Received 14 April 2023, accepted 10 May 2023, date of publication 17 May 2023, date of current version 25 May 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3277250*

RESEARCH ARTICLE

# GAN Neural Networks Architectures for Testing Process Control Industrial Network Against Cyber-Attacks

**KRZYSZTOF ZARZYCKI[1], PATRYK CHABER[1], KRZYSZTOF CABAJ[2], MACIEJ ŁAWRYŃCZUK[1], PIOTR MARUSAK[1], ROBERT NEBELUK[1], SEBASTIAN PLAMOWSKI[1], AND ANDRZEJ WOJTULEWICZ[1]**

[1]Faculty of Electronics and Information Technology, Institute of Control and Computation Engineering, Warsaw University of Technology, 00-665 Warsaw, Poland

[2]Faculty of Electronics and Information Technology, Institute of Computer Science, Warsaw University of Technology, 00-661 Warsaw, Poland

Corresponding author: Maciej Ławryńczuk (Maciej.Lawrynczuk@pw.edu.pl)

This work was supported by the Laboratory for Vulnerability Analysis (LaVA) of Stationary and Mobile IT Devices and Algorithms and Software funded by the National Centre for Research and Development under Grant CYBERSECIDENT/488240/IV/NCBR/2021.

**ABSTRACT** Protection of computer systems and networks against malicious attacks is particularly important in industrial networked control systems. A successful cyber-attack may cause significant economic losses or even destruction of controlled processes. Therefore, it is necessary to test the vulnerability of process control industrial networks against possible cyber-attacks. Three approaches employing Generative Adversarial Networks (GANs) to generate fake Modbus frames have been proposed in this work, tested for an industrial process control network and compared with the classical approach known from the literature. In the first approach, one GAN generates one byte of a message frame. In the next two approaches, expert knowledge about frame structure is used to generate a part of a message frame, while the remaining parts are generated using single or multiple GANs. The classical single-GAN approach is the worst one. The proposed one-GAN-per-byte approach generates significantly more correct message frames than the classical method. Moreover, all the generated fake frames have been correct in two of the proposed approaches, i.e., single GAN for selected bytes and multiple GANs for selected bytes methods. Finally, we describe the effect of cyber-attacks on the operation of the controlled process.

**INDEX TERMS** GAN neural networks, cyber-security, cyber-attacks, industrial network.

## I. INTRODUCTION

Cyber-security is increasingly important these days. Protection of computer systems and networks against malicious attacks is important in all computer systems, particularly in industrial networked control systems [1]. Such systems are usually very complicated since they are composed of many components, i.e., the controlled processes themselves, Programmable Logic Controllers (PLCs) [2], Supervisory Control and Data Acquisition Systems (SCADAs) [3]. A successful cyber-attack may cause significant economic losses or even destruction of controlled processes, e.g., power stations, refineries or waste-water treatment plants. Therefore, it is necessary to test the vulnerability of process control industrial networks against possible cyber-attacks.

One popular method of testing network and program resilience is error injection [4], [5]. It involves intentionally writing errors into the program code or sending messages containing errors to the network. Errors can be injected into the system, such as process malfunctions or valve failures. A neural model may detect these errors [6]. A digital twin approach is discussed in [7]; it is possible to observe the influence of various errors. Let us note that in this approach, the so-called error space grows exponentially as the system's complexity increases. The problem with a vast error space can be solved using a testing technique called fuzzing, which is

The associate editor coordinating the review of this manuscript and approving it for publication was Roberto Nardone.

an automated or semi-automated software testing technique that involves providing incorrect, unexpected or random data as input to the system [8], [9], [10], [11]. An effective fuzzer generates such input data that is sufficiently correct not to be directly rejected by the parser but is at the same time sufficiently incorrect to cause unexpected system behavior. The fuzzer can generate valid queries and responses and then fuzz them using mutations and permutations of messages from the network [12]. Transformed messages are finally sent to the system. The EtherCat protocol is considered in [13], [14]. A fuzzer for the Modbus protocol is presented in [15]. A new approach to the vulnerability analysis of smart grid protocols using fuzzing is presented in [16], focusing on inter-protocol test generation. Before creating tests, protocols are classified into three categories. Based on the classification, tests are then created. A fuzzing method that uses both the input grammar of the testing software and the information extracted from the system is described in [14]; the Modbus protocol and SCADA software have been used. An advanced mutation method in the Modbus protocol is considered in [17].

Nowadays, neural networks [18], [19], [20], [21], [22] can also be used for fuzz testing [23]. GANs are of particular note [24]. GANs consist of two independently operating neural networks. The first one, called a discriminator, learns to recognize certain entities. The second one, called a generator, learns to generate these entities. The generator tries to outsmart the discriminator. The discriminator tries to prevent this with samples of true and false (generated) entities. Finally, the discriminator is discarded and the trained generator can be used to create synthetic entities, difficult to distinguish from the real ones. GANs have found many applications: recognition of facial expression [25], art generation [26], image transformation, e.g., aging or rejuvenating faces on photos [27], image [28] and textures in video games enhancement [29], traffic flow prediction [30], text to image synthesis [31], melody generation [32], designing an urban landscape [33].

GANs can also find various applications in the cybersecurity field. They may be used for password generation [34], steganography [35] and malware generation [36]. GANs can be successfully used to generate synthetic messages of industrial protocols for fuzzing. RapidFuzz [10] is a model that employs both mutation-based fuzzing and generative capabilities of GANs to test fuzz program code. A Mask Fuzzer based on GANs [37] has been proposed for testing Modbus TCP protocol. Authors of [38] use classic GANs and Wesserstein GANs (WGANs) to find vulnerabilities in Modbus TCP and EtherCat protocols. Similar work conducted for the DNP3 protocol resulted in the development of CGFuzzer [39], which employs Long Short-Term Memory (LSTM) neural networks-based generation of frames. It is important to note that in the works presented in the literature, the emphasis is on finding potential vulnerabilities and security holes in industrial protocols.

The classical approach to generating fake communication frames, named the single-GAN method [38], has an essential disadvantage because it gives many formally incorrect frames. Hence, we are motivated to develop new frame generation schemes that give formally correct frames but badly affect the control system. Therefore, we test four approaches to fuzz data collected from a laboratory process control network to perform an attack that is not easily detectable. The first one is the classical approach, whereas the authors propose three others. The first two approaches consist in generating a message frame using GANs. The first one [38], in which a single-GAN model is employed to generate one message frame, is done for comparison. In the second approach, one GAN generates one byte of a message frame. In the next two approaches, expert knowledge about frame structure is used to generate part of a message frame; in contrast, the remaining parts are generated using a single (the third approach) or multiple (the fourth approach) GANs. The article aims to present the experiments conducted in an example complex process control network, compare the tested approaches, and describe the advantages of the mechanisms that work. It is important to note that a cyber-attack may result in stopping the controller (i.e., hanging up), which is simple to detect. Moreover, a cyber-attack may lead to improper process operation that is difficult for the operator to identify. To sum up, the contributions of this work are:

1) Three new GAN-based fake frame generation methods for the Modbus protocol are presented. The detailed structure of each GAN employed is presented.
2) Using a laboratory test bed, we perform experiments to assess the effectiveness of new and classical methods. The test bed uses hardware and software components typical of industrial control systems.
3) We assess the effectiveness of all considered methods considering the controlled system's performance.
4) As discussed in this work, the developed methods perform differently but are much better than the classical single-GAN method.

At first, Section II discusses the classical GAN-based neural network architecture for testing industrial networks against cyber-attacks and introduces three new GAN-based structures. Next, Section III describes the experimental test bed and Section IV presents the obtained results; the efficiency of the classical structure and three new ones is compared. Finally, Section V concludes this work.

## II. EMPLOYED GAN STRUCTURES

In this work, we consider the following scenario: the GAN generates attacks that are not easily detected (they do not cause the PLC controller to crash). Additionally, we use a real-world example of a simple industrial network, where the variety of messages sent between various devices is limited. Most messages are either measurement requests or commands with calculated values of manipulated signals. Therefore, we have decided to modify the fuzzing method.

We focus on an approach to data generation where expert knowledge about network data is required. Thus, creating and analyzing a training set for the GAN model requires expertise and at least basic knowledge of the devices connected to the network. We aim to design a tool to carry out cyber-attacks on the network and manipulate the messages flowing in it so that the controlled process connected to the network works inefficiently. At the same time, there must be no obvious error information that a SCADA operator could observe. To develop a GAN that performs the task described above, we perform the following steps:

1) Collecting the training data set for the GAN model.
2) Preprocessing of the data, i.e., message selection and normalization.
3) Training GAN models in four different configurations (the first one is taken from the literature while the following three original ones are proposed in this work):
    a) *a single-GAN model* used to generate one message frame [38],
    b) *one-GAN-per-one-byte model* where one GAN is used to generate one byte of a message frame,
    c) *a single-GAN model for selected bytes*; a GAN is used to generate only a part of the network message frame; the remaining parts of the frame are generated based on expert knowledge,
    d) *multiple GAN models for selected bytes*; multiple GANs are used to generate selected bytes of the frame, and the remaining parts of the frame are generated based on expert knowledge.
4) Testing the messages generated by the models in a laboratory.

A large data set of messages flowing through the laboratory network has been collected. For the data acquisition, an artificial data flow has been introduced. This network traffic mainly consists of the following:

- communication between SCADA and PLCs using SLMP protocol,
- communication between PLCs and HMIs using Melsec Connection,
- communication between PLCs using PROFINET and Modbus TCP,
- other communication, e.g., between GXWorks software and PLC using Melsec Connection.

This traffic has been collected in the form of PCAP files. The first step to creating a training data set for GAN is to analyze the entire set of messages and select the most interesting ones from the point of view of launching an attack. We have decided that conducting the study for the Modbus TCP network protocol would be worthwhile since it is frequently used in industrial control networks [1]. In the laboratory stand, communication via Modbus TCP is carried out between two PLCs in master-slave configuration and the controlled process. The master controller receives measurements of the current state of the process from the slave controller. Proportional-Integral-Derivative (PID) and Dynamic Matrix Control (DMC) controllers [40], [41] have been implemented in the master controller, which calculates the current values of the manipulated variables. These signals are then sent to the slave controller, which finally sends them to the controlled process. Therefore, the training data set has been created with the message frames having the following characteristics:

a) Modbus TCP protocol frames,
b) queries,
c) containing the Write Multiple Registers function code, as these are concerned with potentially sending the values of the manipulated variable to the controlled process.

To summarize, only frames with the structure shown in Fig. 1 remain in the training set. A typical message in the training dataset has the following structure:

- Bytes 1 and 2 are the transaction ID number. The former takes values from 0 to 255; the latter always takes the value 1,
- Bytes 3 and 4 are the protocol ID number; they always take the value 0,
- Bytes 5 and 6 are the length of the message frame; they always take values 0 and 19,
- Byte 7 is the device ID number; it has the value 0,
- Byte 8 is the Write Multiple Registers function code, which always takes the value 16,
- Bytes 9 and 10 are the initial number of registers where new values are written; they have the value 0,
- Bytes 11 and 12 are the number of registers where new values are to be written; they take values 0 and 6,
- Byte 13 is the doubled number of registers; it has the value of 12,
- Bytes 14-25 are the values to be written to the registers; they can take values from 0 to 255, while the values of only the last four bytes are changed during system operation.

After appropriate training using the available data set, the GAN model is supposed to generate messages with similar content. Then, those messages can be sent to the system, replacing the signals of the manipulated variable generated by the controller.

Before the models can be trained, the collected data set has to be preprocessed. In order to complete this task, it is necessary:

a) to convert the data from the hexadecimal numeric format to the decimal one,
b) to normalize the data to take values between $-1$ and 1.

Fig. 2 presents the general concept of using the GAN; its top part shows the training of a GAN. Real Modbus protocol messages, i.e., messages collected from the laboratory network, are converted from hexadecimal to decimal number format. In parallel, random numbers are sent to the generator's input. The generator produces synthetic messages, which, along with the real messages, are sent to the input of the discriminator. The discriminator evaluates whether the synthetic data are real or artificial, comparing them with

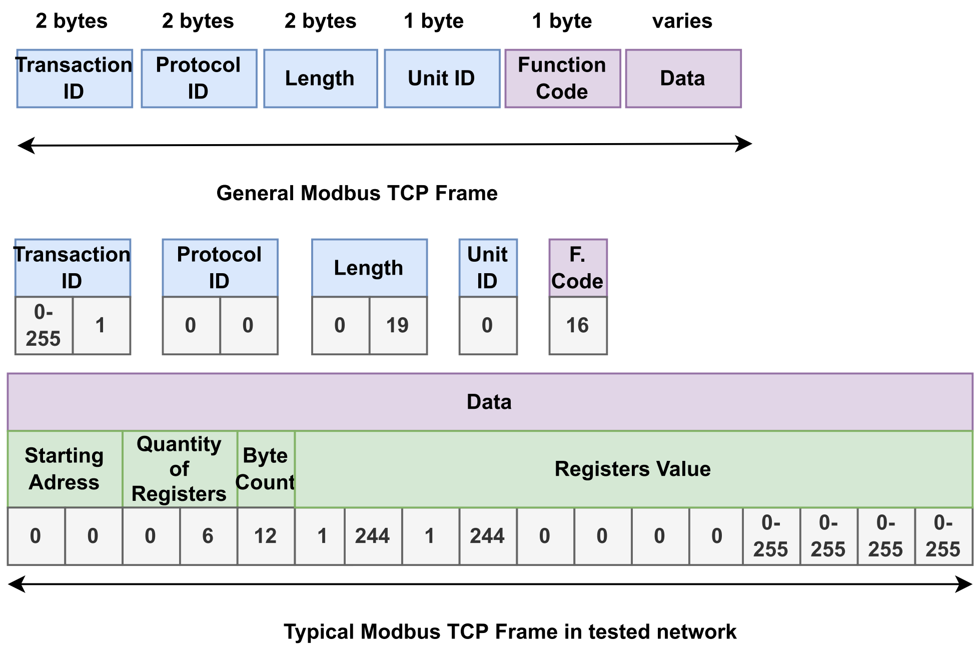


**FIGURE 1.** The general form of a Modbus protocol message frame (top figure) and the structure of a typical frame found in a training dataset (bottom figure).

real messages. The backpropagation algorithm is then used to compute the gradients of the cost functions of both the discriminator and generator with respect to the weights of the network. The entire procedure is repeated many times until optimal weight values are found, allowing the generator to consistently synthesize messages indistinguishable from the real ones. The operation of the trained neural model in the process control network is shown in the lower part of Fig. 2. The discriminator is discarded and not used. Random numbers are fed into the generator and synthetic Modbus protocol frames are produced. They are then converted from decimal to hexadecimal format, scaled and finally sent to the process control network, where they are transmitted to the PLC.

### A. STRUCTURE 1: SINGLE-GAN APPROACH

The first considered model has the structure shown in Fig. 3. Five hidden layers for the generator and three hidden layers of the discriminator have been chosen based on the work presented in [38]. The number of weights in each layer is denoted as a product of two numbers: the first represents the number of inputs of the layer and the second represents the number of its outputs. The number of weights has been chosen experimentally; the best results have been achieved for $n_1^G = 32$, $n_2^G = 64$, $n_3^G = 128$, $n_4^G = 64$, $n_5^G = 32$ for the generator and $n_1^D = 128$, $n_2^D = 64$ for the discriminator.

The generator receives as the input x random input signals previously normalized accordingly. These signals then pass through a linear hidden layer and the ReLU activation function layer. This situation is repeated three more times. Finally, the signals go to the last linear layer; its output y contains numbers corresponding to the values of 25 bytes of the message frame. The last layer is an activation function with a hyperbolic tangent to guarantee that the output values are between $-1$ and 1.

The discriminator is a network that has fewer layers. Its input receives 25 values produced by the generator. They eventually pass through two pairs of hidden layers, i.e., linear and ReLU ones, to reach the final fully connected layer. The signal coming out of the final layer passes through the last layer of activation, which is a sigmoidal function, to guarantee that the discriminator's output is between 0 and 1. This number can be interpreted as the probability that the frame that went to the input of the discriminator is correct.

### B. STRUCTURE 2: ONE-GAN-PER-ONE-BYTE APPROACH

Next, we consider a multi-model neural structure. This concept is as follows. A separate generator should produce each byte of a Modbus frame. Thus, during training, the training data set has been divided into 25 subsets; and each subset is used to train a pair of submodels, i.e., a generator and a discriminator.

We have noted that some submodels have a more difficult task, as they are supposed to generate variable values within specific ranges. In their case, the same complex generator structure is used as described earlier, with only one difference. Each generator takes one random number as the input and has only one numerical value as the output. This structure is named GAN Complex (GC). To make a comparison with a single GAN structure fair, each generator has several weights defined by $n_1^G = 32$, $n_2^G = 64$, $n_3^G = 128$, $n_4^G = 64$, $n_5^G = 32$. Other submodels, whose task is to match fixed

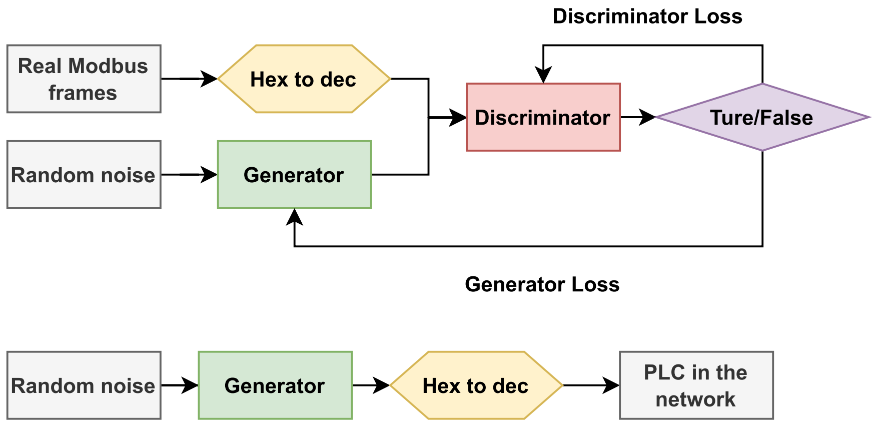


**FIGURE 2.** The general GAN model structure.

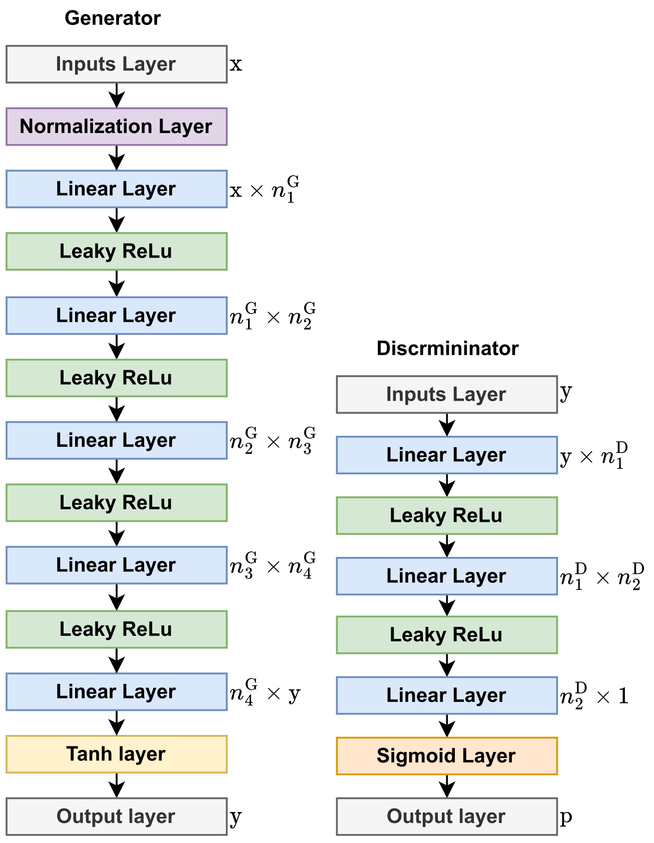


**FIGURE 3.** Single-GAN model structure.

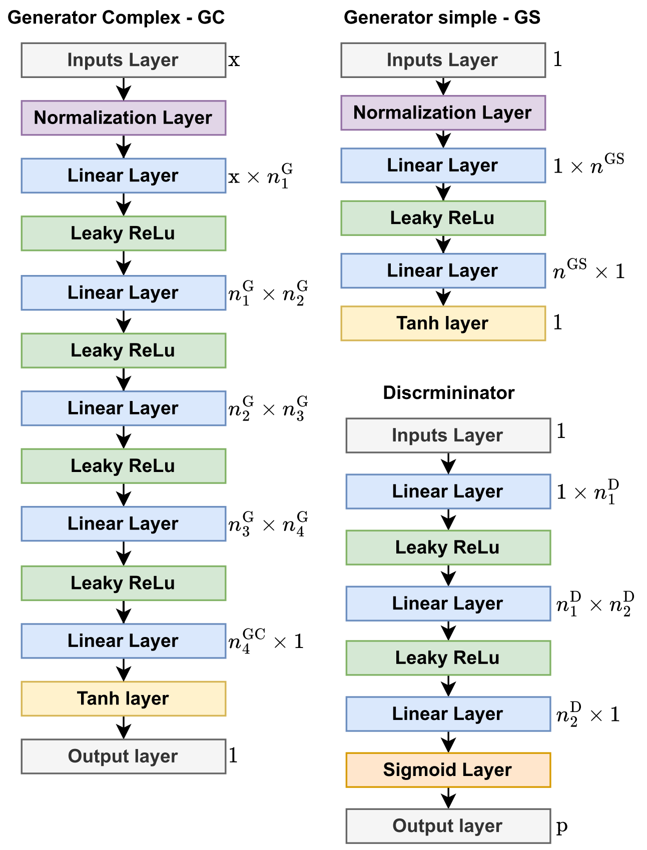


**FIGURE 4.** One-GAN-per-one-byte model structure: the structure of individual models.

values, have a much simpler task. Therefore, we have decided they would have a more straightforward structure, i.e., only one pair of hidden layers (linear and ReLU). This structure is named GAN Simple (GS). Each GS has the number of neurons chosen experimentally; the best results achieved are for $n^{GS} = 64$. All discriminators have the same structure and take a single numerical value as the input. The number of weights is also the same, i.e., $n_1^D = 128$, $n_2^D = 64$. Fig. 4 shows the structure of individual models. On the other hand, Fig. 5 shows the concept of operation of the trained model by sending synthetic messages to the PLC. The input vector $x$ is divided into 25 scalars, each containing a single byte in the frame. Each of the scalars enters a GAN submodel. The GC model is used for the 0th, 21st, 22nd, 23rd and 24th bytes. On the other hand, for the 1st, 2nd, …, 20th bytes, we use less complex GS models. Each GAN submodel generates a single scalar value. Those values are then combined into an

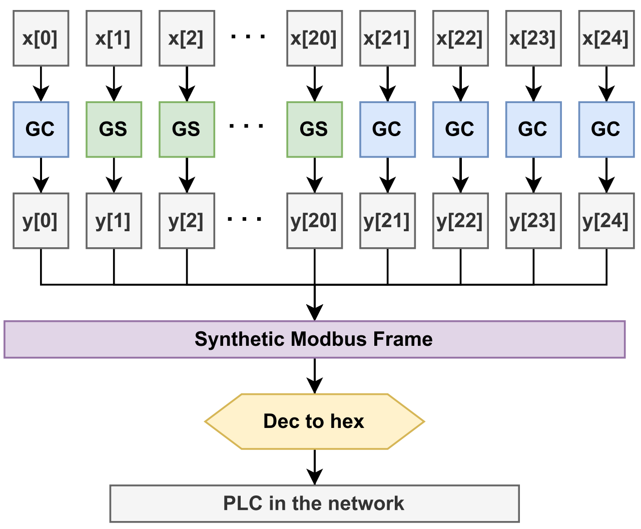


**FIGURE 5. One-GAN-per-one-byte model structure: the concept of operation.**

output vector *y*. Finally, the *y* vector values are transformed from decimal to hexadecimal format and sent to PLC.

### C. STRUCTURE 3: SINGLE-GAN FOR SELECTED BYTES APPROACH

In this approach, we have assumed there is no need for their generation when no changes in individual byte values are observed in the network traffic. The GAN model is used only to generate the last 4 bytes of the frame. The structure of both the discriminator and the generator is the same as in the approach with the GAN model presented in Fig. 3; it differs only in the number of numerical values at the output of the generator and the input to the discriminator. On the right side of Fig. 6, one can see that the last four elements (related to the 21st, 22nd, 23rd and 25 bytes of the frame) of the input vector enter a single GC model. The GC model generates a fragment of the synthetic Modbus frame. The network message fragment generated in this way is then supplemented by constant values shown on the left side of Fig. 6. Only the first byte of the message is problematic here, which is the Transaction ID of the message. This is a sequential number, so we have increased it by 1 with each subsequent message sent.

### D. STRUCTURE 4: MULTIPLE GANs FOR SELECTED BYTES APPROACH

The last developed structure is a combination of structures 2 and 3. In this case, four GAN models are used to generate the four last message bytes, i.e., the control signal for heaters HL and HR (two bytes each). On the right side of Fig. 1, one can see that four scalar input values (related to the frame's 21st, 2nd, 23rd and 24th bytes) enter four GC models. Each model generates a scalar value. The whole synthetic frame is created by appending the generated bytes to a part of the frame, which can be seen on the left side of Fig. 7. Finally, the synthetic frame is transformed from decimal to hexadecimal format and sent to PLC. This allows for both confidence about the correctness of the frame and flexibility in terms of the generated values of considered signals.

## III. EXPERIMENTAL SETUP: STRUCTURE OF PROCESS CONTROL INDUSTRIAL NETWORK USED FOR TESTS

### A. PROCESS CONTROL INDUSTRIAL NETWORK TEST BED

The structure of a process control industrial network test bed considered in our study is depicted in Fig. 8. Two Mitsubishi Electric FX5U PLCs are used. The controlled process is connected directly to one of the PLCs, serving as an interface, while the other one contains the actual implementation of control algorithms. Two PLCs communicate using the Modbus TCP protocol. Two Human Machine Interface LCD panels are connected for on-site process visualization. Additionally, the Adroit SCADA system gathers, manages and visualizes information obtained from all PLCs included in this industrial network.

The thermal process shown in Fig. 9 consists of two heaters, four fans, and five thermal sensors. In this setup, two control algorithms are implemented on the PLC. The first one is a PID controller used to control the temperature measured by the Temperature Left (TL) sensor by manipulating the power of Heater Left (HL); the FLU fan makes the controlled process equally fast to heat up and cool down. The second control algorithm is a DMC controller, which controls the temperature measured by the Temperature Right (TR) sensor by manipulating the power of Heater Right (HR); the FRU fan provides a constant airflow. A physical barrier is placed between those two sets of elements to reduce the cross-influence of airflow between those two control loops. It is important to notice that the flow rates of FLU and FRU fans are constant throughout all experiments. The considered process is quite difficult to control since the two described control loops are not separated, i.e., there still exists a coupling between two control loops via the aluminum plate that holds all elements mentioned above.

During the experiments, it is assumed that the attacker obtains access to the communication between two PLCs used to connect the thermal process to the SCADA system. The Modbus TCP communication sends and receives four main messages: Read Input Registers request and reply, and Write Holding Registers request and reply. We have decided that the only forged message type is the Write Holding Registers request, as it does not require low-level TCP message injections that could be impossible to achieve because of the network structure. Other Modbus messages sent in this setup include Read Input Registers and responses to those mentioned above. Only the selected Write Holding Registers type of message allowed for independency on the current state of communication and allowed to influence the actual state of the control loop.

In addition, sending other types of messages to the network defeats the purpose of the attack, as sending replies and read

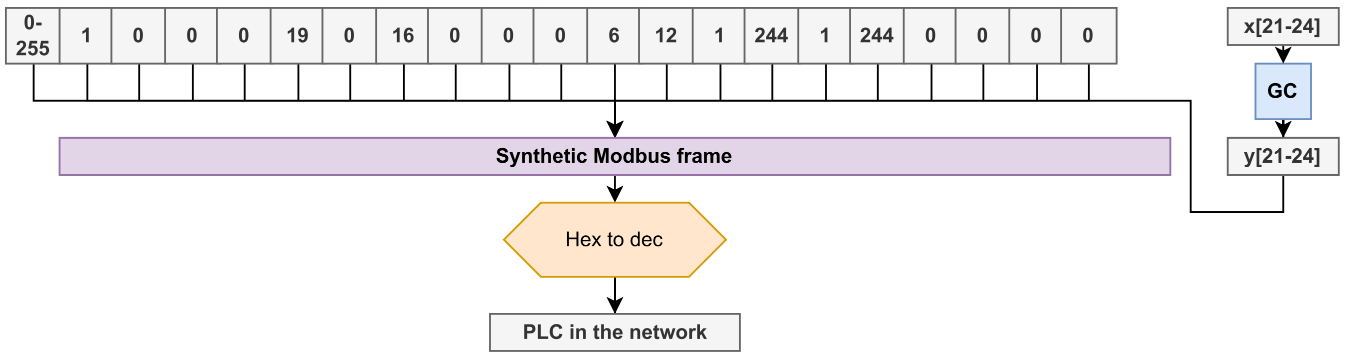


FIGURE 6. Single-GAN model for selected bytes model structure.

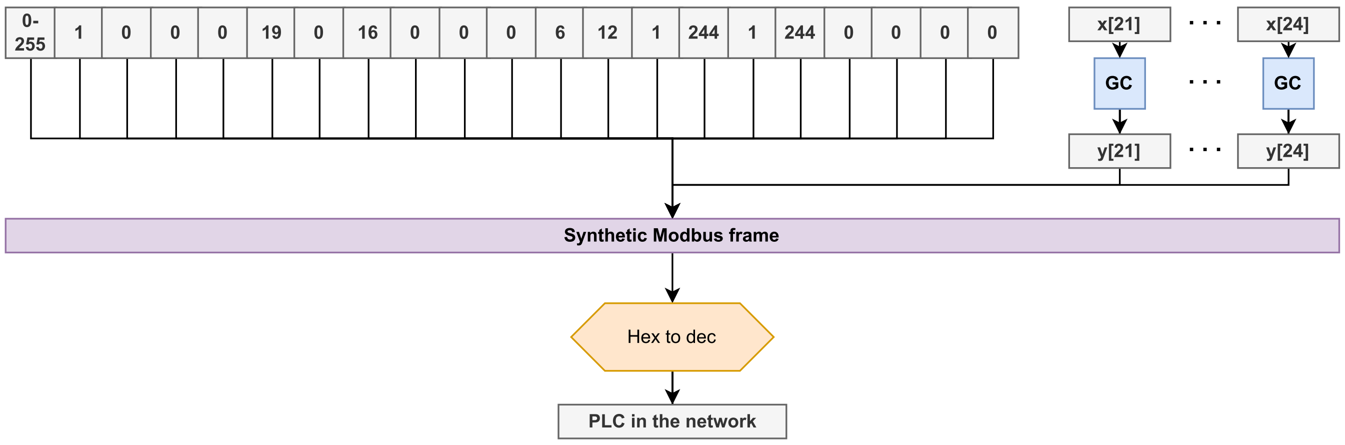


FIGURE 7. Multiple GAN models for selected bytes model structure.

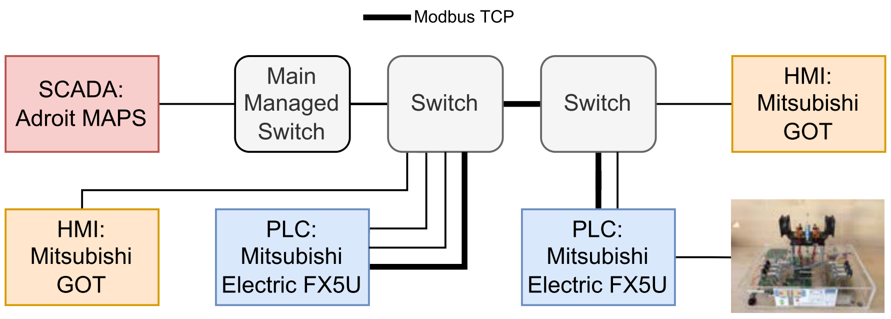


FIGURE 8. Structure of the subset of the considered experimental setup, including protocols used by each device.

requests will not result in changes to the control system. This injection can influence the control loops so that control quality is reduced or even the controlled process is destabilized.

Fig. 10 shows the considered example messages. In those messages, the range of data changes is relatively low. Specifically, out of 25 bytes of the message, 20 bytes are constant. The first byte represents the transaction identifier and increments for each message type separately; thus, its overall spread is uniform. The last four bytes represent two heaters' heating power (in percentiles) (stored as words). Considering that during the nominal operation of this setup, only values of the manipulated variables are changing, i.e., the power of heater HL and HR, only the last 4 bytes and the transaction identifier byte are not constant. When there is no interference, the manipulated variables (MVs), controlled process variables (PVs) and setpoint values (SPs) are as shown in Fig. 11. A SCADA system has been designed to collect and archive data received from the laboratory stand to verify the effectiveness of the attacks. The system has been created using SCADA MAPS software distributed by Mitsubishi Electric.

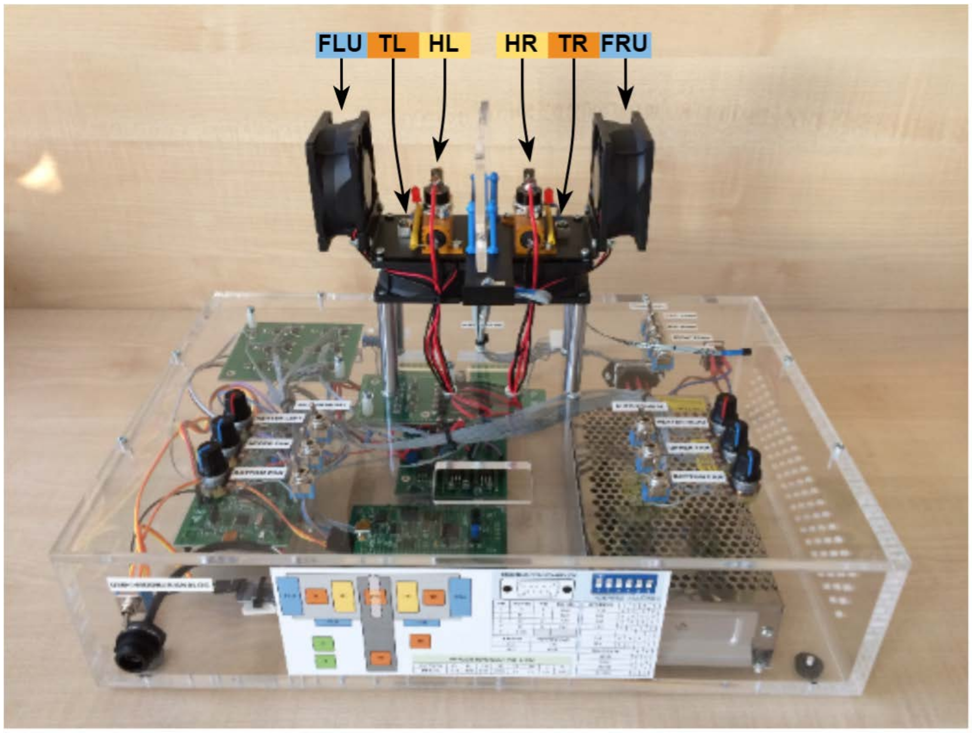
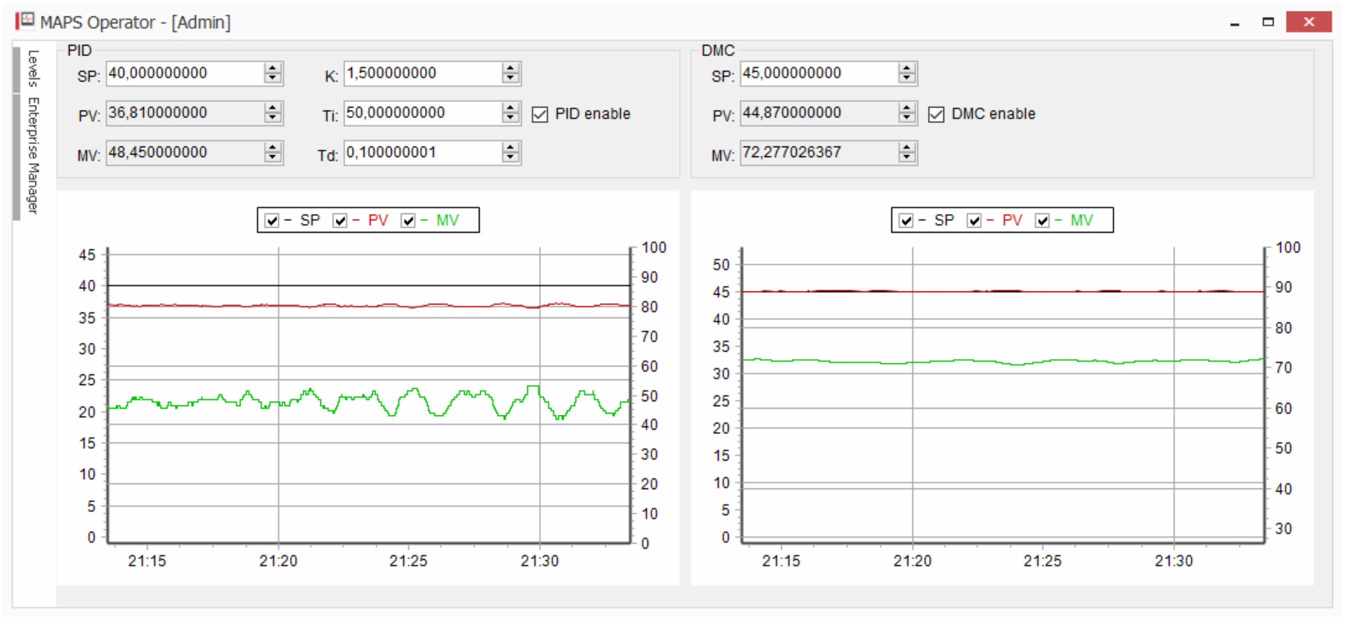


**FIGURE 9.** Benchmark thermal process used in the experimental setup.

```
 78 1 0 0 0 19 0 16 0 0 0 6 12 1 244 1 244 0 0 0 0 1 226 1 94
128 1 0 0 0 19 0 16 0 0 0 6 12 1 244 1 244 0 0 0 0 1 245 1 111
102 1 0 0 0 19 0 16 0 0 0 6 12 1 244 1 244 0 0 0 0 1 236 1 99
236 1 0 0 0 19 0 16 0 0 0 6 12 1 244 1 244 0 0 0 0 1 163 1 198
```

**FIGURE 10.** The considered messages.


MAPS Operator - [Admin]

PID
SP: 40,000000000
PV: 36,810000000
MV: 48,450000000
K: 1,500000000
Ti: 50,000000000
Td: 0,100000001
☑ PID enable

DMC
SP: 45,000000000
PV: 44,870000000
MV: 72,277026367
☑ DMC enable

**FIGURE 11.** The nominal operating point of the controlled process, as seen on the SCADA screen.

### B. IMPLEMENTATION AND TRAINING OF NEURAL STRUCTURES

GAN models have been trained using the Python language with the PyTorch toolbox. A PC equipped with an Nvidia Tesla T4 graphics card, 16 GB RAM and an Intel Xenon 2.20 GHz processor has been used. Various optimization algorithms have been tested, such as Adam, Root Mean Square Propagation (RMSProp) and Stochastic Gradient Descent. The best results have been achieved when Adam and RMSProp have been chosen to train generators and discriminators, respectively. Binary Cross Entropy has been chosen as the cost function. The learning rate (LR) for optimization in both algorithms has been chosen as equal to 0.0001. Larger LR values cause the discriminator to learn too quickly and outperform the generator. Lower values require significantly higher numbers of training epochs. An approach in which LRs of both networks are not equal has also been tested; however, it proved unsuccessful. We obtain the following results:

a) When the generator's LR exceeds the discriminator's LR, the discriminator cannot consistently detect synthetic data.
b) When the generator's LR is lower than the discriminator's LR, the generator cannot deceive the discriminator.

Having chosen the optimal LRs values, the appropriate number of training epochs has been selected. For the GANs with GC type of generator, we have trained the model for 500 epochs. On the other hand, the much more shallow GANs with GS generators have been trained only for 100 epochs. In both cases, increasing those numbers of epochs has not proven to result in more accurate models.

## IV. RESULTS OF EXPERIMENTS

Each considered GAN-based structure has generated as many as 10000 random messages and all approaches' efficiency has been evaluated. To do that, we have conducted experiments in which we introduced artificial messages generated using the approaches discussed in this Section. Messages have been injected into the traffic via a custom Python script from the newly connected PC; we assume that we have access to the network, though obtaining this access is case-specific. It is worth noting that no connection is broken or removed from the network. Selected example results of experiments for the tested approaches are described in Sections IV-A-IV-D while Section IV-E contains a summary and a comparison of obtained results.

### A. EFFICIENCY OF STRUCTURE 1 (SINGLE-GAN APPROACH)

In a single-GAN approach, some problems may be anticipated due to the randomness of the generated data, so there is a risk that some messages would be rejected by the controller for formal reasons, i.e., the message would not comply with the Modbus protocol definition. However, in practice, the FX5 Modbus implementation accepts some incorrect messages without raising any errors. Therefore, the single-GAN approach can still be useful, despite its inaccuracy.

The results of an example attack in the considered process control network described in Section III are shown in Fig. 12. The attack starts at 22:42. The attack causes a shift in the process operating point. Due to the high inertia of the controlled process, the control system does not destabilize, but only the mentioned operating point shifts. At the same time, it can be seen that both PID and DMC control algorithms react with reasonable changes in the manipulated signals to compensate for the shift in the operating point. Note that such operation of control algorithms makes detecting attack attempts more difficult.

### B. EFFICIENCY OF STRUCTURE 2 (ONE-GAN-PER-ONE-BYTE APPROACH)

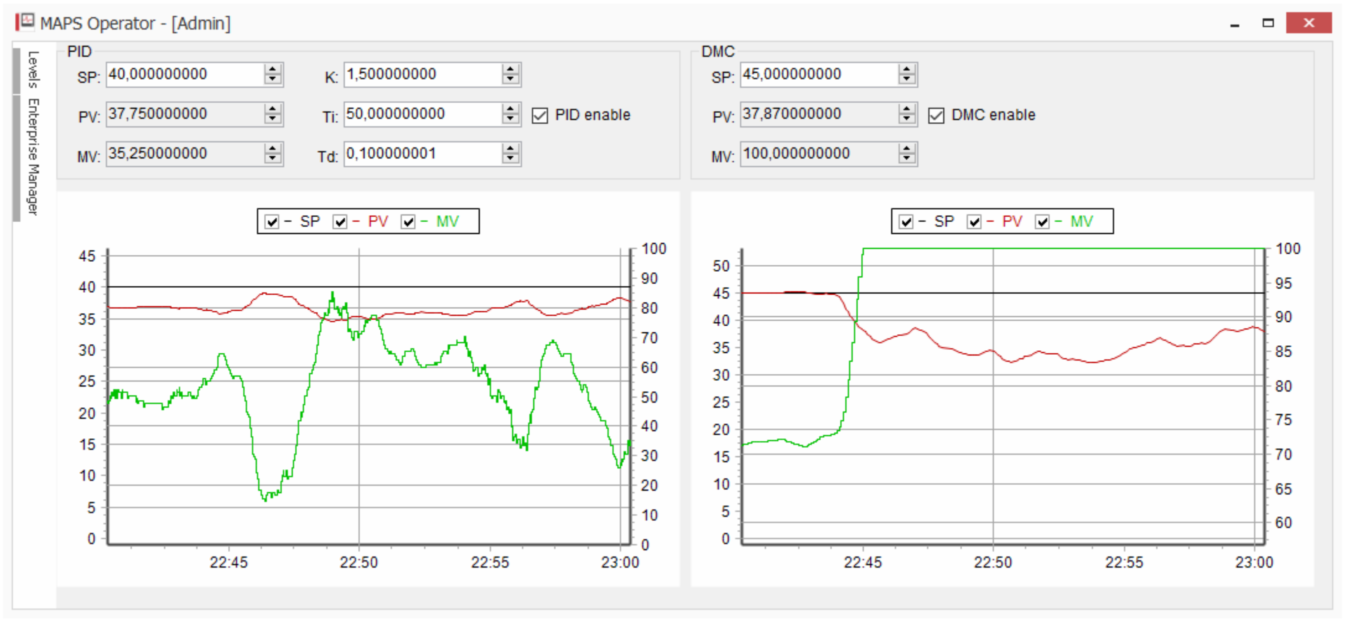
The second tested structure involves using the one-GAN-per-one-byte approach to generate Modbus messages. From the point of view of carrying out the attack, the scenario is identical to the previous experiment; only the model is changed. The messages are again filtered before being sent to test the ability to generate malicious data rather than testing the quality of the network for the generation of formally correct data, which can be improved by increasing the quality of training.

Fig. 13 shows an example results of the attack. The attack starts around 21:32. Similarly to the results presented in Fig. 12, we can see a shift in the operating point of the processes other than expected. However, there is no destabilization of the control system that could potentially damage the process. It is worth noting that compared to the first approach, this yields a higher rate of formally correct and is thus accepted by the PLC, Modbus messages, as described in Section IV-E.

### C. EFFICIENCY OF STRUCTURE 3 (SINGLE-GAN MODEL FOR SELECTED BYTES APPROACH)

Next, we consider the single-GAN for selected bytes approach. The network generates only selected Modbus message elements in this structure, while the rest are created directly based on the Modbus protocol. The attack has been carried out similarly to the previously described attacks. The difference is in the modification of the way the messages are determined by swapping the model used.

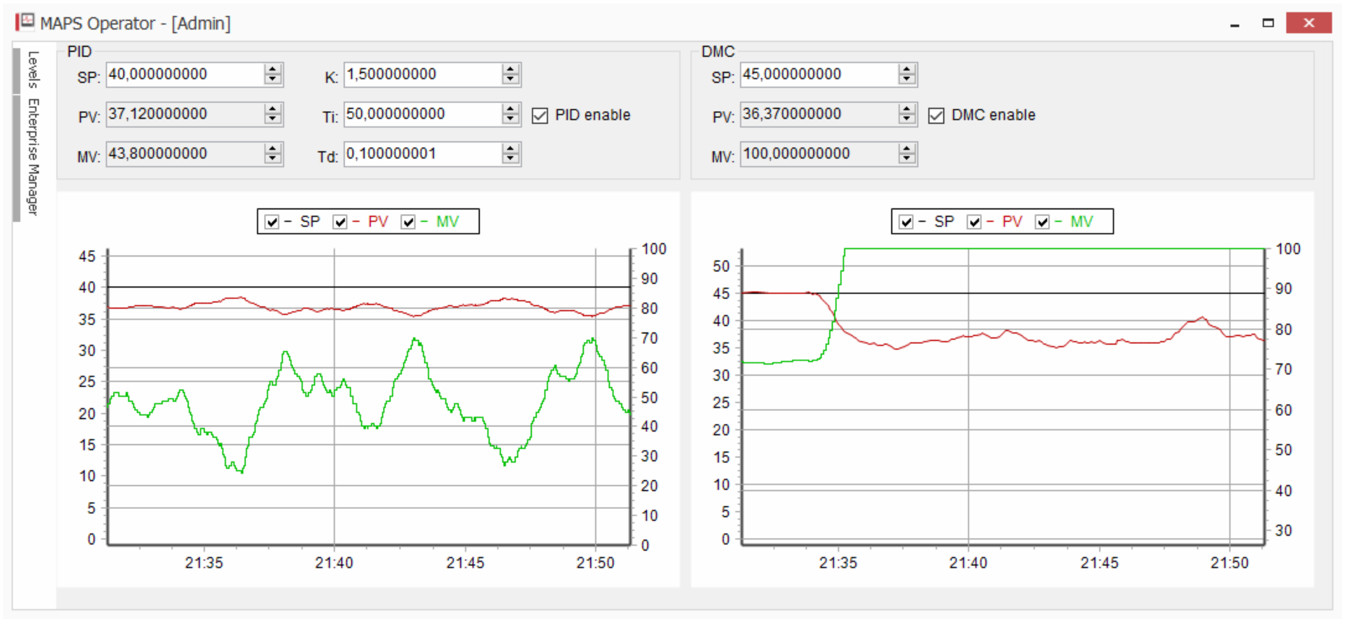
Example results of the experiment are presented in Fig. 14. The attack starts around 22.11. As with the previous attacks, a shift in the operating point can be seen, once again, without any clear system destabilization. However, it is worth noting that, compared to previous approaches, the PID controller has lost its ability to stabilize. At the same time, the DMC controller generates a constant maximum manipulated variable value. In this approach, the Modbus protocol message is mostly constant (in the case of *Transaction ID*, it is continuously and predictably incremented). At the same time, only the last 4 bytes of the frame are randomized. The last 4 bytes



PID
SP: 40,000000000
PV: 37,750000000
MV: 35,250000000
K: 1,500000000
Ti: 50,000000000
Td: 0,100000001
☑ PID enable

DMC
SP: 45,000000000
PV: 37,870000000
MV: 100,000000000
☑ DMC enable

**FIGURE 12.** **Efficiency of structure 1 (a single-GAN model): the reaction of the controlled process to an attack.**



PID
SP: 40,000000000
PV: 37,120000000
MV: 43,800000000
K: 1,500000000
Ti: 50,000000000
Td: 0,100000001
☑ PID enable

DMC
SP: 45,000000000
PV: 36,370000000
MV: 100,000000000
☑ DMC enable

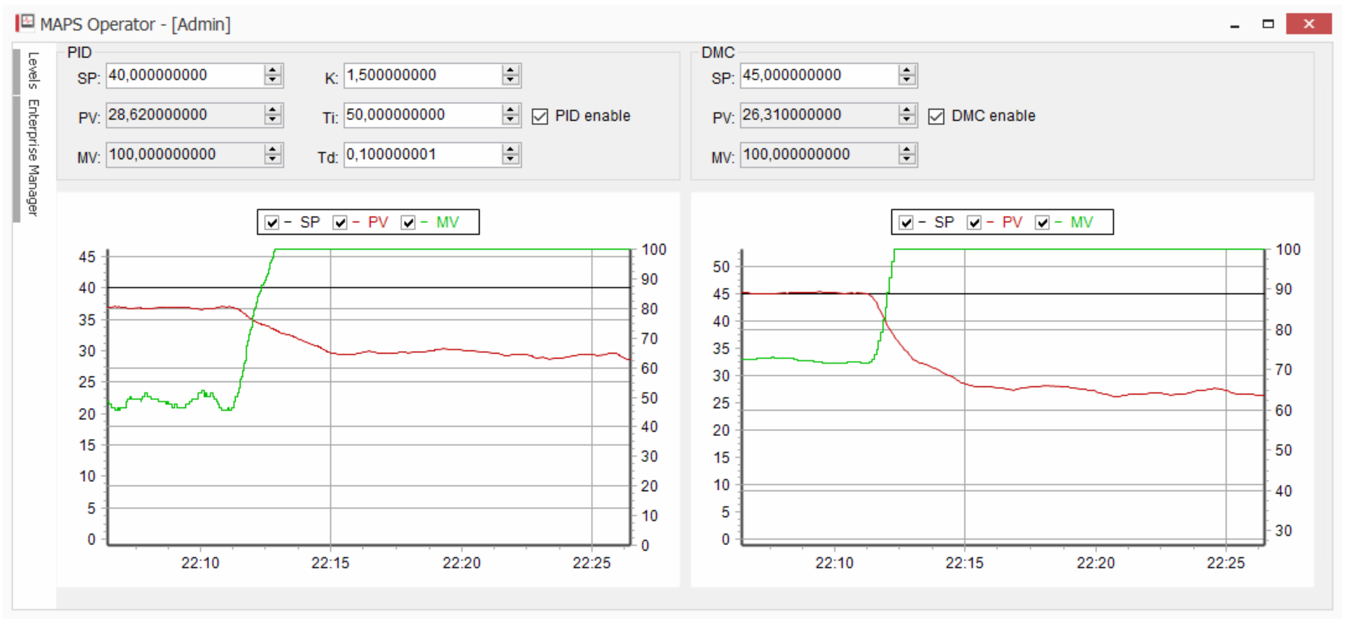
**FIGURE 13.** **Efficiency of structure 2 (one-GAN-per-one-byte model): the reaction of the controlled process to an attack.**

indicate the control of heater HL (the first two bytes listed) and HR (the last two listed), affecting the individual control loops. However, it is important that the values of the other fields of the message are not changed, which translates into its error-free formal form.

### *D. EFFICIENCY OF STRUCTURE 4 (MULTIPLE GAN MODELS FOR SELECTED BYTES APPROACH)*

The last described experiment utilizes a set of neural networks to generate each previously selected bytes of the Modbus message. The selected bytes include two bytes for the control signal of the heater HL and two bytes for the control signal of the heater HR. Therefore, as in the third structure, there is no possibility that the created message is faulty, i.e., it does not comply with the specification of Modbus protocol.

Example results are shown in Fig. 15. They are similar to the ones obtained in the case of the first and the second GAN approaches, i.e., there is a significant change in the operating point for the control loop with the DMC controller. In contrast, the PID control loop is visibly disturbed but not


MAPS Operator - [Admin]

PID

SP: 40,000000000
PV: 28,620000000
MV: 100,000000000
K: 1,500000000
Ti: 50,000000000
Td: 0,100000001
☑ PID enable

DMC

SP: 45,000000000
PV: 26,310000000
MV: 100,000000000
☑ DMC enable

**FIGURE 14.** **Efficiency of structure 3 (a single-GAN model for selected bytes): the reaction of the controlled process to an attack.**

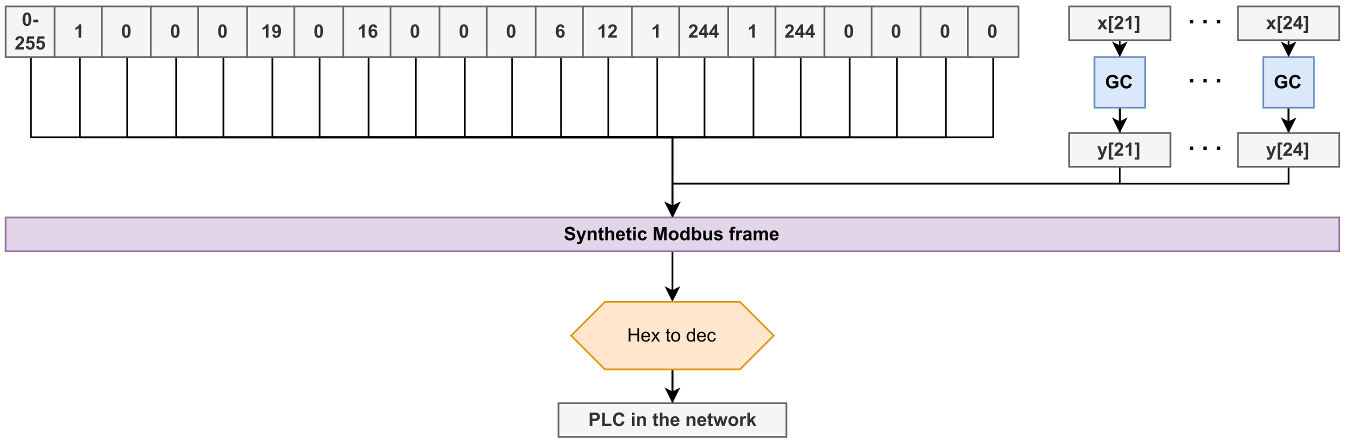


**FIGURE 15.** **Efficiency of structure 4 (multiple GAN models for selected bytes): the reaction of the controlled process to an attack.**

so much it loses control over the process. In fact, the PID controller loses any ability to control the process. The PLC is flooded with Modbus messages generated by the neural network, rendering messages from the PID controller impossible to affect the process.

### *E. COMPARISON OF TESTED APPROACHES*

The first two GAN approaches, i.e., structures 1 and 2, generate a full Modbus protocol frame. The last two approaches, i.e., structures 3 and 4, combine a Modbus frame defined explicitly according to the protocol specification, where selected fields take random GAN-generated values. While the first two approaches could potentially generate invalid data, such as by drawing an invalid value for the *Protocol ID* field or the length of the data being transmitted inside, this is not the case with the models for selected bytes, as the range of values that can be sent to the process actuators is arbitrary (provided it is within two bytes).

In order to test the ability of each GAN approach to generate correct Modbus messages, numerous experiments have been conducted. Random messages have been generated using each GAN approach and it has been tested how many messages would be rejected. As a result of the experiments in which 10000 random messages have been sent, the single-GAN approach generates 4453 incorrect messages, the one-GAN-per-byte approach generates 1592 incorrect messages, while the other two approaches, i.e., a single-GAN model for selected bytes and multiple GAN models for selected bytes, have not generated any incorrect messages. Hence, the efficiency of the single-GAN approach is the

**TABLE 1.** Results for GAN configurations considered.

| GAN configuration | Efficiency |
|---|---|
| Single-GAN model | 55.47% |
| One-GAN-per-one-byte model | 84.08% |
| Single-GAN model for selected bytes | 100% |
| Multiple GAN models for selected bytes | 100% |

worst one (55.47%), the one-GAN-per-byte structure gives better results (84.08%). In contrast, a single-GAN model for selected bytes and multiple GAN models for selected bytes give 100% efficiency. The efficiency of the obtained results for different GAN configurations is compared in Table 1. It should be noted, however, that the messages in the last two approaches under consideration must explicitly take into account the limitations and definition rules of the Modbus protocol.

The attacks conducted by generating the whole Modbus message (Fig. 12 and 13), are characterized by chaotic signals, whereas when only a part of this message is generated (Fig. 14 and 15), the new operating point could be considered stable. This results from including information about the value of the control signals that drive fans FLU and FRU in the first two approaches while setting those as constant in the later attempts. Even though neural networks are trained to generate constant values of FLU and FRU signals (as those have been constant in the training and validation data), these models struggle to generate constant, non-zero values in the fields of the messages they are expected to. By modifying the higher byte of the control signal word, even by 1, the fan significantly changes the system's airflow, thus changing its cooling rate. In the approaches where only the selected set of bytes is generated, the airflow is always constant. Thus, the system's operating point is changed to the one used while collecting training and validation data.

It has to be underlined that the presented and described process behavior is similar across multiple experiments for each model configuration.

## V. CONCLUSION

Three approaches employing GANs to generate fake Modbus frames have been proposed in this work, tested in the example industrial process control network and compared with the classical approach that is known from the literature. The classical single-GAN approach is the worst one (it has 55.47% efficiency). The one-GAN-per-byte approach generates significantly more correct message frames than the classical method (it has 84.08% efficiency). Moreover, all the generated fake frames have been correct in single-GAN model for selected bytes and multiple GAN models for selected bytes methods (they have 100% efficiency). However, in these two approaches, only a part of the frame is generated by either one or multiple GANs. The remaining part of the frame is generated using expert knowledge about the Modbus protocol.

The classical GAN approach is chosen to test machine learning abilities regarding modeling communication protocols. Due to the generative character of this structure, it should learn the definition of the protocol based on a large number of data provided during the training phase. The quality of generated messages should be similar regardless of the communication protocol. Unfortunately, many of the generated frames are incorrect. Motivated by the poor performance of the classical approach, we designed three GAN-based structures. In particular, two human-made approaches give 100% efficiency. It is important to stress that these structures use GANs to generate parts of the frames where the data change, i.e., containing values of the manipulated and controlled variables which occur in control systems. We take advantage of GANs because they are able to generate data that are similar to real data but are likely to cause a malfunction of the control system and are intentionally introduced during the cyber-attack. Therefore, our GAN-based approach makes it possible to test the vulnerability of the network control system.

Interestingly, the correct, fake frames do not cause unstable control system operation during the tests. In the case of both PID and DMC controllers, they manage to work reasonably. However, the operating point shifts, sometimes significantly. It means that the operation of the control system is not optimal and the attack can cause economic losses. Moreover, as the control system still works, it is more difficult to detect the attack attempt than in the case when the operation of the control system collapses.

## DATA AVAILABILITY

The datasets generated during and/or analyzed during the current study are available from the corresponding author upon reasonable request.

## CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

## REFERENCES

[1] E. D. Knapp and J. T. Langill, *Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems*. Amsterdam, The Netherlands: Elsevier, 2015.

[2] Y. Zhang, Z. Sun, L. Yang, Z. Li, Q. Zeng, Y. He, and X. Zhang, "All your PLCs belong to me: ICS ransomware is realistic," in *Proc. IEEE 19th Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom)*, Guangzhou, China, Dec. 2020, pp. 502–509.

[3] L. Rosa, T. Cruz, P. Simoes, E. Monteiro, and L. Lev, "Attacking SCADA systems: A practical perspective," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, Lisbon, Portugal, May 2017, pp. 741–746.

[4] R. Chen, X. Li, H. Zhong, and M. Fei, "A novel online detection method of data injection attack against dynamic state estimation in smart grid," *Neurocomputing*, vol. 344, pp. 73–81, Jun. 2019.

[5] H. Li, J. Zhang, and X. He, "Design of data-injection attacks for cyber-physical systems based on Kullback–Leibler divergence," *Neurocomputing*, vol. 361, pp. 77–84, Oct. 2019.

[6] Q. Xiong and S. Zhang, "Fault injection and diagnosis for civil aircraft cabin pressure control system," in *Proc. Prognostics Syst. Health Manage. Conf. (PHM-Chongqing)*, Chongqing, China, Oct. 2018, pp. 314–318.

[7] D. Orive, N. Iriondo, A. Burgos, I. Saráchaga, M. L. Álvarez, and M. Marcos, "Fault injection in digital twin as a means to test the response to process faults at virtual commissioning," in *Proc. 24th IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Zaragoza, Spain, Sep. 2019, pp. 1230–1234.

[8] M. Cho, H. Jin, D. An, and T. Kwon, "Evaluating code coverage for kernel fuzzers via function call graph," *IEEE Access*, vol. 9, pp. 157267–157277, 2021.

[9] L. H. Park, S. Chung, J. Kim, and T. Kwon, "GradFuzz: Fuzzing deep neural networks with gradient vector coverage for adversarial examples," *Neurocomputing*, vol. 522, pp. 165–180, Feb. 2023.

[10] A. Ye, L. Wang, L. Zhao, J. Ke, W. Wang, and Q. Liu, "RapidFuzz: Accelerating fuzzing via generative adversarial networks," *Neurocomputing*, vol. 460, pp. 195–204, Oct. 2021.

[11] Y. Yu, Z. Chen, S. Gan, and X. Wang, "SGPFuzzer: A state-driven smart graybox protocol fuzzer for network protocol implementations," *IEEE Access*, vol. 8, pp. 198668–198678, 2020.

[12] A. G. Voyiatzis, K. Katsigiannis, and S. Koubias, "A modbus/TCP fuzzer for testing internetworked industrial systems," in *Proc. IEEE 20th Conf. Emerg. Technol. Factory Autom. (ETFA)*, Luxembourg City, Luxembourg, Sep. 2015, pp. 1–6.

[13] K. O. Akpinar and I. Ozcelik, "A standalone gray-box EtherCAT fuzzer," in *Proc. 2nd Int. Symp. Multidisciplinary Stud. Innov. Technol. (ISMSIT)*, Ankara, Turkey, Oct. 2018, pp. 1–4.

[14] B. Wu, L. Yun, X. Jin, B. Liu, and G. Wei, "Study on the fuzzing test method for industrial supervisory control configuration software based on genetic algorithm," in *Proc. 11th Int. Conf. Rel., Maintainability Saf. (ICRMS)*, Hangzhou, China, Oct. 2016, pp. 1–6.

[15] K. Katsigiannis and D. Serpanos, "MTF-storm: A high performance fuzzer for modbus/TCP," in *Proc. IEEE 23rd Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Turin, Italy, vol. 1, Sep. 2018, pp. 926–931.

[16] S. Kim, W. Jo, and T. Shon, "A novel vulnerability analysis approach to generate fuzzing test case in industrial control systems," in *Proc. IEEE Inf. Technol., Netw., Electron. Autom. Control Conf.*, Chongqing, China, May 2016, pp. 566–570.

[17] Y. Mubai and F. Jingqi, "Improved modbus/TCP multi-dimensional fuzzing test method," in *Proc. Chin. Control Decis. Conf. (CCDC)*, Nanchang, China, Jun. 2019, pp. 3233–3237.

[18] C. C. Aggarwal, *Neural Networks and Deep Learning: A Textbook*. Cham, Switzerland: Springer, 2018.

[19] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Oxford Univ. Press, 1995.

[20] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. Jesúus, *Neural Network Design*, 2014. [Online]. Available: https://hagan.okstate.edu/NNDesign.pdf

[21] S. Haykin, *Neural Networks and Learning Machines*. Upper Saddle River, NJ, USA: Pearson, 2009.

[22] M. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015. [Online]. Available: http://neuralnetworksanddeeplearning.com/

[23] Y. Lai, H. Gao, and J. Liu, "Vulnerability mining method for the modbus TCP using an anti-sample fuzzer," *Sensors*, vol. 20, no. 7, p. 2040, Apr. 2020.

[24] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[25] A. Chen, H. Xing, and F. Wang, "A facial expression recognition method using deep convolutional neural networks based on edge computing," *IEEE Access*, vol. 8, pp. 49741–49751, 2020.

[26] M. Mazzone and A. Elgammal, "Art, creativity, and the potential of artificial intelligence," *Arts*, vol. 8, no. 1, p. 26, Feb. 2019.

[27] G. Antipov, M. Baccouche, and J.-L. Dugelay, "Face aging with conditional generative adversarial networks," 2017, pp. 2089–2093, *arXiv:1702.01983*.

[28] B. Xu, D. Zhou, and W. Li, "Image enhancement algorithm based on GAN neural network," *IEEE Access*, vol. 10, pp. 36766–36777, 2022.

[29] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. C. Loy, "ESRGAN: Enhanced super-resolution generative adversarial networks," in *Proc. Comput. Vis.-ECCV Workshops*, Munich, Germany, 2018, pp. 63–79.

[30] H. Zheng, X. Li, Y. Li, Z. Yan, and T. Li, "GCN-GAN: Integrating graph convolutional network and generative adversarial network for traffic flow prediction," *IEEE Access*, vol. 10, pp. 94051–94062, 2022.

[31] Y. Cai, X. Wang, Z. Yu, F. Li, P. Xu, Y. Li, and L. Li, "Dualattn-GAN: Text to image synthesis with dual attentional generative adversarial network," *IEEE Access*, vol. 7, pp. 183706–183716, 2019.

[32] Y. Yu, A. Srivastava, and S. Canales, "Conditional LSTM-GAN for melody generation from lyrics," 2019, *arXiv:1908.05551*.

[33] J. S. Wijnands, K. A. Nice, J. Thompson, H. Zhao, and M. Stevenson, "Streetscape augmentation using generative adversarial networks: Insights related to health and wellbeing," *Sustain. Cities Soc.*, vol. 49, Aug. 2019, Art. no. 101602.

[34] Z. Xia, P. Yi, Y. Liu, B. Jiang, W. Wang, and T. Zhu, "GENPass: A multi-source deep learning model for password guessing," *IEEE Trans. Multimedia*, vol. 22, no. 5, pp. 1323–1332, May 2020.

[35] W. Tang, S. Tan, B. Li, and J. Huang, "Automatic steganographic distortion learning using a generative adversarial network," *IEEE Signal Process. Lett.*, vol. 24, no. 10, pp. 1547–1551, Oct. 2017.

[36] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on GAN," 2017, *arXiv:1702.05983*.

[37] W. Sun, B. Zhang, J. Ding, and M. Tang, "MaskFuzzer: A MaskGAN-based industrial control protocol fuzz testing framework," in *Proc. IEEE Int. Conf. Smart Internet Things (SmartIoT)*, Suzhou, China, Aug. 2022, pp. 51–57.

[38] Z. Li, H. Zhao, J. Shi, Y. Huang, and J. Xiong, "An intelligent fuzzing data generation method based on deep adversarial learning," *IEEE Access*, vol. 7, pp. 49327–49340, 2019.

[39] Z. Yu, H. Wang, D. Wang, Z. Li, and H. Song, "CGFuzzer: A fuzzing approach based on coverage-guided generative adversarial networks for industrial IoT protocols," *IEEE Internet Things J.*, vol. 9, no. 21, pp. 21607–21619, Nov. 2022.

[40] J. Maciejowski, *Predictive Control With Constraints*. Harlow, U.K.: Prentice-Hall, 2002.

[41] P. Tatjewski, *Advanced Control of Industrial Processes, Structures and Algorithms*. London, U.K.: Springer, 2007.

**KRZYSZTOF ZARZYCKI** was born in Pruszków, Poland, in 1993. He received the B.Sc. degree in automatic control and robotics from the Faculty of Mechatronics, Warsaw University of Technology, in 2017, and the M.Sc. degree in automatic control and robotics from the Faculty of Electronics and Information Technology, Warsaw University of Technology, in 2020.

He has been with the Institute of Control and Computation Engineering, Warsaw University of Technology, since 2020, where he is currently an Assistant Professor. His research interests include algorithms for industrial process control, mainly advanced model predictive control (MPC) and the applications of artificial intelligence as a process control and modeling tool.

**PATRYK CHABER** was born in Warsaw, Poland, in 1990. He received the B.Sc. and M.Sc. degrees in computer science and the Ph.D. degree in automatic control and robotics from the Faculty of Electronics and Information Technology, Warsaw University of Technology, in 2012, 2014, and 2018, respectively. He is currently an Assistant Professor with the Institute of Control and Computation Engineering, Warsaw University of Technology. He is the author or coauthor of more than 15 publications, including journal articles, conference papers, and reports. His research interests include advanced control algorithms, particularly MPC algorithms, microcontrollers, real-time process control, and modeling.

**KRZYSZTOF CABAJ** received the M.Sc., Ph.D., and D.Sc. (Habilitation) degrees in computer science from the Faculty of Electronics and Information Technology, Warsaw University of Technology (WUT), in 2004, 2009, and 2019, respectively. He is currently a Professor with WUT. He is a Former Instructor of Cisco certificated academy courses, such as CCNA routing and switching, CCNA security, and CCNP with the International Telecommunication Union Internet Training Centre (ITU-ITC). His research interests include network security, honeypots, dynamic malware analysis, data-mining techniques, the IoT, and industrial control systems security. He is the author or coauthor of over 80 publications and a supervisor of more than 35 B.Sc. and M.Sc. degree theses in the field of information security. He participated in over a dozen research projects, among others, for EU, ESA, Samsung, U.S. Army, and U.S. Air Force. He is the Co-Leader of the Computer Systems Security Group, Institute of Computer Science.

**MACIEJ ŁAWRYŃCZUK** was born in Warsaw, Poland, in 1972. He received the M.Sc., Ph.D., and D.Sc. degrees in automatic control from the Faculty of Electronics and Information Technology, Warsaw University of Technology, in 1998, 2003, and 2013, respectively.

He has been with the Institute of Control and Computation Engineering, Warsaw University of Technology, since 2003. He was a Teaching Assistant, from 2003 to 2004, an Assistant Professor, from 2004 to 2015, an Associate Professor, from 2015 to 2022, and a Full Professor, since 2022. He is the author or coauthor of seven books and more than 190 other publications, including more than 70 journal articles. His research interests include advanced control algorithms, particularly model predictive control (MPC) algorithms, setpoint optimization algorithms, soft computing methods, particularly neural networks, modeling, and simulation.

**PIOTR MARUSAK** was born in Warsaw, Poland, in 1974. He received the M.Sc. degree in computer science and the Ph.D. and D.Sc. degrees in automatic control from the Faculty of Electronics and Information Technology, Warsaw University of Technology (WUT), in 1997, 2003, and 2020, respectively.

He has been with the Institute of Control and Computation Engineering, WUT, since 2002, where he is currently an Assistant Professor. He is the author or coauthor of more than 120 publications. His research interests include nonlinear control, fuzzy modeling and control, fault-tolerant control, advanced process control, model predictive control (MPC) algorithms, and the cooperation of setpoint optimization with MPC algorithms.

**ROBERT NEBELUK** was born in Warsaw, Poland, in 1992. He received the B.Sc. degree in automatic control and robotics from the Faculty of Production Engineering, Warsaw University of Technology, in 2015, and the M.Sc. degree in automatic control and robotics from the Faculty of Electronics and Information Technology, Warsaw University of Technology, in 2019. He is currently an Assistant Professor with the Institute of Control and Computation Engineering, Warsaw University of Technology. His research interests include advanced algorithms for industrial process control, particularly model predictive control algorithms, nonlinear control, and modeling.

**SEBASTIAN PLAMOWSKI** was born in Milanówek, Poland, in 1976. He received the M.Sc. and Ph.D. degrees in automatic control from the Faculty of Electronics and Information Technology, Warsaw University of Technology, in 2000 and 2006, respectively. Since 2015, he has been with the Institute of Control and Computational Engineering, Warsaw University of Technology, where he is currently an Assistant Professor. At the same time, he has been working in the industry for leading automation companies, since 2000, and he has been with the Research and Development Department, Emerson, since 2011. His research interests include process diagnostics, control quality, DCS and SCADA systems, programming in embedded environments, cyber security, control algorithms, fuzzy modeling and model predictive control (MPC) algorithms, and process optimization.

**ANDRZEJ WOJTULEWICZ** was born in Pruszków, Poland, in 1988. He received the degree from the Faculty of Electronics and Information Technology, Warsaw University of Technology, and the Ph.D. degree from the Institute of Automation and Applied Informatics, 2020. His research interests include FPGA-based MPC control systems and industrial automation system's design and implementation. In addition to his college and career interests, he is interested in RC modeling and sound visualization.