Networked Automation Systems: a new cryptographic scheme

Públio M. Lima [*] **Lilian K. Carvalho** [*] **Marcos V. Moreira** [*]

[*] *COPPE - Electrical Engineering Program, Universidade Federal do Rio de Janeiro, Cidade Universitária, Ilha do Fundão, Rio de Janeiro, 21.945-970, RJ, Brazil,*
Emails: publio@poli.ufrj.br, lilian.carvalho@poli.ufrj.br, moreira.mv@poli.ufrj.br

Abstract: One of the main concerns about implementing networked automation systems is ensuring its security against cyber attacks. In this paper, we consider networked automation systems abstracted as Discrete-Event Systems (DES), and consider cyber attacks where a malicious agent eavesdrops a network communication channel with the objective to gather information about the system behavior. It is important to remark that network security strategies used in Information Technology (IT) cannot be straightforwardly used in industrial networks, since the control of automation systems usually requires small communication delays, which limits the size of the data that can be transmitted in the communication network. In this paper, we introduce a new cryptographic scheme based on events, called event-based cryptography, where an event is defined as any change in the binary signals transmitted in the channel. We also present a necessary and sufficient condition that the event-based encryption function must satisfy to be used in the cryptographic scheme proposed in this paper, and present a class of encryption functions that can be modeled by Mealy automata. We also present procedures for the implementation of the event-based encryption function, and illustrate all results with a practical example.

Keywords: Cryptography, Security, Discrete-Event Systems, Automation.

1. INTRODUCTION

Modern engineering solutions consider the implementation of networked automation systems, which consist of systems that integrate computing and communication capabilities to monitor and control physical processes. Since modern automation systems use network communications, then one of the main concerns about using this type of system is ensuring its security against cyber attacks (Gou et al., 2013; Singh and Singh, 2015; He et al., 2016; Da Xu et al., 2014). In this paper, we propose a method to ensure the confidentiality of the transmitted data in communication networks of automation systems abstracted as Discrete Event Systems (DESs), *i.e.*, only the sender and the intended receiver must be able to understand the transmitted data in the network (Stallings, 2006).

Several works in the literature propose strategies to model and thwart cyber attacks in the context of DESs (Thorsley and Teneketzis, 2006; Goes et al., 2017; Carvalho et al., 2018; Zhang et al., 2018; Lima et al., 2018; Su, 2018; Lima et al., 2019, 2021), and address the problem of ensuring security, which can be defined as the prevention of damages caused to the system by an attacker that is capable of altering resources of the system, *e.g.*, altering sensor readings or enabling actuators. Another way of protecting the system from cyber attacks in the context of DESs is to ensure that the attacker is not capable of estimating, from the observed data transmitted in the communication channel, the sequence of events executed by the system. The problem of a malicious observer eavesdropping the communication has been mainly addressed in the DES community as the problem of opacity (Lin, 2011; Yin and Lafortune, 2015; Jacob et al., 2016; Tong et al., 2018; Barcelos and Basilio, 2018; Lafortune et al., 2018).

A more general way of protecting data in communication channels is to use cryptography (Stallings, 2006; Kurose and Ross, 2011; Fritz et al., 2019). In Fritz et al. (2019), the authors propose the use of cryptography in a networked automation system composed of a plant and a controller, modeled as DES. A controller encryption scheme is proposed to secure the communication and the information inside the controller. Since, the encryption method presented in Fritz et al. (2019) is based on operations with large prime numbers, then the size of the transmitted data increases, which can increase the delay of transmission. This can compromise the use of this encryption scheme, since the control of industrial systems usually requires small communication delays.

More recently in Lima et al. (2020), the property of confidentiality of Discrete Event Systems is introduced, where the authors propose the use of an event-based encryption function to ensure that an attacker is unable to

⋆ This work has been partially supported by the National Council for Scientific and Technological Development - Brasil (CNPq) - under grants 305267/2018-3, 431307/2018-0, and 436672/2018-9, FAPERJ, and the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

correctly estimate that a secret sequence has been executed by the system. A method of verification of this property is also proposed.

In this paper, we propose the implementation of an encryption scheme that does not alter the transmission data size or structure. In order to do so, we define events associated with changes in the binary signals transmitted in the communication channel, and present a cryptography method based on these events, called event-based cryptography. To this end, we define an event encryption function and present a necessary and sufficient condition that it must satisfy to be used in the proposed cryptographic scheme. We also present a class of encryption functions that can be modeled by Mealy automata. Finally, we illustrate the implementation of the proposed encryption scheme in a practical example.

This paper is organized as follows. In Section 2, some preliminary concepts are presented. In Section 3, a defense strategy based on event-based cryptography is proposed. We also present in Section 3, a necessary and sufficient condition for the existence of an event encryption function that can be used in the proposed cryptographic scheme. In Section 4, a class of encryption functions that can be modeled by a deterministic finite-sate Mealy automaton is presented, and Encryption and Decryption Mealy automata are introduced. In Section 5, a practical example is used to illustrate the implementation of the cryptographic scheme proposed in this work. Finally, in Section 6, the conclusions are drawn.

2. PRELIMINARIES

Let $G = (X, \Sigma, f, x_0)$ be a deterministic automaton, where X is the set of states, Σ is the finite set of events, $f : X \times \Sigma \to X$ is the transition function, and $x_0 \in X$ is the initial state of the system. Let $\Gamma_G : X \to 2^{\Sigma}$ be the active event function, where $\Gamma_G(x) = \{\sigma \in \Sigma : f(x,\sigma) \text{ is defined}\}$, for all $x \in X$. The domain of the transition function f can be extended to $X \times \Sigma^\star$, where $\Sigma^\star$ denotes the Kleene-closure of Σ, as usual: $f(x,\varepsilon) = x$, and $f(x, s\sigma) = f(f(x,s),\sigma)$, for all $s \in \Sigma^\star$, and $\sigma \in \Sigma$, where ε denotes the empty sequence. The language generated by G is defined as $L(G) = \{s \in \Sigma^\star : f(x,s) \text{ is defined}\}$. Let $s \in \Sigma^\star$, then $|s|$ denotes the length of s.

A deterministic Mealy automaton is defined as $\mathcal{G} = (X, \Sigma, \Sigma_{out}, f, h, x_0)$, where (X, Σ, f, x_0) is a deterministic automaton, with Σ representing the input events, Σ_{out} is the finite set of output events, and $h : X \times \Sigma \to \Sigma_{out}$ is the output transition function (Mealy, 1955; Esmoris et al., 2005). The interpretation regarding the transitions of a Mealy automaton is as follows: when the system is in state $x \in X$, and a feasible event $\sigma \in \Sigma$ occurs, then it makes the transition to state $f(x,\sigma)$ and, in that process, emits event $h(x,\sigma) \in \Sigma_{out}$ (Cassandras and Lafortune, 2008).

A function $g : \Sigma^\star \to \tilde{\Sigma}^\star$ is said to be prefix-preserving if for any pair of sequences $s_1, s_2 \in \Sigma^\star$, whose longest common prefix is s', then their images $g(s_1), g(s_2) \in \tilde{\Sigma}^\star$ have the same prefix $g(s')$ (Wonham and Cai, 2019).

The exclusive disjunction operator, denoted by $\oplus$, is defined for two binary numbers $a, b \in \{0,1\}$ as $a \oplus b = 0$, if $a = b$, and $a \oplus b = 1$, if $a \neq b$. The exclusive disjunction can be applied to two vectors of binary numbers $\underline{x}_1$ and $\underline{x}_2$ if the number of entries of the vectors is equal. In this case, the *i-th* element of vector $\underline{x} = \underline{x}_1 \oplus \underline{x}_2$ is defined as $x^i = x_1^i \oplus x_2^i$, where x_j^i is the *i-th* element of x_j, for $j = 1, 2$.

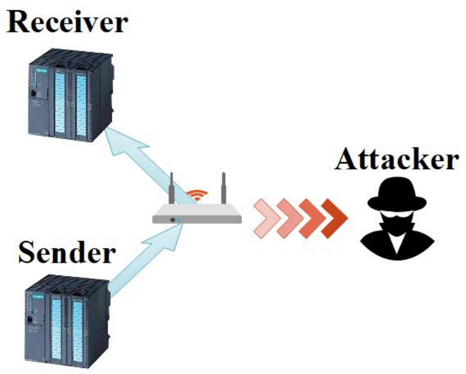

Fig. 1. Eavesdropping attack in the communication channel between sender and receiver.

The set of natural numbers is denoted by $\mathbb{N}$ and $\mathbb{Z}_1 = \{0,1\}$.

3. EVENT-BASED CRYPTOGRAPHIC SCHEME

In this paper, we address the problem of ensuring confidentiality of the transmitted data in the communication network of an automation system, *i.e.*, only the sender and the intended receiver must be able to understand the message transmitted in the network (Stallings, 2006; Kurose and Ross, 2011; Lima et al., 2020). We consider that the communication is carried out using a wired or wireless network. The sender and the receiver can represent different entities in an industrial network, *e.g.*, the sender may be an industrial plant and the receiver a controller or supervisor, or the sender may be a controlled system and the receiver a diagnoser or observer. The data transmitted from sender to receiver is a vector of binary numbers $\underline{u} = [u_1 \ u_2 \ \dots \ u_m]^T \in \mathbb{Z}_1^m$, where $m \in \mathbb{N}$ is the dimension of vector $\underline{u}$.

We consider that the communication channel between sender and receiver is vulnerable to attacks, as shown in Figure 1, where the attacker can observe the data transmitted in this channel. The attacker eavesdrops the binary vector $\underline{u}$ with the objective of estimating the system state or its dynamic behavior. In this paper, we do not consider that the attacker can modify the data transmitted in the attacked channel, *i.e.*, the attacker performs only passive attacks (Stallings, 2006).

In order to ensure confidentiality of the transmitted data, we need to use some kind of encryption. Several methods of encryption are proposed in Information Technology, where the main objective is to keep the information secret, regardless of the size of the transmitted data. However, in industrial networks, avoiding the increase in the data size is important to do not delay the transmission of the information to the receiver. In this paper, we propose an event-based cryptographic method that keeps the structure of the transmitted data, avoiding the increase in the transmission delay. In order to do so, we first define an event as any change in at least one of the entries of vector $\underline{u}$. Let $\underline{u} = [u_1 \; u_2 \; \dots \; u_m]^T$ be the current observed vector in the channel, and $\underline{u}' = [u'_1 \; u'_2 \; \dots \; u'_m]^T$ be the next observed vector. Then, an event σ is coded by vector

$$\underline{\sigma} = \underline{u} \oplus \underline{u}'.$$

Notice that the i-th entry of vector $\underline{\sigma}$, σ_i, is equal to one if there is a change from u_i to u'_i, and it is zero otherwise, *i.e.*, vector $\underline{\sigma}$ represents the changes in the values of the entries of the observed vector $\underline{u}$.

Remark 1. A similar representation of events has been proposed in Moreira and Lesage (2019) for the identification of DESs with the aim of fault detection. In the event representation proposed in Moreira and Lesage (2019), the authors consider that the change in the reading of an entry of vector $\underline{u}$ can be negative, when its value goes from 1 to 0, or positive, when its value goes from 0 to 1, distinguishing the falling and rising edges of the signal, respectively. In this paper, events are defined as changes in the entries of vector $\underline{u}$, independently if it is a rising or a falling edge of the binary signal, which allows the occurrence of two equal consecutive events. This facilitates the definition of the encryption and decryption functions proposed in this work. □

Example 1. Let us consider that three binary signals are transmitted in the communication channel, *i.e.*, $\underline{u} = [u_1 \; u_2 \; u_3]^T$, and that the following three different vectors have been observed: $\underline{u}_1 = [0 \;\; 0 \;\; 0]^T$, $\underline{u}_2 = [1 \;\; 0 \;\; 0]^T$, and $\underline{u}_3 = [0 \; 1 \; 0]^T$. Then, when the change from vector $\underline{u}_1$ to $\underline{u}_2$ is observed, an event σ_1, associated with the change of the first entry of the observed vector, is generated. This event is coded as $\underline{\sigma}_1 = \underline{u}_1 \oplus \underline{u}_2 = [1 \; 0 \; 0]^T$. If, in the sequel, there is a change to vector $\underline{u}_3$, then a new event σ_2 is generated, coded as $\underline{\sigma}_2 = \underline{u}_2 \oplus \underline{u}_3 = [1 \; 1 \; 0]^T$. It is important to remark that, since each event represents a change in at least one binary value of $\underline{u}$, without distinguishing a rising edge from a falling edge, then, if the sequence of observed vectors is $\underline{u}_1 \underline{u}_2 \underline{u}_1$, the associated sequence of events is $\sigma_1 \sigma_1$. □

In this paper, we propose the cryptographic scheme depicted in Figure 2, where each observation of an event by the sender is ciphered using an event encryption function. We denote by plain event, any event not modified by the encryption function, and by cipher event those modified by the encryption function. We also use this terminology for sequences and languages. In the cryptographic scheme depicted in Figure 2, after the observation of an event σ by the sender, a cipher event σ_c is generated, and then, transmitted to the receiver. At the receiver's site, σ_c is decrypted and the plain event σ is recovered. Notice that, in this scheme, it is assumed that an event is encrypted and transmitted to the receiver after each new event observed by the sender. This assumption is needed to avoid delays in the actions that the receiver may need to perform, such as control commands, or informing the occurrence of a fault to the system operator. Thus, in order to implement the cryptographic scheme proposed in Figure 2, there must exist an encryption and a decryption function that correctly encrypts each plain event observed by the sender, and then decrypts the corresponding cipher event.

Let Σ be the set of all observed events, and Σ_c be the set of all possible signal changes of a vector in $\mathbb{Z}_1^m$, *i.e.*, the set of all possible events generated from a vector in $\mathbb{Z}_1^m$.

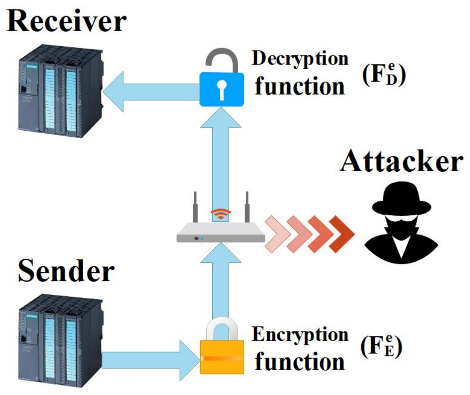

Fig. 2. Cryptographic scheme.

Then, the event encryption function $F^e_E : \Sigma^\star \rightarrow \Sigma_c \cup \{\varepsilon\}$ generates the cipher event $\sigma_c = F^e_E(s\sigma)$, corresponding to the plain event $\sigma \in \Sigma$, after the occurrence of the plain sequence $s\sigma \in \Sigma^\star$. Notice that if no event is observed by the sender, then no cipher event can be generated, which implies that $F^e_E(\varepsilon) = \varepsilon$. If $F^e_E(s\sigma) = F^e_E(s'\sigma)$ for all $s, s' \in \Sigma^\star$, then the cipher event σ_c depends only on the last event observed by the sender σ, and F^e_E is called a static encryption function. On the other hand, if σ_c depends on σ and on the sequence of events s observed by the sender before the occurrence of σ, then F^e_E is called a dynamic encryption function. It is not difficult to see that dynamic encryption functions are more difficult to be broken by the attacker than static encryption functions.

In order to obtain the cipher sequence s_c transmitted to the receiver after the observation of a sequence s by the sender, let us define the encryption function $F_E : \Sigma^\star \rightarrow \Sigma_c^\star$, recursively as $F_E(\varepsilon) = \varepsilon$ and $F_E(s\sigma) = F_E(s)F^e_E(s\sigma)$, for all $s \in \Sigma^\star$ and $\sigma \in \Sigma$. Notice that, F_E satisfies the assumption that a cipher event is generated after the observation of each event of s. Thus, $|s_c| = |s|$. In addition, the following property can be obtained.

Theorem 1. Encryption function $F_E : \Sigma^\star \rightarrow \Sigma_c^\star$ is prefix-preserving.

Proof: The proof is straightforward from the definition of F_E and of a prefix-preserving function. ■

According to Theorem 1, if two sequences $s, s' \in \Sigma^\star$ have the same prefix $\bar{s}$ of length p, then $s_c = F_E(s)$ and $s'_c = F_E(s')$ have the same cipher prefix $F_E(\bar{s})$ of length p.

At the receiver's site, after the transmission of the cipher event σ_c, the original plain event σ must be recovered. Thus, it is necessary to define an event decryption function $F^e_D : \Sigma_c^\star \rightarrow \Sigma \cup \{\varepsilon\}$, such that $F^e_D(\varepsilon) = \varepsilon$, and $F^e_D(F_E(s\sigma)) = \sigma$, *i.e.*, the event decryption function is capable of recovering the correct plain event $\sigma \in \Sigma$ after the observation of the cipher sequence $s_c\sigma_c = F_E(s\sigma) \in \Sigma_c^\star$. It is important to remark that if a dynamic encryption function is used, then a dynamic decryption function is needed to recover the last plain event transmitted in the channel.

In the sequel, we present a necessary and sufficient condition for the existence of an event decryption function F^e_D, given an event encryption function F^e_E.

Theorem 2. Let F^e_E and F_E be an event encryption function and its associated encryption function, respectively. Then, there exists an event decryption function F^e_D such that $F^e_D(F_E(s\sigma)) = \sigma$ if, and only if, $F^e_E(s\sigma') \neq F^e_E(s\sigma'')$, for all $\sigma' \neq \sigma''$ and $s \in \Sigma^\star$.

Proof: Since, according to Theorem 1, F_E is prefix-preserving, then $F_E(s\sigma')$ and $F_E(s\sigma'')$ have the same prefix $s_c = F_E(s)$. Thus, if $F^e_E(s\sigma') = F^e_E(s\sigma'')$, for $\sigma' \neq \sigma''$, then the same cipher sequence $s_c\sigma_c$ is transmitted to the receiver independently of the last event observed by the sender. Thus, it is impossible to discover the correct plain event that has occurred in the system, and $F^e_D(s_c\sigma_c)$ is not uniquely defined.

Let us consider now that $F^e_E(s\sigma') \neq F^e_E(s\sigma'')$, for all $\sigma' \neq \sigma''$ and $s \in \Sigma^\star$. Since F_E is prefix-preserving, then after the observation of sequence $s_c = F_E(s)$, the unique way of distinguishing the occurrence of event σ' from σ'' is that these events are mapped to different events by the event encryption function F^e_E. Since, by hypothesis, $F^e_E(s\sigma') \neq F^e_E(s\sigma'')$, for all $\sigma' \neq \sigma''$, then it is always possible to discover the correct plain event by using the inverse mapping after the observation of $s_c\sigma_c$, which implies that there exists a decryption function F^e_D such that $F^e_D(F_E(s\sigma)) = \sigma$. ■

In the sequel, we present a method to obtain an event encryption function F^e_E that satisfies the condition of Theorem 2, modeled by a Mealy automaton.

4. ENCRYPTION AND DECRYPTION MEALY AUTOMATA

In this section, we introduce the class of event encryption functions F^e_E that can be represented by a deterministic finite-state Mealy automaton $\mathcal{G}_E = (X_E, \Sigma, \Sigma_c, f_E, h_E, x_{0,E})$ satisfying the following conditions:

C1. $f_E(x, \sigma)$ and $h_E(x, \sigma)$ are defined for all $x \in X_E$ and $\sigma \in \Sigma$;

C2. $h_E(x, \sigma) \neq h_E(x, \sigma')$, for all $x = f_E(x_{0,E}, s) \in X_E$ and $\sigma \neq \sigma'$, where $\sigma, \sigma' \in \Sigma$.

In the Mealy automaton $\mathcal{G}_E$, each transition, represented by $f_E(x, \sigma)$, is labeled with a plain event $\sigma \in \Sigma$, and $h_E(x, \sigma)$ represents the encryption of event σ into event σ_c, after the occurrence of a sequence s that leads the system to state $x = f_E(x_{0,E}, s)$. This implies that the function F^e_E that is represented by $\mathcal{G}_E$, is such that each cipher event obtained after the occurrence of sequence $s\sigma$ can be represented by $F^e_E(s\sigma) = h_E(f_E(x_{0,E}, s), \sigma)$, and $\mathcal{G}_E$ has finite state. It is also important to remark that, since F^e_E is defined for all $\Sigma^\star$, then $\mathcal{G}_E$ must have complete transition function and output transition function, *i.e.*, $f_E(x, \sigma)$ and $h_E(x, \sigma)$ are defined for all $x \in X_E$ and $\sigma \in \Sigma$ (Condition **C1**). In order to be used in the cryptographic scheme proposed in this paper, it is also necessary the existence of an event decryption function F^e_D associated with F^e_E. Since $F^e_E(s\sigma) = h_E(f_E(x_{0,E}, s), \sigma)$, for all $s \in \Sigma^\star$ and $\sigma \in \Sigma$, then, if Condition **C2** is satisfied, we have that $F^e_E(s\sigma) \neq F^e_E(s\sigma')$, for all $\sigma \neq \sigma'$. Thus, according to Theorem 2, there exists an event decryption function F^e_D such that $F^e_D(F_E(s\sigma)) = \sigma$.

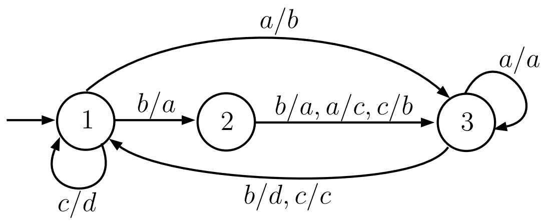

Fig. 3. Encryption Automaton $\mathcal{G}_E$

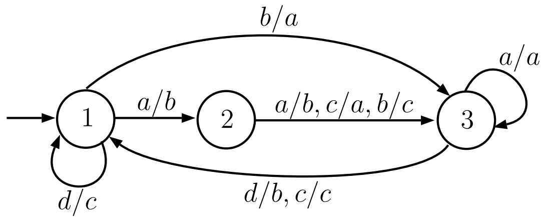

Fig. 4. Decryption Automaton $\mathcal{G}_D$

In the next example, we present an event encryption function F^e_E described by a deterministic finite-state Mealy automaton $\mathcal{G}_E$.

Example 2. Let $\Sigma = \{a, b, c\}$ and $\Sigma_c = \{a, b, c, d\}$ be the set of plain events and cipher events, respectively. Consider the Mealy automaton $\mathcal{G}_E = (X_E, \Sigma, \Sigma_c, f_E, h_E, x_{0,E})$, depicted in Figure 3. Notice that, automaton $\mathcal{G}_E$ has complete transition functions f_E and h_E. In addition, $h_E(x, \sigma) \neq h_E(x, \sigma')$, for all $\sigma \neq \sigma'$ and $x \in X_E$. Thus, the event encryption function $F^e_E(s\sigma) = h_E(f_E(x_{0,E}, s), \sigma)$, for all $s\sigma \in \Sigma^\star$, can be used in the cryptographic scheme proposed in this paper.

Let us consider now that the plain sequence $s = abc$ is observed by the sender. Then, according to $\mathcal{G}_E$ presented in Figure 3, the cipher sequence transmitted to the receiver is $s_c = F^e_E(a)F^e_E(ab)F^e_E(abc) = h_E(1, a)h_E(3, b)h_E(1, c) = bdd$. □

If $\mathcal{G}_E$ satisfies Conditions **C1** and **C2**, then the plain sequence $s \in \Sigma^\star$ can be recovered from the cipher sequence $s_c \in \Sigma_c^\star$ using the Mealy automaton $\mathcal{G}_D = (X_E, \Sigma_c, \Sigma, f_D, h_D, x_{0,E})$, obtained from $\mathcal{G}_E$, where $f_D : X_E \times \Sigma_c \to \Sigma$ and $h_D : X_E \times \Sigma_c \to \Sigma$. The transition function f_D is defined as $f_D(x, \sigma_c) = f_E(x, \sigma)$, where $\sigma_c = F^e_E(s\sigma)$, and $x = f_E(x_{0,E}, s)$, and the output transition function h_D is defined as $h_D(x, \sigma_c) = \sigma$. Notice that, differently from f_E, f_D is not necessarily complete on its domain.

Example 3. Let us consider automaton $\mathcal{G}_E$ depicted in Figure 3. Then, automaton $\mathcal{G}_D$, obtained from $\mathcal{G}_E$, is presented in Figure 4. If the cipher sequence $s_c = bdd$ is observed at the receiver's site, then the original plain sequence is given by $s = F^e_D(b)F^e_D(bd)F^e_D(bdd) = h_D(1, b)h_D(3, d)h_D(1, d) = abc$. □

Remark 2. It is important to remark that the cryptographic scheme proposed in this work, based on the construction of an encryption automaton $\mathcal{G}_E$, can represent classical encryption functions from Information Technology such as the monoalphabetic Caesar cipher and the polyalphabetic Vigenere cipher. More details about Caesar and Vigenere ciphers are presented in Stallings (2006). □

In the sequel, we present the encryption and decryption procedures to ensure the confidentiality of the data in the attacked network channel.

4.1 Encryption and decryption procedures

In order to implement the cryptographic scheme proposed in this paper, it is necessary that the sender and the receiver are synchronized in the sense that, initially, both devices must know the correct initial binary vector $\underline{u}_0$. Thus, in the first step of the encryption procedure the observed vector $\underline{u}_0$ is communicated to the receiver. Then, any modification in the observed vector at the sender site, interpreted as an event occurrence, is encrypted by using encryption automaton $\mathcal{G}_E$, and communicated to the receiver as a modification in the last transmitted binary vector. Thus, the objective of the encryption procedure is to generate the encrypted vector $\underline{u}_c$ and transmit it to the receiver. The encryption process is presented in Algorithm 1.

Algorithm 1. Encryption procedure

```
 1: $\underline{u} \leftarrow \underline{u}_0$, where $\underline{u}_0$ is the first observation of the vector
    of binary signals to be transmitted.
 2: $\underline{u}_c \leftarrow \underline{u}_0$.
 3: $s \leftarrow \varepsilon$.
 4: Transmit $\underline{u}_c$ to the receiver.
 5: Wait for the observation of a vector $\underline{u}' \neq \underline{u}$.
 6: Compute event $\sigma$ coded as $\underline{\sigma} = \underline{u} \oplus \underline{u}'$.
 7: Compute cipher event $\sigma_c = F_E^e(s\sigma) = h_E(f_E(x_0, s), \sigma)$,
    using $\mathcal{G}_E$, coded as $\underline{\sigma}_c$.
 8: $s \leftarrow s\sigma$.
 9: $\underline{u} \leftarrow \underline{u}'$
10: $\underline{u}_c \leftarrow \underline{u}_c \oplus \underline{\sigma}_c$.
11: Transmit $\underline{u}_c$ to the receiver.
12: Return to Step 5.
```

The decryption procedure is presented in Algorithm 2. The objective of the decryption procedure is to discover the binary vector $\underline{u}$ that was observed by the sender from the last transmitted encrypted vector $\underline{u}_c$. The first observed vector communicated to the receiver is not decrypted, since it is equal to the first vector observed by the sender $\underline{u}_0$, and is used to synchronize both devices. Then, when a different binary vector is observed at the receiver site, the cipher event σ_c is computed. After that, the corresponding plain event σ is recovered from σ_c using the decryption automaton $\mathcal{G}_D$, which allows the receiver to know the binary vector $\underline{u}$ that was observed by the sender.

Algorithm 2. Decryption procedure

```
1: $\underline{u} \leftarrow \underline{u}_c$, where $\underline{u}_c$ is the first vector received.
2: $s_c \leftarrow \varepsilon$.
3: Wait for the observation of a vector $\underline{u}_c' \neq \underline{u}_c$.
4: Compute cipher event $\sigma_c$ coded as $\underline{\sigma}_c = \underline{u}_c \oplus \underline{u}_c'$.
5: Compute plain event $\sigma = F_D^e(s_c\sigma_c) =$
   $= h_D(f_D(x_0, s_c), \sigma_c)$, using $\mathcal{G}_D$, coded as $\underline{\sigma}$.
6: $s_c \leftarrow s_c\sigma_c$.
7: $\underline{u} \leftarrow \underline{u} \oplus \underline{\sigma}$.
8: $\underline{u}_c \leftarrow \underline{u}_c'$.
9: Return to Step 3.
```

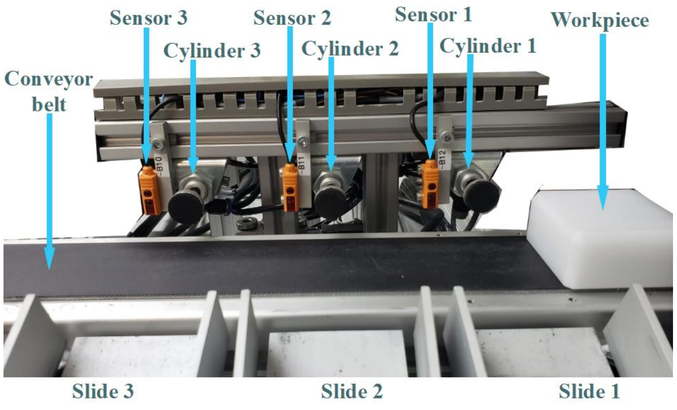

Fig. 5. Sorting unit

Remark 3. It is possible to increase the security of the cryptographic scheme proposed in this work when the receiver already knows the initial binary vector of the system $\underline{u}_0$, since, in this case, it is not necessary to transmit any vector $\underline{u}$ directly to the receiver without using encryption. In this case, an arbitrary vector in $\mathbb{Z}_1^m$ could be transmitted as the first vector, and then, the cipher events would be calculated as the exclusive disjunction of the last two binary vectors communicated to the receiver. Then, using the plain event σ and the already known vector $\underline{u}_0$, the correct vector $\underline{u}$ could be easily found. □

5. PRACTICAL EXAMPLE

In order to illustrate the cryptographic scheme proposed in this paper, let us consider a sorting unit system composed of a conveyor belt, three presence sensors, three pneumatic cylinders, and three slides, as shown in Figure 5. A workpiece is always placed at the beginning of the conveyor, at the right of Figure 5, and is moved to the left until it is sorted by one of the pneumatic cylinders of the system. The function of the sorting unit is to separate workpieces according to their type. The workpieces can be metallic, white plastic or black plastic. Metallic workpieces are pushed to Slide 1, white plastic workpieces are pushed to Slide 2, and black plastic workpieces are pushed to Slide 3. We assume that only one workpiece can be on the conveyor at a time.

Let s_1, s_2, and s_3, be the binary signals associated with the three presence sensors 1, 2 and 3, respectively, and let us consider that the communication channel that transmits the observation of these signals is attacked by an intruder that eavesdrops the information of the channel. In this case, the sender is a Remote Terminal Unit (RTU) that communicates the sensor readings to a Programmable Logic Controller (PLC), that is the receiver. Thus, the vector of binary signals that must be communicated to the PLC is given by $\underline{u} = [s_3\ s_2\ s_1]^T$. Since only one workpiece can be in the sorting unit system at a time, then there are only four possibilities for vector $\underline{u}$, that depends on the position of the workpiece on the conveyor belt: *(i)* the workpiece is in front of Sensor 1, $\underline{u}_1 = [0\ 0\ 1]^T$; *(ii)* the workpiece is in front of Sensor 2, $\underline{u}_2 = [0\ 1\ 0]^T$; *(iii)* the

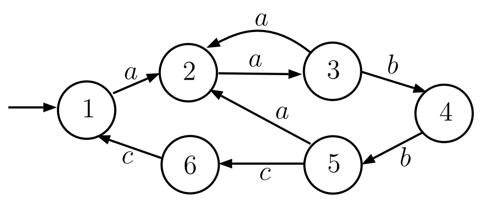

Fig. 6. Automaton representation of the sorting unit G.

Table 1. Sensor readings table of Example 1

State	Vector $\underline{u}$
1	$[0\ 0\ 0]^T$
2	$[0\ 0\ 1]^T$
3	$[0\ 0\ 0]^T$
4	$[0\ 1\ 0]^T$
5	$[0\ 0\ 0]^T$
6	$[1\ 0\ 0]^T$

workpiece is in front of Sensor 3, $\underline{u}_3 = [1\ 0\ 0]^T$; or *(iv)* none of the sensors detect the workpiece $\underline{u}_0 = [0\ 0\ 0]^T$.

The events of the system are associated with changes in the entries of vector $\underline{u}$. For example, if a change from vector $\underline{u}_0$ to $\underline{u}_1$ is observed, an event a, associated with the change of the third entry of $\underline{u}$ is generated. This event is coded as $\underline{a} = \underline{u}_0 \oplus \underline{u}_1 = [0\ 0\ 1]^T$. Similarly, if a change from vector $\underline{u}_0$ to $\underline{u}_2$ is observed, an event b, coded as $\underline{b} = [0\ 1\ 0]^T$ is generated, and if a change from vector $\underline{u}_0$ to $\underline{u}_3$ is observed, an event c, coded as $\underline{c} = [1\ 0\ 0]^T$, is generated. Notice that the set of all possible events is, in this example, equal to $\Sigma = \{a, b, c\}$.

The behavior of the sorting unit can be described by automaton G, depicted in Figure 6. In the initial state 1 of automaton G, there is no workpiece in the sorting unit. Thus, vector $\underline{u} = [0\ 0\ 0]^T$. Then, a workpiece is placed on the conveyor, and Sensor 1 is the unique sensor that can detect the presence of the workpiece, generating event a, which leads the system to state 2 of G, and vector $\underline{u} = [0\ 0\ 1]^T$ is observed. In state 2, the workpiece can be removed by cylinder 1, or it can continue on the conveyor belt. In both cases, Sensor 1 will stop detecting the workpiece and a new occurrence of event a will be generated, leading the system to state 3. Then, in state 3, there are two possibilities. If the workpiece was removed by Cylinder 1, then a new piece can be placed on the conveyor and Sensor 1 will detect its presence after some time, generating event a. However, if the workpiece was not removed by Cylinder 1, then it continues on the conveyor, and after some time it is detected by Sensor 2, generating event b. In state 4, the piece can be removed by cylinder 2 or continue on the conveyor belt, leading the system to state 5. In state 5 there may be no workpiece on the sorting unit, *i.e.*, the workpiece was pushed to Slide 2, making the system wait for a new observation of event a, or the workpiece is on the conveyor belt and will be detected by Sensor 3, generating event c. Finally, in state 6, the workpiece need to be removed by Cylinder 3, generating a new occurrence of event c, and the sorting unit returns to its initial state. The states of G are associated with the sensor readings $\underline{u}$, according to Table 1.

Now, let us consider an encryption function F_E represented by the encryption automaton $\mathcal{G}_E$, depicted in Figure 3. In this case, the set of output events of $\mathcal{G}_E$ is equal

Table 2. Event codification table

Event	Event codification
a	$[0\ 0\ 1]^T$
b	$[0\ 1\ 0]^T$
c	$[1\ 0\ 0]^T$
d	$[0\ 1\ 1]^T$

to the cipher event set $\Sigma_c = \{a, b, c, d\}$, where d represents the simultaneous change of the second and third entries of $\underline{u}$. Event d is, therefore, coded as $\underline{d} = [0\ 1\ 1]^T$. The cipher events are presented in Table 2. Notice that, even though the system cannot perform event d, it is defined for the output event set Σ_c, *i.e.*, set Σ_c can include events that are not possible to be observed in the system.

Consider that the following plain sequence of binary vectors $\underline{u}$ is observed by the RTU:

$$\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}$$

In order to ensure the confidentiality of the transmitted data, let us consider that the encryption procedure described in Algorithm 1 is used, with event encryption function described by $\mathcal{G}_E$ of Figure 3. According to Step 1 of Algorithm 1, the first vector $\underline{u}_0 = [0\ 0\ 0]^T$ is communicated to the receiver to synchronize it with the sender. Then, according to Steps 4 and 5, after the observation of vector $[0\ 0\ 1]^T$, the plain event a is generated. Then, in Step 6, using $\mathcal{G}_E$, the cipher event b is computed and vector $[0\ 1\ 0]^T$ is transmitted to the receiver in Step 3. Following the steps of Algorithm 1, the complete plain sequence $s = aabbaabbcc$ is ciphered as $s_c = badacadabc$. Thus, the following cipher sequence of vectors u_c is the one observed by the intruder:

$$\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\end{bmatrix}$$

Notice that the cipher sequence is different from the plain sequence of binary vectors observed by the sender. In addition, there are readings of vector $\underline{u}_c$ that do not correspond to the normal behavior of the system, *e.g.*, vector $[1\ 1\ 1]^T$ represents that there are three workpieces in the sorting unit at the same time, which, by hypothesis, is not possible in the normal behavior of the system. Therefore, an attacker would be unable of estimating or understanding the sensor readings from the transmitted data, even if the attacker knows the model of the system.

At the receiver's site, each cipher event is decrypted using decryption automaton $\mathcal{G}_D$. In Step 1 of Algorithm 2, the first vector communicated to the receiver is equal to $\underline{u}_0 = [0\ 0\ 0]^T$, and sender and receiver become synchronized. Then, in Step 3, after the observation of the binary vector $[0\ 1\ 0]^T$, the cipher event b is computed. Then, using $\mathcal{G}_D$ depicted in Figure 4, it is possible to recover the plain event a from the cipher event b. Applying the decryption process in the complete sequence $s_c = badacadabc$ we obtain the original plain sequence $s = aabbaabbcc$. Since s can be recovered from s_c, and $\underline{u}_0$ is known by the receiver, then the original sequence of binary vectors observed by the sender can be completely discovered by the receiver.

6. CONCLUSIONS

In this paper, we propose the implementation of a defense strategy, based on event-based cryptography, in order to prevent a malicious agent from getting information transmitted between two entities in an industrial network. In order to do so, we propose a new method to represent events coded as changes in the transmitted vector of the industrial communication channel. Then, we introduce event-based encryption functions, and present an encryption Mealy automaton to represent a class of encryption functions. We also present the decryption Mealy automaton associated with the encryption automaton to recover the plain event observed by the sender. We present procedures for the encryption and decryption of events in an industrial network, and use a practical example to illustrate the results presented in this paper.

REFERENCES

Barcelos, R.J. and Basilio, J.C. (2018). Enforcing current-state opacity through shuffle in event observations. *IFAC-PapersOnLine*, 51(7), 100–105.

Carvalho, L.K., Wu, Y.C., Kwong, R., and Lafortune, S. (2018). Detection and mitigation of classes of attacks in supervisory control systems. *Automatica*, 97, 121–133.

Cassandras, C.G. and Lafortune, S. (2008). *Introduction to discrete event systems*. Springer, Secaucus, NJ.

Da Xu, L., He, W., and Li, S. (2014). Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, 10(4), 2233–2243.

Esmoris, A., Chesñevar, C.I., and González, M.P. (2005). TAGS: A Software Tool for Simulating Transducer Automata. *International Journal of Electrical Engineering & Education*, 42(4), 338–349.

Fritz, R., Fauser, M., and Zhang, P. (2019). Controller encryption for discrete event systems. In *2019 American Control Conference (ACC)*, 5633–5638. IEEE, PHILADELPHIA, CA, USA.

Goes, R.M., Kang, E., Kwong, R.H., and Lafortune, S. (2017). Stealthy Deception Attacks for Cyber-Physical Systems. In *Proceedings of the 56th IEEE Conference on Decision and Control*, 4224–4230. Melbourne, Australia.

Gou, Q., Yan, L., Liu, Y., and Li, Y. (2013). Construction and strategies in IoT security system. In *2013 IEEE international conference on green computing and communications and IEEE internet of things and IEEE cyber, physical and social computing*, 1129–1132.

He, H., Maple, C., Watson, T., Tiwari, A., Mehnen, J., Jin, Y., and Gabrys, B. (2016). The security challenges in the IoT enabled cyber-physical systems and opportunities for evolutionary computing & other computational intelligence. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, 1015–1021.

Jacob, R., Lesage, J.J., and Faure, J.M. (2016). Overview of discrete event systems opacity: Models, validation, and quantification. *Annual Reviews in Control*, 41, 135–146.

Kurose, J.F. and Ross, K.W. (2011). *Computer networking: a top-down approach*. Addison Wesley.

Lafortune, S., Lin, F., and Hadjicostis, C.N. (2018). On the history of diagnosability and opacity in discrete event systems. *Annual Reviews in Control*, 45, 257–266.

Lima, P.M., Alves, M.V.S., Carvalho, L.K., and Moreira, M.V. (2021). Security of cyber-physical systems: Design of a security supervisor to thwart attacks. *IEEE Transactions on Automation Science and Engineering*, 1–12.

Lima, P.M., Carvalho, L.K., and Moreira, M.V. (2018). Detectable and Undetectable Network Attack Security of Cyber-physical Systems. *IFAC-PapersOnLine*, 51(7), 179–185.

Lima, P.M., Alves, M.V.S., Carvalho, L.K., and Moreira, M.V. (2019). Security Against Communication Network Attacks of Cyber-Physical Systems. *Journal of Control, Automation and Electrical Systems*, 30(1), 125–135.

Lima, P.M., Carvalho, L.K., and Moreira, M.V. (2020). Confidentiality of cyber-physical systems using event-based cryptography. In *21st IFAC World Congress 2020*, 1761–1766. Berlin, Germany.

Lin, F. (2011). Opacity of discrete event systems and its applications. *Automatica*, 47(3), 496–503.

Mealy, G.H. (1955). A method for synthesizing sequential circuits. *The Bell System Technical Journal*, 34(5), 1045–1079.

Moreira, M.V. and Lesage, J.J. (2019). Fault diagnosis based on identified discrete-event models. *Control Engineering Practice*, 91, 104101.

Singh, S. and Singh, N. (2015). Internet of Things (IoT): Security challenges, business opportunities & reference architecture for E-commerce. In *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)*, 1577–1581.

Stallings, W. (2006). *Cryptography and network security, 4/E*. Pearson Education India.

Su, R. (2018). Supervisor synthesis to thwart cyber attack with bounded sensor reading alterations. *Automatica*, 94, 35–44.

Thorsley, D. and Teneketzis, D. (2006). Intrusion Detection in Controlled Discrete Event Systems. In *Proceedings of the 45th IEEE Conference on Decision and Control*, 6047–6054. San Diego, CA, USA.

Tong, Y., Cai, K., and Giua, A. (2018). Decentralized Opacity Enforcement in Discrete Event Systems Using Supervisory Control. In *2018 57th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, 1053–1058.

Wonham, W.M. and Cai, K. (2019). *Supervisory Control of Discrete-Event Systems*. Communications and Control Engineering. Springer International Publishing, Cham, 1 edition.

Yin, X. and Lafortune, S. (2015). A new approach for synthesizing opacity-enforcing supervisors for partially-observed discrete-event systems. In *2015 American Control Conference (ACC)*, 377–383.

Zhang, Q., Li, Z., Seatzu, C., and Giua, A. (2018). Stealthy Attacks for Partially-Observed Discrete Event Systems. In *2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, volume 1, 1161–1164.