2026 IEEE/SICE International Symposium on System Integration (SII)
January 11-14, 2026. Cancún, México

Secure Supervisory Control of Discrete Event Systems using Homomorphic Encryption*

Ana Clara P. Gonçalves[1], Patrícia N. Pena[2] and Lucas V. R. Alves[3]

***Abstract*— This paper addresses the critical challenge of securing Cyber-Physical Systems (CPSs) against passive communication attacks, specifically focusing on systems modeled as Discrete Event Systems (DES). We introduce a methodology, based on the Elliptic Curve ElGamal (EC-ElGamal), to protect the confidentiality of DES-based CPS using homomorphic encryption. Our approach employs the computational intractability of the Elliptic Curve Discrete Logarithm Problem (ECDLP) to fortify supervisory control systems operating over vulnerable communication channels. The work details the EC-ElGamal protocol, from key generation to encryption and decryption, emphasizing its direct adaptation for ensuring robust data confidentiality within distributed DES architectures. A practical example demonstrating the application of this methodology in a DES context is implemented and tested on an ESP32 microcontroller.**

I. INTRODUCTION

Cyber-physical systems (CPSs) have a distributed structure that combines both cyber and physical components. Examples of these include smart grids, flexible manufacturing systems, and autonomous vehicles [1]. With the growth in cyberattacks over the years, these CPSs are susceptible to invasion. In an attack, the actuators or sensors in the feedback control loop are exploited while the controller is compromised, and the system is turned into a state of vulnerability [2].

Within this context, a CPS can be modeled as a system with discrete states. Many researchers have explored this modeling formalism because of its ability to detect both passive and active attacks [3].

In this paper, we focus on passive attacks, in which the intruder intercepts the communication routes in the system to gather secret information. Using an encryption system called Elliptic Curve ElGamal (EC-ElGamal), which is a public-key encryption scheme based on its security strength from the computational intractability of the Elliptic Curve Discrete Logarithm Problem [4]–[7].

Within this context, the logical behavior of a DES can be represented mathematically. We adopt a matrix-based approach for modeling and controlling Discrete Event Systems as described in [8]. This model represents the system's state and events using binary vectors and matrices. Operations such as state updates are typically represented as multiplications of a binary matrix with a binary vector. However, due to the binary nature of these components, a matrix-vector multiplication can be re-formulated as a series of logical sums and conditional checks. The additive homomorphic properties of EC-ElGamal are perfectly suited for these logical sum operations.

While more powerful schemes like Fully Homomorphic Encryption (FHE) support both addition and multiplication, they have much higher memory usage, making them impractical for embedded systems like the ESP32, which have more limited resources. Standard non-homomorphic lightweight ciphers could secure the communication channels but require the supervisor to decrypt the plant observations before processing [3]. Other recent work has focused on developing specialized Resilient Homomorphic Encryption (RHE) schemes, which are designed to actively neutralize additive attacks on ciphertexts, thus ensuring system availability even when under active attack [9].

The remainder of this paper is organized as follows: Section 2 provides essential preliminaries on Languages and Automata, Supervisory Control Theory, and EC-ElGamal. Section 3 presents our proposed methodology. Section 4 details the experimental setup and implementation on the ESP32. Finally, Section 5 concludes the paper.

II. PRELIMINARIES

A. Languages and Automata

Alphabet Σ is a finite collection of symbols. From this Σ, a finite sequence of symbols is known as a word s (or string). The Kleene closure, Σ^*, encloses all finite-length words formed from Σ, including the empty word ϵ. A language L is a subset of Σ^*, and $\overline{L}$ describes the set of all prefixes of strings in L. A string $t \in \Sigma^*$ is a prefix of a string $s \in \Sigma^*$ if s can be formed by concatenating t with another string $u \in \Sigma^*$, as defined by the equation $s = tu$. [10].

A Deterministic Finite Automaton (DFA) is defined as $G = (Q, \Sigma, \delta, q_0, Q_m)$ [10], where:

- Q is a finite set of states;
- Σ is the alphabet;
- $\delta : Q \times \Sigma \longrightarrow Q$ is the state transition function;
- $q_0 \in Q$ is the initial state;
- $Q_m \subseteq Q$ is the set of marked states.

An automaton G generates a language $\mathcal{L}(G)$ (all possible sequences of events from q_0) and marks a language $\mathcal{L}_m(G)$ (sequences leading to marked states) [10].

*This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001 and Fapemig P/APQ-06580-24.

[1]Ana Clara P. Gonçalves is with the Graduate Program in Electrical Engineering - Universidade Federal de Minas Gerais - Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brazil anacpg@ufmg.br

[2]Patrícia N. Pena is with Department of Electronic Engineering - Universidade Federal de Minas Gerais - Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brazil ppena@ufmg.br

[2]Lucas V. R. Alves is with Technical College - Universidade Federal de Minas Gerais - Av. Antônio Carlos 6627, 31270-901, Belo Horizonte, MG, Brazil lucasvra@ufmg.br

B. Supervisory Control Theory

Ramadge and Wonham established the mathematical foundation for Supervisory Control Theory (SCT). This framework proved crucial for the systematic analysis and design of controllers specifically tailored for discrete event systems. [11]. SCT serves as a formal methodology for regulating the behavior of DES, aiming to ensure their safe and efficient operation in adherence to predefined specifications. The uncontrolled behavior of a system is represented by a finite automaton, often termed the "plant," which delineates its physical capabilities and all potential behavior. [11].

Events within these systems are categorized into two types: controllable (Σ_c) and uncontrollable (Σ_u). Controllable events are those that the supervisor has the authority to enable or disable, whereas uncontrollable events cannot be interfered with by the supervisor. The ultimate objective in SCT is to synthesize a supervisor capable of enforcing a non-blocking desired closed-loop behavior by disabling only controllable events.

Let G be a plant and B be a specification, the necessary and sufficient condition for the existence of a non-blocking supervisor S for G, such that $\mathcal{L}_m(S/G) = \mathcal{L}_m(G||B) = \mathcal{K}$, is that $\mathcal{K}$ is controllable with respect to $\mathcal{L}(G)$ and Σ_u, that is, $\overline{\mathcal{K}}\Sigma_u \cap \mathcal{L}(G) \subseteq \overline{\mathcal{K}}$. If the supervisor is not controllable, then the supremal controllable sublanguage of the desired language, denoted by $Sup\mathcal{C}(\mathcal{K}, G)$, must be computed.

C. Elliptic Curve Fundamentals and Point Arithmetic

The concept of elliptic curves for cryptography was introduced by Victor Miller in 1985 [4] and Neal Koblitz in 1987 [5]. Their work demonstrated that existing public-key algorithms could be adapted to elliptic curve structures [6].

To work with the algebraically convenient form of an elliptic curve shown in (1), we define it over a field K and impose the condition that the characteristic of K is not the number 2 or 3:

$$E_K : y^2 = x^3 + ax + b. \tag{1}$$

In this equation, a and b are coefficients belonging to the field K. The cubic on the right side of the equation must have no multiple roots. The set of solutions (x, y) to this equation, along with a point at infinity, constitutes the points on the curve. This set of points forms a group with the point at infinity as the identity element. For cryptographic applications, K is typically a finite field $G_pF(q)$ where $q = p^n$ [5]. An elliptic curve equation is defined as

$$y^2 + cxy + dy = x^3 + ax + b. \tag{2}$$

For points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ on this curve, the coordinates of their sum $P_3 = (x_3, y_3)$ are given by eqs. (3) to (5):

$$\alpha = \begin{cases} (y_2 - y_1)/(x_2 - x_1), & \text{if } P_1 \neq P_2 \\ (3x_1^2 + a - cy_1)/(2y_1 + cx_1 + d), & \text{if } P_1 = P_2 \end{cases} \tag{3}$$

$$x_3 = -x_1 - x_2 + \alpha^2 + c\alpha \tag{4}$$

$$y_3 = -cx_3 - d - y_1 + \alpha(x_1 - x_3) \tag{5}$$

To obtain the addition rules for the simplified curve $y^2 = x^3 + ax + b$, we simply substitute c=0 and d=0 into the general formulae presented above (eqs. (3) to (5)) [5].

These point arithmetic operations enable scalar multiplication, denoted as kP, which is the process of adding point P to itself k times ($P+P+\cdots+P$, k) [6]. This multiplication represents the most computationally intensive component within Elliptic Curve Cryptography (ECC) systems [6].

D. The EC-ElGamal Encryption Protocol

The ElGamal public-key encryption scheme, originally conceived by Taher ElGamal in 1985 [7], is adaptable to the structure of elliptic curves. This adaptation is conceptually understood as a key transfer mechanism within the broader Diffie-Hellman key agreement framework [6].

For an entity, referred to as A, the key generation process involves several steps [7]. First, the public parameters are selected and agreed upon by all communicating parties: a large prime p, an elliptic curve E defined over K, and a generator point $G_p \in E(K)$. The order n of the cyclic subgroup generated by G_p is also determined to ensure cryptographic strength. Following this, Entity A selects a random integer k_A from the interval $[1, n-1]$, which serves as its strictly confidential private key. The public key, a point on the elliptic curve, is then computed by performing scalar multiplication of the generator point G_p by the private key k_A:

$$A = k_A G_p. \tag{6}$$

This public key A (often represented as a tuple (E,G_p,A)) is then disseminated publicly, for instance, by being listed on a Public Key server.

To encrypt a message M for A, another entity, referred to as B, performs a series of operations [7]. The plaintext message M is first converted into a point $P_M = (M_1, M_2)$ on the elliptic curve E, a mapping that must be reversible. Entity B then chooses a random ephemeral key k_B from the interval $[1, n-1]$ and computes a temporary point B by performing scalar multiplication of the generator point G_p by k_B:

$$B = k_B G_p. \tag{7}$$

A shared secret point S_AB is then derived by computing using Entity A's public key A and Entity B's ephemeral key k_B:

$$S_{AB} = k_B A = k_B(k_A G_p) = (x_S, y_S). \tag{8}$$

Finally, the ciphertext components are calculated by multiplying the coordinates of the message point P_M with the respective coordinates of the shared secret point S_AB:

$$C_{M1} = x_S \cdot M_1 \tag{9}$$

$$C_{M2} = y_S \cdot M_2 \tag{10}$$

The resulting ciphertext, a pair $(B, (C_M1, C_M2))$, is then transmitted to Entity A.

Upon receiving the ciphertext $(B, (C_M1, C_M2))$, Entity A performs the necessary steps to decrypt the message [7]. It first re-computes the shared secret point S_AB by using its private key k_A and the received temporary point B:

$$S_{AB} = k_A B = k_A(k_B G_p) = (x_S, y_S). \tag{11}$$

With the shared secret point re-established, Entity A can recover the original message point coordinates by performing modular division of the ciphertext components by the coordinates of $S_A B$:

$$M_1 = C_{M1}/x_S \tag{12}$$

$$M_2 = C_{M2}/y_S \tag{13}$$

The original message M is then reconstructed from the recovered point $P_M = (M_1, M_2)$, completing the decryption process.

E. Homomorphic Encryption with EC-ElGamal

A homomorphic encryption (HE) scheme is an encryption form that permits the performance of functions on encrypted data by a third party like a service provider. These actions will preserve the same format of the data and features of the function [12].

For an additively homomorphic encryption (HE) scheme, the sum of two messages, m_1 and m_2, can be computed on their encrypted forms, $E(m_1)$ and $E(m_2)$, respectively, to obtain $E(m_1 + m_2)$, all without knowing the content of the original messages [12].

This additive property of homomorphic encryption is particularly well-suited to the mathematical structure of the Elliptic Curve ElGamal (EC-ElGamal) cryptosystem. While EC-ElGamal is classified as a Partially Homomorphic Encryption (PHE) scheme, the addition scheme is supported. This operation is highly efficient and directly supports the type of logical summation and matrix-vector multiplication required for processing DES.

III. PROPOSED METHODOLOGY

In some cases, due to unprotected communication channels, the system might be vulnerable to an attacker. It is considered a passive intruder. This eavesdropper can have the capabilities of knowing both the system model G and the supervisor S, and it can detect the events in Σ_c [13].

However, in some recent work, the assumption is that the attacker does not know the supervisors [14]–[16]. To compensate for the attacker's limited knowledge of the supervisor's model, it could engage in passive learning by collecting a language O of event observations. This data would then be used to estimate the supervisor's structure, enabling the attacker to successfully and covertly inflict damage upon the system [16].

Regardless of whether the attacker knows the supervisor's internal model or not, the utilization of EC-ElGamal's homomorphic properties ensures data confidentiality. The structure of the supervisory control system with cryptography is shown in Figure 1. This diagram illustrates a closed-loop system with insecure network communication between a plant and a supervisor. To ensure confidentiality, observations from the plant are encrypted before transmission. Also, disablements, which are control commands from the supervisor, are sent in an encrypted format and decrypted before being applied to the plant.

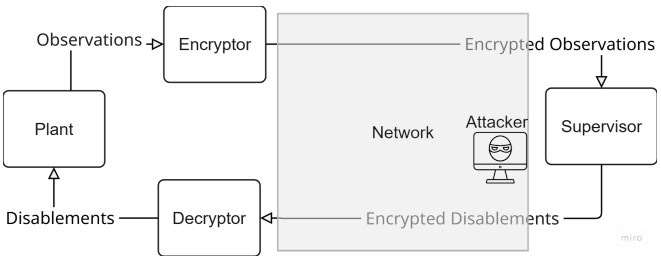


Fig. 1: Control System with Cryptography

A. Encoding and Decoding for Discrete Event Systems

States and events can be represented in binary components (e.g., a state is active/inactive, an event occurred/did not occur). In order to apply the EC-ElGamal cryptography, this information needs to be translated into points on an elliptic curve.

A '0' (inactive) is typically mapped to the identity element (Point at Infinity) and behaves like a zero in elliptic curve addition (e.g., P+O=P), while a '1' (active) is mapped to the generator point G_p of the chosen elliptic curve group. This binary encoding allows logical operations and linear algebra to be performed homomorphically through the additive properties of the elliptic curve group.

After homomorphic operations, the supervisor needs to retrieve the actual data. So, the resulting point must be converted back to its representation. This is achieved by mapping the identity element to 0 and any other non-zero point to 1. We define the encoding function, $\phi(m)$, which maps a binary message $m \in \{0, 1\}$ to a point on the elliptic curve:

$$\phi(m) = \begin{cases} O, & \text{if } m = 0 \\ G_p, & \text{if } m = 1 \end{cases} \tag{14}$$

where O is the identity element (Point at Infinity) and G_p is the generator point of the elliptic curve group.

Conversely, the decoding function, $\phi^{-1}(P)$, maps a point P back to its binary representation. This process is formalized by the following equation:

$$\phi^{-1}(P) = \begin{cases} 0, & \text{if } P = O \\ 1, & \text{if } P \neq O \end{cases} \tag{15}$$

This pair of functions, ϕ and ϕ^{-1}, works because the Point at Infinity (O) is the group's identity element ($P + O = P$), ensuring that homomorphic additions correctly mimic the logical OR required for the matrix-vector sums.

B. EC-ElGamal Integration

The EC-ElGamal protocol is an asymmetric cryptosystem whose security is based on the computational difficulty of the discrete logarithm problem [17]. All of the phases done in the algorithm are detailed below.

1) Key Generation: It generates both a public and a private key for both encryption and decryption processes.

This involves selecting an elliptic curve and its generator point, then randomly generating a private key k_{priv} and computing the corresponding public key $P_{pub} = k_{priv} \cdot G_p$. The private key is securely stored in an isolated module that is resistant to tampering, while the public key is made available for encryption by the plant's sensing units.

2) Encryption of DES Data: The public key is used to encrypt information. In this case, the plant's current state vector. Each binary value (0 or 1) of the vector components is individually encrypted using the EC-ElGamal protocol.

An ephemeral key r is randomly selected for each encryption, and the ciphertext for each bit $m \in \{0, 1\}$ consists of two components: $C_1 = r \cdot G_p$ and $C_2 = r \cdot P_{pub} + m \cdot G_p$.

This results in an encrypted vector of ciphertexts, where each ciphertext corresponds to a bit of the original vector. This is represented by Algorithm 1.

Algorithm 1: Encrypt

Input: Binary message $m \in \{0, 1\}$, elliptic curve group $\mathcal{G}$, generator G_p, public key P_{pub}.
Output: Ciphertext (c_1, c_2).
Initialize random number generator $\mathcal{R}$;
$ephemeral\_key \leftarrow$ random integer in $[1, \text{order}(G_p) - 1]$;
$c_1 \leftarrow ephemeral\_key \cdot G_p$;
$s\_point \leftarrow ephemeral\_key \cdot P_{pub}$;
if $m = 0$ **then**
  $c_2 \leftarrow s\_point$;
end
else
  $c_2 \leftarrow s\_point + G_p$;
end
return (c_1, c_2);

3) Decryption of DES Data: The decryption process is performed by using the private key information.

For each ciphertext (c_1, c_2), it computes $S' = k_{priv} \cdot c_1$. The original message point $m \cdot G_p$ is recovered by computing $P_m = c_2 - S'$. The point P_m is then mapped back to a binary integer using a decoding function, reconstructing the original DES vector. This process is represented by Algorithm 2, and the decoding function is defined by 15.

Algorithm 2: Decrypt

Input: Ciphertext (c_1, c_2), private key k_{priv}.
Output: Decrypted binary message m.
$s\_point\_recomputed \leftarrow k_{priv} \cdot c_1$;
$message\_point \leftarrow c_2 - s\_point\_recomputed$;
$m \leftarrow \phi^{-1}(message\_point)$;
return m;

4) Homomorphic Addition of Ciphertexts: The EC-ElGamal scheme supports homomorphic addition. Given two ciphertexts (c_{1a}, c_{2a}) encrypted as m_a and (c_{1b}, c_{2b}) encrypted as m_b, a ciphertext encrypting $m_a + m_b$ can be obtained by adding their respective components: $(c_{1a} + c_{1b}, c_{2a} + c_{2b})$. This operation, which we denote as $\oplus$, allows for secure aggregation of binary state vector components without decryption.

$$C_a \oplus C_b = (c_{1a} + c_{1b}, c_{2a} + c_{2b}) \tag{16}$$

5) Homomorphic Matrix-Vector Multiplication for State Update: Many operations, such as calculating the next state based on current state and event or determining which events are enabled, can be modeled as matrix-vector multiplication. In Algorithm 3, a homomorphic addition is applied to the encrypted vector components.

Algorithm 3: homomorphic_mat_vec (Matrix-Vector Multiplication)

Input: Encrypted vector $\mathbf{x}_{enc}$, clear-text binary matrix $\mathbf{M}$.
Output: Encrypted result vector $\mathbf{y}_{enc}$.
Initialize $\mathbf{y}_{enc}$ as a vector of encrypted zeros (using Encrypt(0));
for *each row i in* $\mathbf{M}$ **do**
  $c_{row\_sum} \leftarrow$ Encrypt(0);
  for *each column j in* $\mathbf{M}$ **do**
    if $\mathbf{M}[i][j] = 1$ **then**
      $c_{row\_sum} \leftarrow c_{row\_sum} \oplus \mathbf{x}_{enc}[j]$;
    end
  end
end
$\mathbf{y}_{enc}[i] \leftarrow c_{row\_sum}$;
return $\mathbf{y}_{enc}$;

In Algorithm 4, the matrix-vector multiplication is used to compute the subsequent system state based on the current encrypted state and the active event.

Algorithm 4: next_state (Homomorphic System Update)

Input: Encrypted current state $\mathbf{x}_{curr}$, clear-text event vector $\mathbf{e}_{clear}$, B-matrices $\mathbf{B}_{\text{event}}$ (clear-text).
Output: Encrypted next state $\mathbf{x}_{next}$.
Initialize $\mathbf{x}_{next}$ as a vector of encrypted zeros;
for *each event e_{name} corresponding to index idx* **do**
  if $\mathbf{e}_{clear}[idx] = 1$ **then**
    $\mathbf{B} \leftarrow \mathbf{B}_{e_{name}}$; `//Retrieve matrix for this event`
    $\mathbf{c}_{\text{contrib}} \leftarrow$ homomorphic_mat_vec$(\mathbf{x}_{curr}, \mathbf{B})$;
    for *each component i of* $\mathbf{x}_{next}$ **do**
      $\mathbf{x}_{next}[i] \leftarrow \mathbf{x}_{next}[i] \oplus \mathbf{c}_{\text{contrib}}[i]$;
    end
  end
end
return $\mathbf{x}_{next}$;

Similarly, Algorithm 5 function employs the matrix-vector multiplication with the requirements matrix (R-matrix) to determine enabled events in the encrypted domain.

Algorithm 5: get_enabled_events (Homomorphic Check)

Input: Encrypted current state $\mathbf{x}_{curr}$, Requirements matrix $\mathbf{R}$ (clear-text).
Output: Encrypted vector of enabled events $\mathbf{e}_{\text{enabled}}$.
$\mathbf{e}_{\text{enabled}} \leftarrow$ homomorphic_mat_vec$(\mathbf{x}_{curr}, \mathbf{R})$;
return $\mathbf{e}_{\text{enabled}}$;

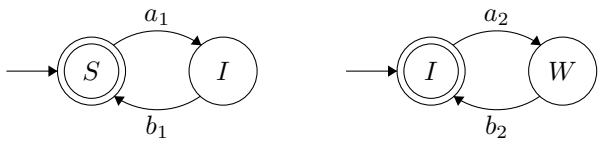
This methodology ensures confidentiality, though its practical security relies on strict ephemeral key non-reuse and hardening against hardware side-channel attacks, which are key areas for future work.

IV. CASE STUDY

The small factory [19] is a well-known toy example composed of two machines connected by a single-unity buffer. While the first machine processes the product and places it in the buffer, the second machine takes the product from the buffer and performs another task. The safety specification is to ensure that the first machine does not place a product in a full buffer and that the second machine does not try to remove a product from the buffer if it is empty, as seen in Figure 2.

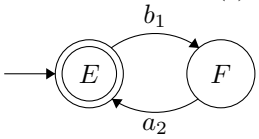
Fig. 2: Small Factory Diagram

The automata representing the dynamic behavior of the machines (M_1 and M_2), as well as the specification, are shown in Figures 3a, 3b and 3c.



(a) Machine M_1

(b) Machine M_2



(c) Specification B

Fig. 3: Automata for the machines (top) and the specification (bottom). (I: Idle, W: Working, E: Empty, F: Full).

It is possible to obtain a non-blocking monolithic supervisor that implements $Sup\mathcal{C}(\mathcal{K}, G)$, as shown in Figure 4. This

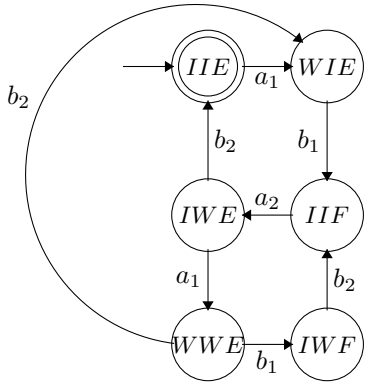


Fig. 4: Small Factory Supervisor

case study was conducted using an ESP32 microcontroller, which is a low cost and widely-used solution in IoT. Also, it was developed in C++ using the Arduino framework and the mdeb-tls library for cryptography.

The chosen elliptic curve is SECP256R1 [20], which is used for digital certificates, Transport Layer Security (TLS) protocols, and blockchain technology. It provides a 256-bit security level and is favored in digital signature algorithms.

Upon initialization, the small factory specific configuration, including its transition rules (B-matrices) and requirement rules (R-matrix), is defined directly in the code.

In B-matrices, each event has an associated B-matrix that describes the state changes it can cause. A 1 at position B[i][j] indicates a possible transition from state j to state i.

- Event a_1: Transitions from state `IIE` to `WIE` and from `IWE` to `WWE`.

$$\mathbf{B}a_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Event b_1: Transitions from state `WIE` to `IIF` and from `WWE` to `IWF`.

$$\mathbf{B}b_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

- Event a_2: Transitions from state `IIF` to `IWE`.

$$\mathbf{B}a_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Event b_2: Transitions from state `IWE` to `IIE`, from state `IWF` to `IIF`, and from `WWE` to `WIE`.

$$\mathbf{B}b_2 = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

The R-matrix defines which events are enabled in which states. A 1 at position R[i][j] means that event i is enabled when the automaton is in state j.

The R-matrix for the small factory is defined as follows:

$$\mathbf{R} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

The supervisor's initial state is then encrypted using the public key. Furthermore, the system simulates the sequence of events. It computes the subsequent state and enabled actions in the ciphertext domain using homomorphic operations. At the end, the encrypted state vector is decrypted using the private key, and the outcome is compared against the non-encrypted data to verify the code.

Imagine the system is in state `IIF`, which physically represents that Machine 1 has finished its task and that the buffer is now full. The active event is a_2, representing Machine 2 starting its work by taking the item from the buffer. The supervisor must securely determine the factory's next state and which actions are now permissible.

First, the state vector for `IIF`, $\mathbf{x}_{\text{current}} = [0,0,0,1,0,0]^T$, is encrypted and sent to the supervisor. For the fourth bit, it uses the public key and a randomly generated 256-bit ephemeral key (e.g., r = 0x1A2B...C3D4) to produce a two-part ciphertext. In the ciphertext domain, the supervisor performs a homomorphic matrix-vector multiplication using this encrypted vector and the public B-matrix for event a_2.

The next state is calculated for each component y_i as a sum over all potential transitions for the active event:

$$y_i = \sum_{k=0}^{\text{num\_events}-1} \left(\sum_{j=0}^{\text{num\_states}-1} \mathbf{B}_{e_k}[i][j] \text{ if } \mathbf{x}_{\text{current}}[j] \text{ is active} \right) \cdot \text{if } \mathbf{e}_{\text{active}}[k] \text{ is active.} \quad (17)$$

A second homomorphic multiplication is then performed between this new encrypted state and the public R-matrix to determine the new set of enabled events:

$$z_k = \sum_{j=0}^{\text{num\_states}-1} \mathbf{R}[k][j] \text{ if } \mathbf{x}_{\text{current}}[j] \text{ is active.} \quad (18)$$

Upon decryption, the results are validated. The next state vector is $[0,0,1,0,0,0]^T$, correctly identifying the new state as `IWE` (Machine 2 is working and the buffer is now empty). The enabled events vector is $[1,0,0,1]^T$. This indicates that events a_1 (Machine 1 can start a new part) and b_2 (Machine 2 can finish its task) are now enabled, which perfectly matches the logical operation of the factory. This correspondence validates that the homomorphic operations correctly preserve the system's logic.

Each transition takes around 2600ms on the ESP32, a cost dominated by the homomorphic matrix-vector multiplications (∼2430ms) rather than encryption (∼86ms) or decryption (∼84ms). This computational cost scales with the number of states (worst-case $O(N^2)$), making the current implementation suitable only for non-critical systems and highlighting this bottleneck as the primary scalability constraint.

V. CONCLUSIONS

This paper introduces and validates a methodology for securing supervisory control of Discrete Event Systems using EC-ElGamal. The decrypted state from the homomorphic computation was identical to the non-encrypted state,confirming that the cryptography preserve the DES logic. This addresses the challenge of ensuring data confidentiality in CPS that operate in insecure communication channels, protecting them against passive attacks.

Future work must address the $O(N^2)$ computational bottleneck of homomorphic_mat_vec via algorithmic optimization. This includes investigating more efficient homomorphic algorithms required by our DES model and exploring the use of less computationally intensive elliptic curves.

For potential research, one direction is the implementation on a Programmable Logic Controller (PLC). Finally, this methodology should be applied to a more complex model, both in terms of computation and memory, to understand the computational cost as the events and states increase.

REFERENCES

[1] F. Basile, G. De Tommasi, S. Dubbioso, and F. Fiorenza, "An Optimization Approach to Current State Opacity Assessment," in *2024 IEEE 63rd Conference on Decision and Control (CDC)*, Milan, Italy, 2024, pp. 491-497.

[2] C. N. Hadjicostis, S. Lafortune, F. Lin, and R. Su, "Cybersecurity and Supervisory Control: A Tutorial on Robust State Estimation, Attack Synthesis, and Resilient Control," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, Cancún, Mexico, 2022, pp. 3020-3040.

[3] S. Oliveira, A. B. Leal, M. Teixeira, and Y. K. Lopes, "A classification of cybersecurity strategies in the context of Discrete Event Systems," *Annual Reviews in Control*, vol. 56, p. 100907, 2023.

[4] V. S. Miller, "Use of elliptic curves in cryptography," in *Advances in Cryptology - CRYPTO '85*, H. C. Williams, Ed. Berlin Heidelberg: Springer-Verlag, 1986, pp. 417-426.

[5] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203-209, 1987.

[6] K. Rabah, "Elliptic Curve ElGamal Encryption and Signature Schemes," *Information Technology Journal*, vol. 4, no. 3, pp. 299-306, 2005.

[7] T. ElGamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," in *Advances in Cryptology. CRYPTO 1984. Lecture Notes in Computer Science*, G. R. Blakley and D. Chaum, Eds. Berlin, Heidelberg: Springer, 1985, vol. 196, pp. 10-18.

[8] Basile, F., Chiacchio, P., & Coppola, J. (2017). "A matrix-based approach for supervising and controlling timed Discrete Event System." *2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC)*. doi:10.1109/icnsc.2017.8000069

[9] Fauser, M., & Zhang, P. (2025). "A Secure Resilient Homomorphic Encryption Scheme for Control Systems." *IEEE Transactions on Automatic Control*, 70(6), 3711-3726. [cite: 313-317, 1176]

[10] C. G. Cassandras and S. Lafortune, "Introduction to Discrete Event Systems". New York, NY, USA: Springer, 2008.

[11] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event systems," *SIAM J. Control Optim.*, vol. 25, no. 5, pp. 1202–1218, 1987.

[12] ACAR, Abbas et al. "A survey on homomorphic encryption schemes: Theory and implementation." *ACM Computing Surveys (Csur)*, v. 51, n. 4, p. 1-35, 2018.

[13] J. Yao, X. Yin, and S. Li, "Sensor Deception Attacks Against Initial-State Privacy in Supervisory Control Systems," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, Cancún, Mexico, 2022, pp. 4839-4845.

[14] L. Lin, R. Tai, Y. Zhu and R. Su,"Heuristic Synthesis of Covert Attackers Against Unknown Supervisors," 2021 60th IEEE Conference on Decision and Control (CDC), Austin, TX, USA, 2021, pp. 7003-7008.

[15] R. Tai, L. Lin, Y. Zhu and R. Su, "Synthesis of the Supremal Covert Attacker Against Unknown Supervisors by Using Observations," in IEEE Transactions on Automatic Control, vol. 68, no. 6, pp. 3453-3468, June 2023.

[16] R. Tai, L. Lin, and R. Su, "Security verification against covert learning attackers," *Automatica*, vol. 177, p. 112344, 2025.

[17] Hussein, Haval I., and Wafaa M. Abduallah. "An efficient ElGamal cryptosystem scheme." International Journal of Computers and Applications 43.10 (2021): 1088-1094.

[18] S. Oliveira, A. B. Leal, M. Teixeira, and Y. K. Lopes, "Integrity of Cyber-Physical Discrete Event Systems under covert actuator attacks," *IFAC PapersOnLine*, vol. 58, no. 1, pp. 198-203, 2024.

[19] W. M. Wonham, "Supervisory Control of Discrete-Event Systems". Toronto, Canada: Systems Control Group, Department of Electrical & Computer Engineering, University of Toronto, 2014.

[20] İşler, Onur. "Implementation and Performance Evaluation of Elliptic Curve Cryptography over SECP256R1 on STM32 Microprocessor. " Cryptology ePrint Archive (2024).