employed Scikit learn [42] to perform the classification step. It is a well-known Python implementation of different ML models and support functions. Moreover, it allows for the extraction of class probabilities instead of classification outputs from the models. We employed this strategy to compute the final score of our benchmarking systems.

5 *SimProcess* at work: the EPIC case study

We evaluated *SimProcess* on EPIC [1], a state-of-the-art power grid testbed. We chose a power grid because it is widely employed in security research in ICS [12]. Moreover, we also needed ground truth data from a real plant; this requirement reduced the possible scenario to consider since datasets coming from real ICS are still scarce in the literature [16]. We note that our framework can also be used in other ICS domains.

For our case study, we use the methodology and seven-stage pipeline introduced in Figure 2. Section 5.1 describes the dataset we use to test *SimProcess* (first pipeline stage). Then, Section 5.2 shows some insight into the fine-tuning of *SimProcess* to provide the best results on this scenario (second to sixth pipeline stages). Next, Section 5.3 tests the overall framework and discusses the results (last pipeline stage).

5.1 Dataset Generation

This section describes the EPIC ground-truth dataset (Section 5.1.1), how we replicate in different simulators a portion of the real testbed (Section 5.1.2), and how we augmented the fidelity by adding noise to our simulations (Section 5.1.3). A summary of the different time-series that compose our dataset is summarized in Table 1. The dataset is then used as input signals in our pipeline.

5.1.1 Ground-Truth Datasets. We needed a dataset containing physical process measurements from a real-world industrial plant or testbed. We found an adequate dataset from a paper on EPIC presented in 2019 [1]. The EPIC datasets include eight scenarios in which the system runs under different conditions, such as different numbers of active generators and loads. The dataset time series are recorded from sensors and actuators, capturing 30 minutes of operation per scenario. For each case, network traffic data is preserved in packet capture files, while system readings are stored in CSV files, offering a view of the physical process behavior.

EPIC is a sophisticated platform designed to emulate real-world power grid operations. Developed by iTrust, the testbed includes four interconnected stages, as summarized in Figure 3: Generation, Transmission, Microgrid, and Smart Home, providing a comprehensive environment for studying smart grid dynamics [50].

EPIC enables testing of various operational conditions, such as synchronization of generators, integration of renewable energy

Table 1: Summary of the employed datasets. Each noise has been applied to both simulators (Sim.), namely Pandapower and Mosaik.

Source	Real	Noise	Parameters
EPIC [1]	✔	✔	-
Plain Simulation	✘	✘	-
Sim. + uniform	✘	✔	σ_u=0.01
Sim. + gaussian1	✘	✔	σ_g=0.01
Sim. + gaussian2	✘	✔	σ_g=0.05
Sim. + poisson	✘	✔	σ_p=0.01, λ_p=1.5
Sim. + laplace	✘	✔	σ_l=0.01
Sim. + pink	✘	✔	σ_p=0.01
Sim. + GMM	✘	✔	σ_g=0.02
Sim. + gaussian + uniform	✘	✔	σ_g=0.01, σ_u=0.01
Sim. + laplace + uniform	✘	✔	σ_l=0.01, σ_u=0.01
Sim. + laplace + poisson	✘	✔	σ_l=0.01, σ_u=0.01, λ_p=1.5
Sim. + VRAE	✘	✔	$input\_weight$ = 0.99

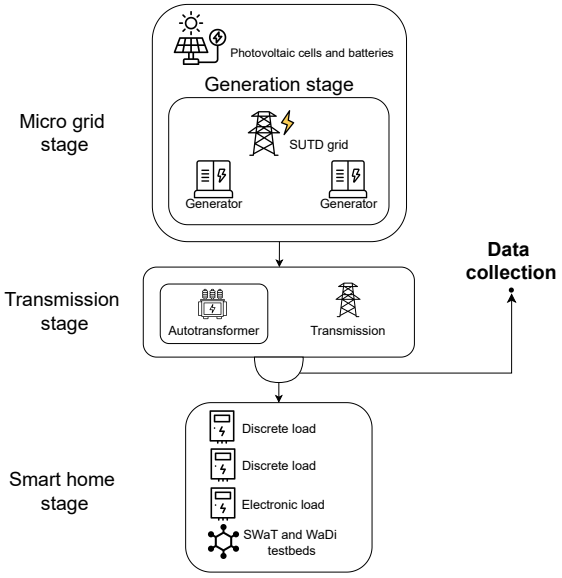


Figure 3: EPIC testbed visual description.

sources, and dynamic load management. It utilizes PLCs for control and supervision, while a SCADA system ensures centralized monitoring. This characterizes flexibility, making it a realistic system for analysis and comparison. Among the many case studies, EPIC has been used to study the impacts of power supply interruption and cyberattacks [26].

Since the EPIC dataset contains a small number of samples with respect to the ones we can generate with the simulators, we decided to increase, when needed, the number of samples by performing a data augmentation preprocessing on the final generated features by using SMOTE [10]. In addition to increasing the real sample size in the dataset, this step helps prevent overfitting and enhance the generalizability of our model. However, we employed these samples for training, keeping the testing phase clean from these artificial samples.

5.1.2 Simulations. We decided to simulate the underlying physical process of the EPIC testbed using two popular tools for power grid simulations, Pandapower [56] and Mosaik [47]. Both tools rely on rule-based and physics-based modeling rather than data-driven or statistical approaches. This ensures the generated data reflects realistic operational behavior and allows an attacker to interact with the system. We extracted information on its architecture (e.g., number of generators) from documentation describing the system [1, 54]. Since the available references do not mention the base values employed in EPIC, we extracted a rough estimation from the EPIC dataset. In particular, we set a base current of $20A$, a base voltage of $240V$, and a base frequency of $50Hz$. We do not need to precisely match EPIC values since *SimProcess* focuses on the noise applied on top of the physical process and not on the exact values of the process itself. Pandapower and Mosaik do not generate process noise, so the output $x*(t)$ is a theoretical simulation of $x(t)$.

The data generated from the simulation, to be as coherent as possible with EPIC dataset, contains direct measures of *Voltage* ($V1, V2, V3$), *Current* ($I1, I2, I3$), and *Frequency*. Moreover, we compute the *Reactive power* as $S = \sqrt{3}VI$ ($VI \in \{V1, V2, V3\}$), the *Real power* as $P = S\cos(\phi)$, and the *Apparent power* as $Q = S\sin(\phi)$, where $\phi = 120^\circ$ represents the phase of the EPIC testbed, while V and I represent respectively the average values over the three phases for voltage ($V1, V2, V3$) and for current ($I1, I2, I3$).

5.1.3 Noise Simulation. We introduced different approaches for adding various noises $n_i(t)$ to the underlying simulations, to generate a signal $\hat{x}^*(t) = x^*(t) + n_i(t)$ that reproduces as close as possible the genuine signals $\hat{x}(t)$. Then, we use *SimProcess* to rank the possible noises $n_i(t)$ and select the ones that make the simulated data look closer to the real $\hat{x}(t)$.

We select the noise candidates by looking at papers discussing noise in power systems. Characterizing noise from a power grid is complicated since many factors can interfere with the desired signal, depending on internal and external conditions, as described in Section 2.

We started by adding *normal (Gaussian) noise* to the simulated data, which is known to represent simple random deviations introduced by thermal noise [29]. There is no clear definition of the correct standard deviation value for such a noise distribution on electric power systems. As discussed in the literature [9, 23, 48, 57, 62], standard deviations range between 0.002 and 0.15 pu. We choose two values in the middle to model our zero-mean Gaussian noises (0.05 and 0.01).

However, Gaussian noise alone cannot model an electric power system [11, 58, 63]. Actuators, sensors, transmission lines, and external factors generate additional noise that does not follow a Gaussian distribution. Following research in the literature [6, 18, 36] and our intuition, we tested other noise distributions, such as Laplace and uniform noise distribution.

The Laplace noise represents impulse noise generated by electromagnetic interferences and may represent external noise sources [15]. Uniform noise may be present in power systems, due, for instance, to quantization noise [6]. We also considered other noises generated from Poisson and Pink distributions, which may represent the various noise sources of a real power system. Moreover, we employed a GMM which can be regarded as a linear combination

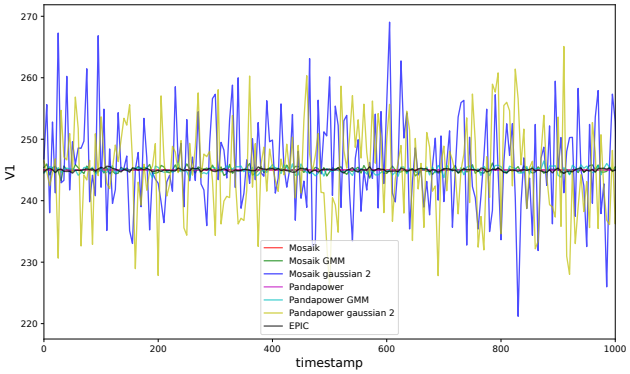


Figure 4: Example of real (EPIC) and simulated (Mosaik, Pandapower) power grid timeseries with some sample noises. Simulations without noise are almost flat and, for the most part, superimposed by others.

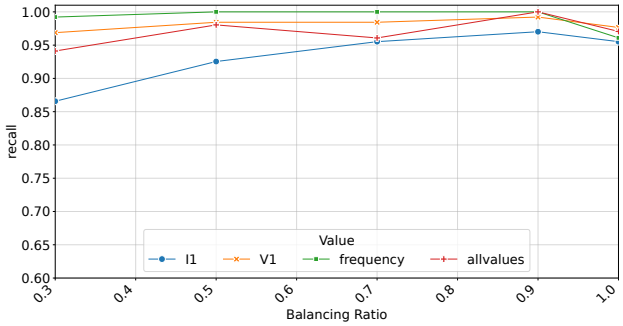


Figure 5: Recall while varying the balance ratio. A higher balancing ratio indicates a more balanced dataset. I1 is the first current phase, while V1 is the first voltage phase.

of k independent Gaussian models and, in theory, can fit any distribution [18]. We extract their parameters from an unprocessed EPIC trace and use it as an additional noise source.

The approaches considered up to now sample the noise errors from the same distribution throughout the process. While this may be the case for certain power grids, external interferences may change over time and based on external factors. A dynamic approach involves using generative methods. An autoencoder can be trained on real data and employed to make simulation data closer to the real process in real-time.

We test this strategy by employing a Variational Recurrent Auto Encoder (VRAE) thanks to its causality that allows the generating samples temporarily dependent on previous elements in the time series. We trained our VRAE employing the same architecture of Chung et al. [14] on the noise estimation generated as depicted in Section 4.1.2. During generation, the VRAE gets as input a clean signal (i.e., without any noise) coming from one of the two simulators.

The output is a new signal resembling the one provided as input, but with additional noise. Since the VRAE only manages noise, we set the input weight to a high value (0.99) to ensure the output is mainly controlled by the signal in input. Our VRAE has been trained for 11 epochs. We refer the reader to the original paper for more details on the model architecture [14].

Figure 4 shows an example of different noises applied to our simulation for the voltage V1, together with the real data from the EPIC dataset.

5.2 *SimProcess* Fine-Tuning

We fine-tune the parameters of *SimProcess* on the EPIC power grid employing the dataset we built as discussed in Section 5.1. We analyze the system parameters, such as the balancing ratio between real and fake data in the training set (Section 5.2.1), the window length (Section 5.2.2), the selection of the number of features to be used for classification (Section 5.2.3), and the ML model to be used (Section 5.2.4).

As a filter $f(\cdot)$ to estimate the noise, we employed a Kalman filter [33]. Based on our tests, we applied two different ϵ values to the window pruning (Equation 2): $\epsilon = 0.1$ for filtering individual values, $\epsilon = 0.3$ for filtering all values simultaneously. This adaptive choice of ϵ provides a trade-off between filtering capabilities and data retention.

A lower ϵ resulted in a more stringent filter, reducing the amount of available data, while a higher ϵ made the filter more permissive, potentially retaining outlier values. The higher ϵ value was chosen for the case where the filter is applied to all values simultaneously since, in this case, a window is retained only if all values within it satisfy the pruning criterion (Equation 2). Thus, a more permissive threshold was necessary to maintain sufficient data for further processing of values.

For the evaluation, we employed the recall metric, which is defined as $Recall = TP/(TP + FN)$, where True Positive (TP) counts real samples correctly identified as real, while False Negative (FN) identifies real samples classified as simulated. Recall measures the TP rate, indicating the rate of real samples classified as real. We perform the experiments on every measure in our dataset, in addition to collecting all the measures together (*allvalues*). For clartity, in most plots we show only results on some representative measurements.

5.2.1 Dataset Balancing. Because the EPIC dataset contains little real data compared to the large amount of data generated from the different simulations, we tried various levels of balancing the dataset. We employed 90% of the real data for training and 10% for testing. Since we can benefit from a huge dataset of simulations, we extract 30% of the data for training and 35% for testing, using stratification to ensure that every noise appears equally.

Then, we selected different balancing levels and trained a Random Forest (RF) classifier on sample settings to monitor the performances. When needed, we employed SMOTE [10] to perform data augmentation on the training data belonging to the real class.

We can see from Figure 5 that the recall increases with the balancing ratio, suggesting, as expected, that increasing the ratio of real samples in the training data allows for a better TPs rate. This is the desired goal of our framework since we want to measure the distance of simulated samples from the real ones that need to be classified correctly as much as possible.

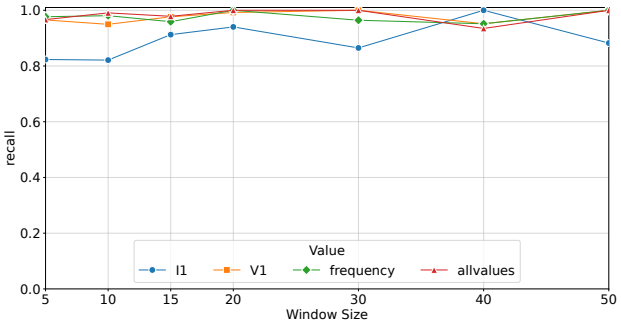


Figure 6: Recall when changing the window size.

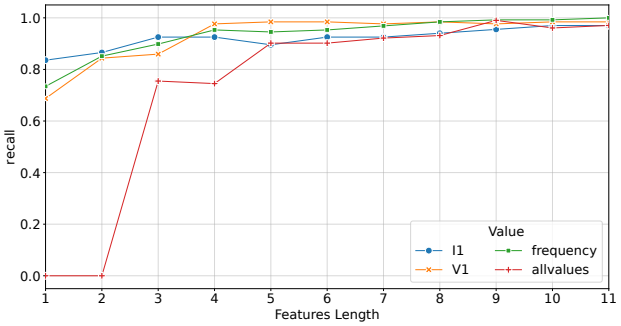


Figure 7: Recall when adding one feature at a time ordered by importance.

5.2.2 Window Length Analysis. As discussed in Section 4.1.3, the window length is the number of samples used to compute the features given to the models as input. Therefore, it represents the minimum duration of the data collection to get the first decision on the physical process under test. Moreover, different window sizes capture different behaviors in the data distributions and provide different samples for training and testing. Each window is classified alone for *SimProcess*, and then all the scores are averaged to obtain the final noise ranking.

We identify the window length by experimenting on the RF classifier with the best results obtained up to now. Results are shown in Figure 6. The recall does not change much while increasing the window size, except for I1, which shows fluctuating values. However, we can see a convergence of high recall for every value considered with a window size of 20, which has been chosen as the default value in our evaluation.

5.2.3 Feature Reduction. Based on the most features classification provided by `tsfresh` [13], we tested their importance by training the model adding one feature at a time, from the most important to the least important, up to 15 features. We tested it for V1, I1, frequency, and all the values together. Results are shown in Figure 7. The first few features influence the measures, while additional features only refine the classification. We see an increase in the recall score, which is particularly pronounced in the first five features. However, the first peak for all the tested features is reached with

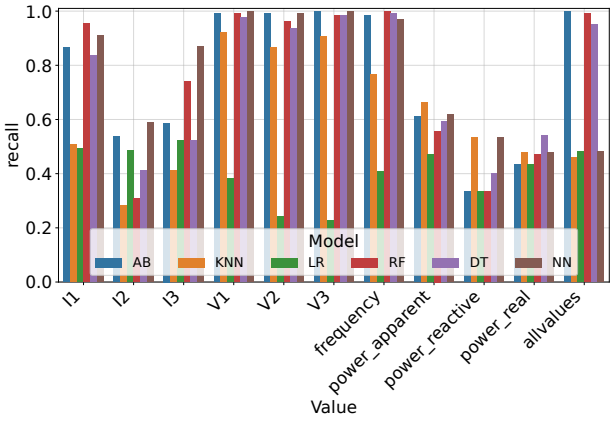


Figure 8: Recall employing different models.

eleven features, which represents the best number of features for our study.

5.2.4 Model Selection. Next in our case study, we merged the results of all the above experiments to identify the most suited model for our task. We formulate the task as a binary classification problem, where training data comes from real and simulated systems. The model is asked to identify between real and simulated data.

We compare different lightweight ML models such as Decision Tree (DT), Logistic Regression (LR), k-Nearest Neighbors (kNN), RF and AdaBoost (AB). We also employ a deep learning approach with a simple Neural Network (NN). For consistency, the NN has been trained on the same set of features. For every model, we fine-tuned the hyperparameters using grid search to maximize the recall.

Figure 8 summarizes the performances of the various models. As explained in Section 4.1.6, we want to reach high recall to ensure a good representation of all the EPIC samples as real. We can see how different models performed quite well, except LR, which failed to identify most of the EPIC sample as real. AB, NN, DT, and RF are the best-performing models, with some differences. For instance, the NN performs well in the current I3, while RF is the best model for the first phase of the current I1. We prefer AB and RF to deep architectures such as the NN that may lead to overfitting, but all of them could be considered as good candidate models for *SimProcess*.

5.3 Results

This section presents the results of our EPIC case study. We show the capabilities of *SimProcess* to provide a fidelity score value to each simulation and for each measure and discuss its strengths and limitations. We analyze the relevance of the employed features in Section 5.3.1, the fidelity scores in Section 5.3.2, and discuss the applicability in a dynamic system in Section 5.3.3.

5.3.1 Features Analysis. Understanding which are the features that impact the most in the classification is essential to comprehend the capabilities of the system. Figure 9 shows the most important features we isolate based on p-values for the voltage V1 case. We can see how the noise mean (i.e., the mean of the estimated noise within each window) is the most important feature, followed by the

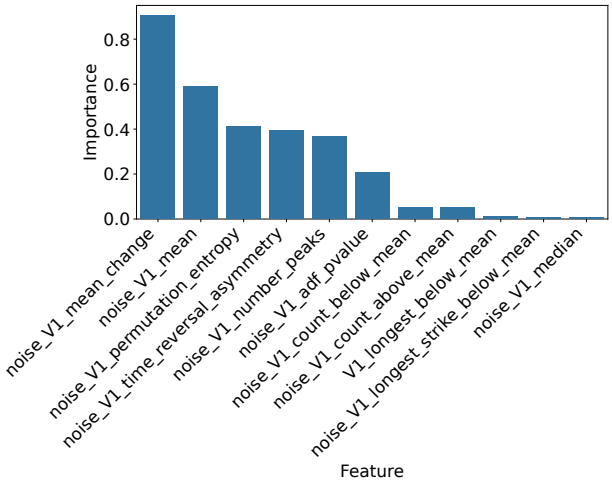
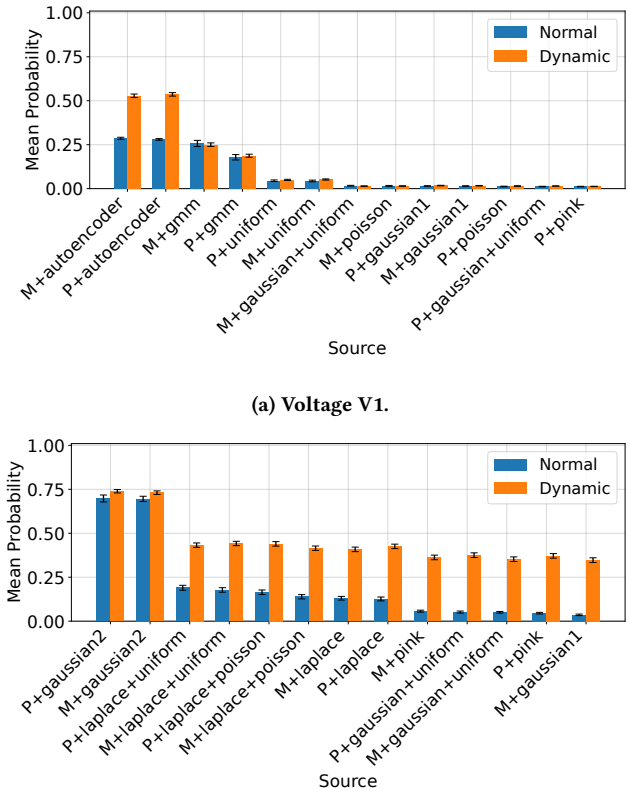


Figure 9: Most important features for V1. Features beginning with "noise" point to features extracted from the noise estimation.



(a) Voltage V1.

(b) Current I1.

Figure 10: Resemblance of a real system of each simulation (M: Mosaik, P: Pandapower) with different noises applied. Blue scores represent normal samples (used for training) discussed in Section 5.3.2, while orange represents the dynamic scenario (not used in training) discussed in Subsection 5.3.3. Error bars indicates the standard error.

permutation entropy of the noise. Five more features extracted from the noise estimate follow suggesting that the noise estimation is practical and its features are meaningful for the classification. Then, the longest above mean feature extracted from the original signal suggests that, even to a smaller extent, meaningful information is extracted from the original signal as well. Similar graphs for the other measurements are available in our GitHub repository.

5.3.2 Simulations Fidelity Scores. Figure 10 shows each window's mean probability of being classified as a real sample (blue bars). High values represent noise that may easily resemble the real data, while low values indicate high chances for the system to recognize the samples as a simulation. In particular, Figure 10a shows the probabilities related to the first voltage phase V1. The graph suggested that GMM is the most well-performing noise for these kinds of measurements with a score of 0.267, even if the autoencoder noise follows with almost the same score (0.265).

Figure 10b represents a different situation for the first current phase I1. Here we can see a high peak of the Gaussian noise with high standard deviation, representing the optimal noise for current values. Then, it is interesting to see that uniform noise follows obtaining good results, and it may indicate errors introduced by the analog-to-digital converters in the PLC.

We cannot plot the graphs for all the measurements for space reasons (the reader will find it in the GitHub repository), but Table 2 summarizes the best noise for each value, indicating the probability for such a sample to be categorized as real. We can see how similar measures share the same noise. The Gaussian distribution well represents current noise, while the voltages are well represented by GMM and autoencoders. In particular, similarly to Figure 10a, V2 and V3 show scores for GMM that are close to the autoencoder.

Our autoencoder also represents frequency noise, while powers follow Gaussian noises, which are probably derived from the current measurements. Interestingly, the power reactive follows a Laplace + Uniform noise, even though other noises such as Poisson and Uniform reach similar probabilities. When considering all values, the GMM is the best distribution to replicate the noise.

Similar measures (e.g., current) share similar noise distributions (e.g., Gaussian), which is at least partially related to the type of sensors employed to collect the measurements. While it is possible to identify the swapping of a sensor with another identical replica through a difference in the generated noise [2, 3], the difference magnitude is more significant when considering different categories of sensors. Still, similar patterns may be observed for probes measuring the same quantity. These considerations suggest that the better strategy for a developer is to take a selective approach and employ different noises for each signal s/he needs to simulate.

5.3.3 Applicability To Dynamic Systems. All the noises discussed in our system have been applied to the plain simulated timeseries $x^*(t)$ as an additional value $n_i(t)$ to generate a signal $\hat{x}^*(t)$ that approximate the real process $\hat{x}(t)$, as discussed in Section 5. However,

Table 2: Summary of the best noises for every measurement. Prob. Indicates the associated probability of misclassification as real, while Delta is the difference in probability in the case of dynamic samples.

Value	Best Noise	Prob.	Delta
voltage V1	GMM	0.267	0.064
voltage V2	Autoencoder	0.397	0.078
voltage V3	Autoencoder	0.356	0.076
current I1	Gaussian2	0.688	0.052
current I2	Gaussian2	0.365	0.074
current I3	Gaussian2	0.525	0.065
frequency	GMM	0.331	0.233
power_apparent	Gaussian2	0.605	0.039
power_reactive	Laplace+Uniform	0.133	0.045
power_real	Gaussian2	0.707	0.073
all values	GMM	0.418	0.119

the mechanisms we employed to add the noise are applicable in real-time and therefore could be used in dynamic simulators that support user interaction and consequent change in the simulated process.

For instance, a developer may allow the attacker to modify the value of a coil in a power grid to open or close a switch. This will disconnect or connect, respectively, a load that has an impact on the simulated process—it will require additional current or release the used current, respectively. This dynamic behavior is a key concept of a high-interaction honeypot, and the *SimProcess* works adequately even in these conditions as shown by the results of the experiment described next.

We modified the developed simulations discussed in Section 5.1.2 to include the effect of opening or closing a switch. We connected a load requesting $4A$ or disconnected a load absorbing $3A$. This does not directly impact the voltage but has a proportional effect on the powers. Then, we employed this new data for testing on our previously trained model (i.e., without including these new signals in the training set) and measured how effective the noises were in these new conditions.

Results are shown in Figure 10 for V1 and I1 (orange bars). I1 is the most influenced measure in this scenario since it is the variable directly influenced by the load change. We see how the Gaussian noise performs better, indicating that our framework can successfully classify the noise instead of the underlying physical process. We also note how probabilities are higher with the fluctuation scenario, but the order is kept almost unchanged. In the case of V1, GMM remains a good representation of the noise with similar probabilities. However, the high increase in the autoencoder suggests that such a generative approach is best suited for dynamic solutions that require more adaptability.

We summarized the probability difference between each noise applied to the normal system and the dynamic scenario in Table 2 in the *Delta* column. We can see the low probability changes, especially for the voltage and current cases. These results show how *SimProcess* is robust to dynamic changes in the system, making it applicable in high-interaction honeypots.

6 Related Work

After the infamous Stuxnet attack in 2010 [20], awareness of ICS security started to rise, and academia and industry began developing and deploying decoy systems to trick the attacker into revealing their attack techniques without compromising the real target. These systems, called ICS honeypots, exhibit different capabilities, from low interaction honeypots [43] to more advanced architectures [17, 30, 31, 46]. While low interaction honeypots are easy to identify [61], researchers also propose solutions to detect high interaction ones [52].

An interesting approach has been taken by HoneyJudge [64]. This framework considers six different parameters to identify real or simulated PLCs. They relate to memory handling, logic execution, and the behavior of the physical process. In particular, the authors propose collecting data from the I/O memory of PLCs to extract features such as sensor noise distributions and process dynamics. HoneyJudge employs physical equations to identify the right time for the data collection from the PLC under test. which needs to be accessible during the profiling. Our solution, instead, does not require equations of the physical process and works on systems that could be slightly different from the real one.

In [2], the authors propose a defensive mechanism against ICS physical attacks. The system can detect swapping or replacement of sensors by extracting a fingerprint from each sensor noise. It extracts the noise by modeling a simple water tank as an LTI system and using the model to extract the noise from the measurements. A similar solution has been proposed by Ahmed et al. [3]. They presented a way to fingerprint sensors based on imperfections and noise. The authors proposed to use this information to detect sensor spoofing attacks. The system works by employing subspace system identification techniques to identify the physical equations governing the system from a dataset of measurements from all the sensors. However, that methodology is suited for LTI systems such as the water treatment system employed in the paper. Still, it does not work properly on dynamic (non-LTI) systems such as a power grid. Other defensive mechanisms have been proposed, alleviating the requirements for differential equations of the system [5, 32]. However, no one ever used noise as a source to benchmark simulation fidelity in the ICS context.

Fuzzy testing has been proposed to identify honeypots [53]. It collects device behaviors through a series of packets used as probes and feeds their responses to a deep learning classifier to detect simulated environments. While this system seems compelling, it represents an orthogonal research line to ours. The authors' solution fingerprints a lack of emulation of industrial devices and their protocols but does not consider the underlying physical process. Moreover, solutions exist to prevent the fuzzy identification of honeypots [60].

Different works discussed the characterization of the noise present in a physical process, particularly in power systems [19]. Many papers characterize the measurement noise as a Gaussian noise [9]. However, other research investigates the presence of other non-Gaussian noises, such as the Laplacian and Gaussian Mixture noise discussed in [18, 36].

7 Conclusion

In this paper, we introduced *SimProcess*, a novel framework to assess the physical process fidelity of an ICS simulation based on the underlying noise of the system and its components. The framework is general purpose and can be adapted to different scenarios, without any assumption on the system to simulate (e.g., no mathematical equations describing the system are needed). We propose a case study based on the EPIC testbed and derive different simulations of its physical process, characterized by the presence of different kinds of noise. After a fine-tuning of the framework on our dataset, we employ *SimProcess* and find out that the simulations employing noise from Gaussian and GMM distributions were the most similar to the real scenario, together with the noise generated by the autoencoder. Moreover, we show how different measures could be characterized by different noises, and our system allows benchmarking different noises to select the most realistic one to be employed in the simulated scenario under development. Finally, we show how *SimProcess* is resistant to process variations and keeps its capabilities of identifying the most suited noise even in the presence of dynamic systems.

Limitations. Despite a quite good amount of ICS datasets available in the literature [16], only a few consist of power grid simulations providing physical measurements [1, 41], while others only contain network traffic [22]. Moreover, not all the testbeds have sufficient documentation allowing for the generation of realistic replicas [41]. In other scenarios, such as water treatment, the situation is not much better. In general, while simulators are more widely used, data from real systems is still scarce. Additionally, our simulations do not perfectly reflect the entire EPIC testbed in all its components. This limitation arises from the complexity of power grids and cyber-physical systems in general, as well as the limited number of tools available to efficiently simulate a real-world scenario. Furthermore, details regarding testbeds and datasets [16] are often lacking, making it difficult to accurately emulate a real system.

Future Work. Testing our *SimProcess* on new datasets shared by the community in the coming years is an interesting path for future research. We also invite researchers who are building testbeds and datasets to share simulations and detailed descriptions of their systems, as this will enable the community to perform research similar to this one. Another promising direction for future work involves exploring the problem from an attacker's point of view. Developing noise fingerprinting systems could allow for the identification of simulations. This would pave the way for the development of anti-honeypot fingerprinting techniques, which may enhance the current state of the art in the field [52]. Finally, new and more complex generative solutions to add noise to simulations can be investigated, even employing physics-aware generative models [65].

Acknowledgments

Work partially funded by the European Union under grant agreement no. 101070008 (ORSHIN project). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them. Daniele Antonioli has been partially supported by the French National Research Agency under the France 2030 label (NF-HiSec ANR-22-PEFT-0009). Massimo Merro has been partially supported by the SERICS project (PE00000014) under the *MUR National Recovery and Resilience Plan*, funded by the EU - NextGenerationEU.

References

[1] Sridhar Adepu, Nandha Kumar Kandasamy, and Aditya Mathur. 2019. Epic: An electric power testbed for research and training in cyber physical systems security. In *Computer Security: ESORICS 2018 International Workshops, CyberICPS 2018 and SECPRE 2018, Barcelona, Spain, September 6–7, 2018, Revised Selected Papers 2*. Springer, 37–52.

[2] Chuadhry Mujeeb Ahmed and Aditya P Mathur. 2017. Hardware identification via sensor fingerprinting in a cyber physical system. In *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE, 517–524.

[3] Chuadhry Mujeeb Ahmed, Jianying Zhou, and Aditya P Mathur. 2018. Noise matters: Using sensor and process noise fingerprint to detect stealthy cyber attacks and authenticate sensors in cps. In *Proceedings of the 34th Annual Computer Security Applications Conference*. 566–581.

[4] Daniele Antonioli and Nils Ole Tippenhauer. 2015. MiniCPS: A toolkit for security research on CPS networks. In *Proceedings of the First ACM workshop on cyber-physical systems-security and/or privacy*. 91–100.

[5] Wissam Aoudi, Mikel Iturbe, and Magnus Almgren. 2018. Truth will out: Departure-based process-level detection of stealthy attacks on control systems. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 817–831.

[6] R Baldick, KA Clements, Z Pinjo-Dzigal, and PW Davis. 1997. Implementing nonquadratic objective functions for state estimation and bad data rejection. *IEEE Transactions on Power Systems* 12, 1 (1997), 376–382.

[7] Giovanni Barbieri, Mauro Conti, Nils Ole Tippenhauer, and Federico Turrin. 2021. Assessing the use of insecure ics protocols via ixp network traffic analysis. In *2021 international conference on computer communications and networks (icccn)*. IEEE, 1–9.

[8] Giuseppe Bernieri, Mauro Conti, and Federico Turrin. 2019. Evaluation of machine learning algorithms for anomaly detection in industrial networks. In *2019 IEEE International Symposium on Measurements & Networking (M&N)*. IEEE, 1–6.

[9] Michael Brown, Milan Biswal, Sukumar Brahma, Satish J Ranade, and Huiping Cao. 2016. Characterizing and quantifying noise in PMU data. In *2016 IEEE Power and Energy Society General Meeting (PESGM)*. IEEE, 1–5.

[10] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[11] Tengpeng Chen, Lu Sun, Keck-Voon Ling, and Weng Khuen Ho. 2019. Robust power system state estimation using t-distribution noise model. *IEEE Systems Journal* 14, 1 (2019), 771–781.

[12] Chen-Ching Liu Chih-Che Sun, Adam Hahn. 2018. Cyber security of a power grid: State-of-the-art. *International Journal of Electrical Power & Energy Systems* 99 (2018), 45–56.

[13] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W Kempa-Liehr. 2018. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package). *Neurocomputing* 307 (2018), 72–77. doi:10.1016/j.neucom.2018.03.067

[14] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. *Advances in neural information processing systems* 28 (2015).

[15] Tommaso Coletta, Bassam Bamieh, and Ph Jacquod. 2018. Transient performance of electric power networks under colored noise. In *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 6163–6167.

[16] Mauro Conti, Denis Donadel, and Federico Turrin. 2021. A survey on industrial control system testbeds and datasets for security research. *IEEE Communications Surveys & Tutorials* 23, 4 (2021), 2248–2294.

[17] Mauro Conti, Francesco Trolese, and Federico Turrin. 2022. Icspot: A high-interaction honeypot for industrial control systems. In *2022 International Symposium on Networks, Computers and Communications (ISNCC)*. IEEE, 1–4.

[18] Stefan Čubonović, Dragan Četenović, and Aleksandar Ranković. 2024. Impact of the non-Gaussian measurement noise on the performance of state-of-the-art state estimators for distribution systems. *Serbian Journal of Electrical Engineering* 21, 1 (2024), 113–133.

[19] Himanshu Dehra. 2018. Characterization of noise in power systems. In *2018 International Conference on Power Energy, Environment and Intelligent Control (PEEIC)*. IEEE, 320–329.

[20] Nicolas Falliere, Liam O Murchu, Eric Chien, et al. 2011. W32. stuxnet dossier. *White paper, symantec corp., security response* 5, 6 (2011), 29.

[21] Javier Franco, Ahmet Aris, Berk Canberk, and A Selcuk Uluagac. 2021. A survey of honeypots and honeynets for internet of things, industrial internet of things, and cyber-physical systems. *IEEE Communications Surveys & Tutorials* 23, 4 (2021), 2351–2383.

[22] Ángel Luis Perales Gómez, Lorenzo Fernández Maimó, Alberto Huertas Celdrán, Félix J García Clemente, Cristian Cadenas Sarmiento, Carlos Javier Del Canto Masa, and Rubén Méndez Nistal. 2019. On the generation of anomaly detection datasets in industrial control systems. *IEEE Access* 7 (2019), 177460–177473.

[23] Qingqing Huang, Leilai Shao, and Na Li. 2015. Dynamic detection of transmission line outages using hidden Markov models. *IEEE Transactions on power systems* 31, 3 (2015), 2026–2033.

[24] Alshaibi Ahmed Jamal, Al-Ani Mustafa Majid, Anton Konev, Tatiana Kosachenko, and Alexander Shelupanov. 2023. A review on security analysis of cyber physical systems using Machine learning. *Materials today: proceedings* 80 (2023), 2302–2306.

[25] Arthur Jicha, Mark Patton, and Hsinchun Chen. 2016. SCADA honeypots: An in-depth analysis of Conpot. In *2016 IEEE conference on intelligence and security informatics (ISI)*. IEEE, 196–198.

[26] Nandha Kumar Kandasamy. 2019. An investigation on feasibility and security for cyberattacks on generator synchronization process. *IEEE Transactions on Industrial Informatics* 16, 9 (2019), 5825–5834.

[27] Petar Kovač, Ardian Pantina, Stjepan Groš, and Damir Sumina. 2023. Development of Programmable Logic Controller Emulator With QEMU. In *IEEE EUROCON 2023-20th International Conference on Smart Technologies*. IEEE, 770–775.

[28] Maryna Krotofil and Dieter Gollmann. 2013. Industrial control systems security: What is happening?. In *2013 11th IEEE International Conference on Industrial Informatics (INDIN)*. IEEE, 670–675.

[29] Paula Lamo, Gustavo A Ruiz, Francisco J Azcondo, Alberto Pigazo, and Christian Brañas. 2023. Impact of the noise on the emulated grid voltage signal in hardware-in-the-loop used in power converters. *Electronics* 12, 4 (2023), 787.

[30] Efrén López-Morales, Carlos Rubio-Medrano, Adam Doupé, Yan Shoshitaishvili, Ruoyu Wang, Tiffany Bao, and Gail-Joon Ahn. 2020. Honeyplc: A next-generation honeypot for industrial control systems. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 279–291.

[31] Marco Lucchese, Francesco Lupia, Massimo Merro, Federica Paci, Nicola Zannone, and Angelo Furfaro. 2023. Honeyics: A high-interaction physics-aware honeynet for industrial control systems. In *Proceedings of the 18th International Conference on Availability, Reliability and Security*. 1–10.

[32] Yuan Luo, Long Cheng, Yu Liang, Jianming Fu, and Guojun Peng. 2021. Deepnoise: Learning sensor and process noise to detect data integrity attacks in CPS. *China Communications* 18, 9 (2021), 192–209.

[33] Alfian Ma'arif, Iswanto Iswanto, Aninditya Anggari Nuryono, and Rio Ikhsan Alfian. 2019. Kalman filter for noise reducer on sensor readings. *Signal and Image Processing Letters* 1, 2 (2019), 50–61.

[34] Jan Machowski, Zbigniew Lubosny, Janusz W Bialek, and James R Bumby. 2020. *Power system dynamics: stability and control*. John Wiley & Sons.

[35] David Makovoz. 2006. Noise variance estimation in signal processing. In *2006 IEEE International Symposium on Signal Processing and Information Technology*. IEEE, 364–369.

[36] R Martínez-Parrales, CR Fuerte-Esquivel, and BA Alcaide-Moreno. 2020. Analysis of bad data in power system state estimation under non-Gaussian measurement noise. *Electric Power Systems Research* 186 (2020), 106424.

[37] Daisuke Mashima, Muhammad M Roomi, Bennet Ng, Zbigniew Kalberczyk, SM Suhail Hussain, and Ee-chien Chang. 2023. Towards automated generation of smart grid cyber range for cybersecurity experiments and training. In *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*. IEEE, 49–55.

[38] Tsega Y Melesse, Valentina Di Pasquale, and Stefano Riemma. 2020. Digital twin models in industrial operations: a systematic literature review. *Procedia Manufacturing* 42 (2020), 267–272.

[39] Ariana Mirian, Zane Ma, David Adrian, Matthew Tischer, Thasphon Chuenchujit, Tim Yardley, Robin Berthier, Joshua Mason, Zakir Durumeric, J Alex Halderman, et al. 2016. An internet-wide view of ICS devices. In *2016 14th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 96–103.

[40] Sinil Mubarak, Mohamed Hadi Habaebi, Md Rafiqul Islam, Farah Diyana Abdul Rahman, and Mohammad Tahir. 2021. Anomaly Detection in ICS Datasets with Machine Learning Algorithms. *Computer Systems Science & Engineering* 37, 1 (2021).

[41] Shengyi Pan, Thomas Morris, and Uttam Adhikari. 2015. Developing a hybrid intrusion detection system using data mining for power systems. *IEEE Transactions on Smart Grid* 6, 6 (2015), 3104–3113.

[42] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[43] Niels Provos. 2003. Honeyd-a virtual honeypot daemon. In *10th dfn-cert workshop, hamburg, germany*, Vol. 2. 4.

[44] R. Rajkumar, L. Lee, I. Sha, and J. A. Stankovic. 2010. Cyber-physical systems: the next computing revolution. In *DAC*. ACM, 731–736.

[45] Luis Salazar, Sebastián R Castro, Juan Lozano, Keerthi Koneru, Emmanuele Zambon, Bing Huang, Ross Baldick, Marina Krotofil, Alonso Rojas, and Alvaro A Cardenas. 2024. A tale of two Industroyers: It was the season of darkness. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 312–330.

[46] Luis Salazar, Efren López-Morales, Juan Lozano, Carlos Rubio-Medrano, and Alvaro A Cardenas. 2024. ICSNet: A Hybrid-Interaction Honeynet for Industrial Control Systems. In *Proceedings of the Sixth Workshop on CPS&IoT Security and Privacy*. 68–79.

[47] Steffen Schütte, Stefan Scherfke, and Martin Tröschel. 2011. Mosaik: A framework for modular simulation of active components in smart grids. In *2011 IEEE First International Workshop on Smart Grid Modeling and Simulation (SGMS)*. IEEE, 55–60.

[48] Di Shi, Daniel J Tylavsky, and Naim Logic. 2012. An adaptive method for detection and correction of errors in PMU measurements. *IEEE Transactions on Smart Grid* 3, 4 (2012), 1575–1583.

[49] Siddhant Shrivastava. 2016. Blackenergy-malware for cyber-physical attacks. *Singapore* 74 (2016), 115.

[50] Ahnaf Siddiqi, Nils Ole Tippenhauer, Daisuke Mashima, and Binbin Chen. 2018. On Practical Threat Scenario Testing in an Electric Power ICS Testbed. In *Proceedings of the 4th ACM Workshop on Cyber-Physical System Security (CPSS '18)*. Association for Computing Machinery, 15–21.

[51] Joseph Slowik. 2019. Evolution of ICS attacks and the prospects for future disruptive events. *Threat Intelligence Centre Dragos Inc* (2019).

[52] Shreyas Srinivasa, Jens Myrup Pedersen, and Emmanouil Vasilomanolakis. 2023. Gotta catch'em all: a multistage framework for honeypot fingerprinting. *Digital Threats: Research and Practice* 4, 3 (2023), 1–28.

[53] Yanbin Sun, Xiaojun Pan, Chao Xu, Penggang Sun, Quanlong Guan, Mohan Li, and Men Han. 2020. Identifying Honeypots from ICS Devices Using Lightweight Fuzzy Testing. *Computers, Materials & Continua* 65, 2 (2020).

[54] Heng Chuan Tan, Md Adeeb Hossain, Daisuke Mashima, and Zbigniew Kalbarczyk. 2024. High-fidelity Intrusion Detection Datasets for Smart Grid Cybersecurity Research. In *2024 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 340–346.

[55] Vanessa Tay, Xinran Li, Daisuke Mashima, Bennet Ng, Phuong Cao, Zbigniew Kalbarczyk, and Ravishankar K Iyer. 2023. Taxonomy of fingerprinting techniques for evaluation of smart grid honeypot realism. In *2023 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, 1–7.

[56] Leon Thurner, Alexander Scheidler, Florian Schäfer, Jan-Hendrik Menke, Julian Dollichon, Friederike Meier, Steffen Meinecke, and Martin Braun. 2018. pandapower—an open-source python tool for convenient modeling, analysis, and optimization of electric power systems. *IEEE Transactions on Power Systems* 33, 6 (2018), 6510–6521.

[57] Praveen Tripathy, Suresh C Srivastava, and Sri Niwas Singh. 2009. A divide-by-difference-filter based algorithm for estimation of generator rotor angle utilizing synchrophasor measurements. *IEEE Transactions on Instrumentation and Measurement* 59, 6 (2009), 1562–1570.

[58] Shaobu Wang, Junbo Zhao, Zhenyu Huang, and Ruisheng Diao. 2017. Assessing Gaussian assumption of PMU measurement error using field data. *IEEE Transactions on Power Delivery* 33, 6 (2017), 3233–3236.

[59] Thomas J Witt and Yi-hua Tang. 2005. Investigations of noise in measurements of electronic voltage standards. *IEEE transactions on instrumentation and measurement* 54, 2 (2005), 567–570.

[60] Yunhao Xu, Chao Li, Daiqi Gu, Zhewei Zhang, Zhe Sun, and Yanfei Song. 2024. A Novel Method for Honeypot Anti-Identification against Modbus Fuzz Testing in Industrial Control Systems. In *2024 IEEE 9th International Conference on Data Science in Cyberspace (DSC)*. IEEE, 599–606.

[61] Mohammad-Reza Zamiri-Gourabi, Ali Razmjoo Qalaei, and Babak Amin Azad. 2019. Gas what? I can see your GasPots. Studying the fingerprintability of ICS honeypots in the wild. In *Proceedings of the fifth annual industrial control system security (ics) workshop*. 30–37.

[62] Jinghe Zhang, Greg Welch, Gary Bishop, and Zhenyu Huang. 2013. A two-stage Kalman filter approach for robust and real-time power system state estimation. *IEEE Transactions on Sustainable Energy* 5, 2 (2013), 629–636.

[63] Ning Zhou, Zhenyu Huang, Da Meng, Stephen T Elbert, Shaobu Wang, and Ruisheng Diao. 2014. *Capturing dynamics in the power grid: Formulation of dynamic state estimation through data assimilation*. Technical Report. Pacific Northwest National Lab.(PNNL), Richland, WA (United States).

[64] Hengye Zhu, Mengxiang Liu, Binbin Chen, Xin Che, Peng Cheng, and Ruilong Deng. 2024. HoneyJudge: A PLC Honeypot Identification Framework Based on Device Memory Testing. *IEEE Transactions on Information Forensics and Security* (2024).

[65] Tetiana Zubatiuk and Olexandr Isayev. 2021. Development of multimodal machine learning potentials: toward a physics-aware artificial intelligence. *Accounts of Chemical Research* 54, 7 (2021), 1575–1585.

A Appendix

A.1 tsfresh Relevant Features

In this section, we list the features from `tsfresh` [13] we kept while extracting features from the original signal that extract information from the noise variation and not from the underlying process.

- $approximate\_entropy(x, m, r)$. Implements a vectorized Approximate entropy algorithm.
- $kurtosis(x)$. Returns kurtosis of x calculated with the adjusted Fisher-Pearson standardized moment coefficient $G2$.
- $lempel\_ziv\_complexity(x, bins)$. Calculate a complexity estimate based on the Lempel-Ziv compression algorithm.
- $longest\_strike\_above\_mean(x)$. Returns the length of the longest consecutive subsequence in x that is bigger than the mean of x.
- $longest\_strike\_below\_mean(x)$. Returns the length of the longest consecutive subsequence in x that is smaller than the mean of x.
- $number\_peaks(x, n)$. Calculates the number of peaks of at least support n in the time series x.
- $permutation\_entropy(x, tau, dimension)$. Calculate the permutation entropy.
- $skewness(x)$. Returns the sample skewness of x calculated with the adjusted Fisher-Pearson standardized moment coefficient $G1$.
- $autocorrelation(x, lag)$. Calculates the autocorrelation of the specified lag.